A STUDY OF MAJOR CODING TECHNIQUES FOR DIGITAL COMMUNICATION

by

RICHARD S. SIMPSON
Professor of Electrical Engineering

and

JOSEPH B. CAIN
Research Associate

January 1969

FINAL REPORT NUMBER 101-102

SYSTEMS ENGINEERING GROUP

BUREAU OF ENGINEERING RESEARCH
UNIVERSITY OF ALABAMA UNIVERSITY, ALABAMA

THE UNIVERSITY OF ALABAMA
1831

A STUDY OF MAJOR CODING TECHNIQUES FOR DIGITAL COMMUNICATION

by

Richard S. Simpson
Professor of Electrical Engineering

and

Joseph B. Cain
Research Associate

Final Report Number 101-102

Prepared for

National Aeronautics and Space Administration
George C. Marshall Space Flight Center
Huntsville, Alabama

Under Contract Number

NAS8-20172

University of Alabama Project Number

22-6509

Bureau of Engineering Research
University of Alabama
University, Alabama
January, 1969

ABSTRACT

The performance and complexity of implementation of the major error-control techniques are analyzed. The white-Gaussian-noise channel and the generalized Gilbert channel are used in the performance evaluations. The results allow comparisons among the major coding techniques to be readily made. A new technique, concatenation with inner-code feedback, is presented, and its performance on the Gaussian channel is evaluated. Also, an upper bound on the average digit error probability of a linear block code and the probability distribution of the burst lengths on the generalized Gilbert channel are derived. These results are useful in evaluating the performance of error-control schemes.

ACKNOWLEDGEMENT

The authors would like to express their appreciation to the Telemetry Systems Branch of Astrionics Laboratory, Marshall Space Flight Center, for the support of this work.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

CHAPTER I

INTRODUCTION

Twenty years ago Shannon[1] proved that if the source rate is less than a quantity called the channel capacity, communication over a noisy channel with an error probability as small as desired is possible with proper encoding. Essentially Shannon's work states that signal power, channel noise, and available bandwidth set a limit only on communication rate and not on accuracy.

Gallager's refinements in the statement and proof of Shannon's theorem have given it the following form.[2]

Coding Theorem. The channel is considered to be discrete and memoryless with an input alphabet $a_1, a_2, \cdots, a_K$, an output alphabet $b_1, b_2, \cdots, b_J$, and transition probabilities $P_{jk} = P(b_j|a_k)$. For any code block length N, any number of code words $M = e^{NR}$ (R is the rate in nats, where 1 nat $= \log_2 e$ bits), and any probability distribution on the use of the code words, there exists a code for which the probability of decoding error is bounded by

$$P_W(e) \le \exp\left[-NE(R)\right] \quad , \tag{1.1}$$

where

$$E(R) = \max_{\rho, \bar{p}} \left[-\rho R - \ln \sum_{j=1}^{J} \left[\sum_{k=1}^{K} p_k P_{jk}^{\frac{1}{1+\rho}} \right]^{1+\rho} \right] \quad , \tag{1.2}$$

and the maximization is over all ρ in the range $0 \leq \rho \leq 1$ and all probability vectors $\overline{p} = (p_1, p_2, \cdots, p_K)$. The component p_j of $\overline{p}$ denotes the probability with which the letter a_j is chosen from the input alphabet. The function $E(R)$ is called the error exponent. It is positive, continuous, and convex for all R in the range $0 \leq R < C$, where C is the channel capacity.

Thus, for any rate R in the range $0 \leq R < C$, the probability of error is an exponentially decreasing function of the code block length N. An arbitrarily small error probability can be achieved by increasing the code block length enough. In many cases a problem which is encountered is that the complexity of the decoder increases exponentially with N. In this case the decoder complexity will become excessively large if N is increased indefinitely. It is desirable that the decoder complexity increase only algebraically with N.

In the twenty years since Shannon's classic paper much research has been directed toward finding efficient and practical coding schemes for various types of noisy channels. Most of the progress toward finding practical schemes has come in the last ten years. From all this research it is now clear that coding can provide significant improvements over certain channels in some applications. Much experimental work is being done, and although coding is not yet in widespread use, the use of coding should become much more widespread in the next few years.

Generally, either a probabilistic or an algebraic approach has been used in coding research. With the probabilistic approach an attempt is made to utilize the probabilistic structure of the data-transmission facility in a decoding algorithm. Often random coding techniques are used in proving properties of the algorithm. The algebraic approach is to find codes which have a certain amount of algebraic structure. This algebraic structure is then used in determining practical encoding and decoding schemes.

The type of coding strategy chosen depends upon the error statistics of the communication channel. Channels are classified into two categories, channels with memory and channels without memory. A channel with memory has errors which tend to occur in bursts. A channel without memory, i.e., memoryless, has independent errors. Coding strategies for the two types of channels are often altogether different in concept.

In the past twenty years a large number of technical papers have been written on coding. However, few books have appeared on the subject, and the ones which have appeared have proved to be very difficult reading for most communication engineers. The nature of these books has been such that much attention is given to mathematical detail in the derivation of the encoding and decoding schemes. However, little attention was given to comparisons of the performance and implementation complexity of these coding schemes, particularly over real channels. This tended to make these books too abstract to suit the needs of most communication engineers. In this work a different approach is used. Much attention is given to the performance and implementation complexity of the major coding schemes including coding schemes for both independent and burst errors. Although it is necessary to go into considerable mathematical detail to present the various encoding and decoding techniques, proofs which are documented elsewhere generally are omitted but are referenced for the reader who desires a more complete treatment. So that the communication engineer may make comparisons among the various techniques, extensive data on probability of error and implementation complexity is given. It is hoped that this approach will be enlightening for those who find the study of coding difficult.

In comparing the performance of coding schemes a channel model must be used. The white-Gaussian-noise channel produces independent errors, and it will be used in the performance comparisons of independent error-correcting codes. It was chosen because the comparisons are easily made, since the loss in rate due to the addition of redundancy is accounted for automatically. Also, it is a good model of the space communication channel. It should be emphasized that some codes which are not efficient on the Gaussian channel may be useful on other channels.

The study of coding for burst-error channels is very important because many real communication channels are of this type. Telephone links, high-frequency channels, and ionospheric-scatter and tropospheric-scatter channels all suffer from burst errors. In studying coding for burst-error channels, the Gilbert channel model will be used. With this model the channel is considered to have a good state and a bad state with specified transition probabilities between the two states. This model was chosen because it is simple, it provides insight into burst-error behavior, and it is a good model for some telephone channels.

Fairly general measures of implementation complexity will be used since the final design will depend upon the application and the type of hardware available. The total number of computations per decoded word is calculated, where a computation is an operation which can be performed in one unit of time such as an addition or a shift of a register. Depending upon the type of multiplication unit used a multiplication may be performed as quickly as an addition, or it may take several times as long. The number of binary storage elements is calculated as a relative measure of decoder complexity. There will be no attempt to estimate the amount of control circuitry involved since the total number of computations and the number of binary storage elements will provide a reasonable estimate of the control circuitry required. The encoder complexity is critical in some applications, and the number of binary storage elements is calculated as a relative measure of encoder complexity. The specific application will determine which of the three measures of complexity is most critical.

An introduction to algebra and basic linear code theory is given in Chapter II. Some readers may find this material difficult reading. A thorough understanding of this section is not necessary as a basis for understanding the comparisons which are made between the various coding schemes. However, an understanding of this chapter is necessary in studying the techniques of implementing these codes.

In Chapters III through VII five of the major coding techniques are studied in detail. These are Bose-Chaudhuri-Hocquenghem (BCH) codes, optimum codes for the Gaussian channel, threshold-decodable codes, concatenated codes, and convolutional codes. In each of these cases performance on the Gaussian channel and implementation complexity are studied. For certain decoding algorithms, most notably sequential decoding of convolutional codes, performance can be computed only by extensive simulations. Due to limitations on computer time such simulations were not performed, but results of simulations performed by others are available and will be discussed.

In Chapter VIII the advantage to be gained by the availability of a feedback channel is examined. After a survey of several feedback schemes, a new technique is presented. This technique is concatenation with inner-code feedback. The error exponents obtained with this technique using three inner coding schemes for the Gaussian channel are presented. The derivation of these results is shown in Appendix D.

In Chapter IX several techniques are presented which are applicable to burst-error channels, and the performance of these techniques on the Gilbert channel is analyzed. These performance calculations are facilitated by the result of Appendix E which contains a derivation of the probability distribution of the burst lengths on a Gilbert channel.

Several other appendices are also included. In one method of decoding BCH codes it is necessary to reduce a matrix to upper triangular form. The number of computations required to perform this operation is calculated in Appendix A. In Appendix B the basic principles of optimum receivers are derived. In Appendix C an upper bound to the average digit error probability in decoding a linear block code is developed. This bound is very useful in comparing concatenated codes with other coding systems.

The main purpose of this investigation has been to develop suitable and easily understood comparisons among the major coding techniques. The ease with which these results may be applied in a comparison is shown in the Conclusion. Some results of this investigation will find other applications. A new technique, concatenation with inner-code feedback, is analyzed, and it is shown that communication at rates very near channel capacity is possible with low error probability. In Appendix C a tight upper bound to the average digit error probability of a linear block code is developed. This bound is very useful in comparing performance of error-control systems. In Appendix E a detailed analysis of the generalized Gilbert channel which has not appeared elsewhere is presented. In addition the probability distribution of the burst lengths on this channel is derived. This distribution is very useful in evaluating the performance of error-control schemes on the Gilbert channel.

CHAPTER II
LINEAR CODES FOR THE DISCRETE CHANNEL

In this chapter the fundamentals necessary for a study of coding will be introduced. Although the treatment will be complete, it will not be rigorous due to the length of such a treatment. More rigorous treatments of the topics discussed in this chapter may be found in Carmichael[3] and Peterson[4]. A block diagram of a typical error-control system is shown in Fig. 2.1. The channel is discrete with q inputs and q outputs, i.e., the channel consists of both the modulator and demodulator as well as the transmission link. The purpose of the encoder and decoder is to reduce the frequency of errors in the data sent from the data source to the user. The encoder takes blocks of k information digits and adds redundant digits in a systematic, controlled manner. Then the decoder can use the added redundancy to correct some of the possible error patterns which may have occurred. Linear code theory is used in finding good codes, in determining their error-correcting capability, and in finding practical encoding and decoding methods. The algebraic properties of codes form the basis for the study of linear codes. Some fundamentals of algebra will be introduced in Part A, and these fundamentals will be applied in Part B in the study of linear codes.

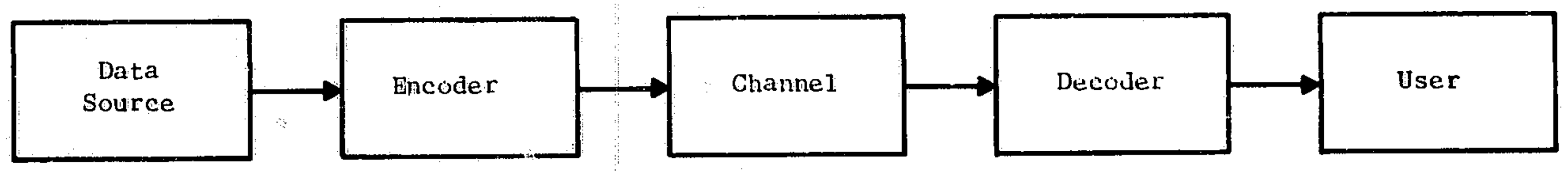

Fig. 2.1. Coded Communication System.

A. Fundamentals of Algebra

Algebraic systems, which are systems satisfying certain rules or laws, play a prominent role in coding theory. The systems of interest to the coding theorist are the group and the field.

Definition 2.1. A group G is a set of elements for which one operation, addition or multiplication, and its inverse, subtraction or division, are defined, and closure holds, i.e., if the operation is applied to two elements of G, the result is an element of G.

Definition 2.2. A field F is a set of elements which is an additive commutative group ($a + b = b + a$), its nonzero elements form a multiplicative commutative group ($ab = ba$), and for which the distributive law ($a(b + c) = ab + ac$) holds.

It is not difficult to find examples of groups and fields. The positive and negative integers with addition as the operation form a group, and the set of positive rational numbers with multiplication as the operation also form a group. The set of all real numbers and the set of all rational numbers both form fields. In each of these examples of groups and fields, the number of elements is infinite.

The Euclidean division algorithm for integers is used quite frequently in developing properties of algebraic systems.

Euclidean Division Algorithm for Integers. For every pair of integers s and d there is a unique pair of integers q, the quotient, and r, the remainder, such that

$$s = dq + r, \quad 0 \le r < |d| \quad . \qquad (2.1)$$

Although the examples of groups and fields which have been given contain an infinite number of elements, the groups and fields of interest to the coding theorist contain a finite number of elements since a code will consist of a finite number of code words and have an alphabet with a finite number of symbols. Under certain conditions the set of integers modulo p forms a field.

Definition 2.3. The set of <u>integers modulo p</u> is the set 0, 1, $\cdots$, p-1, with the following rules for addition and multiplication:

1. $a + b = c$, where c is the remainder term of $\frac{(a+b)}{p}$, $0 \le c \le p-1$.
2. $a \cdot b = d$, where d is the remainder term of $\frac{a \cdot b}{p}$, $0 \le d \le p-1$.

The process in Definition 2.3 of keeping only the remainder (residue) upon division by p is called a reduction modulo p. The set of integers which have the same remainder (residue) after division by p are said to belong to the same residue class.

Definition 2.4. Two integers a and b are said to be congruent modulo p, denoted by

$$a \equiv b \pmod{p} \quad , \qquad (2.2)$$

if they have the same residue after division by p.

It is easily seen that the set of integers modulo p has p residue classes each of which contains an integer, s, in the range $0 \leq s \leq p-1$. It can be shown that the set of integers modulo p forms a field if and only if p is a prime number.[5] This field is called a prime field or a Galois field of p elements, GF(p).

The integers modulo 2, GF(2), is the field most often used in digital communication. Accordingly, most of the codes studied in this work will be binary codes, although some codes will be studied which use symbols from other fields. The field, GF(2), contains only the elements 0 and 1 since addition and multiplication of elements is followed by a reduction modulo 2. Thus, the reader can easily verify that the addition and multiplication tables for GF(2) are those shown in Table 2.1.

Table 2.1. Addition and Multiplication Tables for GF(2).

+	0	1	·	0	1
0	0	1	0	0	0
1	1	0	1	0	1

The only example of a finite field, given thus far is GF(p) with p a prime number. For every number q that is a power of a prime number ($q = p^m$ with p a prime number), there is a field with q elements. In order to introduce the representation of this field several additional definitions are needed.

Definition 2.5. A polynomial with coefficients from GF(p) is called irreducible if it can be factored no further in GF(p) or, in other words, has no roots from GF(p).

Euclidean Division Algorithm for Polynomials. For every pair of polynomials s(X) and d(X), there is a unique pair of polynomials q(X), the quotient, and r(X), the remainder, such that

$$s(X) = d(X)q(X) + r(X) , \qquad (2.3)$$

and the degree of r(X) is less than the degree of d(X).

In a manner analogous to that for integers the concepts of residue classes and congruence also apply to polynomials. It can be shown that if d(X) is a polynomial irreducible over GF(p), the set of polynomials with coefficients from GF(p) modulo d(X) is a representation of $GF(p^m)$, where m is the degree of d(X).

A field formed by taking polynomials with coefficients from GF(p) modulo an irreducible polynomial d(X) of degree m is called an extension field of degree m over GF(p). The field elements are represented by polynomials of degree less than m with coefficients from GF(p). Since the field consists of all polynomials of degree less than m over GF(p), there are p^m elements in the extension field. It is also said that the field $GF(p^m)$ has characteristic p and that GF(p) is a subfield of $GF(p^m)$.

The polynomial $X^{2^m-1} + 1$ can be factored into 2^m-1 linear factors

$$X^{2^m-1} + 1 = (X + \beta_1)\cdots(X + \beta_{2^m-1}) , \qquad (2.4)$$

where the roots β_i are the nonzero elements of $GF(2^m)$. These roots can be expressed as powers of a primitive root α, i. e., the 2^m-1 nonzero field elements may be expressed as

$$\alpha, \alpha^2, \ldots, \alpha^{2^m-1} = 1 = \alpha^0 \quad . \qquad (2.5)$$

If α^i is an element of GF(2^m), the minimum polynomial, $m_i(X)$, of α^i is the monic polynomial of smallest degree with coefficients in GF(2). It can be shown that all the roots of $m_i(X)$ are contained in the sequence $\alpha^i, \alpha^{2i}, \alpha^{4i}, \alpha^{8i}, \ldots$.[6]

As an example the polynomial X^7+1 has three irreducible factors, i.e.,

$$X^7+1 = (X+1)(X^3+X+1)(X^3+X^2+X+1) \quad . \qquad (2.6)$$

If α is a root of X^3+X+1, the other roots are α^2 and α^4. The roots of X^3+X^2+X+1 are α^3, α^5, and α^6, and the root of $X+1$ is $\alpha^7 = 1$. The nonzero elements of GF(2^3) are represented in Table 2.2 by three methods.

Table 2.2. Representation of the Nonzero Elements of GF(2^3).

$\alpha^0 =$	1	$= (100)$
$\alpha^1 =$	α	$= (010)$
$\alpha^2 =$	α^2	$= (001)$
$\alpha^3 =$	$1+\alpha$	$= (110)$
$\alpha^4 =$	$\alpha+\alpha^2$	$= (011)$
$\alpha^5 =$	$1+\alpha+\alpha^2$	$= (111)$
$\alpha^6 =$	$1+\alpha^2$	$= (101)$
$\alpha^7 =$	1	$= \alpha^0$.

In the first column the field elements are represented as powers of a primitive element α. In the second column the field elements are represented as polynomials in α with degree less than three. In the third column the elements are represented as vectors with components equal to the coefficients in the polynomial representation.

The concept of order is an important one.

Definition 2.6. The order, e, of a field element, α, is the least positive integer such that

$$\alpha^e = 1 \quad . \tag{2.7}$$

The order of a primitive element of a field is equal to the number of nonzero field elements.

A property which will be useful in decoding is that with elements from GF(2^m) squaring is a linear operation, i.e., if α and β are elements of GF(2^m),

$$\begin{aligned}(\alpha+\beta)^2 &= \alpha^2 + \alpha\beta + \alpha\beta + \beta^2 \\ &= \alpha^2 + \beta^2 \quad .\end{aligned} \tag{2.8}$$

This result follows from the fact that GF(2^m) has characteristic 2.

Code words are generally represented either by polynomials in X with coefficients from an appropriate field or by vectors with components from an appropriate field. Although the two representations are equivalent, the vector approach is sometimes more convenient in presenting the basic properties of codes while the polynomial approach is useful in determining simple implementation schemes. The material necessary for presenting the polynomial approach has already been introduced; however, some additional definitions are needed before presenting the vector approach.

Definition 2.7. A <u>subgroup</u> H is a subset of elements of a group G such that the elements of H form a group.

To determine if H is a subgroup it is necessary only to check for closure.

Definition 2.8. An <u>n-tuple</u> is an ordered set of n field elements denoted by $(a_1, a_2, \cdots, a_n)$. Addition of n-tuples is defined by

$$(a_1, a_2, \cdots, a_n) + (b_1, b_2, \cdots, b_n)$$
$$= (a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n) \quad . \qquad (2.9)$$

Multiplication of an n-tuple by a field element is defined by

$$c(a_1, a_2, \cdots, a_n) = (ca_1, ca_2, \cdots, ca_n) \quad . \qquad (2.10)$$

With these definitions the set of all n-tuples over a field forms a vector space.

The reader can also easily see that the set of all n-tuples over a field forms a group. If the field elements are from GF(q), there are q^n elements in this group, and thus the group is finite. A subspace is defined to be a subset of a vector space which is also a vector space. This is analogous to the definition of a subgroup. To verify that a subset of a vector space is a subspace, it is necessary only to check for closure under addition and multiplication by scalars.

Definition 2.9. A set of vectors $v_1, v_2, \cdots, v_n$ is <u>linearly dependent</u> if and only if there are scalars $c_1, c_2, \cdots, c_n$ not all zero such that

$$c_1v_1 + c_2v_2 + \cdots + c_nv_n = 0 \quad . \qquad (2.11)$$

A set of vectors which is not linearly dependent is linearly independent.

Definition 2.10. A basis of a vector space is a set of linearly independent vectors such that every vector in the vector space is a linear combination of the basis vectors. If there are n basis vectors, the space has dimension n.

Definition 2.11. The dot product of two n-tuples is defined by

$$(a_1,a_2,\cdots,a_n) \cdot (b_1,b_2,\cdots,b_n) = a_1b_1 + a_2b_2 + \cdots + a_nb_n \ , \quad (2.12)$$

which is a scalar quantity.

Definition 2.12. Two vectors are orthogonal if and only if their dot product is zero.

The set of all n-tuples orthogonal to a subspace V_1 of n-tuples forms a subspace V_2 of n-tuples called the null space of V_1. If the dimension of V_1 is k, the dimension of V_2 is n-k. The concepts of subspaces, basis vectors, orthogonality, and null spaces are very important in establishing the basic theory of linear codes as the reader will see in the next section.

B. Linear Code Theory

Linear codes are most easily introduced by using the vector approach.

Definition 2.13. A linear code is a set of vectors which form a subspace of the vector space of all n-tuples.

Definition 2.14. The Hamming weight of a vector v denoted by w(v) is the number of nonzero components.

Definition 2.15. The Hamming distance between two vectors v_1 and v_2 is the number of positions in which they differ or $w(v_1 - v_2)$.

Since a linear code is a subspace, any combination of two vectors is another code vector and vice versa. Thus, the minimum Hamming distance between all pairs of code vectors for a linear code equals the minimum Hamming weight of its nonzero vectors. The minimum distance of a code is important since it determines the error-correcting properties of the code.

A linear code is a mapping of a k-tuple containing the information symbols into a n-tuple containing both information and parity symbols. A convenient method of describing a linear code V is by using the generator matrix G. To construct this matrix a set of basis vectors for the linear code is found and used as the rows of G. If each code vector is to contain k information digits, then the dimension of V is k, and G has k rows. The code vectors are all possible linear combinations of the rows of G. If the field has q elements, there are q^k code vectors. Such a code is called an (n,k) code. The code vector v corresponding to the k-tuple v_k of information digits can be generated easily from G by the relation

$$v = v_k G \quad . \tag{2.13}$$

A matrix related to G which is also useful is the parity check matrix. The null space of V denoted by V_1 has dimension (n-k). The parity check matrix H is formed by using a basis for V_1 as rows. Since V_1 is the null space of V, all rows of H are orthogonal to all linear combinations of rows of G, or

$$Hv^T = 0 \tag{2.14}$$

for all code vectors v. All solutions to the system of equations (2.14) are code vectors. The parity check matrix can be used to determine how to implement a code as will be shown later in an example. The process of determining the code word from the information digits is called encoding. Usually the encoding process is simple; however, the decoding process is often quite complicated. For short codes such as the Hamming (7,4) code, decoding is not complicated, but such a short code will not give a large decrease in error probability. Large decreases in error probability can be obtained only by using codes with long block lengths and considerable error-correcting capability. The use of such codes in turn requires a decoder of considerable complexity.

One decoding scheme often discussed is maximum-likelihood decoding. Assuming that a binary (n,k) code is being used over the binary symmetric channel and that all code vectors are equally likely, the best decoding rule is to decode the received vector into a code vector that differs from the received vector in the fewest positions. This scheme is called maximum-likelihood or minimum distance decoding. It could be implemented by having a list of possible code vectors at the decoder and determining the Hamming distances between the received vector and the possible code vector. The code vector corresponding to the minimum Hamming distance would be chosen. The complexity of this decoding scheme, however, is proportional to the number of code vectors which is 2^k. For large values of k, the decoder complexity is unreasonable. It is desirable that the complexity of the decoder increase only algebraically with k. The algebraic structure of a code often leads to a decoding scheme of reasonable complexity.

Knowledge of the number of errors that a particular coding and decoding scheme is capable of correcting allows calculation of the probability of error. Assuming that an (n,k) code is used on a binary symmetric channel with transition probability p, the probability that i errors occur in i specified positions of the received word is $p^i(1-p)^{n-i}$. It is also assumed that the coding and decoding scheme is capable of correcting $\alpha(i,n)$ different patterns of i errors in n digits. Then the probability of correct decoding is

$$P(c) = \sum_{i=0}^{n} \alpha(i,n) p^i (1-p)^{n-i} \quad , \tag{2.15}$$

and the probability of error is

$$P(e) = 1 - P(c) \quad . \tag{2.16}$$

For a code which can correct all combinations of t or fewer errors and no combinations of more than t errors, the probability of correct decoding is

$$P(c) = \sum_{i=0}^{t} \binom{n}{i} p^i (1-p)^{n-i} \quad . \tag{2.17}$$

From the maximum-likelihood decoding scheme is is obvious that to correct t errors the minimum Hamming distance between code words must be at least $2t+1$.

The syndrome of the received vector is used in decoding. Letting the received vector, r, be the sum of a transmitted vector, v, and an error vector, e, the syndrome is defined as

$$S = Hr^T \quad . \tag{2.18}$$

The syndrome is a vector with a number of components equal to the number of rows of H which is (n-k). Since

$$\begin{aligned} S &= H(v^T + e^T) \\ &= Hv^T + He^T \\ &= He^T \quad , \end{aligned} \tag{2.19}$$

the reader can see that the syndrome depends on the error vector but not on the code vector. Thus, there is a set of possible received vectors the same size as the set of code words which all have the same syndrome. Each set of received vectors having this property is called a coset of the set of code vectors. The vector in each coset with the smallest Hamming weight will be called the coset leader.

The standard array shown in Fig. 2.2 can be used as a decoding table.

$$\begin{array}{llll} v_1 & v_2 & v_3 & \cdots v_r \\ e_1 & v_2 + e_1 & v_3 + e_1 & \cdots v_r + e_1 \\ e_2 & v_2 + e_2 & v_3 + e_2 & \cdots v_r + e_2 \\ \vdots & \vdots & \vdots & \vdots \\ e_m & v_2 + e_m & v_3 + e_m & \cdots v_r + e_m \end{array}$$

$$v_1 = (0,0,\cdots,0), \; r = 2^k, \; m = 2^{n-k}-1 \quad .$$

Fig. 2.2. The Standard Array.

All code vectors are listed in the first row with the all zeros vector listed first. The first member of the second row is chosen by picking an n-tuple, e_1, of minimum weight from the remaining n-tuples. For $i > 1$ the member of the second row in the i'th column is formed by adding e_1 to v_i, the member of the first row in the i'th column. The first member of the third row is chosen by picking an n-tuple, e_2, of minimum weight from the remaining n-tuples. For $i > 1$ the member of the third row in the i'th column is formed by adding e_2 to v_i. This process is continued until all 2^n n-tuples are listed in the array. Each row is a coset and each member of the first column is a coset leader.

The standard array provides a convenient method of implementing minimum distance decoding. All members of a given row have the same syndrome, and each row has a different syndrome. The first row has a syndrome of all zeros. If the syndrome is zero, it is assumed that no error occurred, and the received vector is accepted. If the syndrome is nonzero, the coset leader corresponding to the syndrome is assumed to be the error vector and is subtracted from the received vector to get the most probable transmitted vector. That this is a minimum distance decoding scheme follows from the fact that coset leaders were chosen to be vectors of minimum weight.

As an example, implementation of the binary Hamming (7,4) code will be discussed. The parity check matrix is chosen as

$$H = \begin{bmatrix} 0&0&0&1&1&1&1 \\ 0&1&1&0&0&1&1 \\ 1&0&1&0&1&0&1 \end{bmatrix} . \qquad (2.20)$$

By using as the i'th column of H the binary representation of the number i, a very simple decoding scheme is obtained. Letting the code vectors be represented by

$$v = (p_1,p_2,v_1,p_3,v_2,v_3,v_4) \quad , \tag{2.21}$$

where p_1, p_2, and p_3 are parity checks, from (2.14) the following equations for parity checks are obtained

$$p_3 = v_2 + v_3 + v_4 \quad , \tag{2.22}$$

$$p_2 = v_1 + v_3 + v_4 \quad , \tag{2.23}$$

and

$$p_1 = v_1 + v_2 + v_4 \quad . \tag{2.24}$$

Decoding is straightforward since if a single error occurs, the syndrome is equal to the column of H corresponding to the position in error, which in this case is the binary representation of the position in error. Thus, if there is an error in the sixth position

$$\begin{aligned} S &= H(v^T + e_6^T) \\ &= He_6^T \\ &= H\begin{bmatrix}0\\0\\0\\0\\0\\1\\0\end{bmatrix} \\ &= \begin{bmatrix}1\\1\\0\end{bmatrix} \quad , \end{aligned} \tag{2.25}$$

which indicates an error in the sixth position. This is a good example where mathematical structure is used to facilitate decoding. Much research in coding theory is directed toward finding codes with a mathematical structure that will lead to a simple decoding algorithm.

An important subclass of linear codes is the class of cyclic codes. These codes have the property that if $v = (a_0, a_1, a_2, \cdots, a_{n-1})$ is a code vector, then $v_1 = (a_{n-1}, a_0, a_1, \cdots, a_{n-2})$, obtained by shifting the components of v cyclically one unit to the right, is also a code vector. Cyclic codes are very important since they are easily implemented and have excellent performance. The polynomial representation is best for these codes because with this representation encoding and decoding methods are readily derived. The code vector v mentioned above can be represented by the polynomial

$$f(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1} \quad . \qquad (2.26)$$

A cyclic code can be specified by a polynomial $g(X)$ that divides $X^n - 1$. If $g(X)$ has degree r, the code has dimension $k = n - r$. The set $g(X)$, $Xg(X)$, $\cdots$, $X^{n-r-1}g(X)$ can be taken as basis vectors. With these basis vectors each vector $f(X)$ is a code vector if and only if it is divisible by $g(X)$. The polynomial $g(X)$ is called the generator of the code. The generator may alternately be specified by specifying the roots a_1, a_2, $\cdots$, a_r of $g(X)$. Then if $m_i(X)$ is the minimum polynomial of a_i, $g(X)$ must be divisible by $m_1(X)$, $m_2(X)$, $\cdots$, $m_r(X)$ and hence by their least common multiple. Thus,

$$g(X) = \text{LCM}\left[m_1(X), m_2(X), \cdots, m_r(X)\right] \quad . \qquad (2.27)$$

Occasionally polynomials will be multiplied resulting in polynomials of degree n or greater. Since it will be desirable to keep the degree of all polynomials less than n, all polynomials will be reduced modulo some polynomial $n(X)$ of degree n. Of course, this is done by dividing by n(X) and keeping only the remainder (residue). By choosing $n(X) = X^n-1$, multiplication by X is the same as a cyclic shift, i.e.,

$$\begin{aligned} Xf(X) &= X(a_0 + a_1X + \cdots + a_{n-1}X^{n-1}) \\ &= a_{n-1} + a_0X + a_1X^2 + \cdots + a_{n-2}X^{n-1} + a_{n-1}(X^n-1) \\ &\equiv (a_{n-1} + a_0X + \cdots + a_{n-2}X^{n-1})\mathrm{mod}(X^n-1) \quad . \end{aligned} \tag{2.28}$$

Linear finite-state switching circuits are of much importance in the implementation of codes, particularly cyclic codes. The three elements used in building these circuits are represented in Fig. 2.3. These elements are the GF(q) adder, the GF(q) storage device, and the device which multiplies its input by a constant A in GF(q). The output of the storage device is the same as its input one unit of time earlier. In the binary case the adder would be a modulo-2 adder, the storage device would be a flip-flop, and the multiplier would be simply the presence or absence of a connection.

Using these elements a circuit can be constructed for multiplying any input polynomial,

$$a(X) = a_0 + a_1X + \cdots + a_{k-1}X^{k-1} + a_kX^k \ , \tag{2.29}$$

by the fixed polynomial,

$$h(X) = h_0 + h_1X + \cdots + h_{r-1}X^{r-1} + h_rX^r \quad . \tag{2.30}$$

The product is

$$\begin{aligned} a(X)h(X) = a_0h_0 &+ (a_0h_1 + a_1h_0)X \\ &+ (a_0h_2 + a_1h_1 + a_2h_0)X^2 + \cdots \\ &+ (a_{k-2}h_r + a_{k-1}h_{r-1} + a_kh_{r-2})X^{k+r-2} \\ &+ (a_{k-1}h_r + a_kh_{r-1})X^{k+r-1} + a_kh_rX^{k+r} \quad . \end{aligned} \tag{2.31}$$

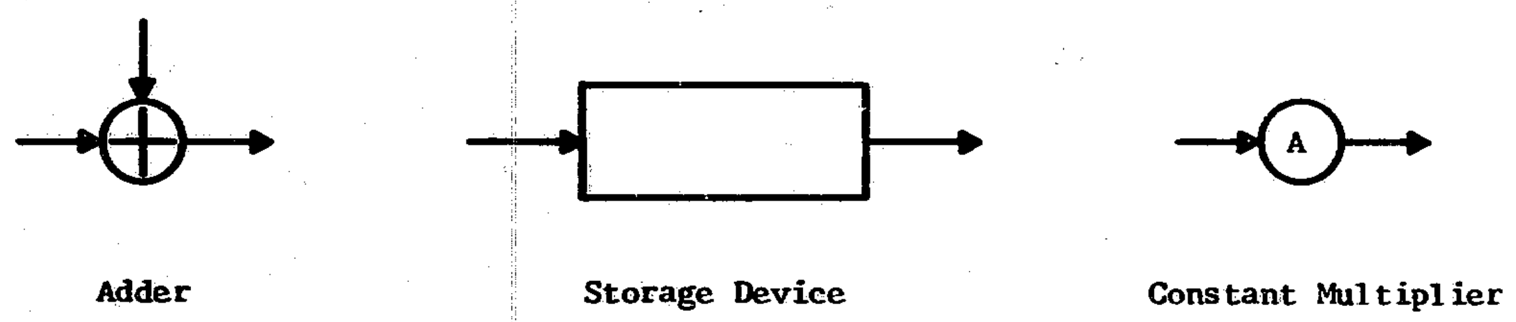

Fig. 2.3. Elements for Building Linear Finite-State Switching Circuits.

A circuit for performing this multiplication is shown in Fig. 2.4. The storage elements are initially set to zero, and the coefficients of a(X) are shifted in, high order first, and followed by r zeros. The product coefficients are developed in the shift register as will be evident from the following discussion. The first symbol a_k is shifted in resulting in an output $a_k h_r$, which is the coefficient of X^{k+r}. At this time the storage devices contain $a_k h_0, a_k h_1, \cdots, a_k h_{r-1}$. When the next input, a_{k-1}, is shifted in, the output is $a_k h_{r-1} + a_{k-1} h_r$, which is the coefficient of X^{k+r-1}. At this time the storage devices contain $a_{k-1} h_0, a_k h_0 + a_{k-1} h_1, \cdots, a_k h_{r-2} + a_{k-1} h_{r-1}$. The operation continues in this manner. After r shifts the term $a_k h_0 + a_{k-1} h_1 + a_{k-2} h_2 + \cdots + a_{k+r-1} h_{r-1} + a_{k-r} h_r$ appears at the output, which is the correct coefficient of X^k.

A linear finite-state switching circuit can also be constructed for dividing the polynomial $d(X) = d_0 + d_1 X + \cdots + d_n X^n$ by the fixed polynomial $g(X) = g_0 + g_1 X + \cdots + g_r X^r$. It is shown in Fig. 2.5. Again the storage elements initially contain zeros. The coefficients of d(X) are shifted in, high order first. The output will be zero for the first r shifts, and the first nonzero output is $d_n g_n^{-1}$, which is the correct first quotient coefficient. The second quotient output will be $g_r^{-1}(d_{n-1} - d_n g_r^{-1} g_{r-1})$, which is also correct as can be verified by long division. For each quotient coefficient q_j, the polynomial $q_j g(X)$ must be subtracted from the dividend. This operation is performed by the feedback connections. After n shifts all quotient coefficients have appeared at the output, and the remainder is left in the shift register.

By combining the two circuits that have just been discussed a single shift register circuit that multiplies by h(X) and divides

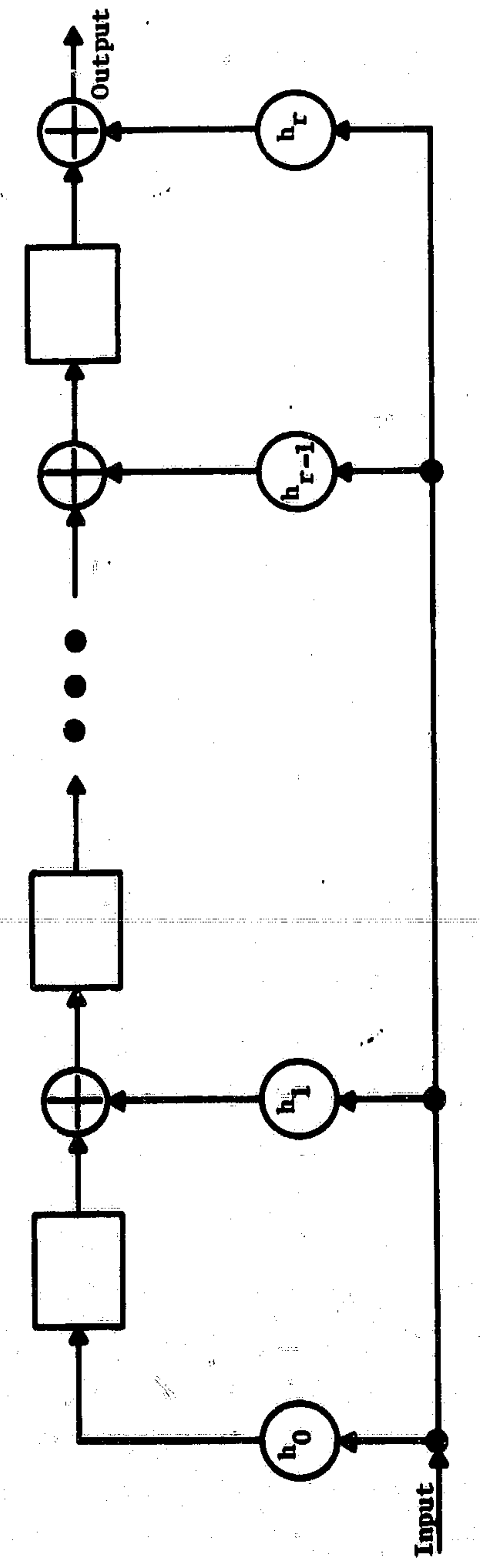

Fig. 2.4. Circuit for Multiplying Polynomials.

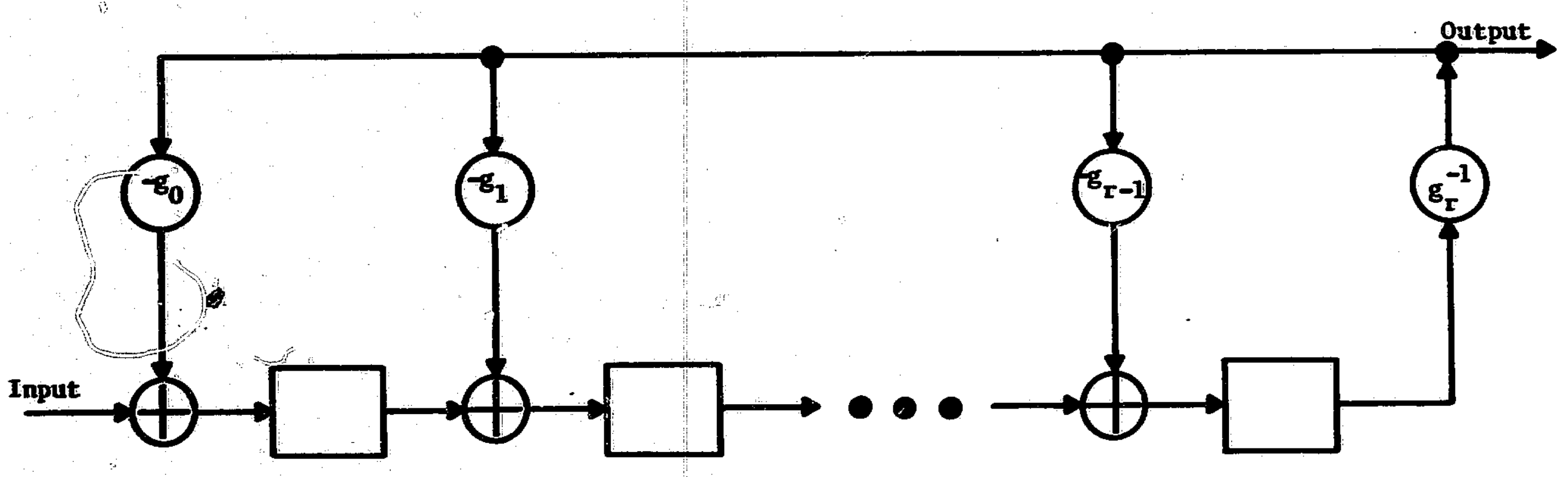

Fig. 2.5. Circuit for Dividing Polynomials.

by g(X) is obtained. This circuit is shown in Fig. 2.6. It is ass"med that the degree of h(X) is no greater than the degree of g(X). This particular circuit will be used later in calculating the syndrome for shortened cyclic burst-error-correcting codes.

By using a special case of Fig. 2.6, a shift-register encoder for a cyclic code can be constructed. Assuming g(X) is the generator polynomial of the code and letting $f_o(X)$ be a polynomial in which the k coefficients of X^{n-1}, $X^{n-2}, \cdots, X^{n-k}$ are arbitrary information symbols and there are no terms of lesser degree, then by (2.3)

$$f_o(X) = g(X)q(X) + r(X) \quad , \qquad (2.32)$$

where r(X) has degree less than n-k, the degree of g(X). Since

$$f_o(X) - r(X) = g(X)q(X) \quad , \qquad (2.33)$$

it is evident that $f_o(X) - r(X)$ is a code vector because it is divisible by g(X). Also since r(X) has degree less than n-k, the information symbols are in $f_o(X)$, and the parity check symbols are in $-r(X)$. Thus, to generate the check symbols for a code word only the remainder, r(X), after dividing $f_o(X)$ by g(X), need be calculated.

Since $f_o(X) = X^{n-k}f(X)$ where f(X) is a polynomial of degree less than k, the calculation of the remainder, r(X), of (2.33) is simplified by using the circuit of Fig. 2.6 to multiply f(X) by $h(X) = X^{n-k}$ and divide by g(X). The circuit for performing this operation is shown in Fig. 2.7. To operate the encoder the k information symbols are shifted in while simultaneously transmitting them over the communication channel. When all information symbols have entered the register, the remainder is stored in the register. The feedback

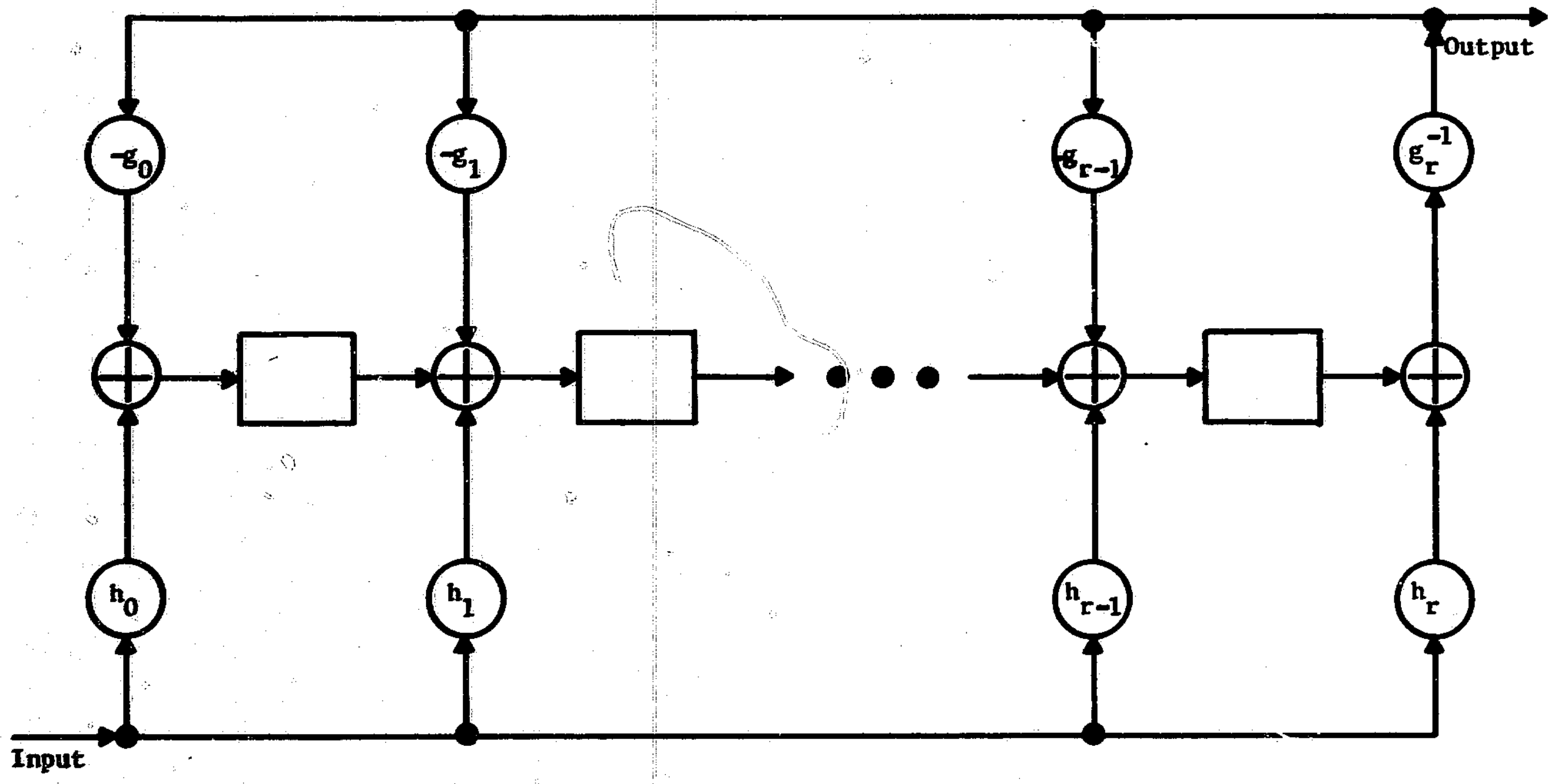

Fig. 2.6. Circuit for Multiplying by h(X) and Dividing by g(X).

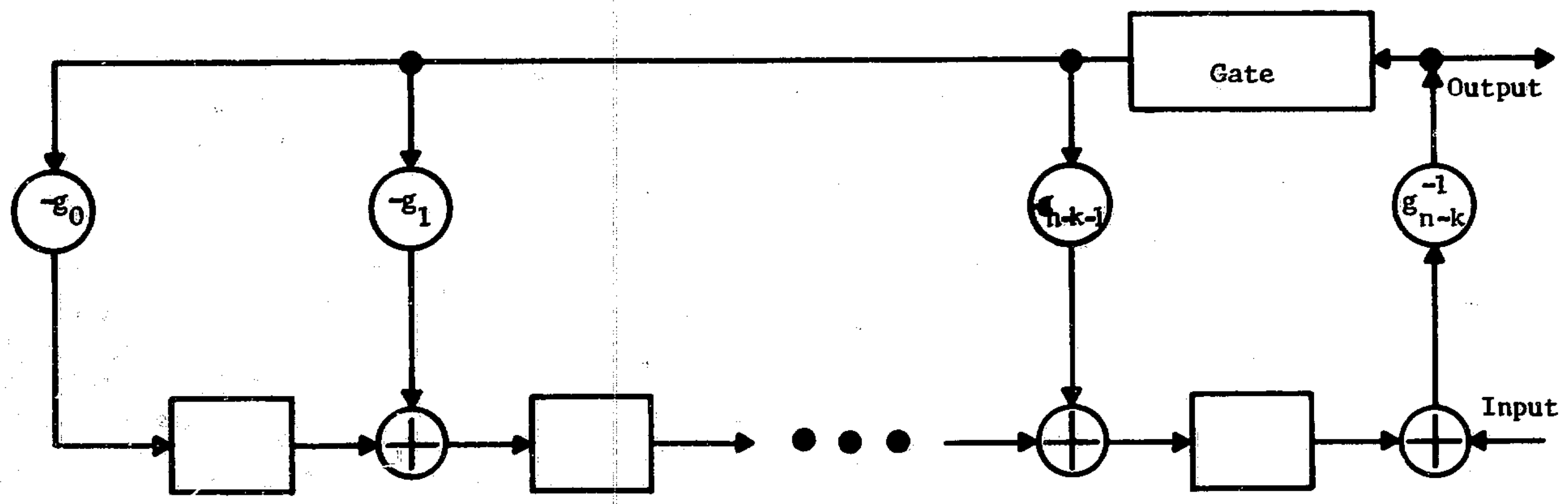

Fig. 2.7. Encoder Using (n-k) Shift Register Stages.

circuit is then disabled at the gate, and the remainder is shifted out with the signs of the symbols being changed as they leave the register. These are the check symbols. Since the circuit of Fig. 2.5 is also a division circuit, it could also be used as an encoder. However, since there is no automatic premultiplication by X^{n-k}, the information symbols would have to be shifted in followed by n-k zeros before the remainder is available. This is a distinct disadvantage, and the circuit of Fig. 2.7 is generally used.

A cyclic code may also be generated by using a feedback shift register of length k. First, the polynomial

$$h(X) = \frac{X^n-1}{g(X)} = h_0 + h_1X + \cdots + h_{k-1}X^{k-1} + X^k \qquad (2.34)$$

is computed and the feedback connections are determined by the coefficients h_0, h_1, $\cdots$, h_k, according to the diagram of Fig. 2.8. If the contents of the registers are shifted left, the input to the right-most stage is

$$a_{i+k} = -\sum_{j=0}^{k-1} h_j a_{i+j} \quad , \qquad (2.35)$$

or

$$\sum_{j=0}^{k} h_j a_{i+j} = 0, \; h_k = 1 \quad . \qquad (2.35)$$

Thus the output of this shift register encoder is a solution of the recurrence relation or difference equation (2.35). Since the solutions generated are dependent on the initial values in the k shift register stages, a total of q^k different solutions are obtained. This is equal to the number of different code words in a linear code over GF(q) with k information symbols per code word. In a

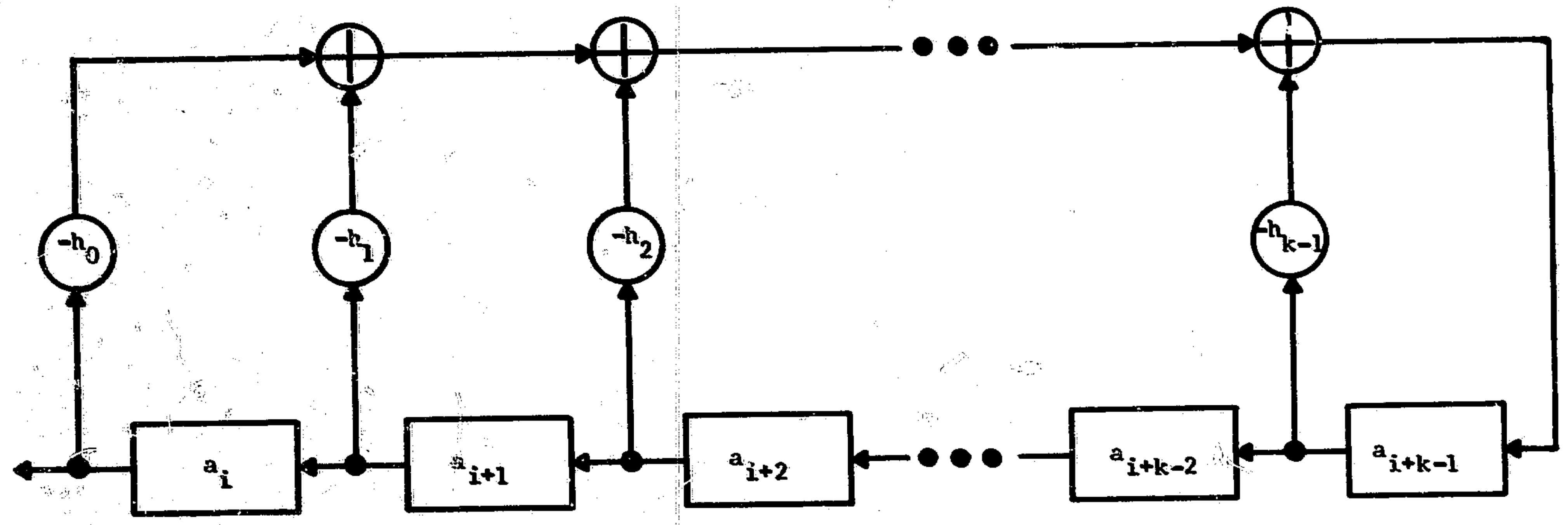

Fig. 2.8. Encoder Using k Shift Register Stages.

quite lengthy proof which will be omitted here, Peterson shows that the solutions of the recurrence relation (2.35) have period n and are code words in the linear code generated by g(X).[7] The operation of this encoder is simple. The k information symbols are stored initially in the k storage devices, and then the encoder is shifted n times. The first k symbols coming out of the encoder are information symbols, and the last n-k symbols are parity check symbols. This method is not used as frequently as the one shown in Fig. 2.7. Generally codes with a high rate are desirable so that $(n-k) < k$; therefore, the encoder of Fig. 2.7 is simpler. In both methods of code generation the higher-order symbols of a polynomial are transmitted first. This convention will always be followed.

In several decoding algorithms it is required to multiply two elements of $GF(2^m)$. A very useful technique for implementing this multiplication described by Berlekamp[8] requires two m-stage registers for storing the multiplier and multiplicand. This method will be demonstrated by an example. Letting α denote an element of $GF(2^5)$ with minimal polynomial $M(X) = X^5 + X^2 + 1$, the multiplicand, U, is stored in the feedback shift register of Fig. 2.9 which is wired to replace U by $U\alpha$ with each left shift. The multiplier, V, is stored in the register of Fig. 2.10 which may be shifted cyclically to the right. The product, $Z = UV$, is accumulated in a register to which the U-register may be added. The procedure is simple. If $V_0 = 1$, the U-register is added to the Z-register. Then both the U and V-registers are shifted and V_0 is examined again to determine whether the U-register is added to the Z-register. After a total of four shifts the multiplication is complete. This method is particularly

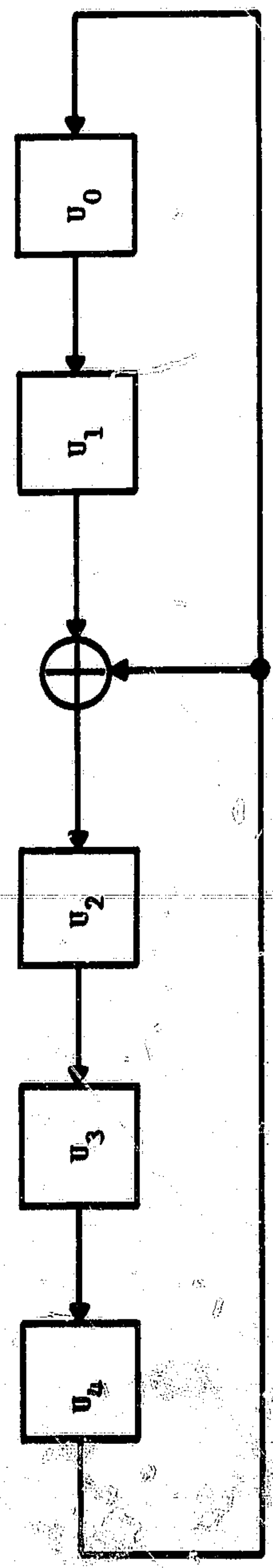

Fig. 2.9. Multiplicand Register.

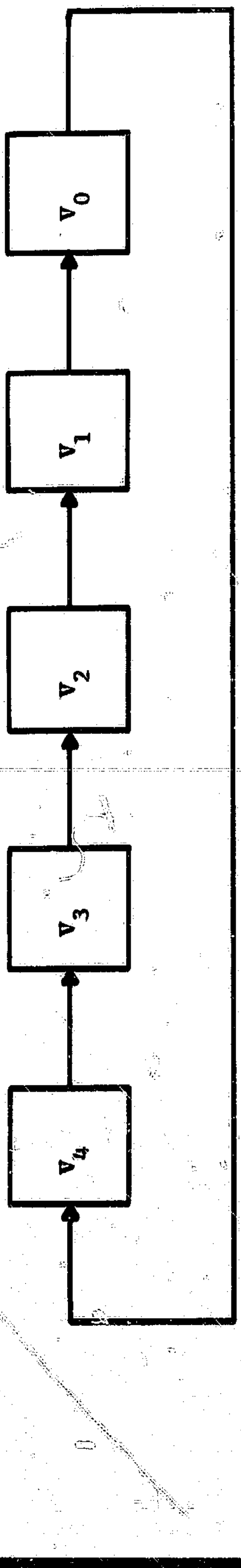

Fig. 2.10. Multiplier Register.

useful since the registers of Figs. 2.9 and 2.10 can be used for storage as well as the multiplication process.

CHAPTER III
BOSE-CHAUDHURI-HOCQUENGHEM CODES

The class of binary codes considered in this chapter were discovered by Bose and Chaudhuri[9] and independently by Hocquenghem[10]. These codes were generalized to GF(q) first by Reed and Solomon[11] for the special case $n = q - 1$ and later by Gorenstein and Zierler[12] for other values of n. This class of codes is generally considered to be one of the best classes of linear codes for the correction of independent errors on the discrete channel. In this chapter the encoding and decoding procedures for the binary Bose-Chaudhuri-Hocquenghem (BCH) codes and their nonbinary generalizations, the Reed-Solomon (RS) codes, will be examined. Also, figures displaying performance and implementation difficulty will be presented making possible comparisons with other coding techniques.

A. Encoding a BCH Code.

BCH codes will be defined in terms of the roots of the generator polynomial.

Definition 3.1: The <u>BCH code</u> with symbols from GF(q) consists of all vectors F(X) over GF(q) for which

$$\alpha^{m_o}, \alpha^{m_o + 1}, \ldots, \alpha^{m_o + d - 2}$$

are roots of F(X) where m_o is any integer and α is any element of GF(q^m). The length, n, of the code is the order of α, and the minimum distance of the code is d.

In this definition GF(q) is called the symbol field, and GF(q^m) is called the locator field of the BCH code. This code is cyclic since it is defined in terms of the roots of its generator polynomial, and it can be shown that the code has minimum distance d.[13]

The most important BCH codes are the binary BCH codes generated by letting α be a primitive element of GF(2^m), $m_o = 1$, $d = 2t+1$, and $n = 2^m - 1$. In this case F(X) is a code vector if and only if

$$\alpha, \alpha^3, \cdots, \alpha^{2t-1}$$

are roots of F(X), and the code is capable of correcting t errors. The generator polynomial of this code is

$$g(X) = \text{LCM}\left[m_1(X), m_3(X), \cdots, m_{2t-1}(X)\right], \qquad (3.1)$$

where $m_i(X)$ is the minimum polynomial of α^i and LCM denotes the least common multiple. The degree of g(X) is n-k, the number of parity checks. Since each $m_i(X)$ has at most degree m, the code has at most mt parity checks. To find the exact number of parity checks for a given code, the degree of g(X) must be determined.

The procedure of designing an encoder for a BCH code of a given length and error-correcting ability is simple. Knowing g(X), the code can be implemented using the k-stage shift register of Fig. 2.3 or the (n-k)-stage shift register of Fig. 2.4.

The Reed-Solomon codes may be defined similarly.

Definition 3.2: The <u>RS code</u> with symbols from GF(q) consists of all vectors F(X) over GF(q) for which

$$\alpha, \alpha^2, \cdots, \alpha^{d-1}$$

are roots of F(X), and α is an element of GF(q). The length, n, of the code is the order of α, and the minimum distance of the code is d. When α is primitive, $n = q - 1$, the largest possible value.

Reed-Solomon codes are very good codes as is evident from the following definition.

Definition 3.3: A maximum code of length n, and minimum distance, d, has exactly the maximum possible number of information symbols, $k = (n + 1 - d)$.

Forney shows that the RS codes are maximum codes.[14] This is an important property since if d/n is held constant and $n \to \infty$, the rate

$$\lim_{n \to \infty} \frac{k}{n} = \lim_{n \to \infty} \frac{n + 1 - d}{n}$$

$$= 1 - d/n \quad , \qquad (3.2)$$

does not approach zero. This is significant because Peterson has demonstrated that the binary BCH codes suffer from a defect of most non-random coding schemes in that if the ratio d/n is held constant as $n \to \infty$, the rate approaches zero.[15] Although it is generally conceded that for $n < 10{,}000$ the binary BCH codes are still good, for very large values of n they are not among the best codes. In overcoming this defect by using RS codes a disadvantage is encountered in that since $n < q$, the size of the field must be increased as n is increased. From this discussion it is obvious that the RS codes should be used when the required length is $n < q$. If $n > q$, the BCH codes are satisfactory providing n is not too large.

A BCH code word can be represented by

$$F(X) = \sum_{i=0}^{n-1} F_i X^i \quad . \tag{3.3}$$

Since F(X) has roots α^{m_o}, α^{m_o+1}, $\cdots$, α^{m_o+d-2}, the set of equations

$$\sum_{i=0}^{n-1} F_i \alpha^{im} = 0, \ m_o \le m \le m_o + d - 2 \tag{3.4}$$

is satisfied. Letting $Z_i = \alpha^i$, $i = 0, 1, \cdots, n-1$, be the locator for position i of the code word, a BCH code word satisfies the system of equations

$$\sum_{i=0}^{n-1} F_i Z_i^{\,m} = 0, \ m_o \le m \le m_o + d - 2 \quad . \tag{3.5}$$

B. Decoding a BCH Code.

In this section three procedures for decoding a binary BCH code will be investigated. A new method discovered by Berlekamp[16] will be investigated first since it is in general more easily implemented than the other methods. The second method to be presented is an algorithm due to Peterson[17] which was historically the first method found for decoding BCH codes. The third method presented is a modification of Peterson's algorithm introduced by Berlekamp.[18]

Assuming that the transmitted binary BCH code word is

$$F(X) = \sum_{i=0}^{n-1} F_i X^i \quad , \tag{3.6}$$

the received word is

$$R(X) = F(X) + E(X)$$

$$= \sum_{i=0}^{n-1} F_i X^i + \sum_{i=0}^{n-1} E_i X^i \; . \qquad (3.7)$$

Since the code word has α^j, $j = 1, 2, \cdots, 2t$ as roots

$$R(\alpha^j) = 0 + \sum_{i=0}^{n-1} E_i \alpha^{ij}$$

$$= \sum_{k=1}^{e} Z_k^j = S_j, \; j = 1, 2, \cdots, 2t \; , \qquad (3.8)$$

where the error locators $Z_1, Z_2, \cdots, Z_e$ denote the positions which are in error. The error locator for the i'th position is α^i. Decoding requires several steps. First, the parity checks, S_j, $j = 1, 2, \cdots, 2t$, must be calculated. Second, the error locations $Z_1, Z_2, \cdots, Z_e$, must be determined from the system of equations

$$\sum_{k=1}^{e} Z_k^j = S_j, \; j = 1, 2, \cdots, 2t \; . \qquad (3.9)$$

These equations have several solutions corresponding to different error patterns in the same coset. The decoder attempts to find a solution with as small a value of e as possible to minimize the probability of error. Third, the errors must be corrected. In decoding a BCH code the second step presents the major problem.

The first step in decoding is to calculate the parity checks, S_j, $j = 1, 2, \cdots, 2t$. This calculation is simplified by using a circuit of the form of Fig. 2.4 to calculate the remainder, $r^{(j)}(X)$, when $R(X)$ is divided by $m_j(X)$, the minimum polynomial of α^j, $j = 1, 2, \cdots, 2t$. Since the code word is a multiple of the minimum polynomial of α^j, $j = 1, 2, \cdots, 2t$,

$$S_j = r^{(j)}(\alpha^j) \quad . \tag{3.10}$$

Once the parity checks are found the error locations $Z_1, Z_2, \cdots, Z_e$ must be found from the system of equations (3.9). Peterson's method involves using matrix reduction methods in this step which becomes quite lengthy. Berlekamp's algorithm is of a recursive nature.

Berlekamp's algorithm. To introduce Berlekamp's algorithm, the error locator polynomial

$$\sigma(X) = \prod_{i=1}^{e} (1 - Z_i X) = 1 + \sum_{j=1}^{e} \sigma_j X^j \tag{3.11}$$

is defined. The σ's and the S's may be related by introducing the generating function

$$S(X) = \sum_{j=1}^{\infty} S_j X^j \quad . \tag{3.12}$$

From (3.9)

$$S(X) = \sum_{j=1}^{\infty} \sum_{i=1}^{e} Z_i^j X^j$$

$$= \sum_{i=1}^{e} \frac{Z_i X}{1 - Z_i X} \quad . \tag{3.13}$$

Multiplying both sides of this equation by $\sigma(X)$ to clear the fractions,

$$S(X)\sigma(X) = \sum_{i=1}^{e} \frac{Z_i X}{1 - Z_i X} \prod_{j=1}^{e} (1 - Z_j X)$$

$$= \sum_{i=1}^{e} Z_i X \prod_{j \neq i} (1 - Z_j X) \quad . \qquad (3.14)$$

Adding $\sigma(X)$ to both sides of this equation gives

$$\left[1 + S(X)\right] \sigma(X) = \sigma(X) + \sum_{i=1}^{e} Z_i X \prod_{j \neq 1} (1 - Z_i X) \quad . \qquad (3.15)$$

If the right-hand side of this equation is represented by $\omega(X)$, where

$$\omega(X) = \sigma(X) + \sum_{i=1}^{e} Z_i X \prod_{j \neq i} (1 - Z_j X) \quad , \qquad (3.16)$$

then (3.15) becomes

$$\left[1 + S(X)\right] \sigma(X) = \omega(X) \quad . \qquad (3.17)$$

Since the decoder knows only the coefficients of the first 2t powers of X in S(X), it does not know S(X), but only S(X) mod X^{2t+1}. The relevant equation is

$$\left[1 + S(X)\right] \sigma(X) \equiv \omega(X) \bmod X^{2t+1} \quad , \qquad (3.18)$$

which Berlekamp calls the Key Equation. Both $\sigma(X)$ and $\omega(X)$ must be determined from this equation given S(X). The polynomials $\sigma(X)$ and $\omega(X)$ both have degrees $\leq e$, the number of errors which actually occurred. In decoding binary BCH codes the polynomial $\omega(X)$ is of no interest.

Since solution of (3.18) as it stands is difficult, the solution is obtained in a recursive manner by considering the sequence of equations

$$(1 + S)\sigma^{(k)} \equiv \omega^{(k)} \bmod X^{k+1} \quad . \tag{3.19}$$

Then for each k = 0, 1, 2, ···, 2t polynomials

$$\sigma^{(k)} = \sum_i \sigma_i^{(k)} X^i \tag{3.20}$$

and

$$\omega^{(k)} = \sum_i \omega_i^{(k)} X^i \tag{3.21}$$

can be found, which are solutions of (3.19). The development of the algorithm for the solution of (3.19), which is quite lengthy, can be found in Berlekamp[19], and only the algorithm will be stated here.

Algorithm 3.1. Berlekamp's Algorithm for Solving the Key Equation over any Field. Initially define $\sigma^{(0)} = 1$, $\tau^{(0)} = 1$, $\omega^{(0)} = 1$, $\gamma^{(0)} = 0$, $D(0) = 0$, $B(0) = 0$. Proceed recursively as follows. Define $\Delta_1^{(k)}$ as the coefficient of X^{k+1} in the product $(1 + S)\sigma^{(k)}$, and let

$$\sigma^{(k+1)} = \sigma^{(k)} - \Delta_1^{(k)} X\tau^{(k)}$$

and

$$\omega^{(k+1)} = \omega^{(k)} - \Delta_1^{(k)} X\gamma^{(k)} \quad .$$

If $\Delta_1^{(k)} = 0$; if $D(k) > \frac{k+1}{2}$; or if $\Delta_1^{(k)} \neq 0$, $D(k) = \frac{k+1}{2}$, and $B(k) = 0$; set

$$D(k+1) = D(k) \quad ,$$
$$B(k+1) = B(k) \quad ,$$
$$\tau^{(k+1)} = X\tau^{(k)} ,$$

and

$$\gamma^{(k+1)} = X\gamma^{(k)}.$$

But, if $\Delta_1^{(k)} \neq 0$, and either $D(k) < \frac{k+1}{2}$ or $D(k) = \frac{k+1}{2}$ and $B(k) = 1$, set

$$D(k+1) = k+1-D(k) ,$$

$$B(k+1) = 1-B(k) ,$$

$$\tau^{(k+1)} = \frac{\sigma^{(k)}}{\Delta_1^{(k)}} ,$$

and

$$\gamma^{(k+1)} = \frac{\omega^{(k)}}{\Delta_1^{(k)}} .$$

In the binary case this algorithm can be simplified.

Algorithm 3.2. Berlekamp's Simplified Algorithm for Binary BCH Codes. Initially define $\sigma^{(0)} = 1$ and $\tau^{(0)} = 1$, Proceed recursively as follows. Define $\Delta_1^{(i)}$ as the coefficient of X^{i+1} in the product $(1+S)\sigma^{(i)}$. Let

$$\sigma^{(2k+2)} = \sigma^{(2k)} + \Delta_1^{(2k)} X\tau^{(2k)}$$

and

$$\tau^{(2k+2)} = \begin{cases} X^2\tau^{(2k)} & \text{if } \Delta_1^{(2k)} = 0 \text{ or if deg } \sigma^{(2k)} > k , \\ \frac{X\sigma^{(2k)}}{\Delta_1^{(2k)}} & \text{if } \Delta_1^{(2k)} \neq 0 \text{ and deg } \sigma^{(2k)} \leq k . \end{cases}$$

The desired error locator polynomial is $\sigma(X) = \sigma^{(2t)}(X)$.

After this algorithm has been performed the error locator polynomial $\sigma(X)$ is obtained. It is evident from (3.11) that the roots of $\sigma(X) = 0$ are the reciprocals of the error locators $X = Z_i^{-1}$, $i = 1, 2, \cdots, e$. The error locators could be found by substituting all n locators α^i, $i = 1, 2, \cdots, n$, into the equation $\sigma(X) = 0$ and picking the locators which satisfy the equation. This lengthy procedure can be avoided by using a method proposed by Chien which will be called a Chien search.[20]

The Chien searcher computes the polynomials $\sigma(\alpha) = \sum_i \sigma_i\alpha^i$, $\sigma(\alpha^2) = \sum_i \sigma_i\alpha^{2i}$, $\sigma(\alpha^3) = \sum_i \sigma_i\alpha^{3i}$, $\cdots$ in a systematic manner. For this purpose it uses t registers. At the k'th step these registers contain the quantities $\sigma_1\alpha^k$, $\sigma_2\alpha^{2k}$, $\cdots$, $\sigma_t\alpha^{tk}$. In the next step the register containing $\sigma_i\alpha^{ik}$ is multiplied by α^i. These multiplications in $GF(2^m)$ are performed simultaneously in a single clock cycle by appropriate feedback connections to the registers containing $\sigma_1\alpha^k$, $\sigma_2\alpha^{2k}$, $\cdots$, $\sigma_t\alpha^{tk}$. As an example, multiplication by the wired constant α in $GF(2^4)$ with $\alpha^4 = \alpha + 1$ may be accomplished by the circuit of Fig. 3.1. Note that the feedback connections are determined by the relation $\alpha^4 = \alpha + 1$. There are similar circuits for multiplication by α^2, α^3, $\cdots$, α^t. At each step the contents of the registers are added to see if they total unity. If this happens at the k'th step, then $\sigma(\alpha^k) = 0$ and α^{-k} is a reciprocal root of the error polynomial. A one is then added to the erroneous digit at location α^{-k}, which is now leaving the buffer. A block diagram of the Chien searcher is shown in Fig. 3.2.

The steps to be followed in decoding a binary BCH code can be outlined now. A block diagram of the entire decoder is shown in Fig. 3.3. The operation of each of the blocks has been explained in detail except for the central Galois Field processor. This

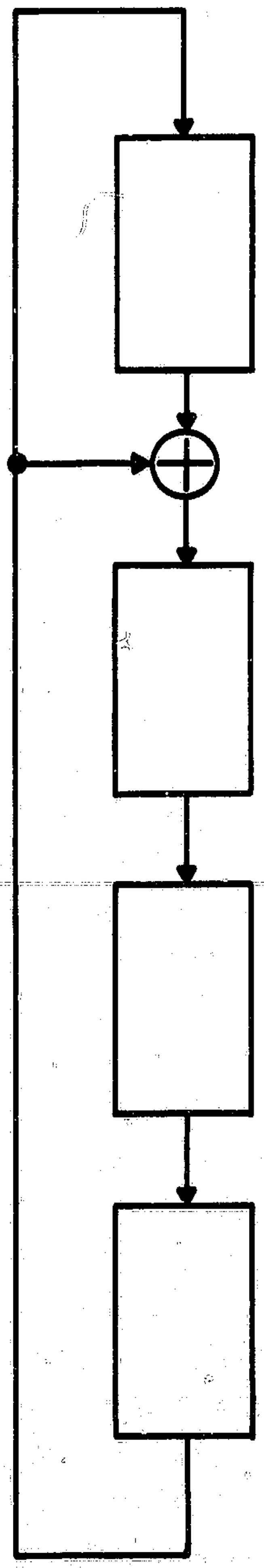

Fig. 3.1. Circuit for Multiplying by α

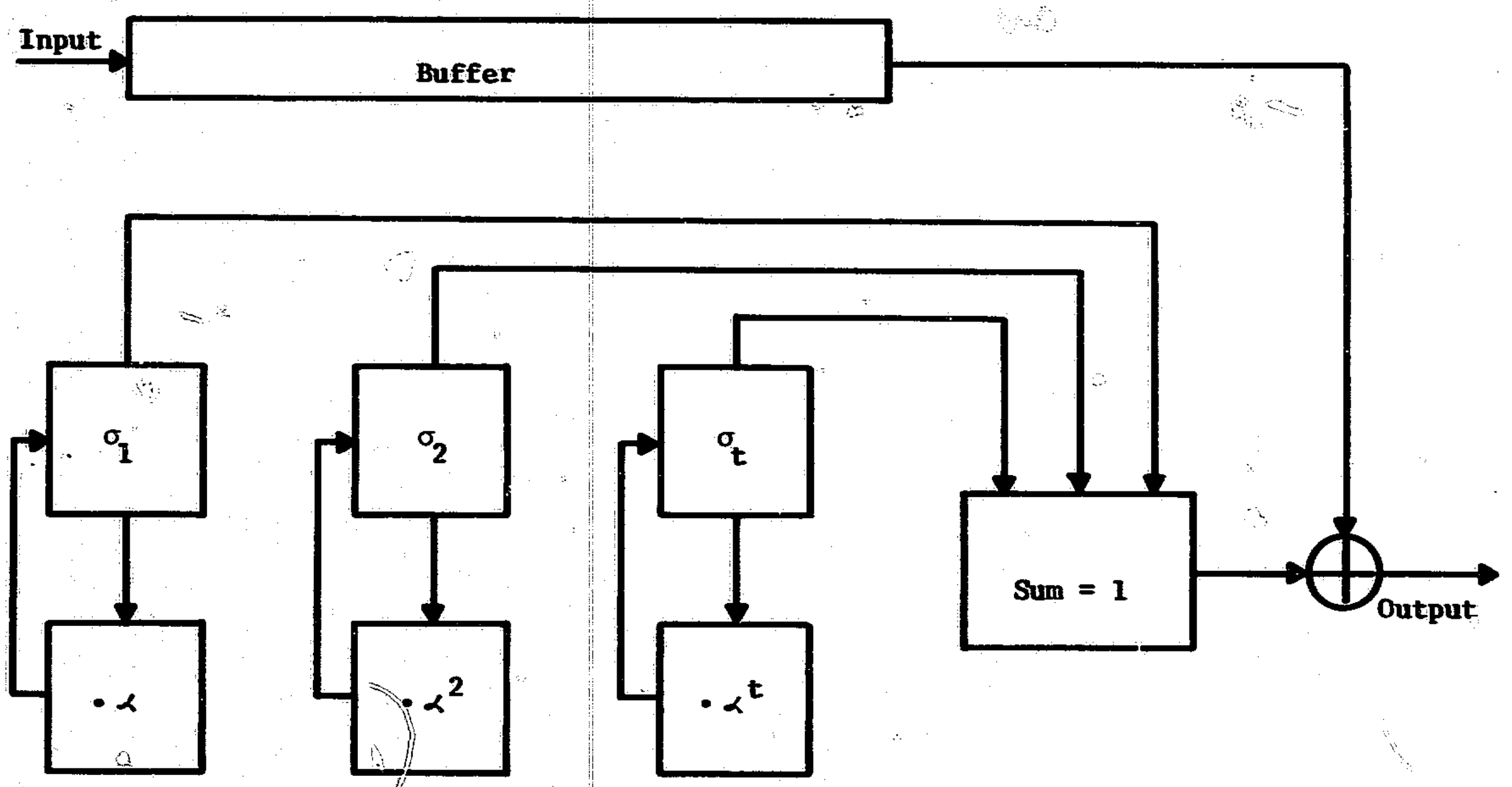

Fig. 3.2. Chien Searcher.

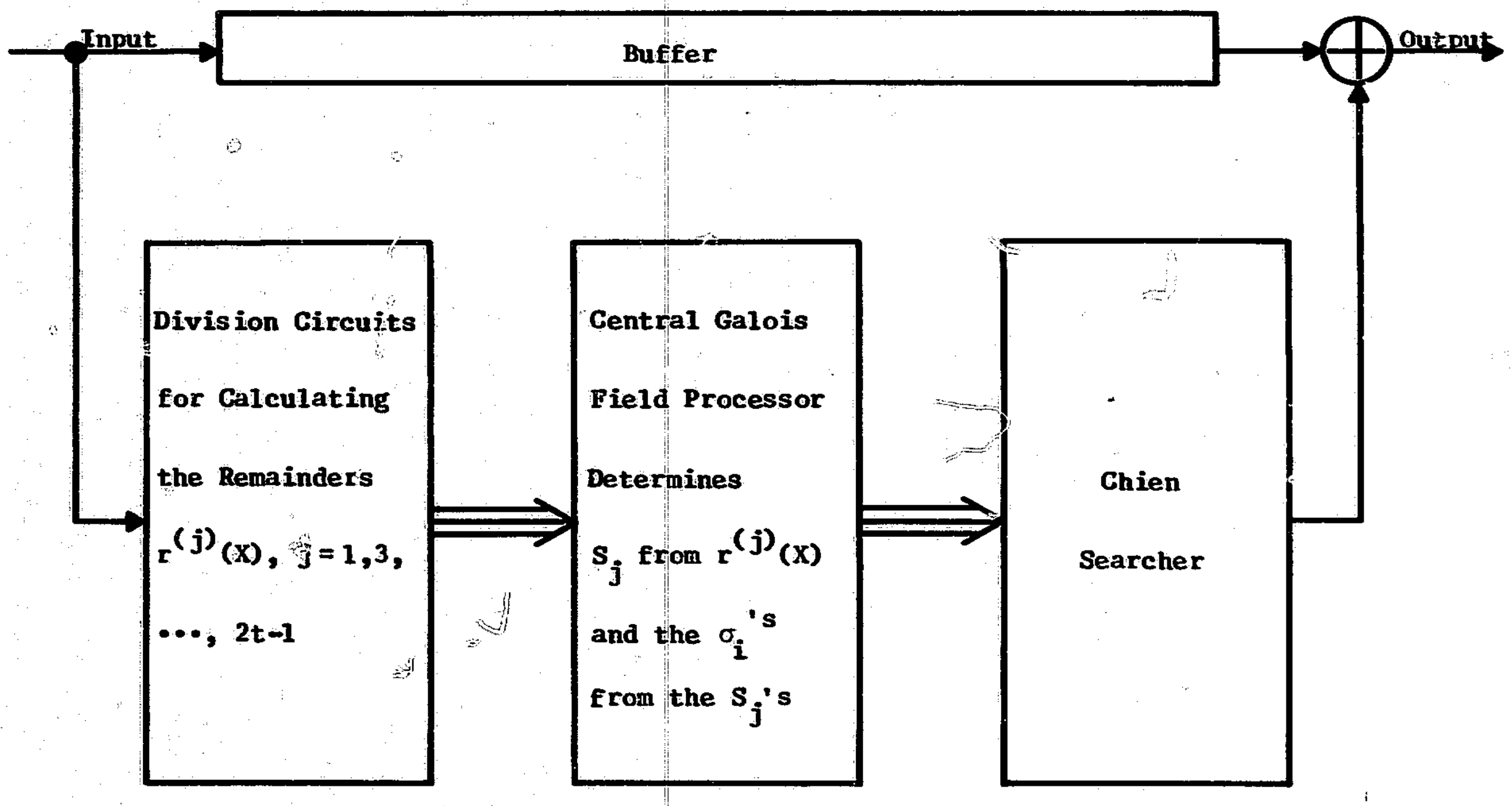
Input
Buffer
Output
Division Circuits for Calculating the Remainders $r^{(j)}(X)$, $j=1,3,\ldots,2t-1$
Central Galois Field Processor Determines S_j from $r^{(j)}(X)$ and the σ_i's from the S_j's
Chien Searcher

Fig. 3.3. Binary BCH Decoder.

processor consists of one master arithmetic and control unit and $(t+1)$ separate slave units, as shown in Fig. 3.4. The slaves are identical, each containing five registers and having the facility to add or multiply two of them together and accumulate the result in its slave accumulator. Identical control commands produced by the master control unit are sent to each slave. Each slave accumulator is connected to a master adder whose output is accessible to the master.

The central processor begins operation when the remainders $r^{(1)}(X)$, $r^{(3)}(X)$, $\cdots$, $r^{(2t-1)}(X)$ are transferred out of the registers at the left of Fig. 3.3 and into the appropriate slave units. One of the registers is then set to the wired constant, α^k, and it computes $r_0^{(k)} + r_1^{(k)}\alpha^k$, α^{2k}, $r_0^{(k)} + r_1^{(k)}\alpha^k + r_2^{(k)}\alpha^{2k}$, α^{3k}, $\cdots$, until it accumulates the parity checks $S_k = r^{(k)}(\alpha^k)$. This requires $(m-1)$ Galois field multiplications. With all slaves working simultaneously the parity checks S_1, S_2, $\cdots$, S_t can be computed in $(m-1)$ clock cycles, and in a similar manner S_{t+1}, S_{t+2}, $\cdots$, S_{2t-1} can be computed in another $(m-1)$ clock cycles. The contents of the slave registers at this point are shown in Fig. 3.5.

Berlekamp's algorithm can be implemented by using five columns of registers. Columns one and two contain the parity checks, column three contains $\sigma_i^{(k)}$, column four contains $\tau_i^{(k)}$, and column five is the accumulator. The error locator polynomial can be found by an 11-step procedure. Let k be one less than the number of times at Step 1.

1. The S columns are shifted upward twice. If $k=0$, set $\sigma^{(0)}=1$, $\tau^{(0)}=1$ and continue. If $k=0$, continue.

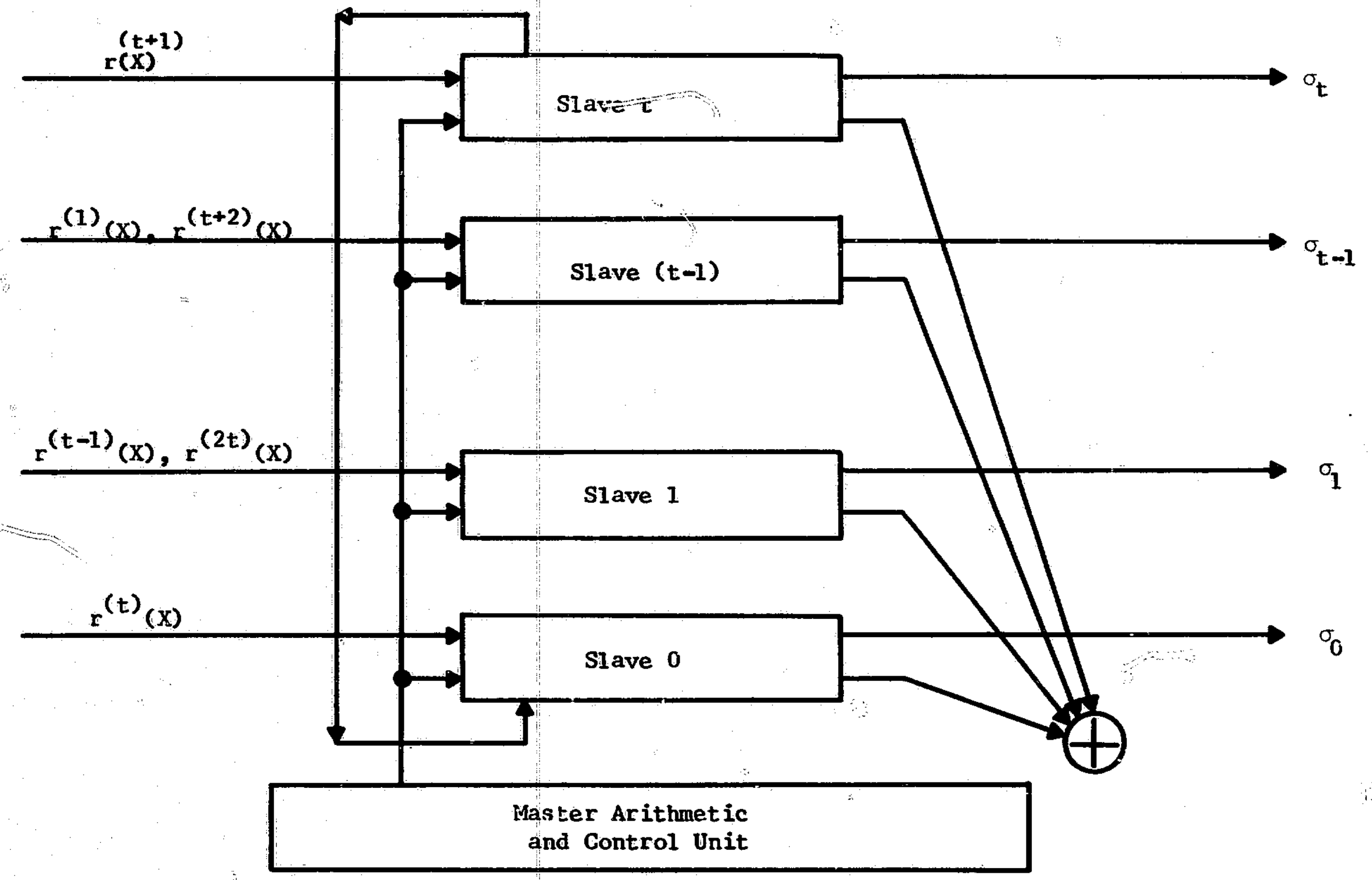

Fig. 3.4. Central Galois Field Processor.

			$\sigma_i^{(k)}$	$\tau_i^{(k)}$	
Slave t	1	S_{t+1}			
	S_1	S_{t+2}			
	S_2	S_{t+3}			
	●	●			
	●	●			
	●	S_{2t-2}			
Slave 2	●	S_{2t-1}			
Slave 1	S_{t-1}	—			
Slave 0	S_t	—			

Fig. 3.5. Contents of the Slave Registers after Computation of the Parity Checks.

2. Each slave computes the product of its second and third registers and places the result in its fifth register.
3. The sum of all registers in the fifth column is placed in the master accumulator. This sum is $\Delta_1^{(2k)}$.
4. The contents of the registers in the fourth column are shifted upward once which corresponds to multiplication of $\tau^{(2k)}$ by X.
5. $\Delta_1^{(2k)}$ is placed in the master multiplier register.
6. Each slave multiplies the contents of its τ-register by the contents of the master multiplier register, adds the result to the σ-register and places this result in the fifth column. After this step the fifth column contains $\sigma^{(2k+2)}$.
7. If $\Delta_1^{(2k)} = 0$ or if deg $\sigma^{(2k)} > k$, go to Step 9, otherwise go to Step 8.
8. The inverse of $\Delta_1^{(2k)}$ is computed and multiplied by the σ-column. The result is placed in the τ-column.
9. The τ-column is shifted upward once. It now contains $\tau^{(2k+2)}$.
10. Transfer $\sigma^{(2k+2)}$ from the fifth column to the third column.
11. If $k = (t-1)$, read out the error locator polynomial $\sigma^{(2t)}(X)$. If $k \neq (t-1)$ go back to Step 1 and increment k by 1.

The contents of the slave registers at a typical stage of the algorithm are shown Fig. 3.6. As the contents of the top-most registers of the first two columns are shifted out they are placed in the bottom register of the adjacent S column.

The buffer must hold the bits of the word being decoded as well as the incoming bits of the next word. A buffer of length at most 2n is needed assuming the decoder decodes at a nearly constant rate, as this one does. Both the division registers at the input of the central processor and the Chien searcher at the output of the central processor must operate in synchronism with the buffer. Since the central processor does not have to be synchronized with the buffer, in an actual decoder buffer requirements could be reduced by making the central processor as fast as practicable. However, it will be assumed that a buffer of length 2n is required.

In computing the decoding delay the basic unit of delay is the time required to perform an addition or shift a register. This basic unit will be called a computation. The method introduced in Chapter II for multiplying the contents of two registers requires m computations. Although using a Bartee and Schneider type multiplication unit would accomplish this with one computation, it would be costly to put one of these units in each slave.[21] The maximum number of computations required in the execution of the algorithm can now be determined. The most lengthy procedure after the test of Step 7 is to perform both Step 8 and Step 9. An inverse is calculated in Step 8. By using a method given in Berlekamp this requires $4m+1$ computations.[22] A tabulation of the number of

S_{2k+2}	S_{2k+t+3}	●	●	●
S_{2k+3}	S_{2k+t+4}	●	●	●
●	●	●	●	●
●	●	●	●	●
●	S_{2t-1}	●	●	●
●	—	●	●	●
●	—	●	●	●
●	1	●	●	●
●	S_1	●	●	●
●	●	●	●	●
●	S_{2k-2}	$\sigma_3^{(2k)}$	$\tau_3^{(2k)}$	$S_{2k-2}\sigma_3^{(2k)}$
S_{2k+t}	S_{2k-1}	$\sigma_2^{(2k)}$	$\tau_2^{(2k)}$	$S_{2k-1}\sigma_2^{(2k)}$
S_{2k+t+1}	S_{2k}	$\sigma_1^{(2k)}$	$\tau_1^{(2k)}$	$S_{2k}\sigma_1^{(2k)}$
S_{2k+t+2}	S_{2k+1}	1	0	S_{2k+1}

Fig. 3.6. Contents of the Slave Registers at a Typical Stage of the Algorithm.

computations required in the various steps of the algorithm is now shown in Table 3.1.

Table 3.1. Number of Computations Required in the Individual Steps of the Berlekamp Decoding Procedure.

Step	Number of Computations
parity checks	$2m(m-1)$
1	$2t$
2	mt
3	t
4	t
5	t
6	$t(m+1)$
7	t
8	$t(5m+1)$
9	t
10	t
11	t
Chien search	n

As is indicated in Table 3.1 each of the Steps 1 through 11 is performed t times. The total number of computations required by the Berlekamp algorithm is the sum of the number of computations required in the individual steps of the algorithm shown in Table 3.1. If the total number of computations required is denoted by N_C, the Berlekamp algorithm requires

$$N_C = n + 2m^2 - 2m + 7mt + 11t \qquad (3.22)$$

computations.

In formulating a measure for decoder complexity the number of binary storage elements will be required. This number will be used as a relative measure in comparing different coding schemes. In general it will give a good indication of the amount of control circuitry required.

The remainders, $r^{(j)}(X)$, are calculated using feedback shift registers with at most m stages. At most t such registers are required. The central processor has $(t+1)$ slaves each containing five m-stage registers. The master unit contains an accumulator register of m stages, a multiplier register of m stages, and two registers for computing multiplicative inverses consisting of $(2m+3)$ stages. Thus, the central processor contains approximately $(5mt+9m+3)$ binary storage elements. The Chien searcher requires t m-stage registers. The number of buffer stages required is 2n. Letting N_{SD} represent the approximate number of binary storage elements required in the decoder, a Berlekamp decoder requires approximately

$$N_{SD} = 2n + 7mt + 9m + 3 \tag{3.23}$$

binary storage elements.

Peterson's algorithm. The Peterson algorithm was the first algorithm proposed for decoding BCH codes, and in some cases it is the simplest to implement. All operations in a Peterson decoder are performed in $GF(2^m)$, the locator field. The decoder has a Galois field adder and a Galois field multiplication unit of the type described by Bartee and Schneider for which a multiplication requires only one computation. Decoding is accomplished in three steps:

1. Finding the parity checks.
2. Finding the error locator polynomial.
3. Performing the Chien search.

The received word,

$$R(X) = \sum_{i=1}^{n} R_i X^i \quad , \tag{3.24}$$

is stored in an n-stage register. The odd numbered parity checks,

$$S_j = \sum_{i=1}^{n} R_i \alpha^{i(n-j)}$$

$$= \sum_{k=1}^{e} Z_k{}^j \text{ , j odd ,} \tag{3.25}$$

can be calculated by an iterative method utilizing the Galois field arithmetic unit, i.e.,

$$S_j = \left[(R_1\alpha^j + R_2)\alpha^j + R_3\right]\alpha^j + R_4 \cdots . \tag{3.26}$$

In this case the locator for position i is $Z_i = \alpha^{n-i}$, because the error locator polynomial will be defined differently. Since

$$(S_j)^2 = \left[\sum_{k=1}^{e} Z_k^j\right]^2 = \sum_{k=1}^{e} Z_k^{2j} = S_{2j} \quad , \tag{3.27}$$

the even numbered parity checks can be calculated by $S_{2j} = (S_j)^2$. To calculate each of the t odd parity checks, (n - 1) multiplications and (n - 1) additions are needed, but to obtain the even numbered parity checks only (t - 1) multiplications are needed. The total number of computations required in Step 1 is then

$$N_{C1} = 2(n-1)t + t - 1$$
$$= 2tn - t - 1 \quad . \qquad (3.28)$$

The parity checks are power-sum symmetric functions and are related to the elementary symmetric functions σ_i by Newton's identities:

$$\begin{aligned}
&S_1 - \sigma_1 = 0 \ , \\
&S_2 - S_1\sigma_1 + 2\sigma_2 = 0 \ , \\
&S_3 - S_2\sigma_1 + S_1\sigma_2 - 3\sigma_3 = 0 \ , \qquad (3.29) \\
&S_4 - S_3\sigma_1 + S_2\sigma_2 - S_1\sigma_3 + 4\sigma_4 = 0 \ , \\
&S_5 - S_4\sigma_1 + S_3\sigma_2 - S_2\sigma_3 + S_1\sigma_4 - 5\sigma_5 = 0 \ , \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots .
\end{aligned}$$

It is possible to solve these equations and obtain correct solutions only when the number of equations used equals or exceeds by one the number of errors that actually occur. Once the elementary symmetric functions σ_i are found, the error locators may be found since they must satisfy the equation

$$(Z_1 - X)(Z_2 - X)(Z_e - X) = \sigma_e - \sigma_{e-1}X + \cdots + \sigma_1(-X)^{e-1} + (-X)^e \ . \quad (3.30)$$

Since the locator field has characteristic two, addition and subtraction are the same operation, and also $2\sigma_2 = \sigma_2 + \sigma_2 = 0$, $4\sigma_4 = 2\sigma_4 + 2\sigma_4 = 0$, etc. Thus, the second of equations (3.29) gives no information about σ_2 while the fourth of equations (3.29) gives no information about σ_4. It is obvious that in solving for the σ_i's only the odd numbered equations of (3.29) should be used. The problem then becomes that of solving the system of equations

$$\begin{bmatrix} 1 & 0 & 0 \cdots\cdots\cdots 0 \\ S_2 & S_1 & 1 \quad 0 \cdots\cdots 0 \\ S_4 & S_3 & S_2 \; S_1 \; 1 \; 0 \cdots 0 \\ \cdots & \cdots & \cdots \\ S_{2t-2} & S_{2t-3} & \cdots\cdots, S_{t-1} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3 \\ \vdots \\ S_{2t-1} \end{bmatrix} \qquad (3.31)$$

Peterson shows that if $e < t$ the rank of the coefficient matrix is $(e+1)$.[23] This rank is determined by reducing the system to upper triangular form as shown in Appendix A.

Assuming $e < t$ ~~the~~ $(e+1) \times (e+1)$ system of equations,

$$\begin{bmatrix} 1 & 0 & 0 \cdots\cdots 0 \\ 0 & S_1^{(1)} & 1^{(1)} \cdots \\ 0 & 0 & S_2^{(2)} \cdots \\ \cdots & \cdots & \cdots \\ 0 & 0 & 0 \cdot 0 \quad S_e^{(e)} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_{e+1} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3 \\ \vdots \\ S_{2e+1} \end{bmatrix} \qquad (3.32)$$

is solved for the σ_i (σ_{e+1} will be zero). If $e = t$, the upper triangular $t \times t$ system of equations obtained from (3.31) is solved. From Appendix A the maximum number of computations involved in Step 2 is

$$N_{C2} = (2t^2+5t)(t-1) + t^2 + (2t-1)(2m-2) + t(t-1)(2t-1)/3$$

$$- (4t+5)(t-1)t/2$$

$$= \frac{4t^3 + 15t^2 - 37t + 24mt - 12m + 12}{6} . \qquad (3.33)$$

The last step in the decoding algorithm is the Chien search. This requires a total of

$$N_{C3} = n \qquad (3.34)$$

computations. Then the total number of computations required in decoding a BCH code using the Peterson algorithm is

$$N_C = \frac{6n + 4t^3 + 15t^2 - 43t + 12tn + 24mt - 12m + 6}{6} \quad . \qquad (3.35)$$

Again it is assumed that a buffer of length 2n is needed. Other storage registers which are needed are (2t - 1) registers for the storage of the parity checks, t(t + 1) storage registers for the application of Gauss's Algorithm, and t registers for the Chien search. All of these registers have m stages. Also needed are 2 m-stage registers for the GF(2^m) adder, and an n-stage register to store the received word while calculating the parity checks. The total number of binary storage elements in a decoder using Peterson's algorithm is

$$N_{SD} = 3n + m(t^2 + 4t + 1) \quad . \qquad (3.36)$$

Berlekamp's modification of Peterson's algorithm. In decoding a binary BCH code using the Peterson method the system of equations (3.31) must be solved. This involves triangularizing the determinant of coefficients which has order t. For large t this step can be quite lengthy. Berlekamp has found a transformation which effectively halves the size of this matrix.[24] Writing the system (3.31) as

$$\begin{bmatrix} S_1 & 1 & 0 & 0 & 0 \cdots 0 \\ S_3 & S_2 & S_1 & 1 & 0 \cdots 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{2t-1} & S_{2t-2} & \cdots & \cdots & S_{t-1} \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_1 \\ \sigma_2 \\ \cdot \\ \cdot \\ \cdot \\ \sigma_t \end{bmatrix} = 0 \ , \qquad (3.37)$$

a set of new variables R_k is introduced which are formed by convolving the sequence S_k with a sequence A_k, i.e.,

$$R_k = \sum_j A_j S_{k-2j}, \text{ where } A_0 = S_0 = 1 \quad ,$$

and (3.38)

$$A_{-i} = S_{-i} = 0, \ i > 0 \quad .$$

Now if to each equation of (3.37) A_1 times the previous equation, A_2 times the equation before that, etc. are added, (3.37) is transformed into

$$\begin{bmatrix} R_1 & 1 & 0 & 0 & 0 \cdots 0 \\ R_3 & R_2 & R_1 & 1 & 0 \cdots 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ R_{2t-1} & R_{2t-2} & \cdots & \cdots & \cdots R_{2t-1} \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = 0 \ . \qquad (3.39)$$

The A_j can be chosen in such a way that the $R_{2i} = 0$ for all $i = 0$ by setting

$$\sum_{j=0}^{i} A_j S_{2(i-j)} = 0 \text{ or } A_i = \sum_{j=0}^{i-1} A_j S_{2(i-j)} \quad . \qquad (3.40)$$

Then given that $A_0 = 1$ the rest of the A_i are defined recursively by (3.40).

Now with $R_{2i} = 0$, the system (3.39) can be separated into two systems of equations. Letting $\lfloor A \rfloor$ represent the greatest integer less than or equal to A and $\lceil A \rceil$ represent the least integer greater than or equal to A, the last $\lfloor t/2 \rfloor$ equations from (3.39) become (3.41) while the first $t - \lfloor t/2 \rfloor$ equations become (3.42). Thus

$$\begin{bmatrix} \cdot & \cdot & \cdot & \cdots & \cdot \\ R_{2t-5} & \cdot & \cdot & \cdots & \cdot \\ R_{2t-3} & R_{2t-5} & \cdot & \cdots & \cdot \\ R_{2t-1} & R_{2t-3} & R_{2t-5} & \cdots & \cdot \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_2 \\ \sigma_4 \\ \cdot \\ \cdot \\ \sigma_{2\lfloor t/2 \rfloor} \end{bmatrix} = 0 \qquad (3.41)$$

and

$$\begin{bmatrix} \sigma_1 \\ \sigma_3 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} R_1 & 0 & 0 & 0 \cdots 0 \\ R_3 & R_1 & 0 & 0 \cdots 0 \\ R_5 & R_3 & R_1 & 0 \cdots 0 \\ \cdot & \cdot & \cdot & \cdots \\ R_{2t-1} & \cdot & \cdots & \cdot R_1 \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_2 \\ \sigma_4 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} . \qquad (3.42)$$

The upper-right-hand corner element of the coefficient matrix of (3.41) is R_1 if t is even or R_3 if t is odd. Once (3.41) is solved for the even numbered σ_i, then (3.42) gives the odd numbered σ_i directly. Berlekamp also shows that for all $i = 0$, $A_{2i} = 0$. This property will reduce the number of calculations needed to perform the transformation.

The parity check calculation and the Chien search are performed in the same manner as they were using Peterson's algorithm. In using Berlekamp's transformation the only modification is in the method for solving for the error locator polynomial from the parity checks. There are four steps in this procedure:

1. Computing $A_1, A_3, \cdots, A_r$, where r is the greatest odd integer less than t/2.
2. Computing $R_1, R_3, \cdots, R_{2t-1}$.
3. Solving (3.41) for $\sigma_2, \sigma_4, \cdots, \sigma_{2\lfloor t/2 \rfloor}$.
4. Solving (3.42) for $\sigma_1, \sigma_3, \cdots$.

In calculating the A_i note that $A_1 = S_2$, A_3 requires one addition and one multiplication, A_5 requires two additions and two multiplications, etc. After some thought it is evident that calculation of A_r requires $\lceil t/6 \rceil$ additions and $\lceil t/6 \rceil$ multiplications. Thus, calculation of the A_i requires a total of

$$N_{C1} = 1 + 2 \sum_{k=1}^{\lceil t/6 \rceil} k$$

$$= 1 + \lceil t/6 \rceil (\lceil t/6 \rceil + 1) \qquad (3.43)$$

computations. Similarly, in calculating the R_i, $R_1 = S_1$, R_3 requires one addition and one multiplication, R_5 requires two additions and two multiplications, etc. Thus, calculation of the R_i requires a total of

$$N_{C2} = 1 + 2 \sum_{k=1}^{t-1} k$$

$$= 1 + (t-1)t \qquad (3.44)$$

computations. In the third step a matrix of order $\lfloor t/2 \rfloor$ must be triangularized. The maximum number of computations is required when the rank of this matrix is $\lfloor t/2 \rfloor$. From Appendix A the number of calculations required is

$$N_{C3} = \frac{4(\lfloor t/2 \rfloor)^3 + 15(\lfloor t/2 \rfloor)^2 - 37(\lfloor t/2 \rfloor) + 24m(\lfloor t/2 \rfloor) - 12m + 12}{6} . \qquad (3.45)$$

By examining the cases for t odd and t even the number of equations in (3.42), $t - \lfloor t/2 \rfloor$, can be written as

$$t - \lfloor t/2 \rfloor = \left\lfloor \frac{t+1}{2} \right\rfloor . \qquad (3.46)$$

The last step requires solution of (3.42), which is straightforward and requires a number of computations equal to

$$N_{C4} = \sum_{k=1}^{\lfloor \frac{t+1}{2} \rfloor} k + \sum_{k=1}^{\lfloor \frac{t+1}{2} \rfloor - 1} k$$

$$= \frac{\lfloor \frac{t+1}{2} \rfloor (\lfloor \frac{t+1}{2} \rfloor + 1) + \lfloor \frac{t+1}{2} \rfloor (\lfloor \frac{t+1}{2} \rfloor - 1)}{2} \quad . \qquad (3.47)$$

The total number of computations required when using this procedure is obtained by adding the number of computations required in computing the parity checks and performing the Chien search to the number of computations required in the four steps of solving for the error locator polynomial. The total number of computations required is

$$N_C = \frac{6n + 12tn + 6t^2 - 12t - 12m + 24}{6}$$

$$+ \frac{6(\lfloor \frac{t+1}{2} \rfloor)^2 + 6(\lceil t/6 \rceil)^2 + 6(\lceil t/6 \rceil) + 4(\lfloor t/2 \rfloor)^3}{6}$$

$$+ \frac{15(\lfloor t/2 \rfloor)^2 - 37(\lfloor t/2 \rfloor) + 24m(\lfloor t/2 \rfloor)}{6} \quad . \qquad (3.48)$$

Again it will be assumed that a buffer of length 2n is needed. Also needed are 2t registers for storage of the parity checks, $\lfloor \frac{t-1}{4} \rfloor$ storage registers to store the A_i's, t registers for storage of the R_i's, $\lfloor t/2 \rfloor (\lfloor t/2 \rfloor + 1)$ registers to solve (3.41), and t registers for the Chien search. All of these registers have m stages. Other registers needed are two m-stage registers for the $GF(2^m)$ adder, and an n-stage register to store the received word while calculating the parity checks. The total number of binary storage elements needed in this decoder is

$$N_{SD} = 3n + 3mt + m(\lfloor t/2 \rfloor)^2 + \lfloor t/2 \rfloor m + \left\lceil \frac{t-1}{4} \right\rceil m + 2m \; . \qquad (3.49)$$

In this section three schemes for decoding binary BCH codes were presented, and the implementation complexity of these schemes was analyzed in detail. For each scheme N_C, the number of computations required, and N_{SD}, the number of binary storage elements required, was determined. These quantities will be useful in making comparisons.

C. The Performance and Complexity of Implementing BCH Codes

In this section the performance and implementation complexity of some representative BCH codes will be compared. The performance comparison will be made using a binary symmetric channel created from the white-Gaussian-noise channel. Also, the performance of coded and uncoded systems will be compared.

The signaling scheme for the white-Gaussian-noise channel is the antipodal signaling scheme with matched-filter reception described in Appendix B. The output of the matched filter is quantized into two levels, i.e. the receiver makes a hard decision as to which of the two possible messages was sent. This combination of signal generator, Gaussian channel, and matched filter can be modeled as a binary symmetric channel. The transition probability of this channel is given by (B.39), which is repeated here as

$$p = \frac{1}{\sqrt{2\pi}} \int_{\sqrt{\frac{2E_s}{N_o}}}^{\infty} e^{-x^2/2} dx \; , \qquad (3.50)$$

where E_s is the received-signal energy per symbol and N_o is the single-sided spectral density in watts/Hz. The transition probability p is equal to the probability of bit error for transmission over this channel. If an (n,k) code is used on this channel, only k of n symbols are information symbols, and thus the received energy per information symbol is

$$E_b = \frac{n}{k} E_s \quad . \tag{3.51}$$

The ratio E_b/N_o is a measure of the energy required by a signaling or coding scheme to communicate an information bit. The purpose of using coding is to achieve a given error probability at a smaller value of E_b/N_o or to achieve a smaller error probability at a given value of E_b/N_o. In calculating the performance of a t-error-correcting BCH code, (2.16) and (2.17) are used with (3.50) and (3.51) to compute $P_W(e)$ as a function of E_b/N_o.

From an energy standpoint block coding over a binary-symmetric channel created from a white-Gaussian-noise channel is not efficient for low rates and high rates when the required error probability is fairly low, say $10^{-10} \leq P_W(e) \leq 10^{-3}$. The high rate codes are efficient for higher values of $P_W(e)$, and the low rate codes are efficient at lower values of $P_W(e)$. The BCH codes which will be studied have medium rates, i.e., codes with two rates, 1/2 and 2/3, are to be studied. The parameters of these codes and their error-correcting capability t are shown in Table 3.2.

Table 3.2. BCH Codes to Be Investigated.

n	k	t
7	4	1
15	11	1
15	7	2
31	21	2
31	16	3
63	45	3
63	30	6
127	85	6
127	64	10
255	171	11
255	131	18
511	340	20
511	259	30
1023	688	36
1023	513	56

The probability of error as a function of E_b/N_o for the rate 1/2 and rate 2/3 codes is shown in Figs. 3.7 and 3.8, respectively. The probability shown is the word error probability, which allows the most general comparison. Also shown in these figures is the probability of bit error for uncoded transmission over this channel as determined by (3.50). Many times the data consists of characters of b bits. The probability of a b-bit word being correct with uncoded transmission is

$$P_W(c) = (1 - p)^b \,, \tag{3.52}$$

where p is given by (3.50). Then the probability of word error for uncoded transmission is

$$P_W(e) = 1 - (1 - p)^b \,. \tag{3.53}$$

Since

$$p < 1 - (1 - p)^b \,, \tag{3.54}$$

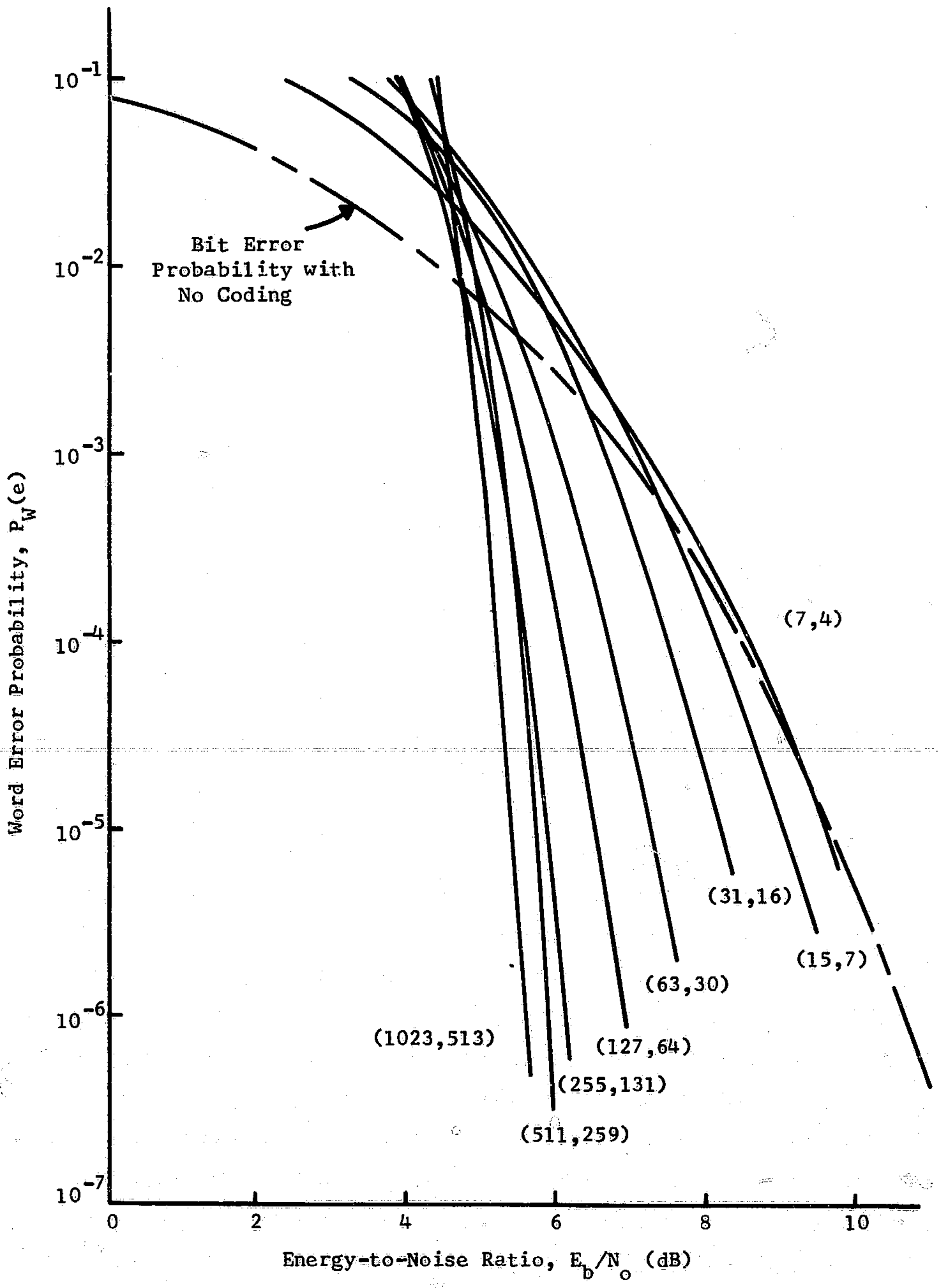

Fig. 3.7. Performance of Rate 1/2 BCH Codes.

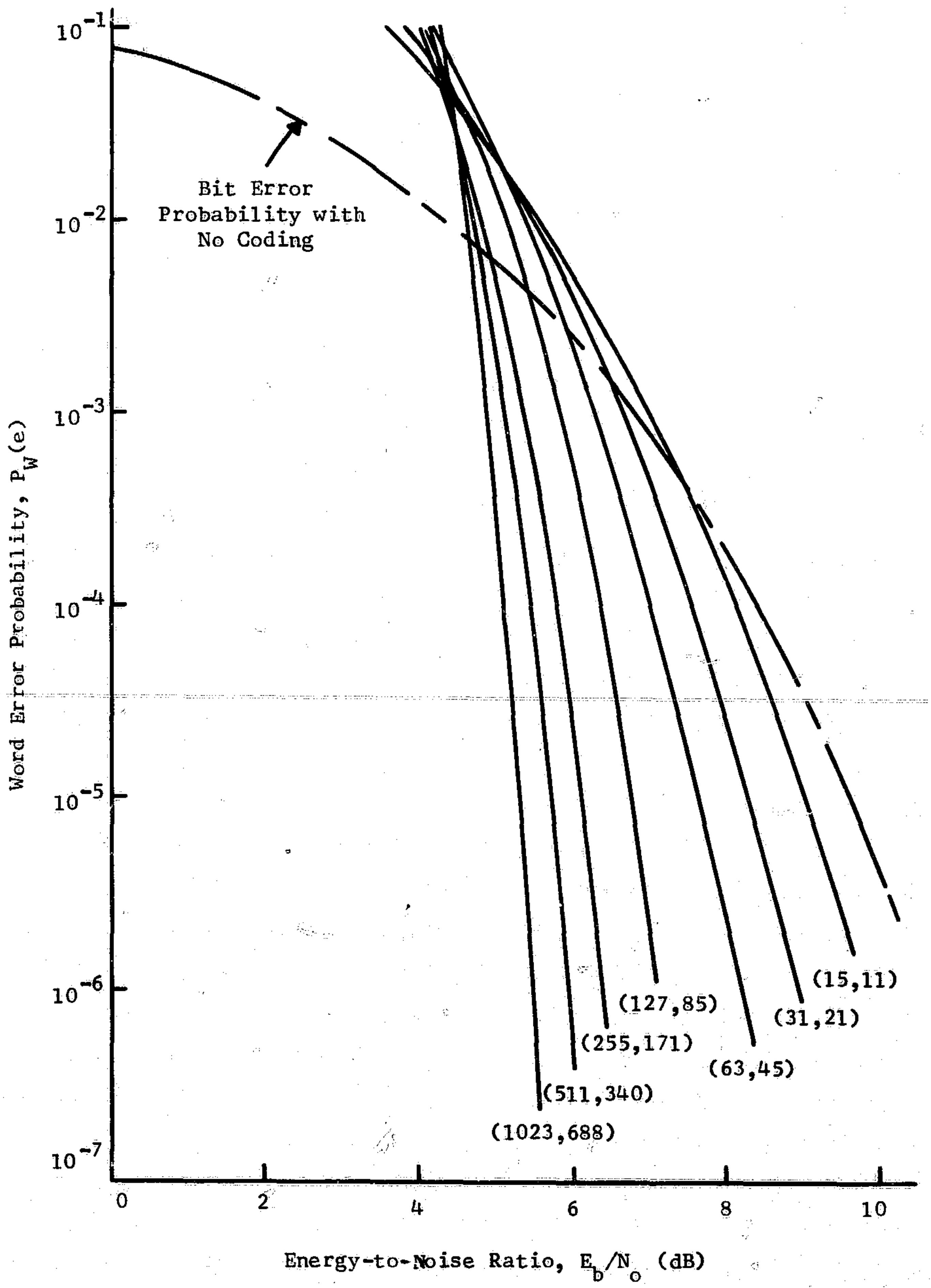

Fig. 3.8. Performance of Rate 2/3 BCH Codes.

the use of coding actually results in a larger decrease in error probability than is indicated in Figs. 3.7 and 3.8, but in the interest of generality the no coding curve determined by (3.50) is shown. If coding is to be applied to a system which uses characters of a specific length, then (3.53) should be used in the comparison.

Several observations can be made from Figs. 3.7 and 3.8. At high values of the specified error probability a smaller value of E_b/N_o is required for no coding than with coded systems. At sufficiently small values of the specified error probability a smaller value of E_b/N_o is required by the coded system than by the uncoded system. Since the probability of error of the coded system decreases faster as a function of E_b/N_o than that of the uncoded system, the coding gain increases as the specified probability of error is decreased. A comparison of Figs. 3.7 and 3.8 reveals that neither rate is better than the other for all block lengths. The code which exhibits the best performance of those shown in Figs. 3.7 and 3.8 is the (1023, 688) code in Fig. 3.8. At a specified error probability of 10^{-5} an uncoded system requires $E_b/N_o = 9.6$ dB, while a system using the (1023, 688) code requires $E_b/N_o = 5.3$ dB for a coding gain of 4.3 dB. However, even at this long block length Shannon's limit ($E_b/N_o = -1.6$ dB) determined by the channel capacity is not closely approached.[25] One reason for this is that by quantizing the output of the matched filter into just two levels much information is lost which results in a degradation of 1.0 to 2.0 dB. Another reason is that the long block length BCH codes are not very good since for a fixed rate $t/n \to 0$ as $n \to \infty$.

In comparing two codes with the same block length, the code with the higher rate will be easier to implement since its error-correction capability is less. Thus, since the rate 1/2 and rate 2/3 codes exhibit approximately the same performance, only the complexity of implementing the rate 2/3 BCH codes will be examined in detail. Also, the encoder for the rate 2/3 codes will be simpler if the encoder of Fig. 2.7 is used. In this case an encoder consisting of

$$N_{SE} = n - k \tag{3.55}$$

shift-register stages will be required.

The number of computations required in decoding these BCH codes as given by (3.22), (3.35), and (3.48) for the three decoders is shown in Fig. 3.9 as a function of block length. The number of computations required by the Berlekamp decoder is much less than the number required by the other two decoders. At a block length of 1023 there is more than an order of magnitude difference and the gap widens with increasing n. This measure gives an indication of the total decoding delay. An indication of the speed advantage required of the decoder is given by the number of computations per decoded information bit which is shown in Fig. 3.10. Note that for $n \geq 127$ the number of computations per decoded information bit decreases with increasing n for the Berlekamp decoder. This quantity increases with n for the other two decoders. At a block length of 1023 both Peterson decoders must have logic which is 100 to 200 times as fast as the incoming information rate, while the Berlekamp decoder must have logic which operates only five to six times as fast. The number of binary storage elements required by these decoders as given by (3.22), (3.36), and (3.49) is shown in Fig. 3.11 as a function of n. For long block lengths the Berlekamp decoder requires the fewest storage elements. For small values of n the Peterson decoders require fewer storage elements, but the difference is small.

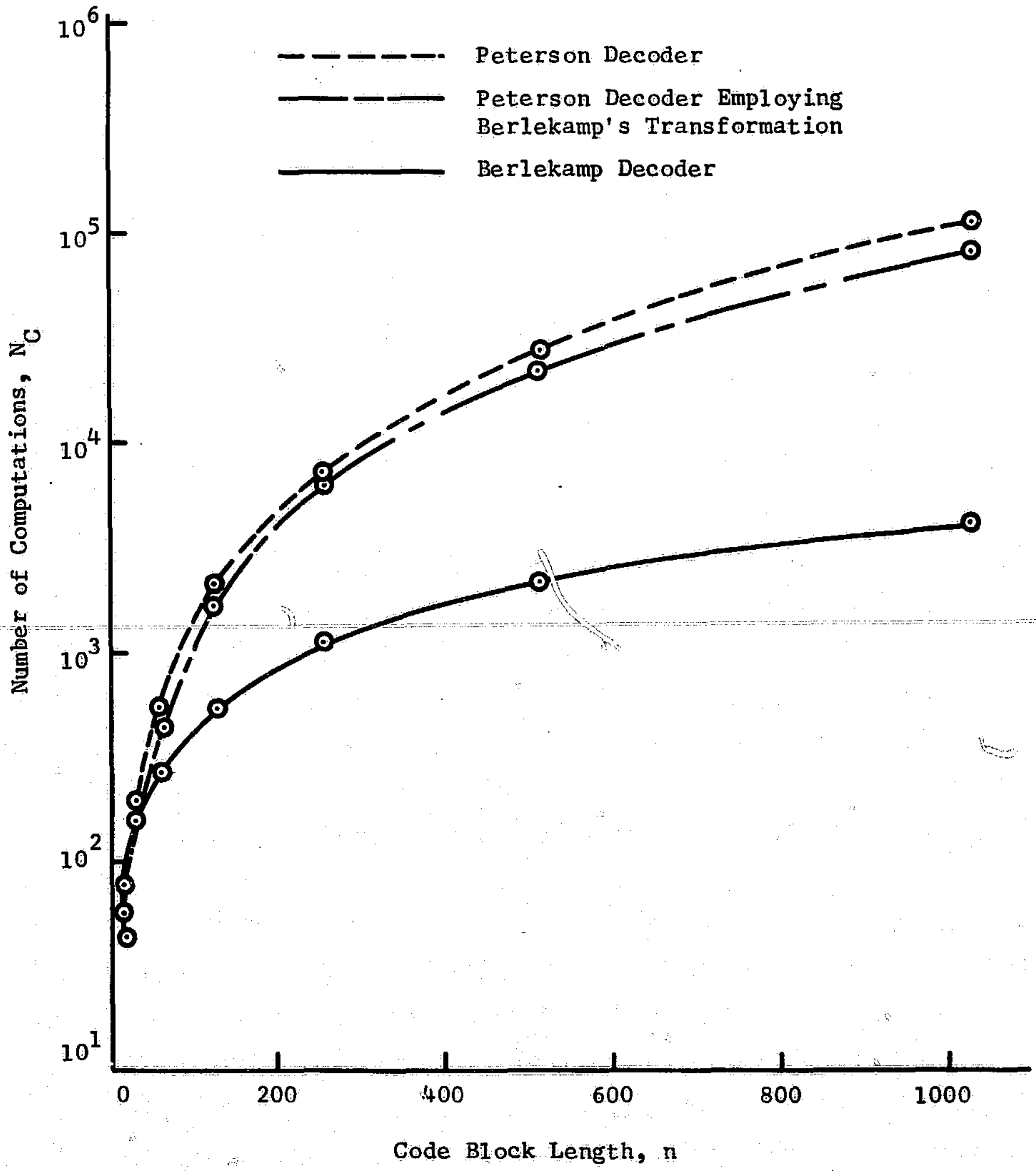

Fig. 3.9. Number of Computations Required in Decoding Rate 2/3 BCH Codes for Three Types of Decoders.

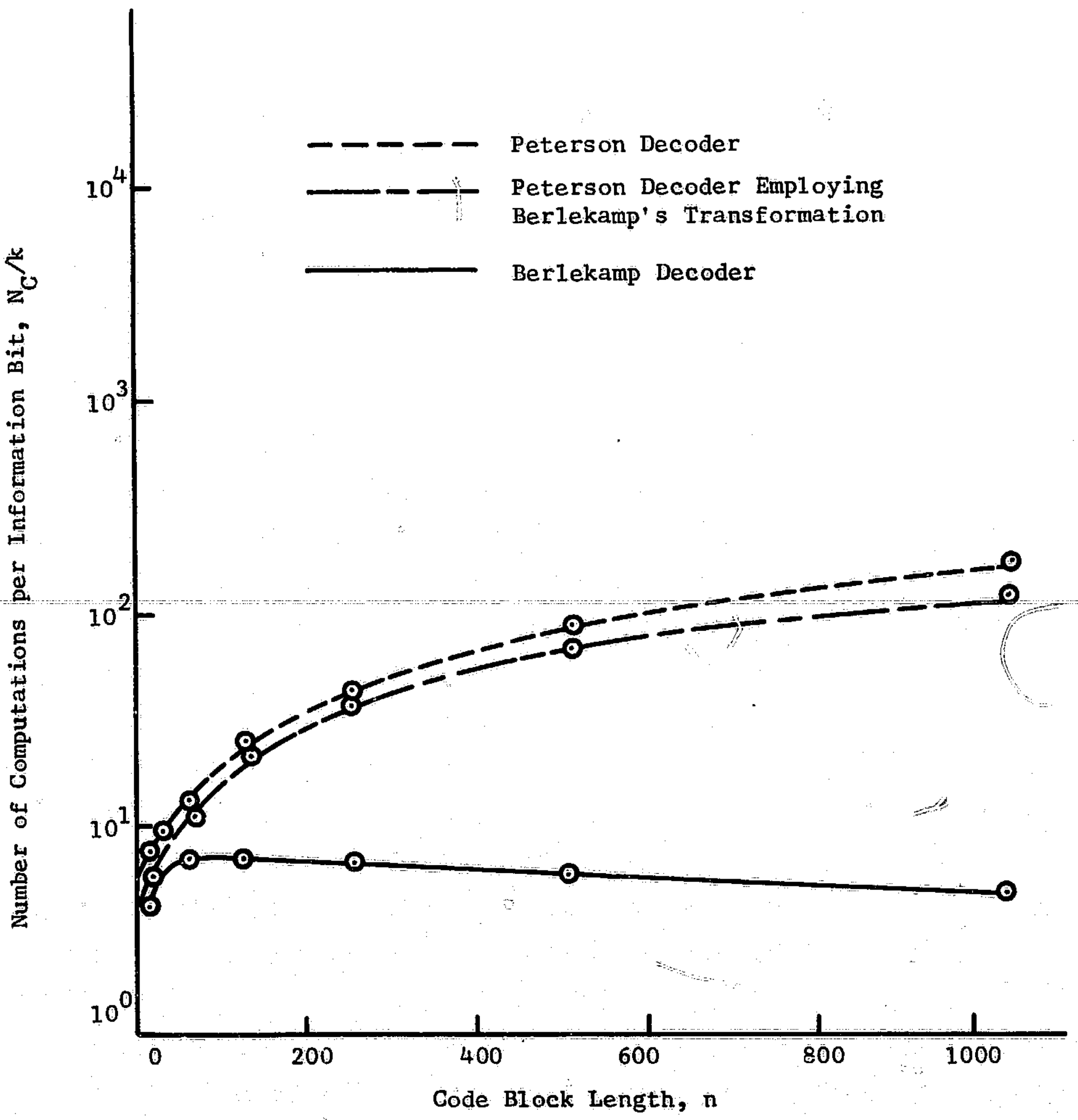

Fig. 3.10. Number of Computations per Information Bit Required in Decoding Rate 2/3 BCH Codes for Three Types of Decoders.

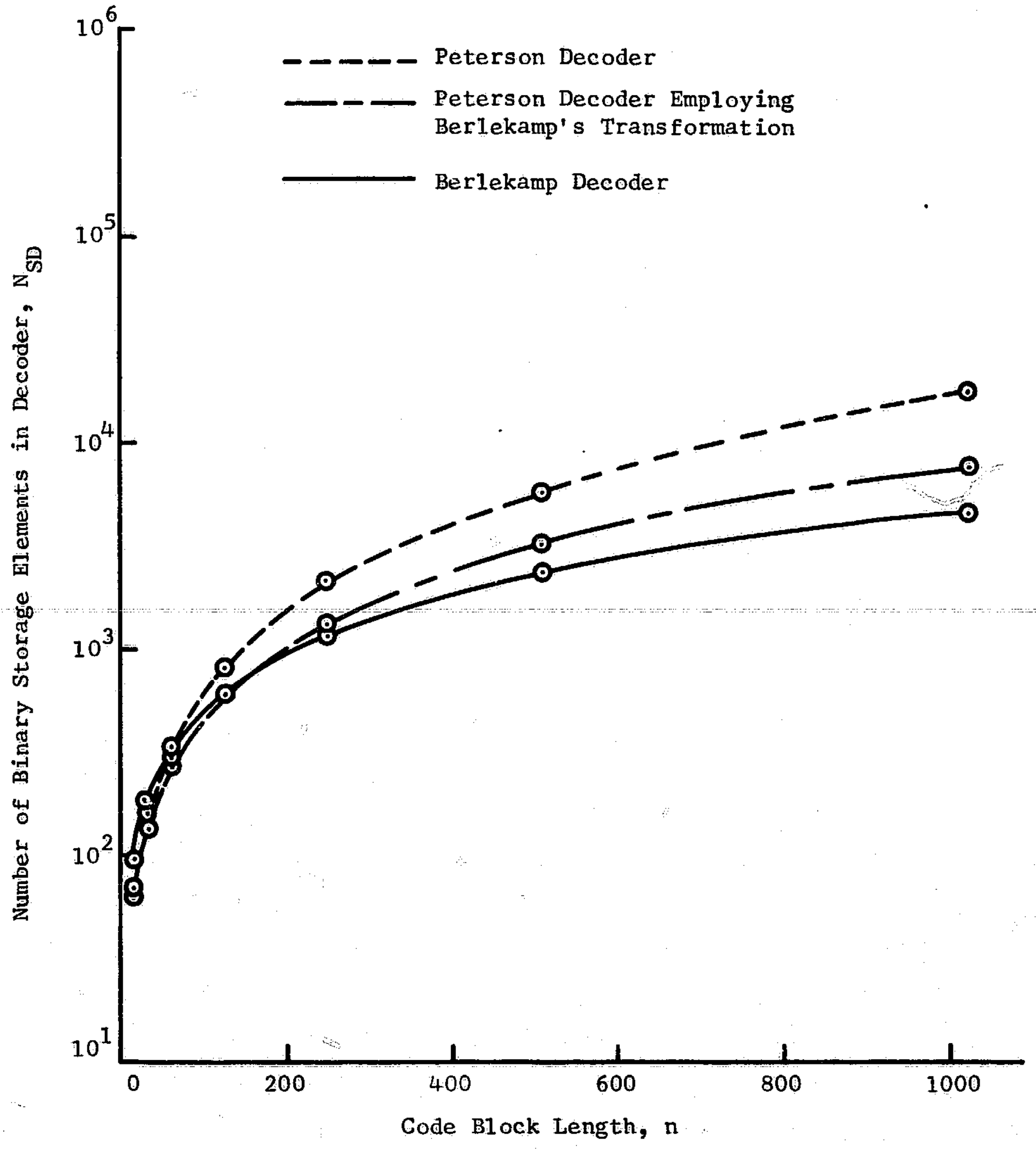

Fig. 3.11. Number of Binary Storage Elements Required in the Decoder for Rate 2/3 BCH Codes for Three Types of Decoders.

At first glance it would seem that the Berlekamp decoder should almost always be used. However, there are simplifications which make the Peterson decoder more attractive in some cases. For small t the system of equations (3.31) can be solved and the necessary determinants evaluated before the decoder is designed, making the Gauss-Jordan reduction unnecessary. With slightly larger t the same technique can be used with the decoder employing Berlekamp's modification of Peterson's algorithm by solving the system (3.41) before the decoder is designed. For large values of t, say $t \geq 10$, the Berlekamp decoder is the most attractive to implement.

CHAPTER IV

OPTIMUM CODES FOR THE GAUSSIAN CHANNEL

In this chapter communication over the white-Gaussian-noise channel will be considered. Block codes will be used with each code word containing k information bits. Decoding is accomplished by an optimum receiver (a correlation receiver). Since a correlation receiver is used, the best signaling strategy is to select code words which are as mutually uncorrelated as possible. Several codes which have low crosscorrelation between code words will be discussed including orthogonal, bi-orthogonal, simplex, and bi-simplex codes.

A. Generation and Decoding of Optimum Codes for the Gaussian Channel

In selecting a waveform to communicate k information bits over the Gaussian channel, a code word is first generated by an encoder. The n symbols of the code word determine the components of an n-dimensional signal vector according to the mapping $0 \rightarrow -1$, and $1 \rightarrow 1$. The transmitted waveform is synthesized from a set n orthogonal waveforms and the components of the signal vector according to (B.20). In this case the orthogonal waveforms are taken to be n time translates of a single basic waveform, $s_o(t)$, and the components of the signal vector are either +1 or -1. If T is the time it takes to send each code word, the set of orthogonal waveforms is $\left\{ s_o(t - jT/n) \right\}$, $j = 0, 1, \cdots, n-1$. The transmitted signal has the form

$$s(t) = \pm s_o(t) \pm s_o(t - T/n) \pm \cdots \pm s_o(t - \frac{n-1}{n} T) , \qquad (4.1)$$

where the sign of each of the n terms is determined by the corresponding code word symbol.

Since a correlation receiver is used, it is obvious that the lowest probability of error is achieved by making the code words as mutually uncorrelated as possible. In searching for codes which have low normalized crosscorrelation coefficients, ρ, between code words, the definition

$$\rho = \frac{\text{number of agreements} - \text{number of disagreements}}{n}, \qquad (4.2)$$

where n is the number of binary symbols per code word , will be used. Codes with small ρ can be constructed from Hadamard matrices, where a Hadamard matrix is a matrix in which each row is orthogonal to every other row. These matrices can be constructed from the recursion relation

$$A_k = \begin{bmatrix} A_{k-1} & A_{k-1} \\ A_{k-1} & \bar{A}_{k-1} \end{bmatrix},$$

where (4.3)

$$A_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

and $\bar{A}_{k-1}$ is obtained from A_{k-1} by replacing each 0 with a 1 and each 1 with a 0.

<u>Orthogonal</u> codes have $\rho = 0$ for all crosscorrelations between code words. An orthogonal code with k information symbols per code word is constructed by using the rows of A_k as code words. The length of the code word is $n = 2^k$.

<u>Bi-orthogonal</u> codes can be constructed by taking a set of orthogonal code words and adding to each word its complement. Each bi-orthogonal code word will then have zero correlation with every other code word

except its complement with which it has a correlation of -1. This code has an advantage over an orthogonal code in that one-half as many symbols per code word are required, leading to a reduction in required bandwidth of one-half (for orthogonal codes $n = 2^k$ while for bi-orthogonal codes $n = 2^{k-1}$).

In a Hadamard matrix generated by (4.3) the first column contains all zeros. A <u>simplex</u> code may be constructed from a Hadamard matrix by deleting this column and taking the code words to be the rows of the resulting matrix. A simplex code has $n = (2^k-1)$ and a crosscorrelation coefficient among all possible pairs of code words equal to $-1/(2^k-1)$. Since it can be shown that this is the minimum attainable crosscorrelation coefficient between 2^k sequences, simplex codes are optimum.[26] For large k the crosscorrelation coefficient is not much less than zero, and in this case these codes will have essentially the same performance as orthogonal codes.

A <u>bi-simplex</u> code is constructed in a manner analogous to the construction of a bi-orthogonal code by taking a set of simplex code words and adding to it the complement of each word. Again a reduction in required bandwidth of approximately one-half is obtained since for a bi-simplex code which has 2^k code words, $n = 2^{k-1}-1$.

The codes discussed above are all readily implemented using the k-stage shift-register encoder illustrated in Fig. 2.8. However, the feedback connections must be such that a maximum-length sequence, i.e., $n = 2^k-1$, is generated. Such a sequence is generated if $h(X) = (X^n-1)/g(X)$ is a primitive polynomial, i.e., all the roots of $h(X)$ are primitive elements of $GF(2^k)$. An excellent discussion of maximum-length shift registers and their applications may be found in Golomb.[27]

A simplex code may be generated by loading the k information bits into the stages of a maximum-length shift register and then shifting the register 2^k-1 times. The bi-simplex code is generated if the first bit is used to determine whether the complemented or uncomplemented output of the register is used. The last k-1 information bits are loaded into a (k-1)-stage maximum-length shift register, and it is shifted 2^{k-1}-1 times. An orthogonal code can be generated by loading the k information bits into a k-stage maximum-length shift register, shifting it 2^k-1 times, and adding a zero as the last bit. To generate a bi-orthogonal code the first bit is transmitted immediately, and the others are loaded into a (k-1)-stage maximum-length shift register. The register is then shifted 2^{k-1}-1 times with the complemented output being taken if the first digit was a zero. Generation of these codes is very simple since it takes at most a k-stage shift register.

B. Probability of Error

In this section expressions for the word error probability of orthogonal and bi-orthogonal codes will be determined. Also curves displaying $P_W(e)$ vs. E_b/N_o for the orthogonal codes will be given.

In calculating the error probability for an orthogonal code, the code words are treated as a set of $M = 2^k$ equal-energy orthogonal signal vectors. If each signal vector has energy E_s,

$$\bar{s}_i \cdot \bar{s}_j = \int_{-\infty}^{\infty} s_i(t) s_j(t)\,dt = E_s \delta_{ij} ; i,j = 0,1,\cdots,M-1 \quad , \qquad (4.4)$$

where

$$\delta_{ij} = \begin{cases} 1, & i = j \, , \\ 0, & i \neq j \, . \end{cases} \qquad (4.5)$$

The unit vector along the j'th coordinate axis and the direction of $\bar{s}_j$ are represented by $\bar{\phi}_j$, i.e.,

$$\bar{s}_j = \sqrt{E_s}\,\bar{\phi}_j, \; j = 0,1,\cdots,M-1 \quad . \tag{4.6}$$

The received vector is denoted by $\bar{r}$. Since

$$\begin{aligned}|\bar{r}-\bar{s}_j|^2 &= |\bar{r}|^2 + |\bar{s}_j|^2 - 2\bar{r}\cdot(\sqrt{E_s}\,\bar{\phi}_j) \\ &= |\bar{r}|^2 + E_s - 2r_j\sqrt{E_s}, r_j = \bar{r}\cdot\bar{\phi}_j \quad ,\end{aligned} \tag{4.7}$$

the optimum decision rule, which is to set $\hat{m} = m_i$ if and only if

$$|\bar{r}-\bar{s}_i| < |\bar{r}-\bar{s}_j|, \text{ all } j \neq i, \tag{4.8}$$

reduces to

$$r_i > r_j, \text{ all } j \neq i \quad . \tag{4.9}$$

If $\bar{s_0}$ is transmitted

$$r_0 = n_0 + \sqrt{E_s} \quad , \tag{4.10}$$

and

$$r_j = n_j, \; j = 1,2,\cdots,M-1 \quad . \tag{4.11}$$

Wozencraft and Jacobs show that if the additive noise is a zero-mean white-Gaussian process with spectral density $N_o/2$, the $\{n_j\}$, $j = 0, 1, \cdots, M-1$, are zero-mean Gaussian random variables with zero covariance and equal variance, $\sigma^2 = N_o/2$.[28] Then

$$\begin{aligned}P(c|m_0, r_0 = \alpha) &= P(n_1 < \alpha, n_2 < \alpha, \cdots, n_{M-1} < \alpha) \\ &= \left[P(n_1 < \alpha)\right]^{M-1} \quad .\end{aligned} \tag{4.12}$$

The last equality follows from the fact that the $\{n_j\}$ are statistically independent and identically distributed.

Since

$$P(r_0) = P(n_0), \; n_0 = r_0 - \sqrt{E_s} \quad , \tag{4.13}$$

then $P(c|m_0)$ is given by

$$P(c \; m_0) = \int_{-\infty}^{\infty} P(c|m_0, r_0 = \alpha) p(r_0 = \alpha) d\alpha$$

$$= \int_{-\infty}^{\infty} p(\alpha - \; E_s) \left[P(n_1 < \alpha) \right]^{M-1} d\alpha$$

$$= \int_{-\infty}^{\infty} p(\alpha - \; E_s) d\alpha \left[\int_{-\infty}^{\alpha} p(\beta) d\beta \right]^{M-1} \; , \tag{4.14}$$

where

$$p(\alpha) = \frac{1}{\sqrt{\pi N_o}} e^{-\alpha^2/N_o} \quad . \tag{4.15}$$

By symmetry

$$P(c|m_j) = P(c|m_0) = P_W(c) \quad . \tag{4.16}$$

Equation (4.14) can be expressed in a slightly different form by making the changes of variable $z = \beta\sqrt{2/N_o}$ and $u = (\alpha - E_s)\sqrt{2/N_o}$ and by replacing E_s by kE_b, where E_b is the received energy per information bit. Then

$$P_W(c) = \int_{-\infty}^{\infty} \frac{e^{-u^2/2}}{\sqrt{2\pi}} du \left[\int_{-\infty}^{u + (2kE_b/N_o)^{1/2}} \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz \right]^{2^k - 1} \; , \tag{4.17}$$

and the probability of word error is

$$P_W(e) = 1 - P_W(c) \quad . \tag{4.18}$$

The word error probability for bi-orthogonal codes can be calculated in the same manner. The set of $M = 2^k$ bi-orthogonal code words is formed from a set of 2^{k-1} orthogonal code words and their complements. Thus only 2^{k-1} correlators are needed assuming that the sign of the output can be determined. The optimum decision rule is to select the correlator whose output has the greatest absolute value and determine its sign. The correct code word is selected if and only if the absolute value of the output of its correlator is greater than the absolute values of the outputs of all other correlators and it has the correct sign.

Assuming that $\bar{s}_0$ is transmitted, the received vector is closer to $\bar{s}_0$ than to $-\bar{s}_0$ if and only if

$$r_0 \geq 0 \ . \tag{4.19}$$

In addition, the received vector is closer to $\bar{s}_0$ than to $\bar{s}_i$ if and only if

$$r_0 > r_i, \ i = 1,2,\cdots,(M/2)-1 \ , \tag{4.20}$$

and it is closer to $\bar{s}_0$ than to $-\bar{s}_i$ if and only if

$$r_0 > -r_i, \ i = 1,2,\cdots,(M/2)-1 \ , \tag{4.21}$$

Given that $\bar{s}_0$ is transmitted and that

$$r_0 = n_0 + \sqrt{E_s} = \alpha > 0 \ , \tag{4.22}$$

the conditional probability of a correct decision is

$$\begin{aligned} P(c|m_0, r_0 = \alpha > 0) &= P(-\alpha < n_1 < \alpha, \cdots, -\alpha < n_{(M/2)-1} < \alpha) \\ &= \left[P(-\alpha < n_1 < \alpha)\right]^{(M/2)-1} \\ &= \left[\int_{-\alpha}^{\alpha} p(\beta)d\beta\right]^{(M/2)-1} \ , \end{aligned} \tag{4.23}$$

where

$$p(\beta) = \frac{1}{\sqrt{\pi N_o}} e^{-\beta^2/N_o} \quad . \tag{4.24}$$

Then

$$P(c|m_0) = \int_0^\infty p(\alpha - \sqrt{E_s}) d\alpha \left[\int_{-\alpha}^{\alpha} p(\beta) d\beta \right]^{(M/2)-1} \quad . \tag{4.25}$$

Again, by symmetry, $P_W(c) = P(c|m_0)$. Making the changes of variable $u = (\alpha - E_s)\sqrt{2/N_o}$ and $z = \beta\sqrt{2/N_o}$, then

$$P_W(c) = \int_{-(2kE_b/N_o)^{1/2}}^{\infty} \frac{e^{-u^2/2}}{\sqrt{2\pi}} du \left[\int_{-u-(2kE_b/N_o)^{1/2}}^{u+(2kE_b/N_o)^{1/2}} \frac{e^{-z^2/2}}{\sqrt{2\pi}} dz \right]^{2^{k-1}-1} \tag{4.26}$$

and

$$P_W(e) = 1 - P_W(c) \quad . \tag{4.27}$$

Viterbi has calculated the bit error probability for orthogonal and bi-orthogonal codes.[29] For this discussion it suffices to point out that for orthogonal and bi-orthogonal codes the bit error probability is approximately related to the word error probability by

$$P_B(e) \approx \frac{1}{2} P_W(e) \quad . \tag{4.28}$$

The exponential behavior of the error probability for orthogonal signaling can be derived by appropriately bounding (4.18).[30] It is convenient to use normalized units in presenting these results.[31] Binary antipodal signaling with a correlation detector is assumed. The code consists of $M = e^{RT}$ code words each of duration T seconds, and the signaling rate is R nats per second. In this case the units of rate are defined to be nats instead of bits. Conversions can be made from the relationships

$$1 \text{ nat} = \log_2 e \text{ bits} = 1.4 \text{ bits} , \qquad 1 \text{ bit} = \ln 2 \text{ nats} = 0.69 \text{ nats} . \tag{4.29}$$

Then

$$R = \frac{\ln M}{T} = \frac{k^*}{T} , \tag{4.30}$$

where $M = e^{k^*}$. The channel capacity for this channel is

$$C = \frac{P_{av}}{N_o} \text{ nats/sec} , \tag{4.31}$$

where P_{av} is the average transmitter signal power in watts and N_o is the single-sided noise spectral density in watts/Hz.

In terms of E_b/N_o, the signal energy-to-noise ratio per information bit, C and be expressed as

$$C = \frac{E_b}{N_o} \frac{\log_2 M}{T} = \frac{E_b}{N_o} \frac{k^*}{T \ln 2} . \tag{4.32}$$

The normalized rate can now be defined as

$$r^* = \frac{R}{C} = \frac{\ln 2}{E_b/N_o} , \tag{4.33}$$

and the normalized length can be defined as

$$n^* = TC = \frac{k^*}{\ln 2} \frac{E_b}{N_o} . \tag{4.34}$$

By bounding (4.18) and using the normalized units, the error probability is bounded by

$$P_W(e) \leq \exp\left[-n^* e(r^*)\right] \quad , \tag{4.35}$$

where

$$e(r^*) = \begin{cases} (1/2 - r^*), & 0 \leq r^* \leq 1/4 \; , \\ (1 - \sqrt{r^*})^2, & 1/4 \leq r^* \leq 1 \; . \end{cases} \tag{4.36}$$

The probability of error approaches zero exponentially with increasing n^* for all normalized rates in the range $0 \leq r^* < 1$, since $e(r^*)$ is positive over that range. The quantity, $e(r^*)$, determines the exponential behavior of the error probability and is called the error exponent.

The probability of word and bit errors for orthogonal and bi-orthogonal codes has been tabulated for a wide range of values of k and E_b/N_o. In Fig. 4.1 the probability of word error as a function of E_b/N_o is shown for k = 2 through k = 8 for bi-orthogonal codes. The performance of bi-orthogonal codes is slightly better than that of orthogonal codes, but the difference is very small. For both types of codes $P_B(e) \approx P_W(e)/2$.

C. Complexity of Implementation

It was indicated previously that correlation detection will be used. In this section a digital implementation scheme is presented, and its complexity is examined. During the j'th bit period, $1 \leq j \leq n$, the transmitted signal is $\pm\, s_o(t - \frac{j-1}{n} T)$ depending upon whether the transmitted bit is a one or a zero. In performing the correlation of the received waveform, r(t), with the i'th message, $s_i(t)$, the form of the result is

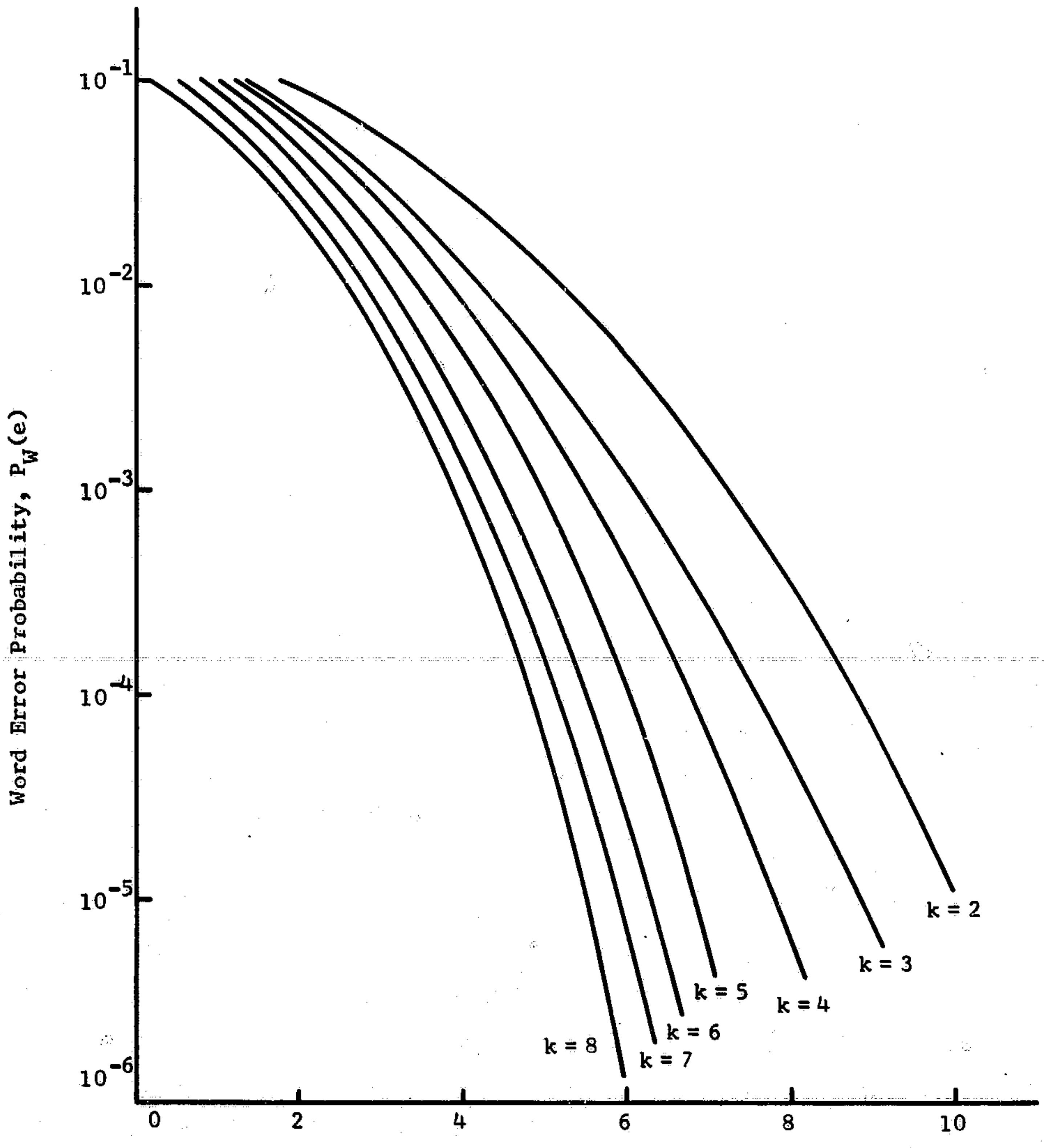

Fig. 4.1. Performance of Bi-orthogonal Codes.

$$\int_0^T r(t)s_i(t)dt = \int_0^T r(t) \sum_{j=1}^{n} \pm s_o(t - \frac{j-1}{n} T)dt$$

$$= \pm\int_0^{T/n} r(t)s_o(t)dt + \cdots$$

$$\pm\int_{\frac{n-1}{n}T}^{T} r(t)s_o(t - \frac{n-1}{n} T)dt \quad . \tag{4.37}$$

This process is simplified by multiplying r(t) by $\sum_{j=1}^{n} s_o(t - \frac{j-1}{n} T)$ and integrating over each subinterval of duration T/n. The integrator output is converted to digital form at the end of each subinterval. Since the signs are determined by the code for $s_i(t)$, the output of the analog-to-digital conversion for the j'th subinterval is either added to or subtracted from the i'th accumulator depending upon whether the j'th symbol of the code word $x^{(i)}$ is "1" or "0". Finally, at the end of the n'th subinterval the numbers in the M accumulators are compared, and the largest is selected as the most probable transmitted code word. A block diagram of this decoding scheme is shown in Fig. 4.2.

The approach described above requires quite a bit of hardware. By using a serial approach, equipment complexity can be reduced considerably at the cost of a longer decoding delay. This consists of storing the n quantized outputs of the integrator for the n subintervals of each code word. The additions and subtractions are then done for each word serially, and the comparison is made with the previous maximum correlation value. A block diagram of this implementation is shown in Fig. 4.3. While correlations are performed on one received word, another word is being

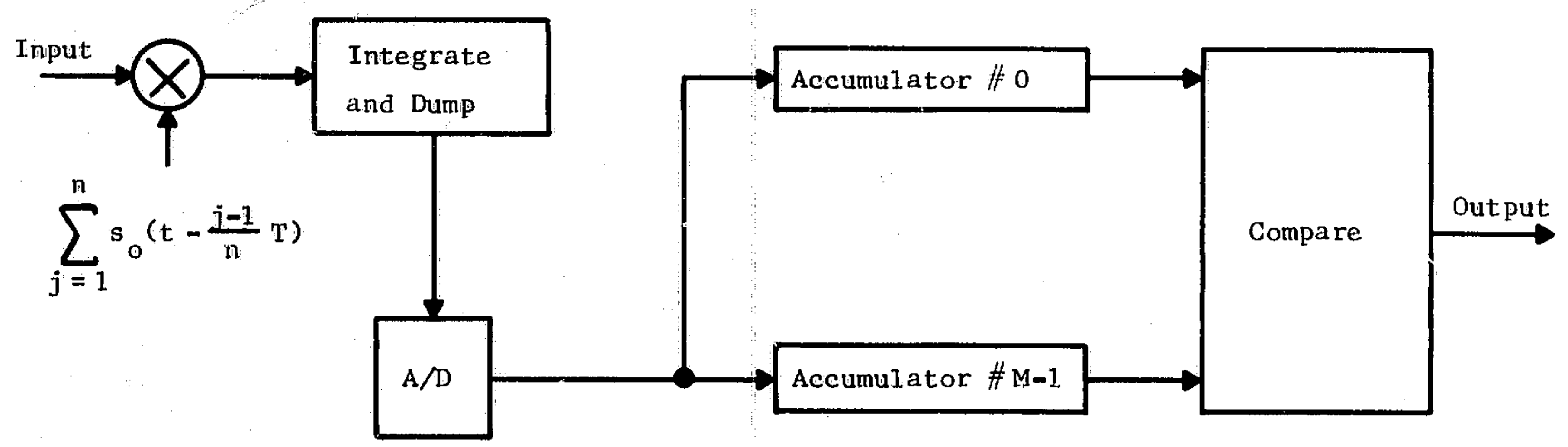

Fig. 4.2. A Parallel Correlation Decoder.

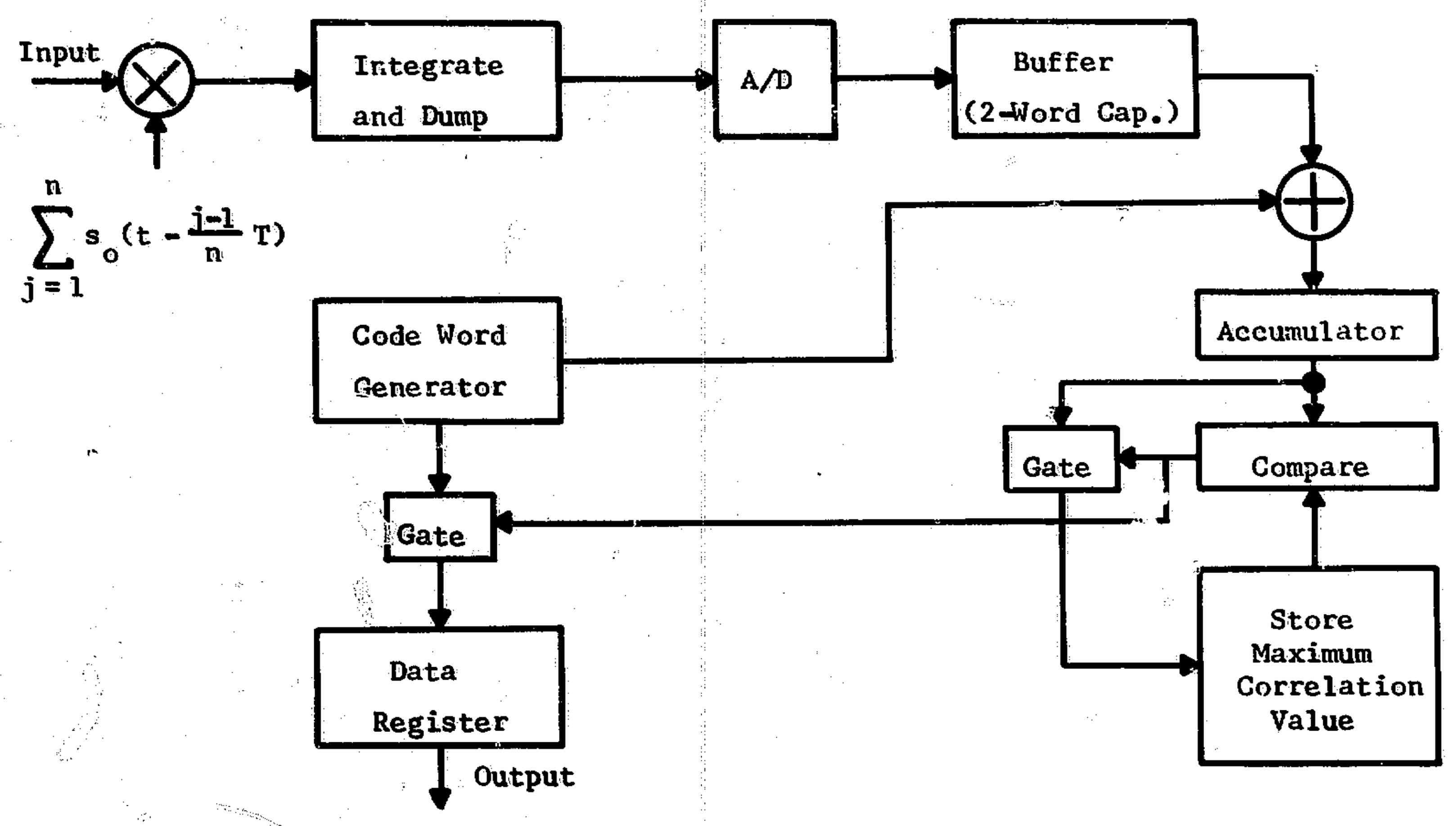

Fig. 4.3. A Serial Correlation Decoder.

received. Thus, a buffer with a two word capacity is needed. Also, two accumulators are needed, one to store the latest correlation value and one to store the previous maximum correlation. The number of stages in these registers is k plus the number of bits in the quantized output of the integrator.

It is shown in Wozencraft and Jacobs that if the signal-to-noise ratio is small enough so that binary signaling is efficient, then two-level quantization of the integrator output results in a performance degradation of about 2 dB while three-level quantization results in a degradation of about 1 dB.[32] This indicates that the quantization does not have to be very fine to obtain a negligible performance degradation. It will be assumed that the integrator output has been quantized to six bits (64 levels). This will result in negligible performance degradation.

The following estimates of decoding delay are derived with the assumption that decoding begins after the outputs of the integrator are stored in the buffer. To decode an orthogonal code, $n = 2^k$ additions for each code word are required, and there are $M = 2^k$ code words. Also, a comparison must be made for each code word. Assuming that an addition and a comparison require approximately the same length of time to complete, decoding requires a total of

$$N_C = 4^k + 2^k \tag{4.38}$$

computations.

In estimating the equipment complexity, the buffer contains two registers each consisting of six parallel registers of 2^k stages for a total of $12 \cdot 2^k$ binary storage elements. The code word generator contains k stages and must be fed by a counter with k stages. Also two accumulators with $k + 6$ stages and a data register of k stages are needed. Thus, the number of binary storage elements required in the decoder is approximately

$$N_{SD} = 3 \cdot 2^{k+2} + 5k + 12 \quad . \tag{4.39}$$

The number of binary storage elements required in the encoder is

$$N_{SE} = k \quad . \tag{4.40}$$

For a bi-orthogonal code the decoder complexity is reduced somewhat. Since $n = 2^{k-1}$, only 2^{k-1} additions per correlation need to be made, and since only one correlation for both a code word and its complement need to be made, a total of $M/2 = 2^{k-1}$ correlations must be made. Thus, a total of 4^{k-1} additions are needed. Since $M/2 = 2^{k-1}$ comparisons are also needed, decoding requires approximately

$$N_C = 4^{k-1} + 2^{k-1} \tag{4.41}$$

computations. Also, since $n = 2^{k-1}$, the total number of binary storage elements required in the buffer is $12 \cdot 2^{k-1}$. The rest of the decoder will have approximately the same complexity as the decoder for an orthogonal code. Thus, the total number of binary storage elements required is approximately

$$N_{SD} = 3 \cdot 2^{k+1} + 5k + 12 \quad . \tag{4.42}$$

The number of binary storage elements required in the encoder is

$$N_{SE} = k - 1 \quad . \tag{4.43}$$

The number of computations required to decode orthogonal and bi-orthogonal codes is shown in Fig. 4.4 as a function of the number of information bits per code word. The number of computations required in decoding a bi-orthogonal code is approximately one-fourth that required in decoding an orthogonal code with the same number of information

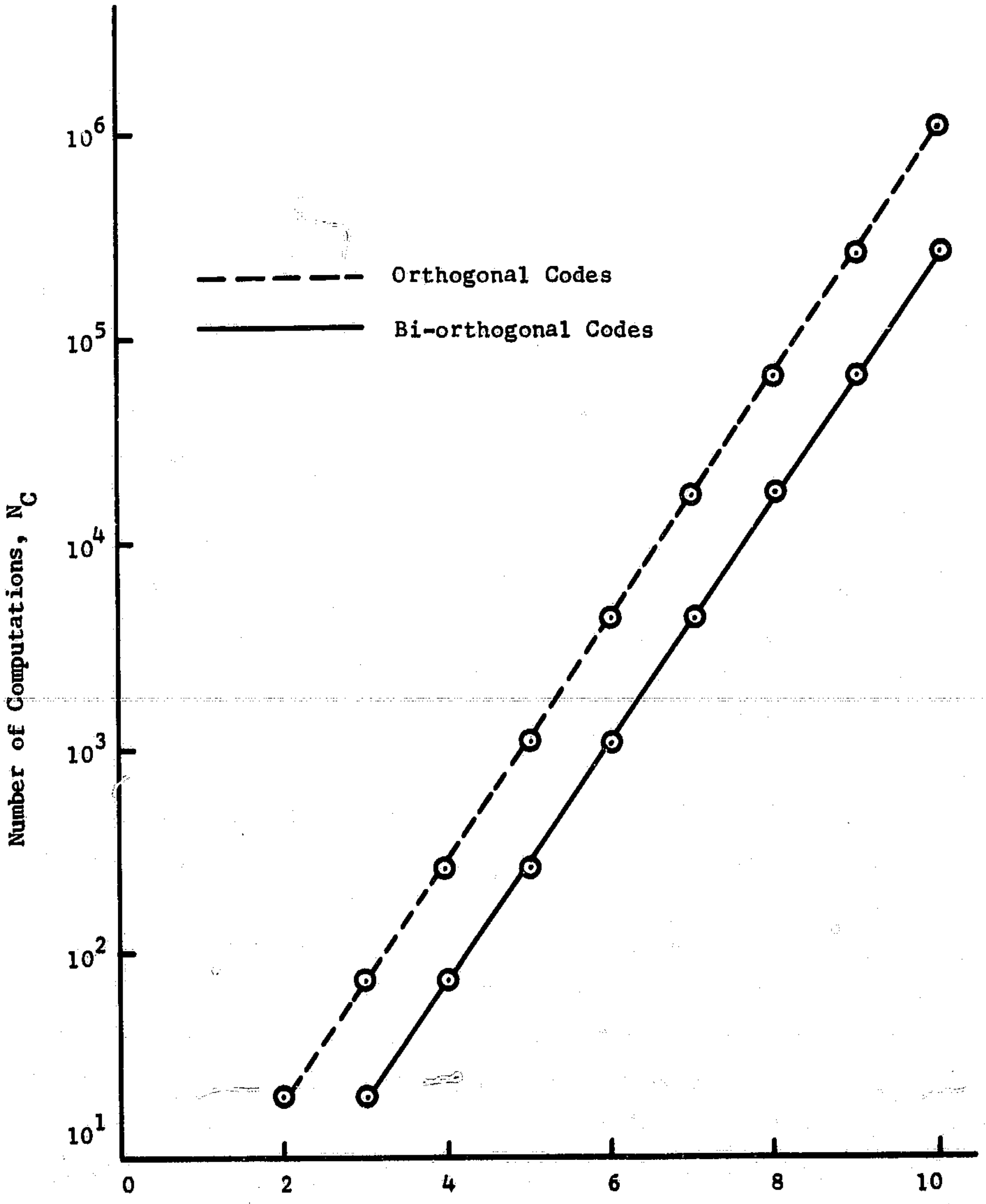

Fig. 4.4. Number of Computations Required in Decoding Orthogonal and Bi-orthogonal Codes.

bits per code word. In Fig. 4.5 the number of computations per decoded information bit is shown as a function of the number of information bits per code word for orthogonal and bi-orthogonal codes. The number of binary storage elements required in the two decoders is shown in Fig. 4.6 as a function of the number of information bits per code word. In this case the number of storage elements required in decoding a bi-orthogonal is approximately one-half the number required in decoding an orthogonal code with the same number of information bits per code word. In all three figures the measures of complexity increase exponentially with k. From Fig. 4.1 it is seen that for large k only a small increase in performance is obtained by increasing k by one. Since increasing k by one more than doubles decoder complexity, a practical upper limit on k or nine is obtained. For example, using a bi-orthogonal code with k = 9 requires that the decoder logic have a speed advantage of approximately 7,000 over the received data rate.

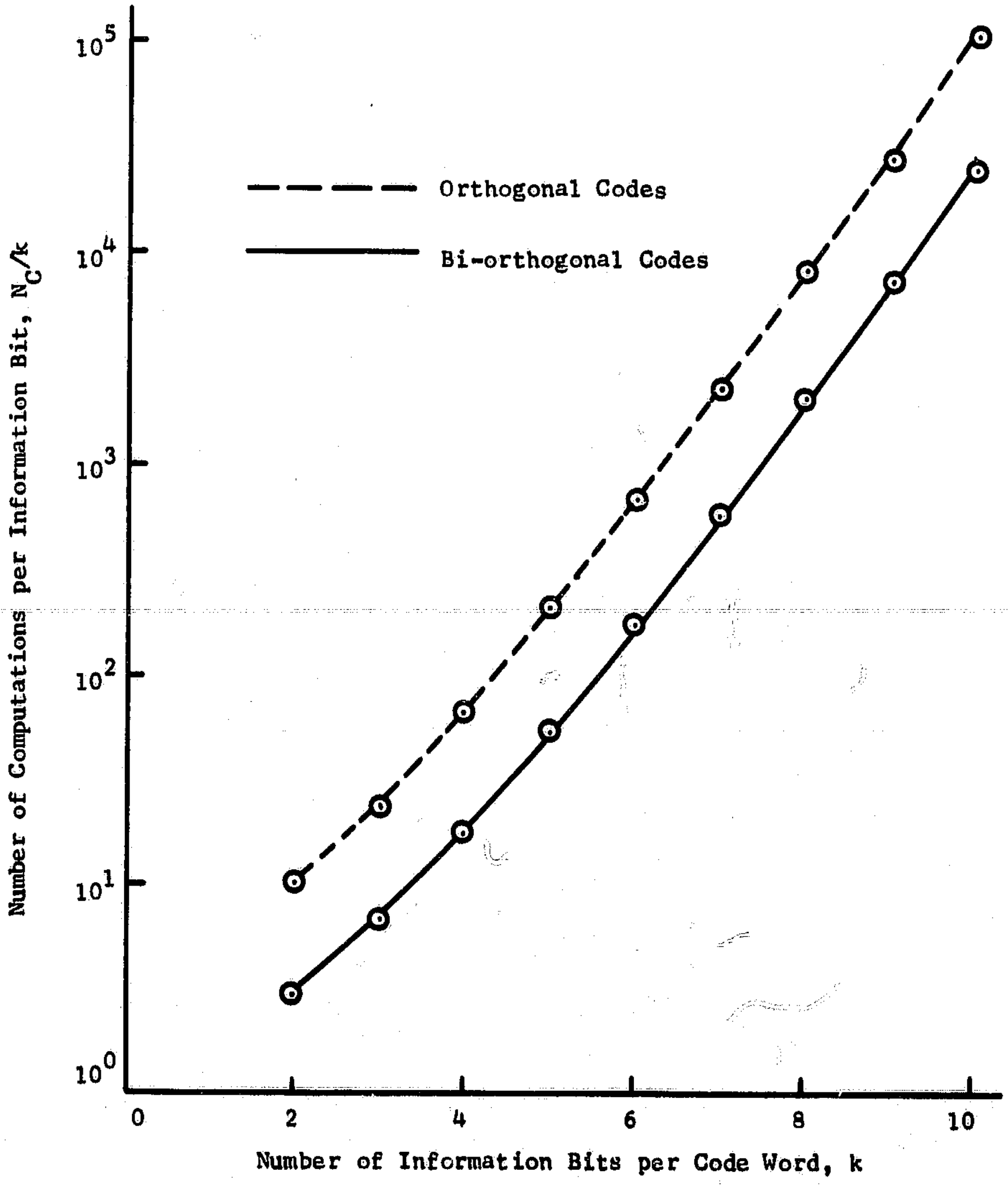

Fig. 4.5. Number of Computations per Information Bit Required in Decoding Orthogonal and Bi-orthogonal Codes.

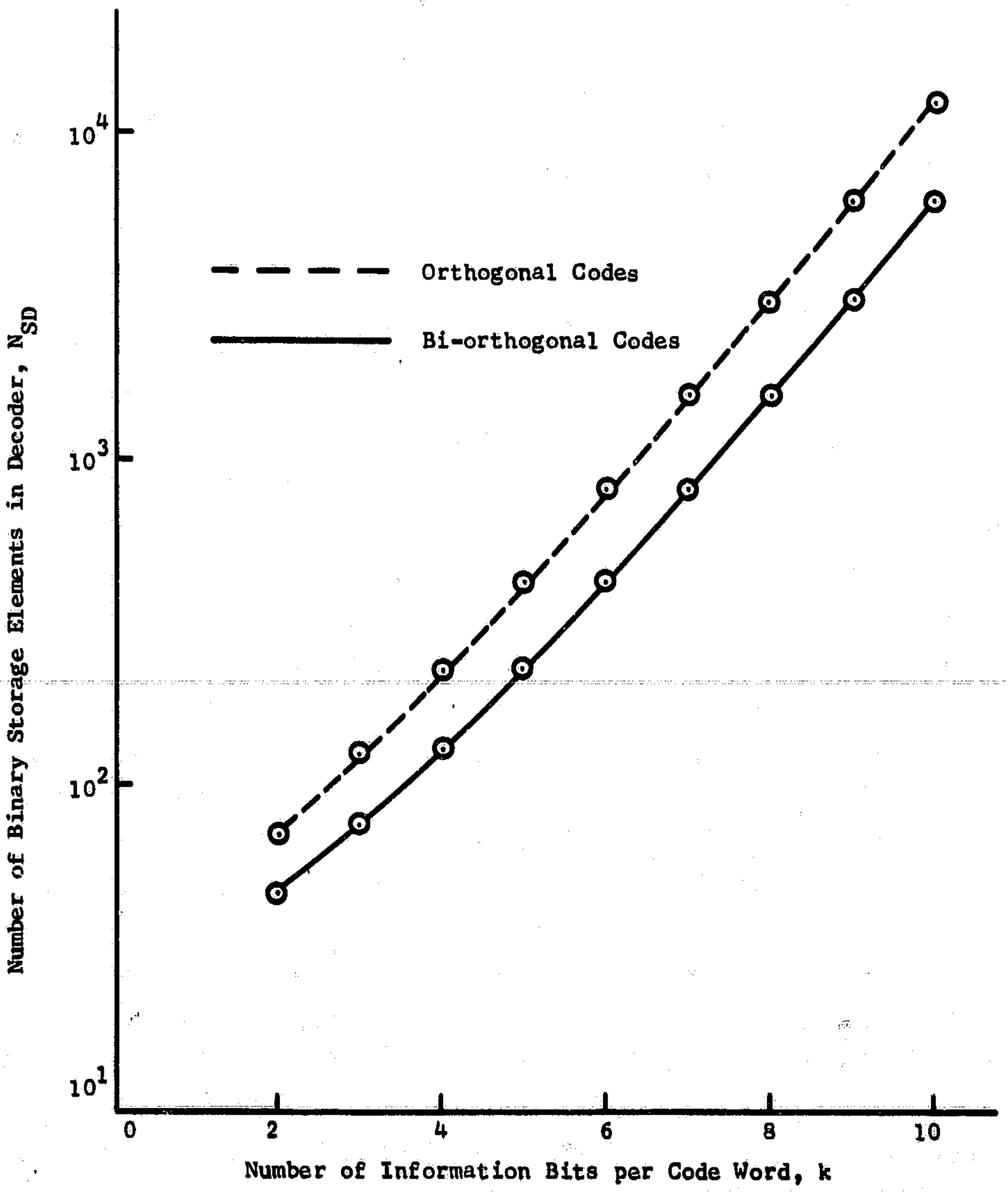

Fig. 4.6. Number of Binary Storage Elements Required in the Decoder for Orthogonal and Bi-orthogonal Codes.

CHAPTER V

THRESHOLD DECODING

Threshold decoding is a technique introduced by Massey by which decoding is accomplished by a majority count of the parity check bits associated with a given information bit.[33] This method is easily implemented, but it can be applied only to codes with special properties. In this chapter the threshold decoding procedure will be described. Also, the performance of some block codes which can be threshold decoded will be presented, and their complexity of implementation will be evaluated.

A. The Threshold Decoding Procedure

All of the definitions and theorems in this section are due to Massey.[34] Two decoding procedures are presented. Majority decoding is an easily implemented procedure, but a-posteriori-probability decoding is a somewhat more complicated decoding scheme which utilizes channel statistics.

Majority decoding. Code words are represented by n-tuples of the form $(f_1, f_2, \cdots, f_n)$, where each f_i is an element of GF(q). It is assumed that the code is in systematic form, i.e.,

$$f_j = \sum_{i=1}^{k} c_{ji} f_i, \quad j = k+1, k+2, \cdots, n \quad , \tag{5.1}$$

where $f_1, f_2, \cdots, f_k$ are information symbols, $f_{k+1}, \cdots, f_n$ are parity check symbols, and the c_{ji} are elements of GF(q) determined by the parity check matrix of the code. Since the channel adds a noise vector

$(e_1, e_2, \cdots, e_n)$ to the transmitted vector $(f_1, f_2, \cdots, f_n)$, the received vector $(r_1, r_2, \cdots, r_n)$ is determined by

$$r_i = f_i + e_i, \; i = 1,2,\cdots,n \; . \tag{5.2}$$

Defining the parity checks,

$$S_j = \sum_{i=1}^{k} c_{ji} r_i - r_j, \; j = k+1, k+2, \cdots, n \; , \tag{5.3}$$

and using (5.1) and (5.2),

$$S_j = \sum_{i=1}^{k} c_{ji} e_i - e_j, \; j = k+1, k+2, \cdots, n \; , \tag{5.4}$$

which is a set of n-k linear equations in the n unknowns $\left\{e_i\right\}$. However, the set of composite parity checks,

$$A_i = \sum_{j=k+1}^{n} b_{ij} S_j \; , \tag{5.5}$$

are more convenient for decoding purposes. From (5.4) and (5.5),

$$A_i = \sum_{j=k+1}^{n} b_{ij} \left[\sum_{h=1}^{k} c_{jh} e_h - e_j \right] . \tag{5.6}$$

This equation may also be written as

$$A_i = \sum_{j=1}^{n} a_{ij} e_j \; , \tag{5.7}$$

where

$$a_{ij} = \begin{cases} \sum_{h=k+1}^{n} b_{ih} c_{hj}, & j = 1,2,\cdots,k \; , \\ b_{ij}, & j = k+1, k+2, \cdots, n \; . \end{cases} \tag{5.8}$$

The majority decoding procedure is a technique of determining the noise digits, e_j, $j = 1, 2, \cdots, n$. If the noise digit, e_j, is included in the equation for the parity check, A_i, it is said that e_j is checked by A_i.

Definition 5.1. A set of J composite parity checks is called orthogonal on e_m if e_m is checked by each member of the set, but no other noise digit is checked by more than one member of the set. Thus, all J equations are affected by e_m, but no other noise digit affects more than one equation.

Theorem 5.1. If $\lfloor J/2 \rfloor$ or fewer of the $\{e_j\}$ that are checked by a set of J parity checks $\{A_i\}$ orthogonal on e_m are nonzero (i.e., there are $\lfloor J/2 \rfloor$ or fewer errors in the corresponding received symbols), then e_m is given correctly as that value of GF(q) which is assumed by the greatest fraction of the $\{A_i\}$. (If no value is assumed by a strict plurality of the $\{A_i\}$, and 0 is one of the several values with most occurrences, the value $e_m = 0$ is used.)

The key to the proof of Theorem 5.1 is recognition that if $\lfloor J/2 \rfloor$ or fewer errors occur, then at most $\lfloor J/2 \rfloor$ of the J parity checks are affected, and therefore at least $\lceil J/2 \rceil$ of the parity checks give the correct value of e_m. The technique described in this theorem is called majority decoding.

For the binary case Theorem 5.1 reduces to the following theorem.

Theorem 5.2. Given a set $\{A_i\}$ of J parity checks orthogonal on e_m, then the majority decoding rule is to choose

$e_m = 1$ if and only if the sum of the A_i (as real numbers) exceeds the threshold value $\lceil J/2 \rceil$.

Majority decoding is easily implemented once a set of orthogonal parity checks is found. However, a useful set of orthogonal parity checks can be found only for codes with special properties.

A-posteriori-probability decoding. Majority decoding is somewhat inefficient since it does not utilize the channel statistics. A-posteriori-probability decoding (APP decoding) is a technique which utilizes additional channel information.

Theorem 5.3. Given a set $\{A_i\}$ of J parity checks orthogonal on e_m and that the noise sequence is additive with digit-to-digit independence, then the decoding rule based on $\{A_i\}$ which determines e_m with the least average probability of error is to choose e_m to be that value V of GF(q) for which

$$\log \left[P(e_m = V) \right] + \sum_{i=1}^{J} \log \left[P(A_i \; e_m = V) \right]$$

is a maximum.

It is evident from Theorem 5.3 that the better performance of an APP decoder will be paid for with increased equipment complexity. The performance of an APP decoding scheme is not easily evaluated since it must be calculated by system simulation.

An APP decoding algorithm suitable for use on the Gaussian channel has been given by Massey.[35] If the set $\{A_i\}$ of J parity checks is orthogonal on e_1, then $e_1 = 1$ is chosen when

$$\sum_{i=1}^{J} w_i \, A_i > T. \tag{5.9}$$

Otherwise, $e_1 = 0$ is chosen. The threshold T is

$$T = \frac{1}{2} \sum_{i=0}^{J} w_i \quad . \tag{5.10}$$

The weights, w_i, have a complicated form. Corresponding to the set $\{A_i\}$, a new set of equations $\{C_i\}$ is defined by

$$C_i = \sum_{j=1}^{n} d_{ij}, \; i = 1,2,\cdots,J \quad , \tag{5.11}$$

where if e_j, $j \neq 1$, is included in the parity check A_i, then

$$d_{ij} = -\ln\left[1 - 2P(e_j = 1)\right] \quad , \tag{5.12}$$

and

$$d_{ij} = 0, \text{ otherwise} \quad . \tag{5.13}$$

The weights are defined from the $\{C_i\}$ by

$$w_i = 2 \ln\left[\coth(C_i/2)\right] \tag{5.14}$$

and

$$w_0 = 2 \ln\left[\coth(C_0/2)\right] \quad , \tag{5.15}$$

where

$$C_0 = -\ln\left[1 - 2P(e_1 = 1)\right] \quad . \tag{5.16}$$

A block diagram of the decoder needed to implement the APP decoding algorithm for the Gaussian channel is shown in Fig. 5.1. The integrate-and-dump filter has two outputs. The polarity of the output of the filter provides an initial estimate of the received digit, and this

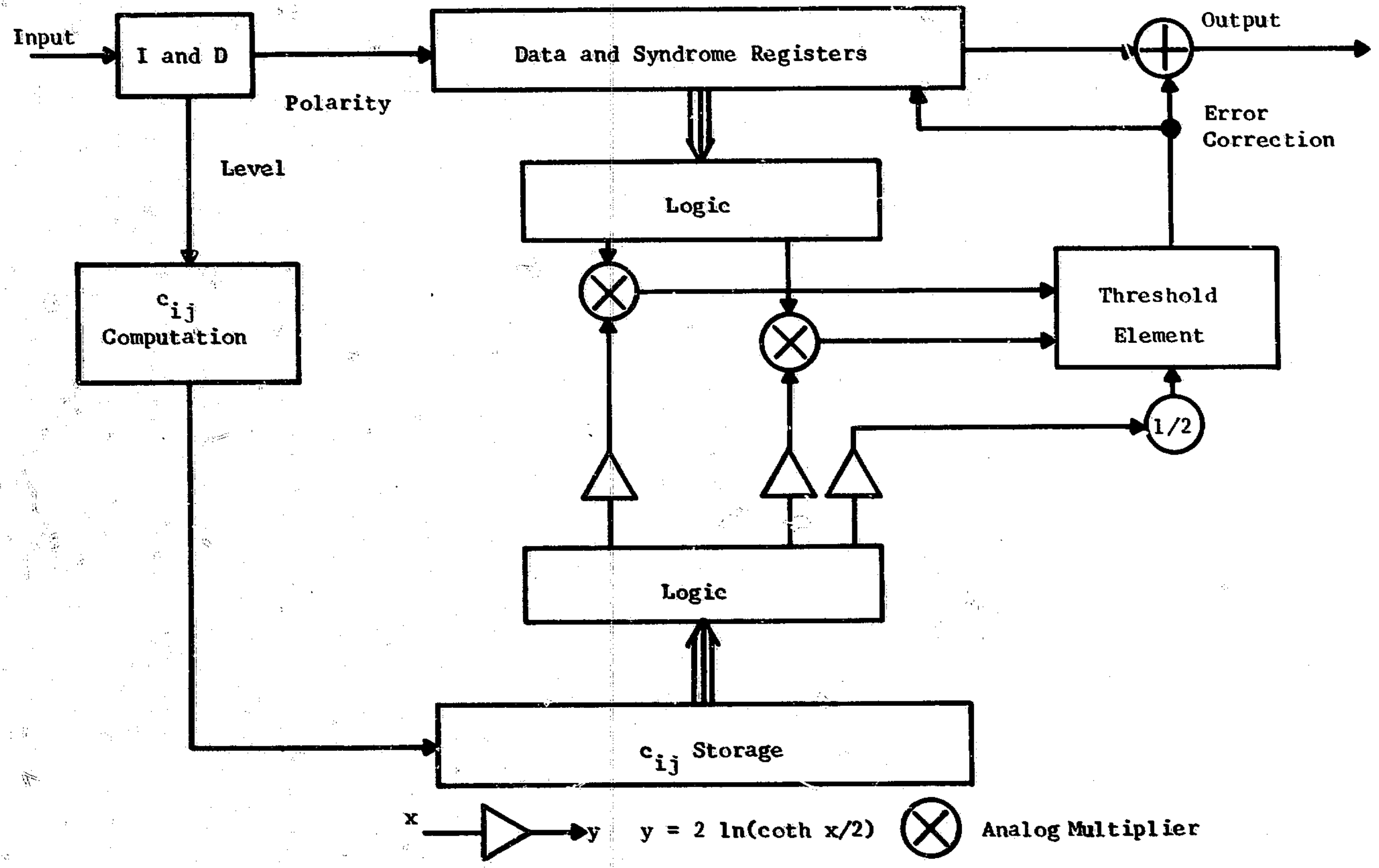

Fig. 5.1. A-Posteriori-Probability Decoder.

binary output is shifted into the data and syndrome registers. The level of the output is used to calculate the probability of this initial estimate being in error. Also, when an error is corrected, the syndrome is modified to remove the effect of the error. Three types of feedback are used with the d_{ij}:

1. Soft feedback. As bits are corrected the d_{ij} are not changed but are simply circulated in the register.
2. Hard decision feedback. After a bit is decoded its error probability is set to zero, hence the corresponding d_{ij} is zero. This causes some difficulty near the end of each decoded block because the weights approach infinity. By assigning a low error probability to each decoded bit this difficulty can be overcome.
3. Full APP feedback. After each bit is decoded the bit error probability after decoding is fed back to compute its d_{ij}.

B. Threshold-Decodable Codes

As mentioned before, only codes with special properties are threshold decodable. Since threshold decoding is so easily implemented, there is much interest in finding codes which are threshold decodable. In describing the required properties of a threshold decodable code, the following definition is helpful.

Definition 5.2. A block (n,k) can be completely orthogonalized if $d-1$ parity checks orthogonal on each e_j, $j = 1, 2, \cdots, k$, can be formed where d is the minimum distance of the code.

It is easily seen from Theorem 5.1 that if majority decoding is used with a code that can be completely orthogonalized and if there are no more than $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors, each of the noise digits corresponding to the k information digits will be correctly evaluated. The k information digits are determined from (5.2). If the parity check digits are needed, they can be obtained through the encoding process of (5.1).

Cyclic codes which can be completely orthogonalized have particularly simple threshold decoding circuits. The simple decoding circuits result from the property that a cyclic code can be completely orthogonalized if and only if d-1 parity checks orthogonal on e_1 can be formed. This follows from the basic property of a cyclic code that any cyclic shift of a code word is another code word. Then the parity checks on e_2 must be of the same form as the parity checks on e_1 with all indices increased cyclically by one. Thus, if d-1 parity checks orthogonal on e_1 can be formed, then d-1 parity checks orthogonal on e_2 can be formed. A similar argument can be made for $e_3, e_4, \cdots, e_k$. If it is not evident at this point how this property will make implementation easier, it will be evident later when implementation is discussed in greater detail.

Complete orthogonalization is generally not possible. However, for some codes it is possible to perform a generalized orthogonalization procedure called L-step orthogonalization. In this procedure starting with the original set of parity checks corresponding to the parity check matrix H, sets of at least d-1 parity checks orthogonal on selected sums of noise bits corresponding to the information bits are formed. Assuming these sums are known (using threshold decoding to estimate each sum) and treating them as additional parity checks, they can be combined with the original parity checks to form a set of parity checks corresponding to a parity check matrix $H^{(1)}$. Similarly $H^{(1)}$ can be transformed into $H^{(2)}$.

Finally, at the L'th step a matrix $H^{(L)}$ is produced from which d-1 parity checks orthogonal on e_1 can be obtained. If this can be done for all e_j, $j = 1,2,\cdots,k$, the code can be L-step orthogonalized. Thus, complete orthogonalization is equivalent to one-step orthogonalization. Although a code can be threshold decoded if it can be L-step orthogonalized, this does not mean the decoder will be simple. For large L the decoder will be unreasonably complex. For codes which correct few errors but which cannot be orthogonalized in a small number of steps, an algebraic decoding algorithm will be simpler to implement. Only codes will be considered which can be orthogonalized in a small number of steps.

Attempts have been made to find classes of codes which can be easily threshold decoded. Weldon's difference-set cyclic c es are completely orthogonalized.[36] Rudolph's projective-geometry codes can be orthogonalized in a small number of steps.[37] The class of codes to be studied here is Weldon's non-primitive Reed-Muller codes which contains both the difference-set codes and the projective-geometry codes as subclasses.[38] The performance of some of the better codes of this class will be examined, and measures of the implementation complexity will be given.

Because the complete treatment of non-primitive Reed-Muller codes is quite lengthy and based on projective geometry, which has not been introduced, the codes will just be defined, and some of their properties given. Derivations of all properties given here may be found in Weldon.[39]

Definition 5.3. If the number v is represented in the form

$$v = b_0 + b_1 2^1 + \cdots + b_{m-1} 2^{m-1} \quad , \qquad (5.17)$$

where b_i denotes a binary digit, then the weight of v is given by

$$w(v) = \sum_{i=0}^{m-1} b_i \quad , \tag{5.18}$$

where real addition is performed in (5.18).

A cyclic code is called non-primitive if every root of the generator polynomial is a power of a given non-primitive element β of $GF(2^m)$. The length, n, of this type of code is the order of β and must divide $2^m - 1$. Letting

$$r = \frac{2^m - 1}{n} \quad , \tag{5.19}$$

then

$$n = \frac{2^m - 1}{r} \quad . \tag{5.20}$$

Now the non-primitive Reed-Muller codes can be defined, and some of their properties can be stated.

Definition 5.4. The cyclic $([2^m-1]/r,k)$ code whose generator polynomial contains all roots α^{rti} such that $w(rti) \leq v$ will be referred to as the v'th order non-primitive Reed-Muller code.

The codes which have

$$r = 2^s - 1 \tag{5.21}$$

for some integer s have special properties. Only these codes will be discussed. In this case, since

$$n = \frac{2^m - 1}{2^s - 1} \quad , \tag{5.22}$$

m/s is an integer which will be denoted by $z+1$. Then

$$n = 2^{sz} + 2^{s(z-1)} + \dots + 2^s + 1 \quad . \tag{5.23}$$

The basic properties of the non-primitive Reed-Muller codes are summarized in the following theorem.

Theorem 5.4. The v'th order non-primitive Reed-Muller codes for which $r = 2^s$ have the following parameters:

$$n = \frac{2^m - 1}{2^s - 1} ,$$

$k =$ number of integers less than $2^m - 1$ which are divisible by $2^s - 1$ whose weight exceeds v,

and

$$d \geq 2^{sh} + 2^{s(h-1)} + \dots + 2^s + 2 ,$$

where $h = v/s$. The generator polynomial of such a code contains as roots every power of α whose exponent has weight v or less and is divisible by $2^s - 1$.

These codes may be $(z-h)$-step orthogonalized where $z = \frac{m}{s} - 1$. Some codes of this class with parameters of interest are shown in Table 5.1.

The codes in Table 5.1 with $z - h = 1$ can be completely orthogonalized. In this case the decoder is particularly simple as shown in Fig. 5.2. When a word is received the D-P switch is placed in the D position, and the k-symbol data sequence is shifted simultaneously into the syndrome and data registers. After the data sequence is in the registers the switch is thrown to position P and the (n-k)-symbol parity check sequence is shifted into the syndrome register forming the syndrome. At this point since the data register contains the syndrome, decoding can be started. The output of the majority gate equals the additive inverse

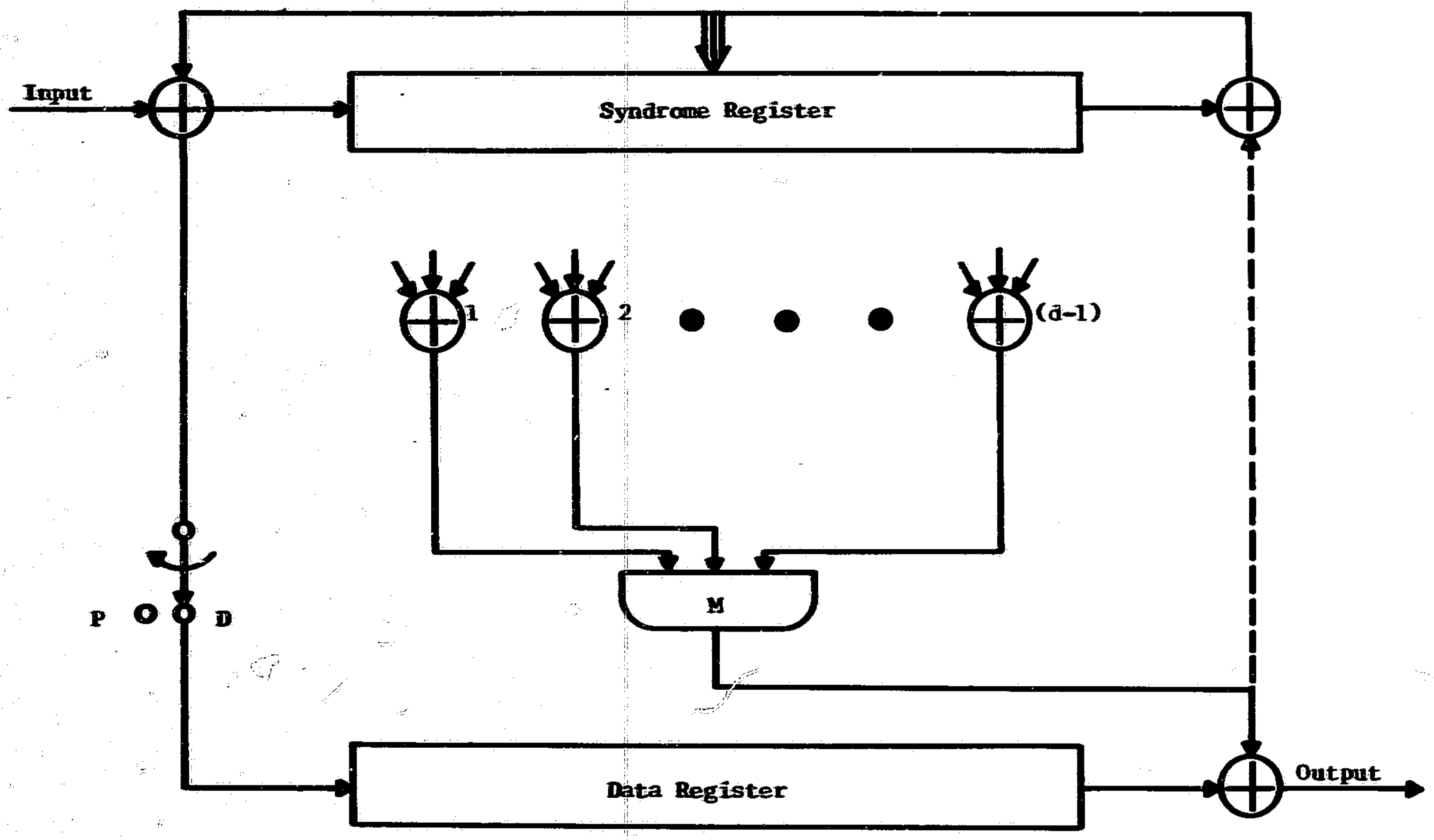

Fig. 5.2. Majority-Logic Decoder for a Code which Can Be Completely Orthogonalized.

Table 5.1. Non-Primitive Reed-Muller Codes.

n	k	d	s	h	m	z	z-h
21	11	6	2	1	6	2	1
73	45	10	3	1	9	2	1
85	68	6	2	1	8	3	2
85	24	22	2	2	8	3	1
273	191	18	4	1	12	2	1
341	315	6	2	1	10	4	3
341	195	22	2	2	10	4	2
585	520	10	3	1	12	3	2
585	184	74	3	2	12	3	1
1057	813	34	5	1	15	2	1
1365	328	6	2	1	12	5	4
1365	1063	22	2	2	12	5	3
1365	483	86	2	3	12	5	2
4161	3431	66	6	1	18	2	1

of the noise digit which was added to first data digit. To correct the first data symbol, the output of the majority gate is simply added to the received symbol. As explained previously, due to the cyclic nature of the code all errors in the received data symbols can be corrected by repeating this process k times. Performance can be improved slightly at virtually no cost by adding the dotted connection and the adder indicated. By doing this the effects of corrected errors are removed from the syndrome. Addition of this connection will allow correction of many error patterns which would not be corrected without the connection.

C. Performance and Complexity of Implementation

The complexity of a majority-logic decoder for codes which are completely orthogonalizable can be estimated in a straightforward manner.

Two syndrome registers and two data registers are used for buffering purposes. While one is being used to decode a particular word, the other will be receiving the next incoming word. Only one majority gate is needed. Each syndrome register has (n-k) shift-register stages while each data register has k stages. Thus, the decoder requires a total of

$$N_{SD} = 2n \tag{5.24}$$

binary storage elements. The decoder also requires one majority gate with

$$M_I = d - 1 \tag{5.25}$$

inputs. The inputs to the majority gate are either directly from the syndrome register or from modulo-2 adders whose inputs are from the syndrome register. The number of computations required to execute the decoding procedure is

$$N_C = k . \tag{5.26}$$

This is extremely advantageous because data rates of the same order of magnitude as the decoder clock cycle can be used. Thus, high data rates are possible. Since the codes are cyclic either of the encoders of Figs. 2.7 and 2.8 can be used. The encoder contains a total of

$$N_{SE} = \begin{cases} k \ , k \leq n-k \ , \\ n-k, k > n-k \ , \end{cases} \tag{5.27}$$

binary storage elements. These four measures of implementation complexity have been evaluated for the codes of Table 5.1 which can be completely orthogonalized and are tabulated in Table 5.2.

Table 5.2. Measures of Implementation Complexity for the Codes of Table 5.1 Which Can be Completely Orthogonalized.

n	k	N_{SD}	M_I	N_C	N_{SE}
21	11	42	5	11	10
73	45	146	9	45	28
85	24	170	21	24	24
273	191	546	17	191	82
585	184	1170	73	184	184
1057	813	2114	33	813	244
4161	3431	8322	65	3431	730

The codes in Table 5.1 with $(z-h) > 1$ can be $(z-h)$-step orthogonalized. In this case the majority gates form a tree with $(z-h)$ levels. The form of this decoder is shown in Fig. 5.3. In the majority-gate-tree each of the majority gates has as its input the outputs of $(d-1)$ majority gates from the level directly above it. Thus, the total number of majority gates needed is

$$M = \sum_{j=0}^{z-h-1} (d-1)^j \quad , \tag{5.28}$$

and each majority gate has

$$M_I = d - 1 \tag{5.29}$$

inputs. The operation of this decoder is similar to the one of Fig. 5.1 which was previously described. Two syndrome and data registers will be used. Thus, the decoder requires a total of

$$N_{SD} = 2n \tag{5.30}$$

binary storage elements. As before the number of computations required to execute the decoding procedure is

$$N_C = k \quad . \tag{5.31}$$

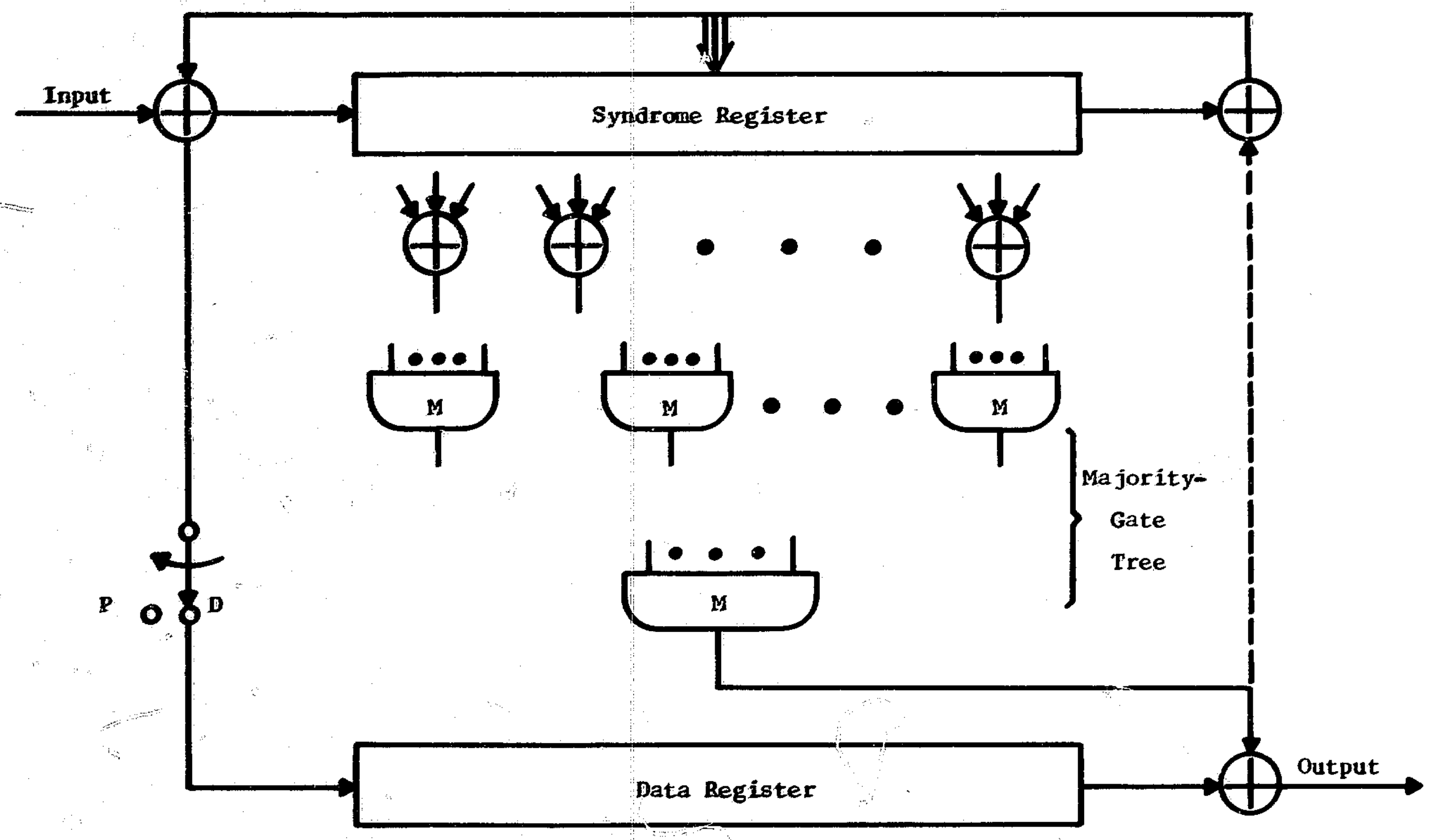

Fig. 5.3. Majority-Logic Decoder for a Code which Cannot Be Completely Orthogonalized.

These five measures of implementational complexity for the codes of Table 5.1 which can be orthogonalized in more than one step are tabulated in Table 5.3.

Table 5.3. Measures of Implementation Complexity for the Codes of Table 5.1 That Are Orthogonalized in More than One Step.

n	k	z-h	N_{SE}	M	M_I	N_{SD}	N_C
85	68	2	17	6	5	170	68
341	315	3	26	31	5	682	315
341	195	2	46	22	21	682	195
585	520	2	65	10	9	1170	520
1365	1328	4	37	156	5	2730	1328
1365	1036	3	302	463	21	2730	1036
1365	483	2	483	86	85	2730	483

In making relative comparisons between majority-logic decoding and decoding methods that have been introduced previously, the majority-logic decoder will appear to be more complex than it actually is. Previous algorithms have been quite complicated, and hence would require extensive control circuitry. Implementing the majority-logic decoding algorithm requires only shifting of shift registers which obviously will result in very little control circuitry.

This complexity of an APP decoder is much greater than that of a majority decoder. The data and syndrome registers and the logic required to compute the orthogonal parity checks $\{A_i\}$ are the same for both decoders. However, the APP decoder must have an integrate-and-dump filter that has a finely quantized output and a table for relating the quantized output to the $\{d_{ij}\}$. Also, the set $\{d_{ij}\}$ must be stored in a register, and logic for computing the $\{A_i\}$ is required. Other elements required are a threshold element with a variable threshold, an analog adder, J analog multipliers, and J nonlinear elements which have an output $y = 2 \ln \left[\coth (x/2)\right]$ in response to an input x. The complexity of this scheme will not be discussed further because its performance cannot be evaluated except by simulation.

The performance of the codes of Table 5.1 over the Gaussian channel when decoded by majority-logic decoding can be calculated from (2.16), (2.17), (3.50), and (3.51). Only five of the codes of Table 5.1 have medium rates which make them efficient for use over a binary symmetric channel created from a white-Gaussian-noise channel. The word error probability as a function of E_b/N_o is shown in Fig. 5.4 for these codes. The small number of useful codes is the main disadvantage of this class of easily implemented codes. It has been observed in simulations that a 1 to 2 dB advantage can be gained by using an APP decoding algorithm rather than a majority decoding algorithm.[40] Of course, this better performance is paid for with an increase in decoder complexity.

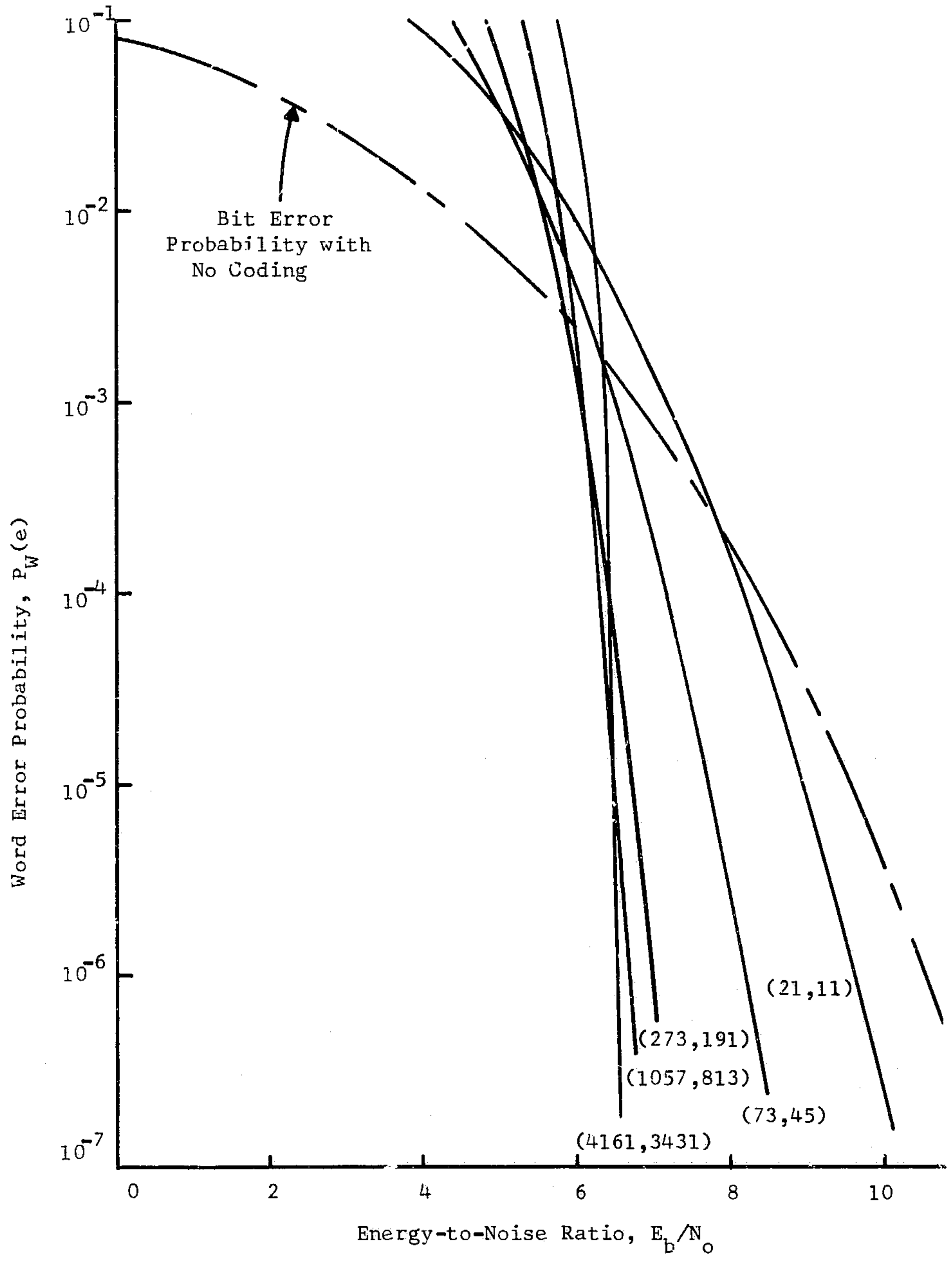

Fig. 5.4. Performance of Non-Primitive Reed-Muller Codes.

CHAPTER VI

CONCATENATED CODES

Recently, Forney introduced the concept of concatenation, i.e., coding is accomplished in two or more stages.[41] The use of two codes with moderate block lengths results in a code with a very long over-all block length for which the probability of error on memoryless channels decreases exponentially with block length. By using two stages of coding, the decoding operation can be performed by two decoders suitable for much shorter codes. Although a certain loss in efficiency occurs with concatenation, coding schemes with very long over-all block lengths can be decoded with a decoder of reasonable complexity.

A. The Process of Concatenation

If a block code of length N and rate R (nats) is used on a discrete, memoryless channel with J inputs and J outputs, the encoder will choose one of e^{NR} messages for each block of data. Since the channel input alphabet is J, the total number of possible input sequences of length N is J^N. The maximum rate is ln J, and the rate of a code is therefore bounded by

$$R \leq R_{max} = \ln J \ . \tag{6.1}$$

If the dimensionless rate r is defined as

$$r = \frac{R}{R_{max}} \ , \tag{6.2}$$

the number of code words in the code can be written as

$$e^{NR} = e^{NrR_{max}} = J^{rN} \ . \tag{6.3}$$

For such a scheme the decoder attempts to choose the correct transmitted message and deliver it to the user. Even though coding has been used to eliminate errors, some errors will inevitably occur. The encoder-channel-decoder combination thus could be viewed as a discrete memoryless channel with e^{NR} inputs and e^{NR} outputs. It is then possible to design a block code of length n and dimensionless rate r for this discrete memoryless channel which has e^{NR} inputs.

The process of concatenation is illustrated in Fig. 6.1. In terms of the original channel, two levels of coding are used which lead to an over-all block length of $N_1 = nN$ with e^{nNrR} code words. The over-all rate is then $R_1 = rR$ (nats). The two codes are called the inner code and the outer code. The combination of inner coder-channel-inner decoder can be renamed a discrete memoryless superchannel. Similarly, the outer coder-inner coder combination can be called a supercoder, and the inner decoder-outer decoder combination can be called a superdecoder. By using concatenation, the decoding problem is much simplified since the two decoders are designed to decode codes of much shorter block length than $N_1 = nN$.

B. Coding Theorem for Concatenation

In this section the coding theorem for concatenation proved by Forney will be discussed. From this theorem it will be clear that a loss in efficiency results from concatenation.

In proving the coding theorem, Forney regarded the superchannel as a discrete memoryless channel to which the coding theorem for discrete memoryless channels could be applied to determine the error exponent. There is a problem, though, because the superchannel is not completely

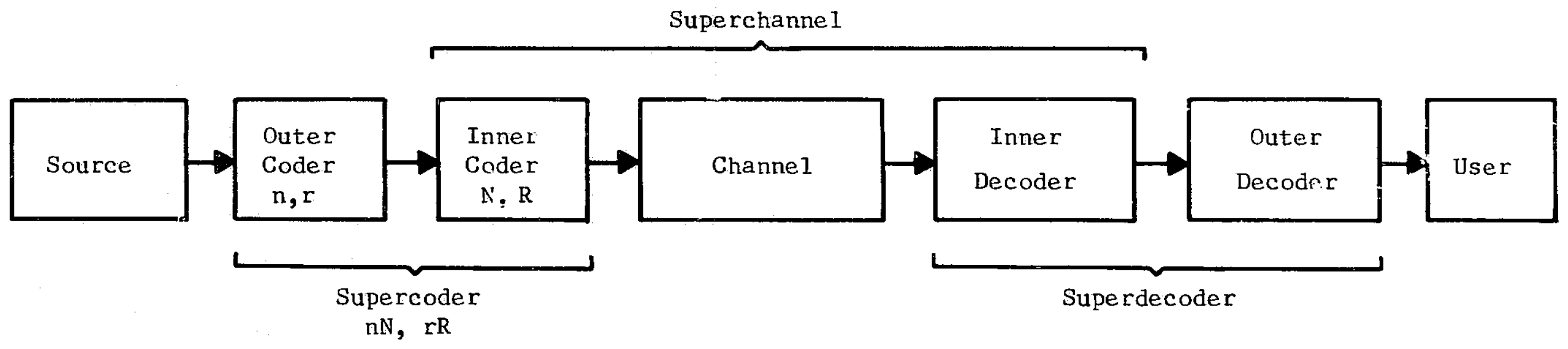

Fig. 6.1. The Process of Concatenation.

specified, since the transition probabilities are not available. He eluded this problem by determining the error exponents for the worst and best superchannels which then lead to the positive and negative statements of the coding theorem.

The theorem is proved by using the true error exponent, E(R), of the inner code, which is defined as follows. If for any N and R the best inner code is defined to be that code for which the average probability of rror p is least, then for R fixed the true error exponent is defined as

$$E(R) = \lim_{N \to \infty} \sup \frac{-\ln p}{N} , \tag{6.4}$$

where for each N the p is that for the best code. The resulting coding theorem given below expresses the error exponents of the over-all concatenation scheme in terms of the true error exponent of the inner code.

Coding Theorem. There exists a concatenation scheme with over-all length $N_1 = nN$ and over-all rate $R_1 = rR$ such that the probability of error is bounded by

$$P_W(e) \leq \exp \left[-N_1 E_{CL}(R_1) \right] . \tag{6.5}$$

The exponent $E_{CL}(R_1)$ is a lower bound to the true error exponent for the over-all concatenation scheme, and it can be calculated from the true error exponent of the inner code, E(R), by

$$E_{CL}(R_1) = \max_{rR = R_1} (1-r) \left[\frac{E(R) + \min \{E(R), R\}}{2} \right] . \tag{6.6}$$

An upper bound to the true error exponent for the concatenation scheme can be calculated by

$$E_C(R_1) = \max_{rR = R_1} (1-r)E(R) \; . \qquad (6.7)$$

In determining the error exponents of (6.6) and (6.7) a maximization is indicated. For a given over-all rate R_1, the maximization is accomplished by varying the inner code rate R and the outer code rate r subject to the constraint $rR = R_1$. An example will serve to illustrate the procedure. First, $E_C(R_1)$ will be constructed. For a fixed inner code rate R, $E_C(R_1)$ is a linear function of R_1. Then

$$E_C(0) = E(R), \; r = R_1 = 0$$

and

$$E_C(R) = 0, \; r = 1 \text{ or } R_1 = R \; .$$

If the straight lines of $E_C(R_1)$ are plotted for all values of R, the exponent is the convex curve which is the upper envelope of the family of straight lines. In Fig. 6.2., E(R) is plotted for the binary symmetric channel with p = 0.01 along with the construction of $E_C(R_1)$. The exponent $E_{CL}(R_1)$ is also a linear function which equals $\left[E(R) + \min\left[E(R), R\right]\right]/2$ at $R_1 = 0$ and zero at $R_1 = R$. Obviously, $E_C(R_1) = E_{CL}(R_1)$ for $E(R) \leq R$. The exponents E(R), $E_C(R_1)$, and $E_{CL}(R_1)$ for the BSC with p = 0.01 are illustrated in Fig. 6.3.

For the Gaussian channel the true error exponent of the inner code is given by (4.36) using normalized units. In this case the maximizations indicated by (6.6) and (6.7) can be performed analytically by using a Lagrange multiplier resulting in

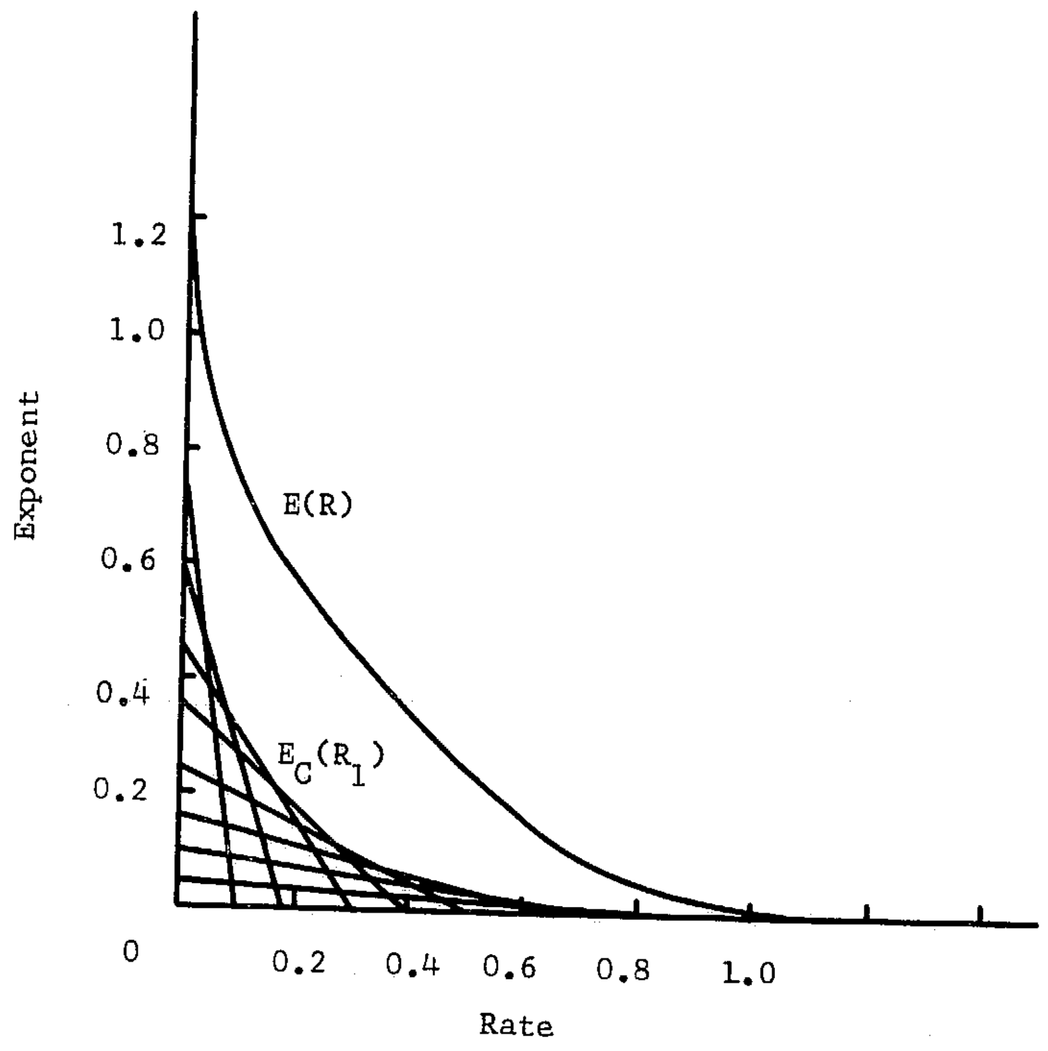

Fig. 6.2. Graphical Construction of $E_C(R_1)$ from E(R).

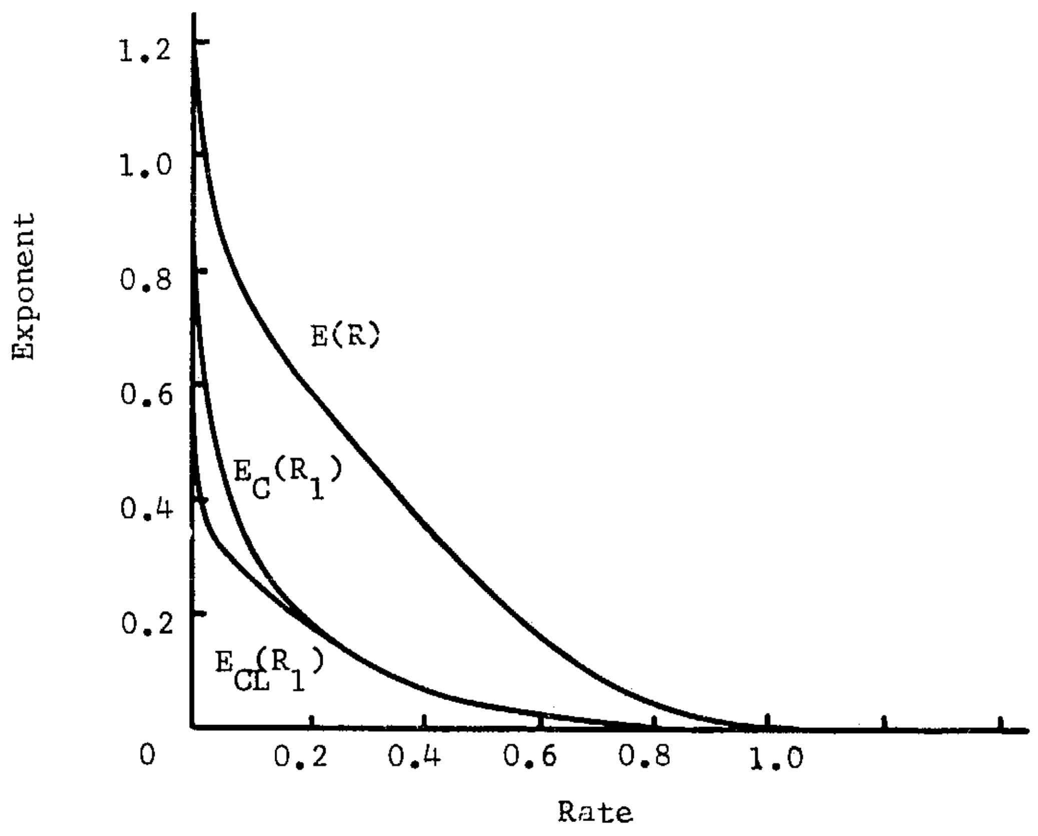

Fig. 6.3. E(R), $E_C(R_1)$, and $E_{CL}(R_1)$ for the Binary Symmetric Channel with $\bar{p} = 0.01$.

$$e_C(r_1) = \begin{cases} (\sqrt{1/2} - \sqrt{r_1})^2, & 0 \le r_1 \le 1/8 , \\ (1 - r_1^{1/3})^3, & 1/8 \le r_1 \le 1 , \end{cases} \tag{6.8}$$

and

$$e_{CL}(r_1) = \begin{cases} (1/4 - r_1), & 0 \le r_1 \le 1/8 , \\ (1 - r_1^{1/3})^3, & 1/8 \le r_1 \le 1 . \end{cases} \tag{6.9}$$

The exponents of (4.36), (6.8), and (6.9) are plotted in Fig. 6.4.

It is obvious from Figs. 6.3 and 6.4 that the error exponent resulting from concatenation is smaller than that of the original channel. This decrease in the exponents is actually a loss in efficiency where the efficiency is defined as

$$\eta(R) = \frac{E_C(R)}{E(R)} . \tag{6.10}$$

The efficiency then is a measure of the increase in over-all length needed with concatenated codes over that of unconcatenated codes. For an efficiency of 0.1 the concatenated code must be ten times longer than the unconcatenated code to achieve the same probability of error. Forney shows that as $R \to C$, $\eta(R) \to 0$. However, he also shows that if $R_1 = C(1-\varepsilon)$, the dropoff in efficiency as $R \to C$ is only linearly with ε so the efficiency slowly goes to zero. The reduced efficiency of using concatenated codes requires the use of a longer over-all block length, but even so the decoding problem is in general much easier since two decoders suitable for decoding much shorter codes can be used.

The coding theorem which has been given is merely an existence theorem. It states that a coding scheme exists which will attain the performance of (6.5), but it does not give such a coding scheme. Forney

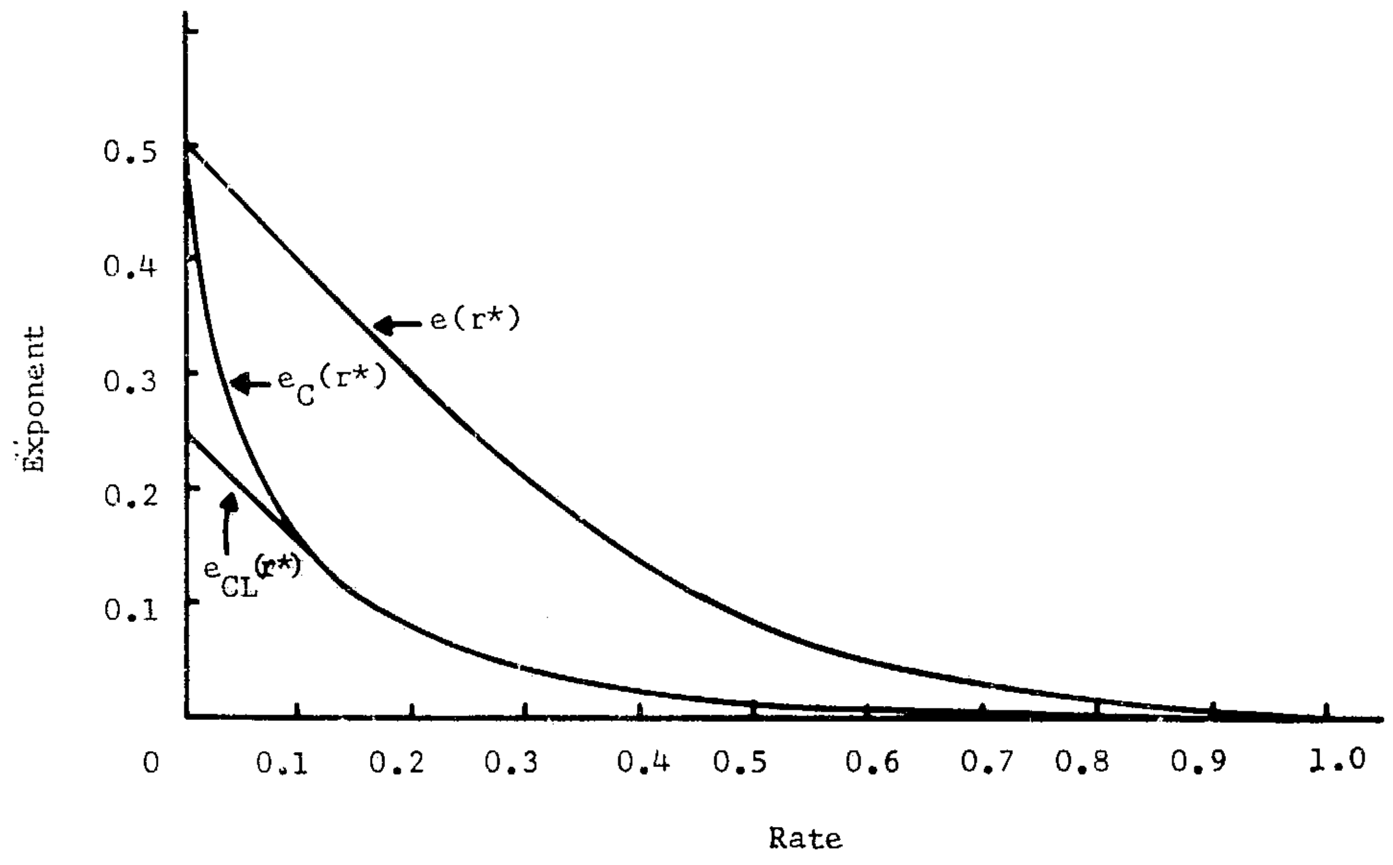

Fig. 6.4. The Error Exponents $e(r^*)$, $e_C(r_1)$, and $e_{CL}(r_1)$ for the White-Gaussian-Noise Channel.

showed that the choices of Reed-Solomon codes as outer codes is a good one since they are maximum codes. In particular, he showed that in using RS codes as outer codes with errors-only decoding (the RS decoder performs error correction only) the error exponent $E_C(R_1)/2$ is attained.[42]

C. Decoding Reed-Solomon Codes

The Reed-Solomon codes were defined in Definition 3.2. They are closely related to the BCH codes. In this section two methods will be discussed for performing errors-only decoding of RS codes. The first method is a generalization of Peterson's method for decoding BCH codes due to Gorenstein and Zierler.[43] The second method is due to Berlekamp and uses Algorithm 3.1.[44]

Gorenstein and Zierler's algorithm. In implementing the Gorenstein and Zierler algorithm the parity checks are first calculated by the iterative method of (3.26). Since a Bartee and Schneider type arithmetic unit is used, calculation of the 2t parity checks requires a total of

$$\begin{aligned} N_{C1} &= 2(n-1)t + t \\ &= 2tn - t \end{aligned} \qquad (6.11)$$

computations. In the nonbinary case (3.8) becomes

$$S_j = \sum_{k=1}^{e} Y_k Z_k^j, \; j = 1,2,\cdots,2t \;, \qquad (6.12)$$

where Y_k is the error value at position Z_k. The decoder must find a solution with as small a value of e as possible. The elementary symmetric functions as defined by (3.30) are related to the parity checks by the system of equations

$$\begin{bmatrix} S_1 & S_2 & S_3 \cdots & S_t \\ S_2 & S_3 & \cdots & \cdots \\ S_3 & \cdots & \cdots & \cdots \\ \vdots & \cdots & \cdots & \cdots \\ S_t & \cdots & \cdots & S_{2t-1} \end{bmatrix} \begin{bmatrix} \sigma_t \\ \sigma_{t-1} \\ \cdot \\ \cdot \\ \sigma_1 \end{bmatrix} = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \cdot \\ \cdot \\ S_{2t} \end{bmatrix} \quad . \tag{6.13}$$

From Appendix A it is seen that the solution of this system requires a maximum of

$$N_{C2} = \frac{4t^3 + 15t^2 - 37t + 24mt - 12m + 12}{6} \tag{6.14}$$

computations. After the σ_i's are found the error locators may be determined by using a Chien search which requires a total of

$$N_{C3} = n \tag{6.15}$$

computations.

Once the error locators are found, they are substituted into the system (6.12) which is now written as

$$\begin{bmatrix} Z_1 & Z_2 \cdots & Z_e \\ Z_1^2 & Z_2^2 \cdots & Z_e^2 \\ \cdots & \cdots & \cdots \\ Z_1^e & Z_2^e \cdots & Z_e^e \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ Y_e \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ S_e \end{bmatrix} \quad . \tag{6.16}$$

To get the powers of the error locators needed in the coefficient matrix of (6.16) requires at most

$$N_{C4} = t(t-1) \tag{6.17}$$

computations. Solution of the system (6.16) requires at most N_{C2} computations. Once the error values are found, they may be corrected as the word is shifted out of the buffer. A total of $N_{C3} = n$ computations are required for this step. The total number of computations required in decoding an RS code with the Gorenstein and Zierler algorithm is

$$N_C = N_{C1} + 2N_{C2} + 2N_{C3} + N_{C4} \tag{6.18}$$

or

$$N_C = \frac{6n + 6nt + 4t^3 + 18t^2 - 43t + 24mt - 12m + 12}{3} . \tag{6.19}$$

Again a buffer length of 2n is assumed. Other storage registers needed are 2t registers for the storage of the parity checks, t(t+1) registers for the solution of (6.16), (t+1) registers for the Chien searcher, and t registers for the storage of the error values. Each of these registers has m stages. An n-stage register is also needed for storing the received word while calculat'ng the parity checks. Finally 2 m-stage registers are needed for the arithmetic unit. The total number of binary storage elements needed is

$$N_{SD} = 3nm + 2t^2m + 6tm + 2m . \tag{6.20}$$

Berlekamp's algorithm. In decoding an RS code using Berlekamp's algorithm a procedure similar to that for the binary BCH code is followed. The remainders and parity checks are calculated, the Key Equation is solved using Algorithm 3.1, and a Chien search is performed. At this point the error locations but not the error values are known. The final step is to solve for the error values and perform the necessary error correction.

From (3.16), $\omega(X)$ is defined as

$$\omega(X) = \sigma(X) + \sum_{i=1}^{e} Z_i X \prod_{j \neq i} (1 - Z_j X) . \tag{6.21}$$

If the error locations are known, then $\omega(X)$ can be evaluated at $X = Z_i^{-1}$ obtaining

$$\omega(Z_i^{-1}) = Y_i \prod_{j \neq i} (1 - Z_j Z_i^{-1}) \quad . \tag{6.22}$$

Thus, the error may be evaluated as

$$Y_i = \frac{\omega(Z_i^{-1})}{\prod_{j \neq i} (1 - Z_j Z_i^{-1})}$$

$$= \frac{Z_i^{(\deg \sigma)} \omega(Z_i^{-1})}{Z_i \prod_{j \neq i} (Z_i - Z_j)} \quad . \tag{6.23}$$

The degree of $\omega(X)$ is equal to the degree of $\sigma(X)$. If the reciprocal polynomial $\tilde{\omega}(X)$ is defined by

$$\tilde{\omega} = X^{(\deg \omega)} \omega(X^{-1}) \quad , \tag{6.24}$$

then (6.24) becomes

$$Y_i = \frac{\tilde{\omega}(Z_i)}{Z_i \prod_{j \neq i} (Z_i - Z_j)} \quad . \tag{6.25}$$

An implementation of this error-correcting procedure in conjunction with the Chien searcher is shown in Fig. 6.5.

The implementation of Algorithm 3.1 for the solution of the Key Equation is more complicated than the implementation of Algorithm 3.2, which was explained in Chapter III. For the RS codes $GF(2^m)$ is both the symbol field and the locator field. The remainders $r^{(1)}(X)$, $r^{(2)}(X)$, ..., $r^{(2t-1)}(X)$, are computed with t m-stage shift registers. However, these are $GF(2^m)$ shift registers, so each stage consists of m binary shift-register stages in parallel. The central processor will be more complicated since eight columns of registers will be needed instead of five as before. The parity checks are computed in the same manner as described in Chapter III. The contents of the slave registers

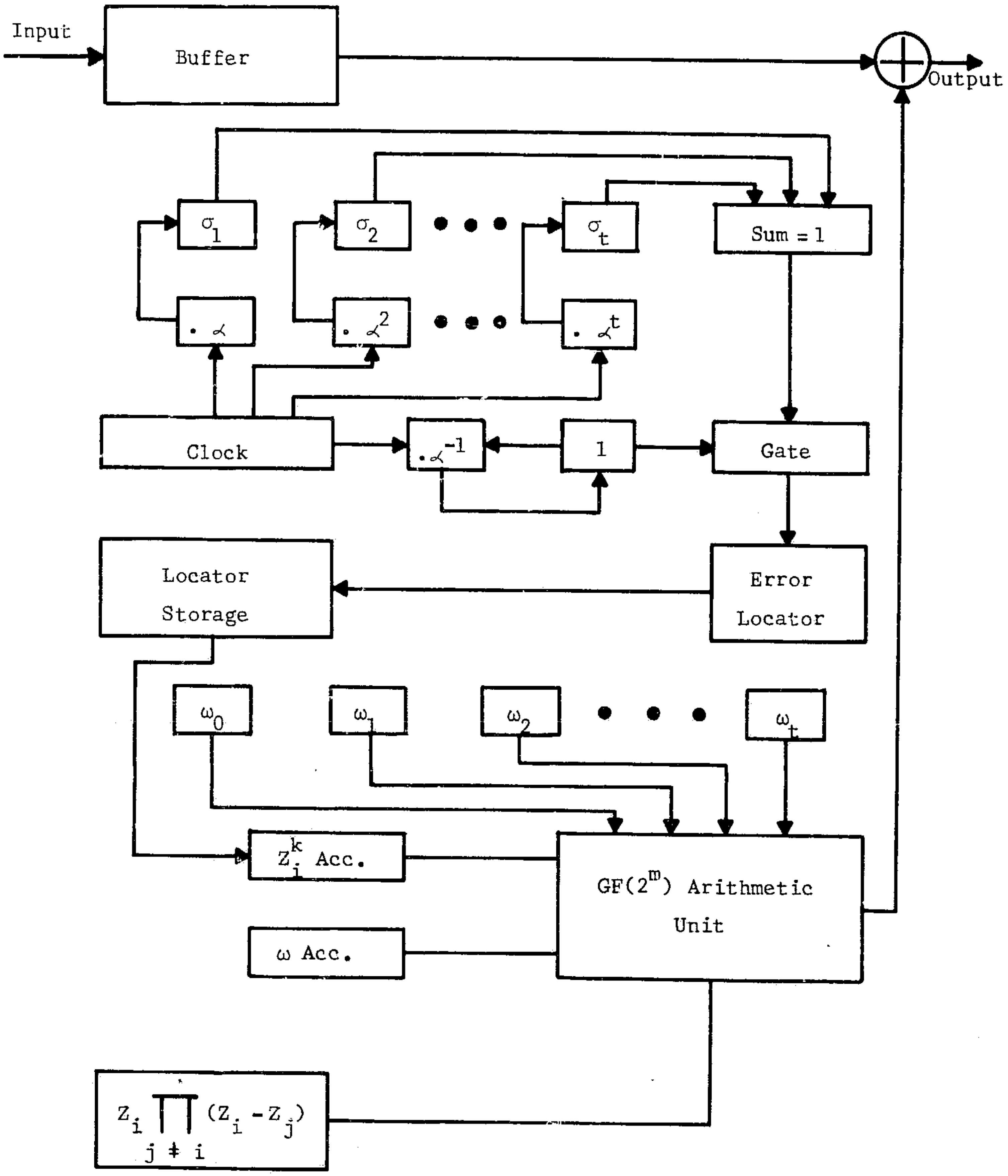

Fig. 6.5. Chien Searcher and Error Corrector.

after the parity checks have been computed are illustrated in Fig. 6.6. The labeling of the columns of register is also shown. The over-all design of the central processor is the same as illustrated in Fig. 3.4 for the binary BCH decoder.

The following procedure can be used for solving the Key Equation by Berlekamp's algorithm. The number k is one less than the number of times at Step 1.

1. The S columns are shifted upward once. If $k = 0$, set $\sigma^{(0)} = 1$, $\tau^{(0)} = 1$, $\omega^{(0)} = 1$, $\gamma^{(0)} = 0$, $D(0) = 0$, $B(0) = 0$, and continue. If $k \neq 0$, continue.
2. Each slave computes the product of its second and third columns and stores the results in the seventh column.
3. The sum of all registers in the seventh column is placed in the master accumulator. This sum is $\Delta_1^{(k)}$.
4. Both the fourth and sixth columns are shifted upward once. This corresponds to multiplication by X.
5. Place $\Delta_1^{(k)}$ in the master multiplier register.
6. Multiply the contents of the fourth column by the contents of the master multiplier register, add these results to the third column, and store the results in the seventh column. The seventh column now contains $\sigma^{(k+1)}$.
7. Multiply the contents of the sixth column by the contents of the master multiplier register, add these results to the fifth column, and store the results in the eighth column. The eighth column now contains $\omega^{(k+1)}$.

			$\sigma_i^{(k)}$	$\tau_i^{(k)}$	$\omega_i^{(k)}$	$\gamma_i^{(k)}$		
Slave t	1	S_{t+1}						
	S_1	S_{t+2}						
	S_2	●						
	●	●						
	●	S_{2t-2}						
Slave 2	●	S_{2t-1}						
Slave 1	S_{t-1}	——						
Slave 0	S_t	——						

Fig. 6.6. Contents of the Slave Registers after Computation of the Parity Checks.

8. If $\Delta_1^{(k)} = 0$; if $D(k) > \frac{k+1}{2}$; or if $\Delta_1^{(k)} \neq 0$, $D(k) = \frac{k+1}{2}$ and $B(k) = 0$, go to Step 9. If not, go to Step 10.

9. Set $D(k+1) = D(k)$, $B(k+1) = B(k)$. Already the calculations $\tau^{(k+1)} = X\tau^{k}$ and $\gamma^{(k+1)} = X\gamma^{(k)}$ have been made by shifting these columns upward once. Go to Step 11.

10. Set $D(k+1) = k+1-D(k)$, $B(k+1) = 1-B(k)$. The contents of the third column are multiplied by the contents of the master multiplier register, and the results are placed in the fourth column. The contents of the fifth column are multiplied by the contents of the master multiplier register, and the results are placed in the sixth column.

11. The contents of the seventh and eighth columns are transferred to the third and fifth columns, respectively. If $k < 2t$, increment k by one and go back to Step 1. If $k = 2t$, read out the error locator polynomial $\sigma^{(2t)}(X)$.

Following this implementation procedure, step-by-step estimation of the number of computations required in the execution of the algorithm will not be difficult. The parity checks are calculated from the remainders in the same manner as they were in Chapter III for the binary BCH codes. It will be assumed that the result of the test of Step 8 directs the decoder to Step 10, since this is the most lengthy procedure. The error-correction procedure implemented as in Fig. 6.5 requires n shifts of a shift register in addition to the calculations needed for calculating the error values. Calculating $\tilde{\omega}(Z_i)$ will require at most (2t-1) multiplications and t additions. Computing the denominator of (6.25) requires at most (t-1) additions and (t-1) multiplications. Of course, to complete the calculation of (6.25) the denominator must be inverted and multiplied by the numerator. This requires 5m+1 computations. At most t such error values must be computed. In Table 6.1 the number of computations required in the various steps of decoding an RS code with this algorithm are listed.

Table 6.1. Number of Computations Required in Each Step of Decoding an RS Code by Berlekamp's Algorithm.

Steps	Number of Computations
parity checks	$2m(m-1)$
1	$2t$
2	$2mt$
3	$2t$
4	$2t$
5	$2t$
6	$2(m+1)t$
7	$2(m+1)t$
8	$2t$
10	$4mt$
11	$2t$
Chien search	n
error correction	$n + 3mt^2 + 2t^2 + 3mt$

The total number of computations required in implementing this algorithm is the sum of the entries of the second column of Table 6.1 which is

$$N_C = 2n + 3mt^2 + 2t^2 + 13mt + 2m^2 + 16t - 2m \quad . \tag{6.26}$$

For storage a buffer of length 2n is again assumed. Of course, each stage consists of m binary storage elements, since elements of GF(2^m) are to be stored. The remainders are calculated with feedback shift registers consisting of at most m^2 stages. At most t such registers are required. The central processor has (t+1) slaves each containing 8 m-stage registers. The master unit contains an accumulator register of m stages, a multiplier register of m stages, and two registers for computing multiplicative inverses consisting of (2m+3) stages. Thus, the central processor has approximately (8mt + 12m +3) binary storage elements. The Chien searcher requires (t+1) m-stage registers. The error corrector needs m(t+1) binary storage elements for storing the locators, m(t+1) storage elements for storing ω_0, ω_1, ω_2, $\cdots$, ω_t, three m-stage accumulators, and an arithmetic unit which can add, multiply, and compute an inverse in GF (2^m). A total of (2mt + 9m + 3) storage elements are required for the error corrector. Thus, the total number of binary storage elements required in an RS decoder using Berlekamp's algorithm is

$$N_{SD} = 2nm + m^2t + 11mt + 23m + 6 \ . \qquad (6.27)$$

D. Performance and Complexity of Implementing Concatenated Codes

Forney suggested two types of inner codes, BCH codes for discrete channels and bi-orthogonal codes for the Gaussian channel. In this chapter the only inner coding scheme to be considered is bi-orthogonal coding for the Gaussian channel.

The number of computations and the number of binary storage elements required in decoding a bi-orthogonal code are given by (4.41) and (4.42), respectively. Since the symbol field of the RS code is GF(2^m), the bi-orthogonal code must have m information symbols. The inner decoder must process all n inner code words before the RS outer decoder can begin to decode. Thus, the number of computations required in decoding a concatenated code using a bi-orthogonal inner code with correlation decoding and a Reed-Solomon outer code with the Gorenstein-Zierler algorithm is

$$N_C = n2^{m-1}(2^{m-1}+1) + \frac{6n + 6nt + 4t^3 + 18t^2 - 43t + 24mt - 12m + 12}{3} \quad . \tag{6.28}$$

The number of binary storage elements required is

$$N_{SD} = 3nm + 2t^2m + 6tm + 7m + 3 \cdot 2^{m+1} + 12 \quad . \tag{6.29}$$

The number of computations required in decoding a concatenated code using a bi-orthogonal inner code with correlation decoding and a Berlekamp decoder for the Reed-Solomon outer code is

$$N_C = n2^{m-1}(2^{m-1} + 1) + 2n + 3mt^2 + 2t^2 + 8mt + 2m^2 + 8t - 2m \quad . \tag{6.30}$$

The number of binary storage elements needed is

$$N_{SD} = 2nm + m^2t + 11mt + 27m + 3 \cdot 2^{m+1} + 18 \quad . \tag{6.31}$$

Since the inner code words each contain m information bits, the inner coder is a shift register with m-1 stages. The outer coder is implemented either by the circuit of Fig. 2.7 or the circuit of Fig. 2.8, depending upon which is simpler. Each stage of this shift-register encoder contains m binary storage elements, since the symbols of the outer code are elements of GF(2^m). The total number of binary storage elements in an encoder for a concatenated code is

$$N_{SE} = \begin{cases} m - 1 + mk \quad , & k \leq n-k \ , \\ m - 1 + m(n-k), & k > n-k \ . \end{cases} \tag{6.32}$$

The performance of a concatenated coding system can be calculated in a straightforward manner. If the word error probability of the bi-orthogonal inner code is denoted by p, this quantity is also the channel error probability for transmission through the superchannel. The values of p as a function of the energy-to-noise ratio on the Gaussian channel are given in Golomb.[45] For a concatenated code, the probability of word error as a function of E_b/N_o can be determined from (2.16), (2.17), and (3.51).

In many applications including space communication the information to be transmitted is grouped into characters. If each of these characters is transmitted by a single inner code word the error probability of interest is the probability of digit error for the outer code. An upper bound to this probability is derived in Appendix C. For an outer code with an odd minimum distance,

$$P_D(e) \leq \sum_{i=t+1}^{n-t-1} \frac{(t+i)}{n} \binom{n}{i} p^i (1-p)^{n-i} + \sum_{i=n-t}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad . \tag{6.33}$$

It is obvious that when errors-only decoding is being used, it is more efficient to use an outer code with an odd minimum distance. The probability of digit error as a function of E_b/N_o is shown in Figs. 6.7 through 6.11 for a number of inner and outer code combinations. The label on each curve is of the form (n,k,m), where n is the length of the outer code, k is the number of information bits per outer code word, and m is the number of information bits per inner code word. Each figure illustrates performance for two coding systems, both of which have the same outer code block length and the same inner code but have different outer code rates. Of the two curves in each figure, the one exhibiting the better performance at low error probabilities was chosen

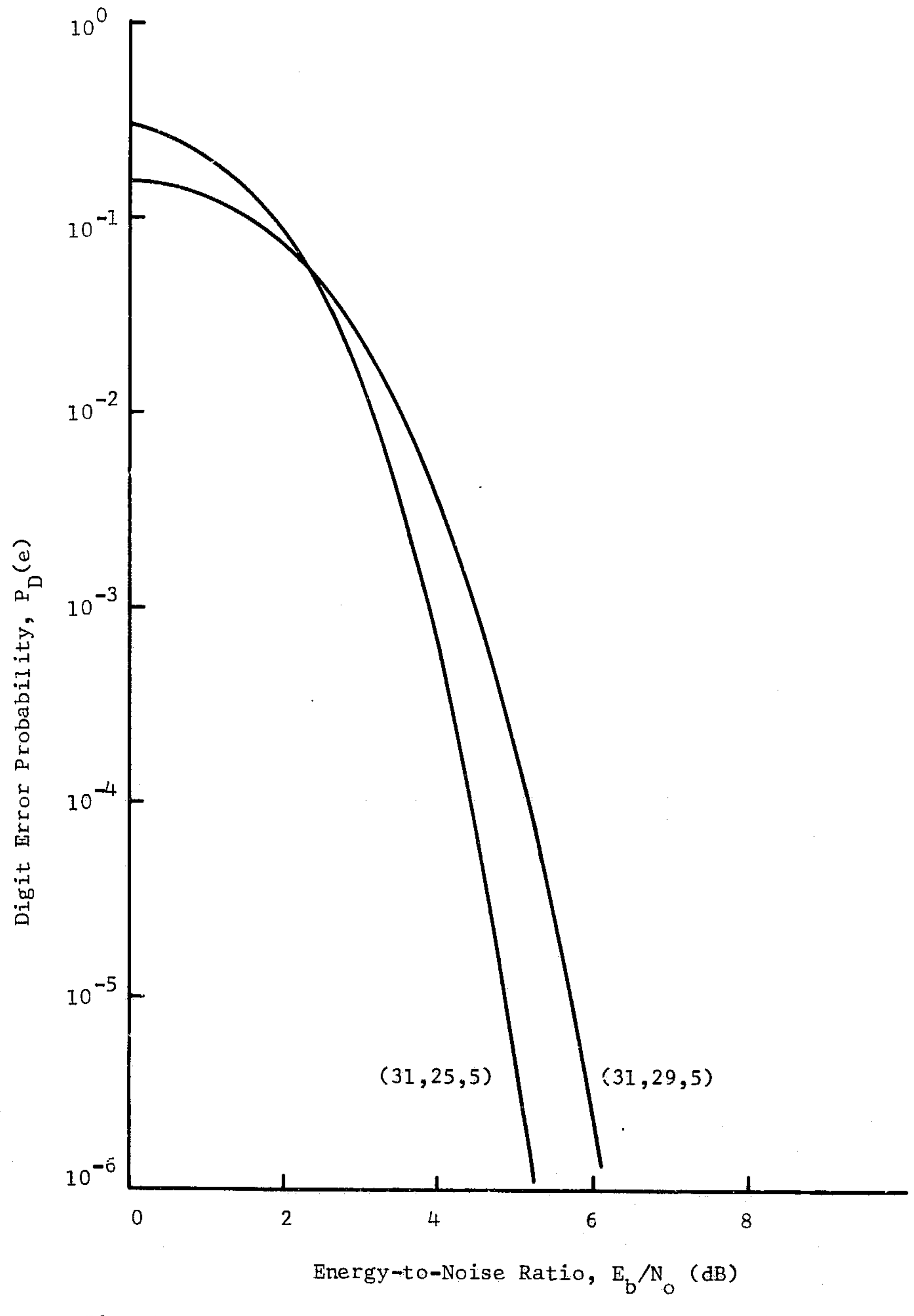

Fig. 6.7. Performance of Concatenation Schemes Utilizing a (16,5) Bi-orthogonal Inner Code with (31,29) and (31,25) Reed-Solomon Outer Codes.

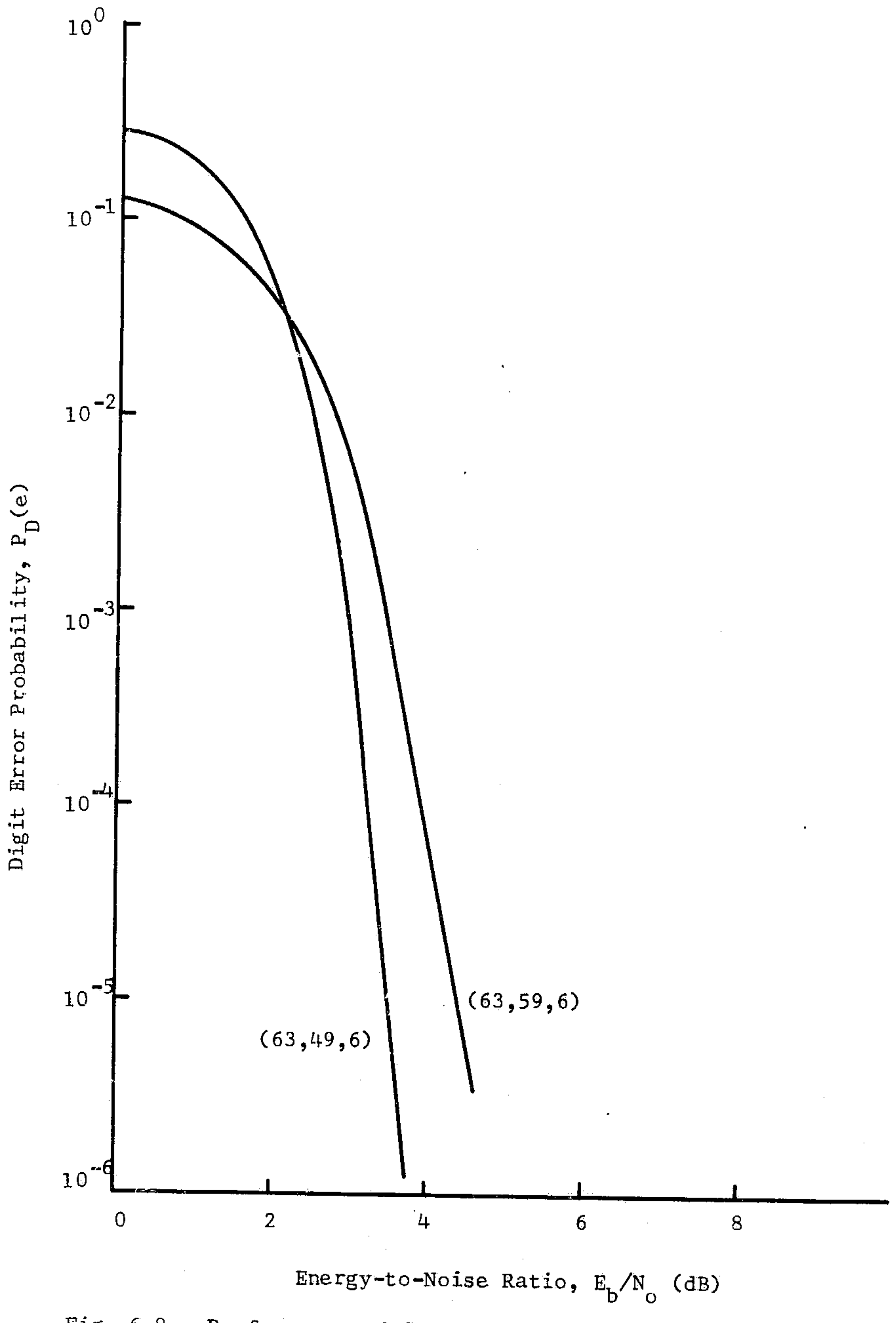

Fig. 6.8. Performance of Concatenation Schemes Utilizing a (32,6) Bi-orthogonal Inner Code with (63,59) and (63,49) Reed-Solomon Outer Codes.

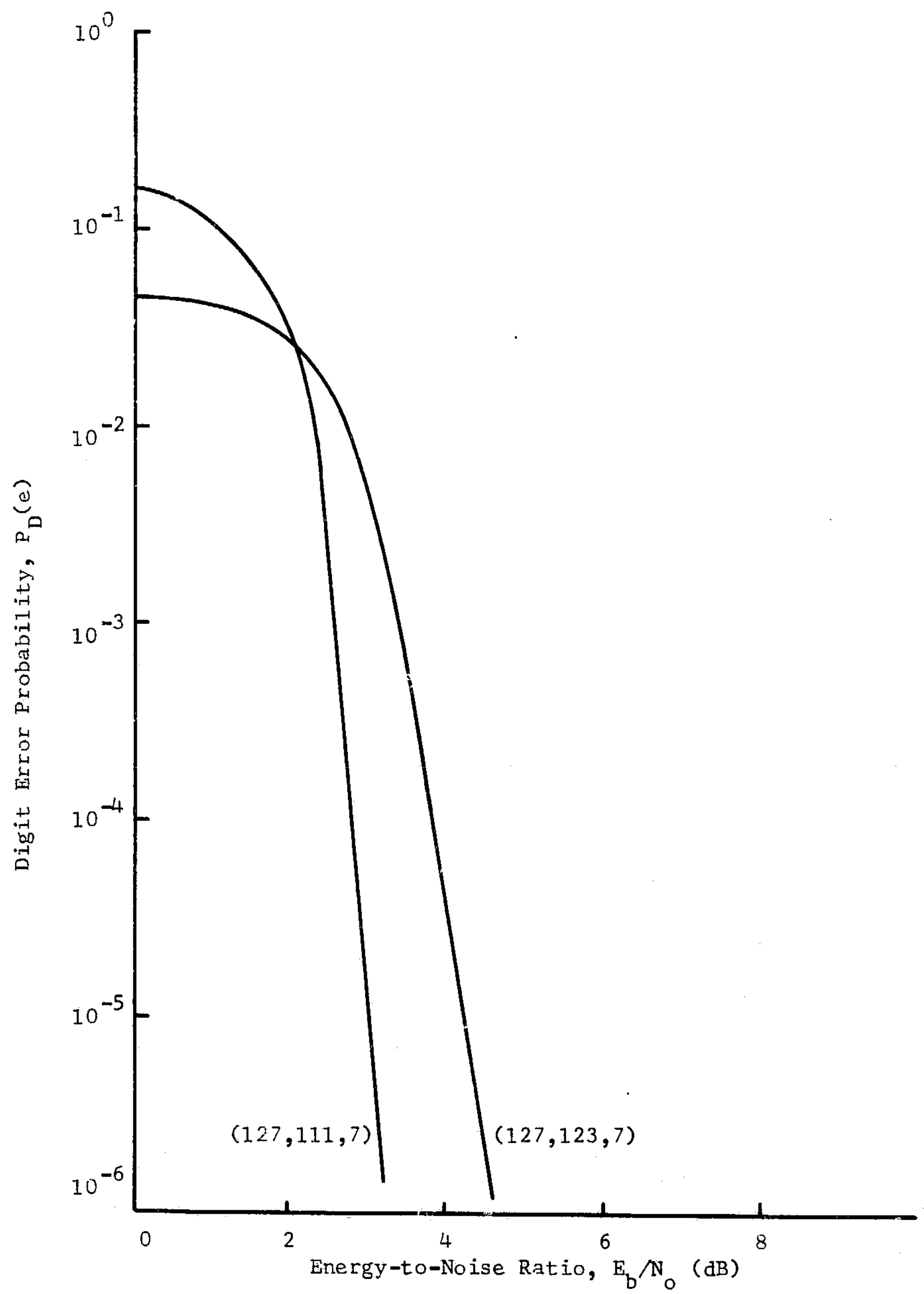

Fig. 6.9. Performance of Concatenation Schemes Utilizing a (64,7) Bi-orthogonal Inner Code with (127, 123) and (127,111) Reed-Solomon Outer Codes.

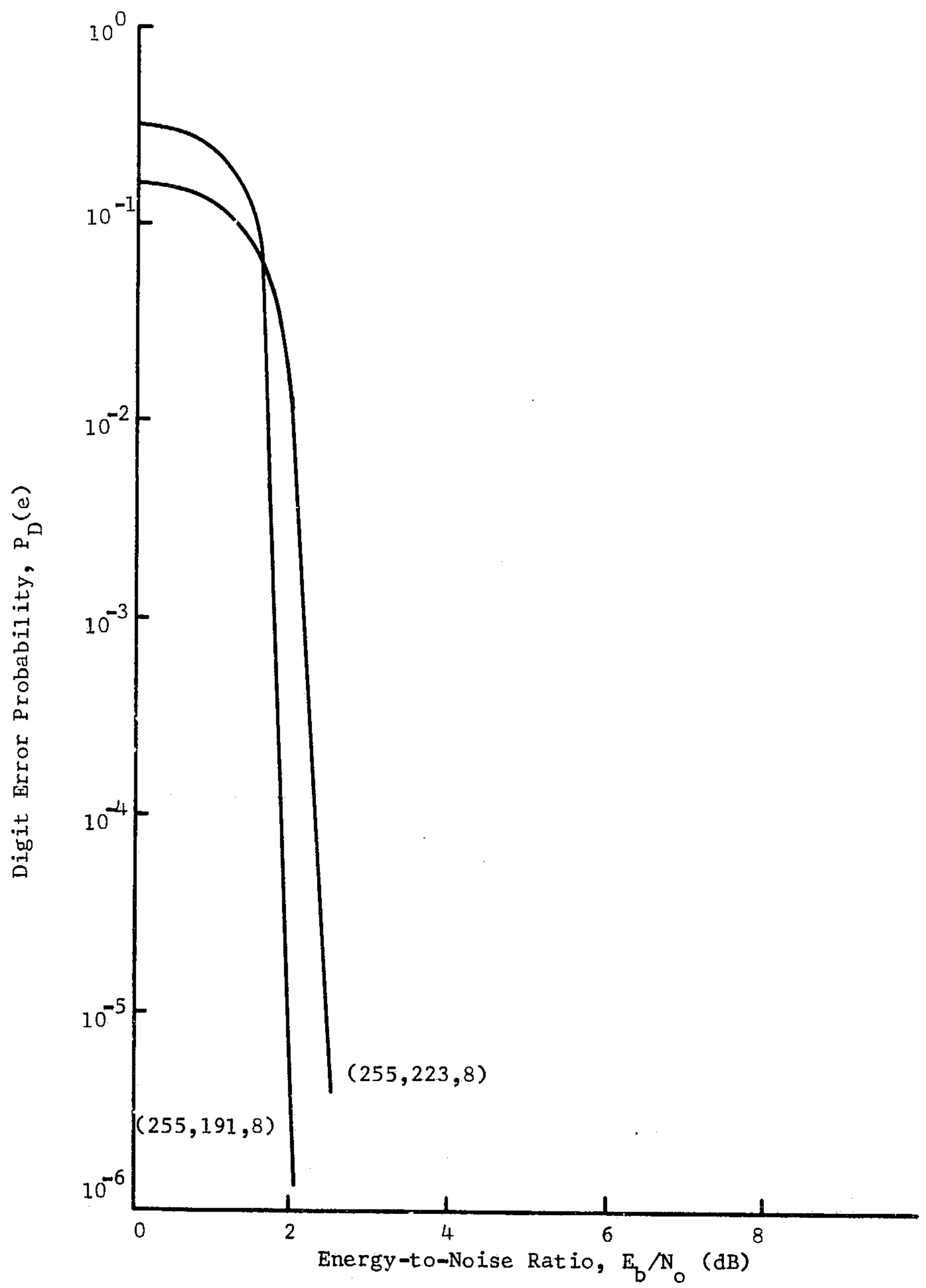

Fig. 6.10. Performance of Concatenation Schemes Utilizing a (128,8) Bi-orthogonal Inner Code with (255,223) and (255,191) Reed-Solomon Outer Codes.

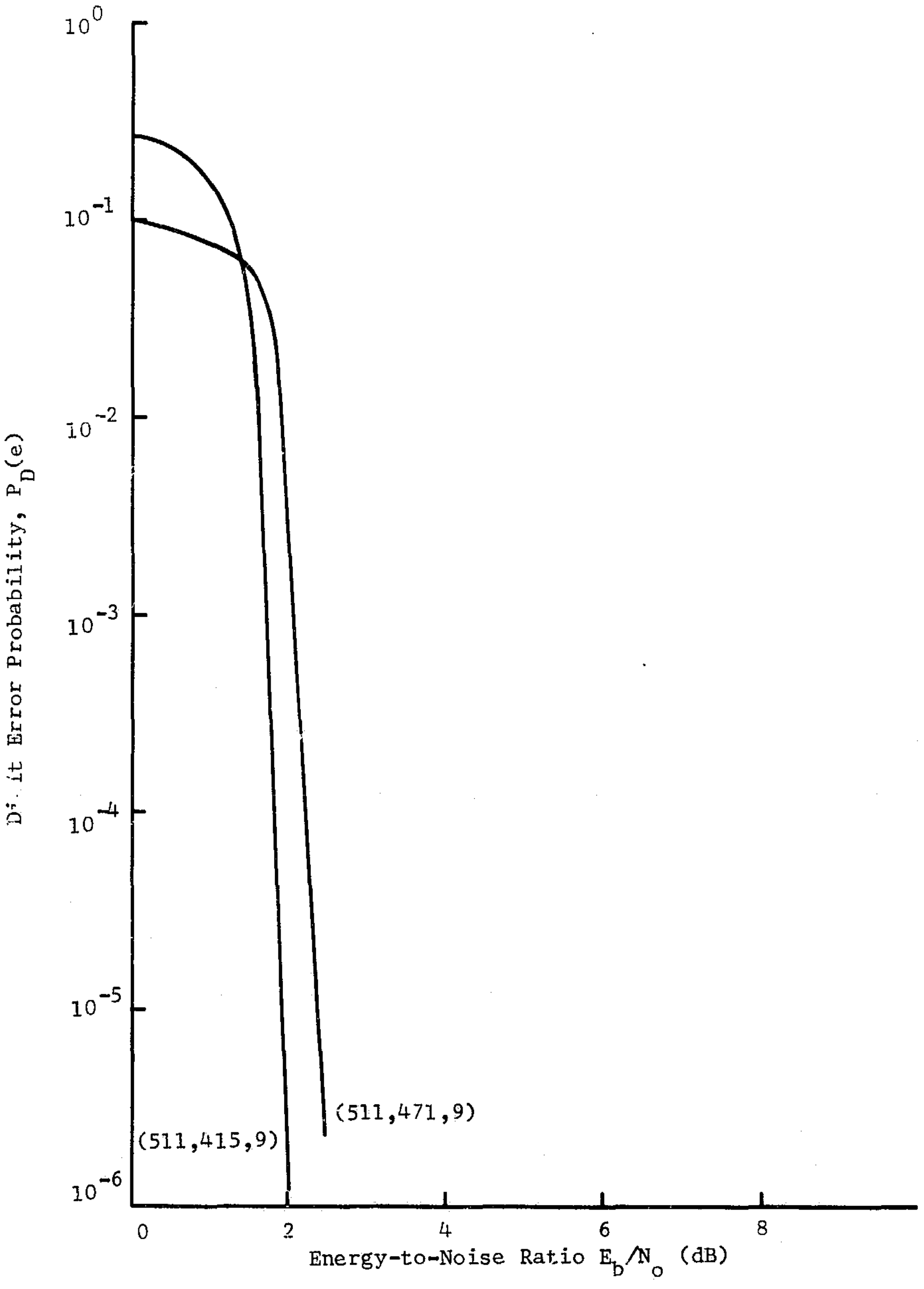

Fig. 6.11. Performance of Concatenation Schemes Utilizing a (256,9) Bi-orthogonal Inner Code with (511,471) and (511,415) Reed-Solomon Outer Codes.

from similar curves as having performance better than or equal to that of all other concatenated codes with the same inner code and the same outer code block length. The other curve in each figure is for a code with a slightly higher outer code rate.

From the standpoint of implementation complexity, outer codes with high rates are desirable. This is evident from Figs. 6.12 through 6.15. In these figures the number of computations per decoded information bit and the number of binary storage elements required for two codes are shown as a function of error-correction capability, t, for the Gorenstein-Zierler and Berlekamp outer decoders. A high rate implies a small value of error correction capability, t. For long outer codes and high values of t, the Berlekamp decoder is much more easily implemented. For shorter codes and small values of t the Peterson decoder can be easily implemented by solving the systems of equations (6.13) and (6.16) before the decoder is designed, thus eliminating the need for a Gauss-Jordan reduction.

Since the complexity of the inner decoder increases exponentially with m, the major portion of equipment complexity is due to the inner decoder for m large. Also, for m large and an outer code with a high rate, the speed factor for the inner-decoder logic must be much larger than that for the outer-decoder logic. The exponential increase of inner-decoder complexity places a practical limit on the over-all block lengths which can be implemented in concatenating a Reed-Solomon outer code with a bi-orthogonal inner code.

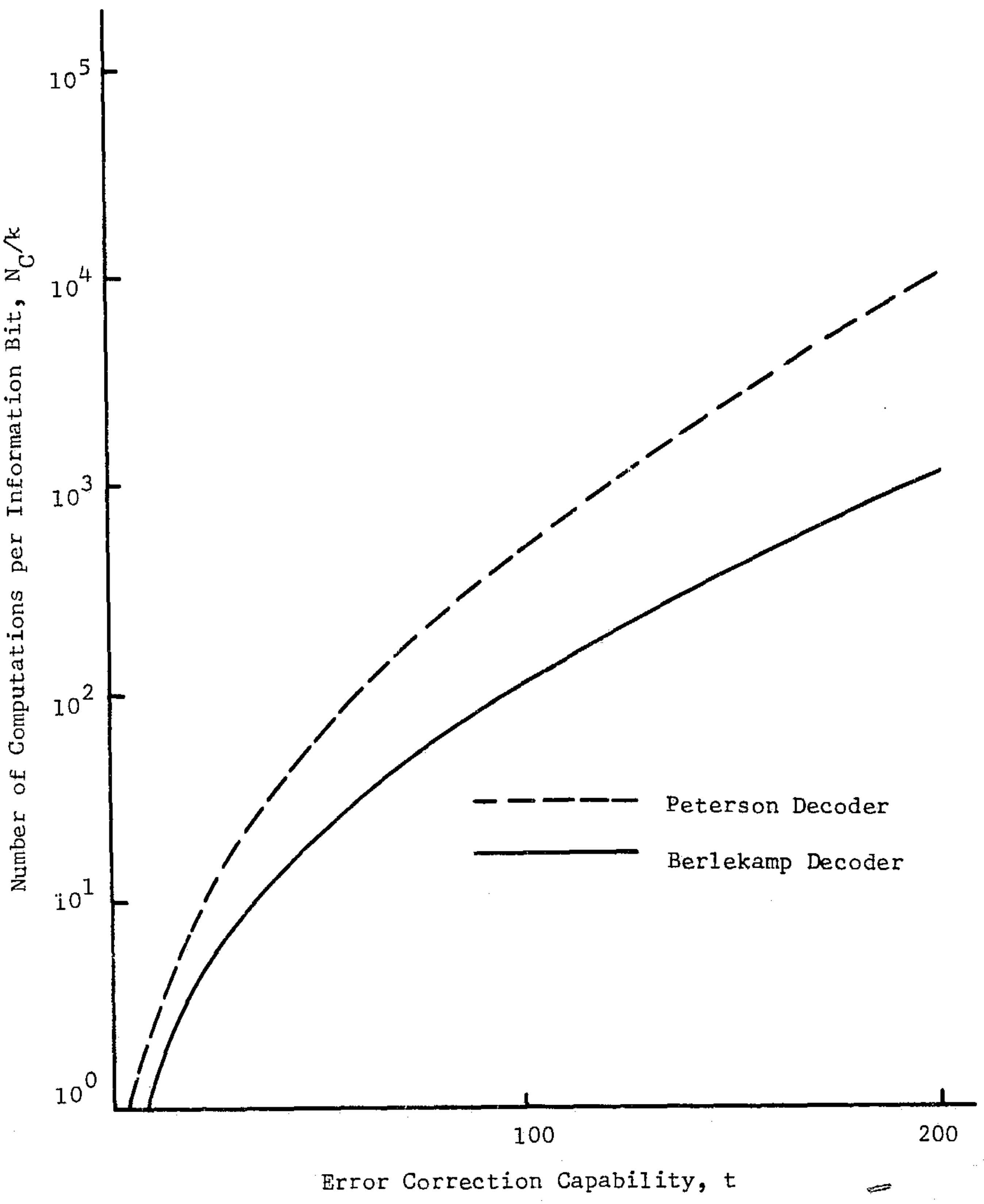

Fig. 6.12. Number of Computations per Information Bit Required in Decoding RS Codes with m = 9 and n = 511.

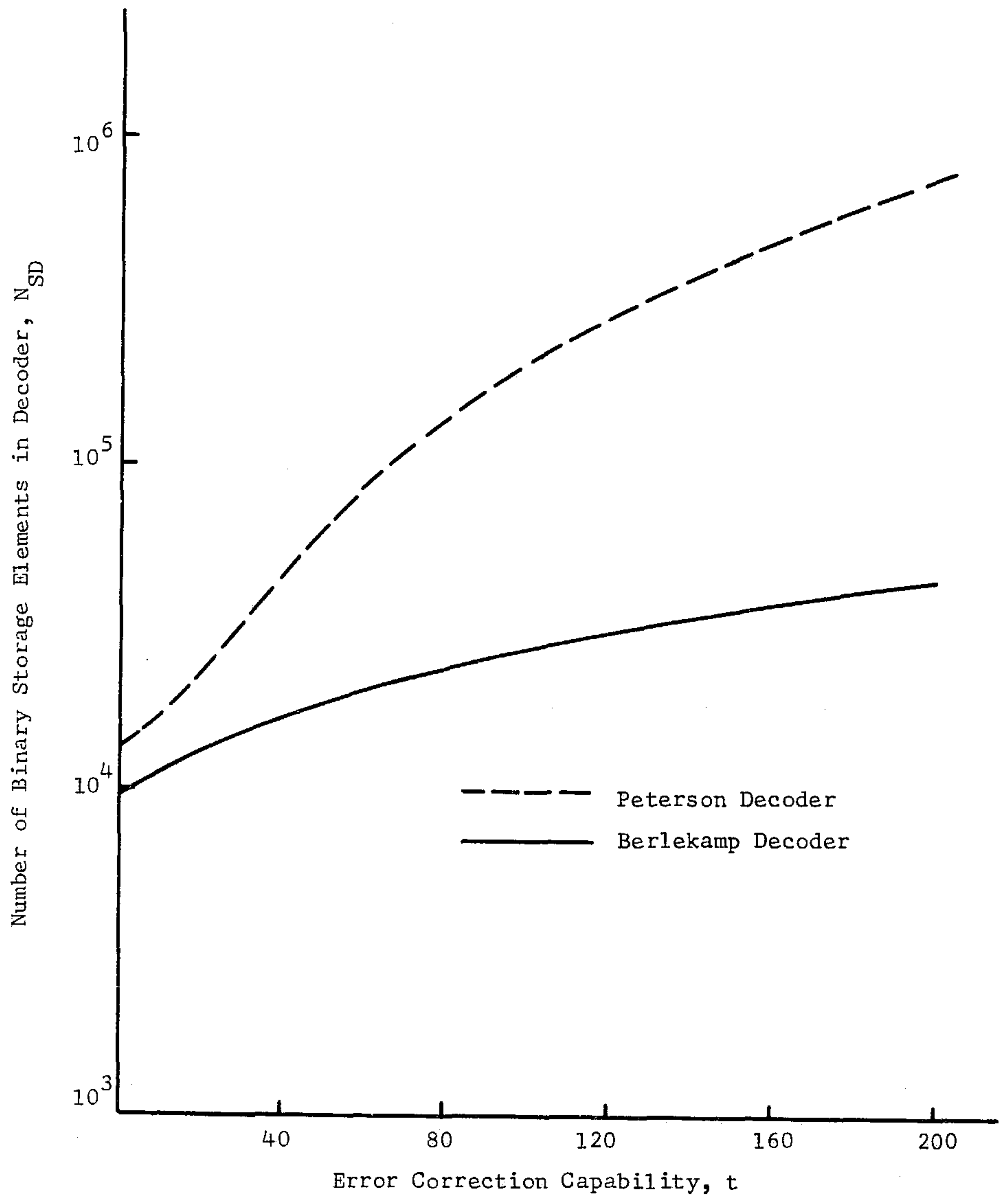

Fig. 6.13. Number of Binary Storage Elements Required in the Decoder for RS Codes with m = 9 and n = 511.

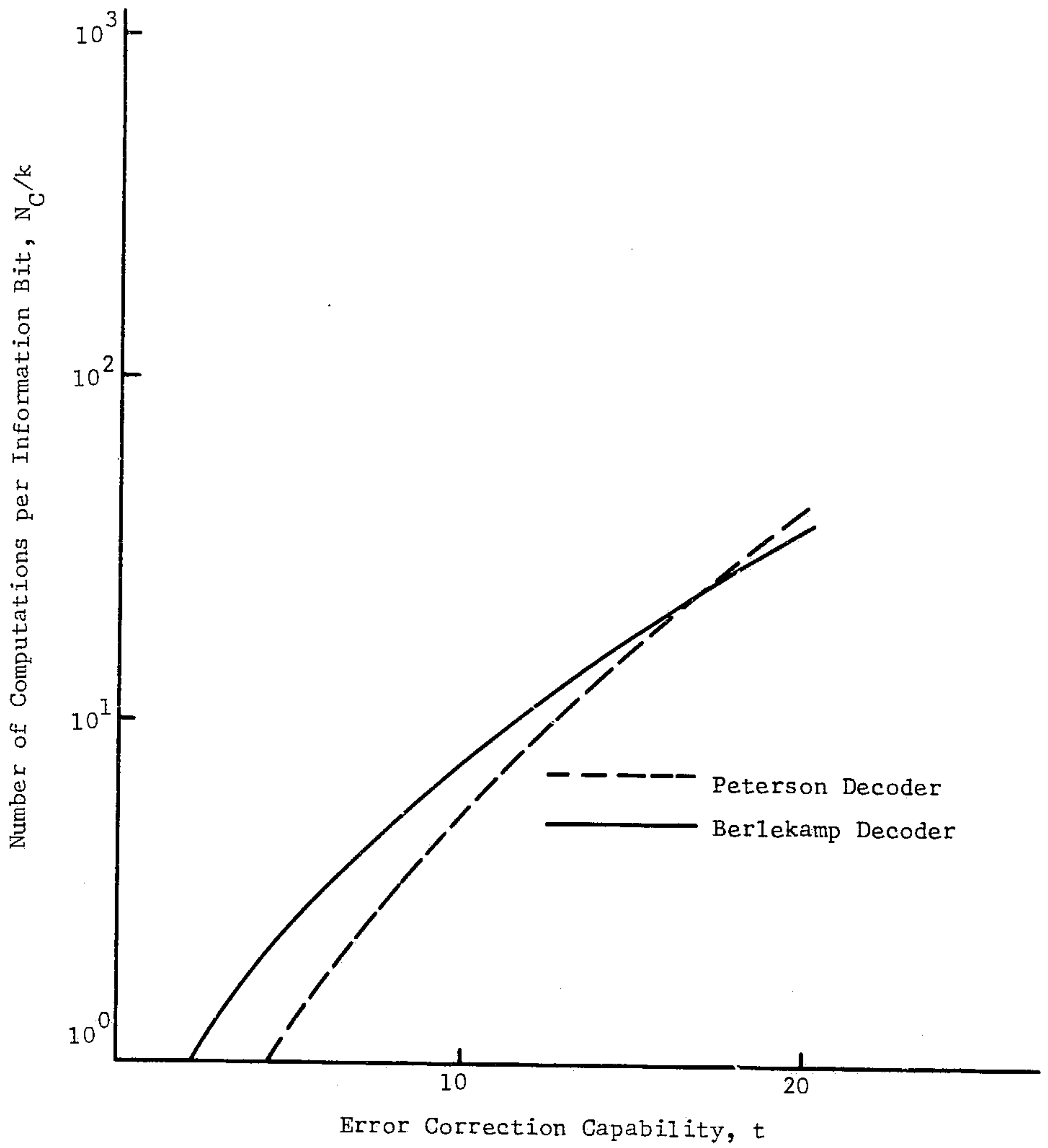

Fig. 6.14. Number of Computations per Information Bit Required in Decoding RS Codes with m = 8 and n = 85.

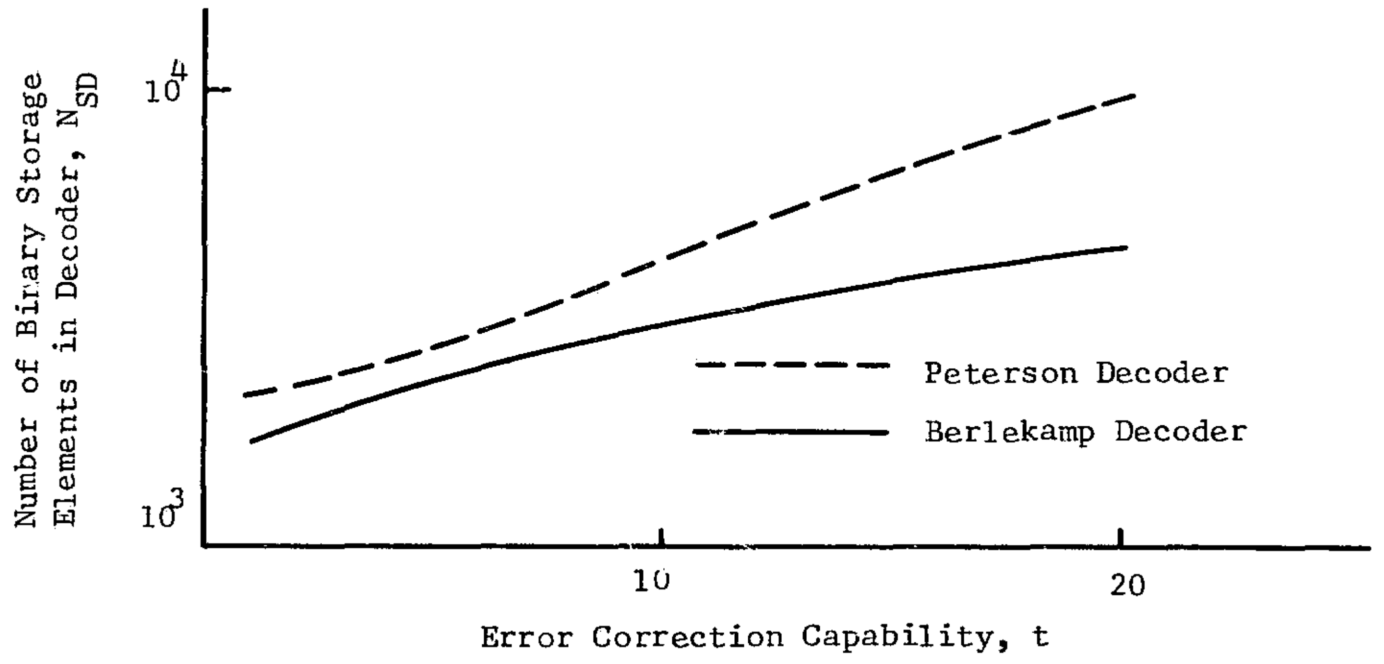

Fig. 6.15. Number of Binary Storage Elements Required in the Decoder for RS Codes with m = 8 and n = 85.

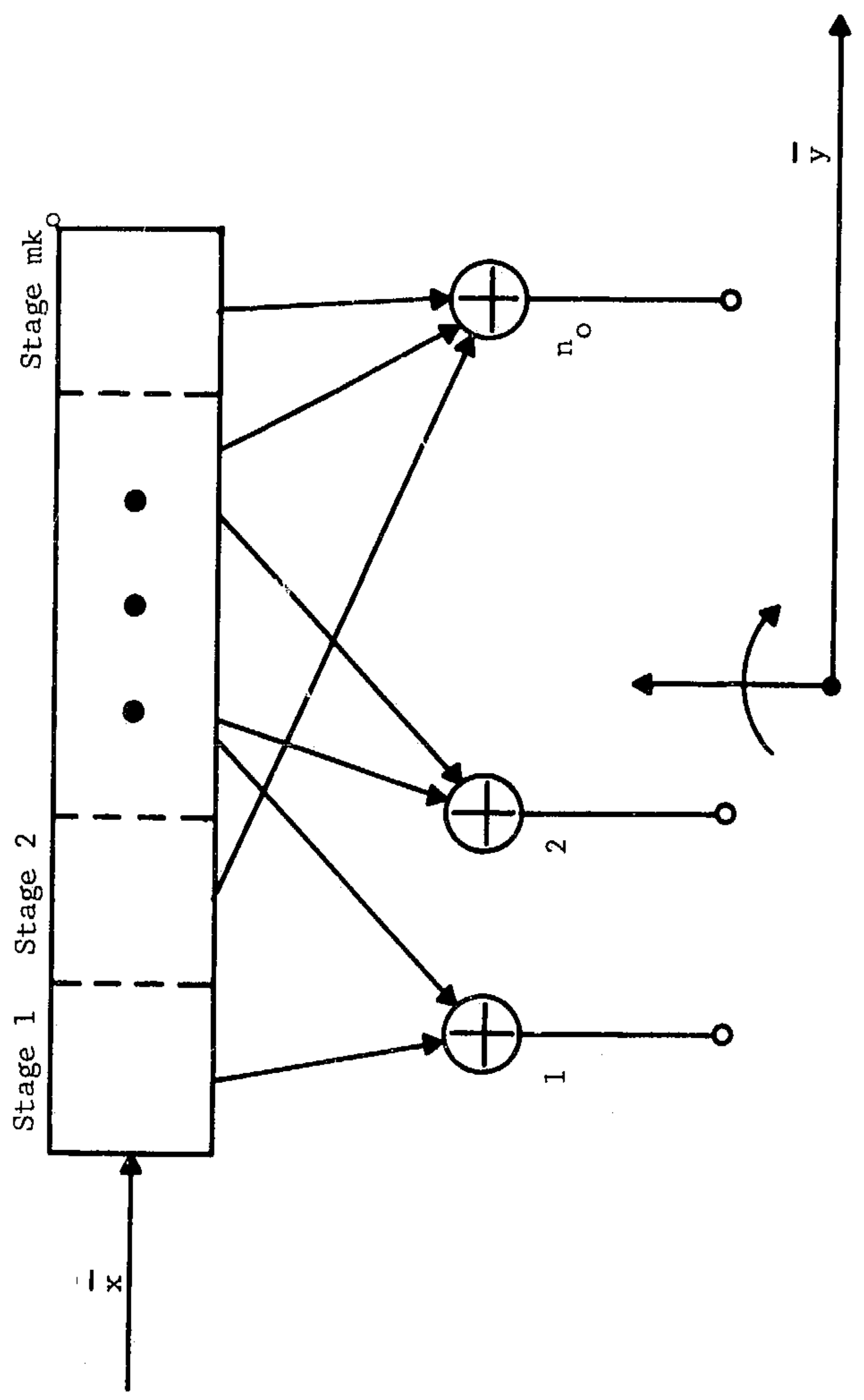

Fig. 7.1. A Convolutional Encoder.

(n_o,k_o) in denoting a convolutional code since it indicates the constraint length of the code. From Fig. 7.1 it is seen that by letting $m = 1$ the encoder simply generates an (n_o,k_o) block code.

The connections between the mk_o stages of the shift register and the n_o modulo-2 adders in Fig. 7.1 are conveniently described by a set of connection vectors. For example, the set of connections from the modulo-2 adders to the j'th stage of the register is described by the vector

$$\bar{g}_j \triangleq (g_{j1}, g_{j2}, \cdots, g_{jn_o}) \quad . \tag{7.1}$$

The component $g_{ji} = 1$ means that the j'th stage of the register is connected to the i'th adder,whereas $g_{ji} = 0$ means that it is not connected. The set of connection vectors can be written as a $k_o n_o m$-component vector

$$\bar{g} \triangleq (\bar{g}_1, \bar{g}_2, \cdots, \bar{g}_{mk_o}) \tag{7.2}$$

referred to as the generator of the code. A simple and practical search procedure for obtaining generators which produce codes with good error-correcting capability has been given by Lin and Lyne.[47]

B. Threshold Decoding of Convolutional Codes

Threshold decoding of convolutional codes is similar to threshold decoding of block codes, i.e., a set of equations orthogonal on a given error digit is used to determine the value of that error digit. Again only certain classes of codes can be decoded with threshold decoding. Codes which can be threshold decoded have been found by Massey[48] and by Robinson and Bernstein[49]. Both majority-logic decoding and APP decoding can be used. The complexity of these decoders is comparable to that of threshold decoders for block codes.

The performance of threshold-decoded convolutional codes can be evaluated only by simulation. Such a simulation has been performed by Neuman and Lumb in which the performance of two convolutional codes over the Gaussian channel was determined.[50] The constraint lengths of the codes were 24 and 44. Both a majority-logic decoder and a hard-decision APP decoder were used with each code. The results of this simulation are shown in Fig. 7.2. At a bit error probability of 10^{-4} only about 0.4 dB gain is achieved by increasing the constraint length from 24 to 44, but about 1.4 dB gain is achieved by using hard-decision APP decoding rather than majority-logic decoding. Of course, the better performance of an APP decoder is obtained through a large increase in decoder complexity as discussed previously. The small increase in performance due to increasing the constraint length is obtained rather cheaply with the addition of some shift register stages and logic. The maximum gain over no coding at a bit error probability of 10^{-4} is 3.0 dB, which is achieved by the (44,22) code with a hard-decision APP decoder.

C. Sequential Decoding of Convolutional Codes

Sequential decoding is a technique whereby the performance of a maximum-likelihood type algorithm can be approached with only a small average number of computations. Since convolutional codes have no fixed block structure, the groups of n_o output bits from an encoder may be thought of as forming branches on a semi-infinite code tree. A transmitted code word corresponds to a path through the code tree. Sequential decoding is a tree-search technique for determining the most likely code-tree path. This is done by comparing the received sequence with possible

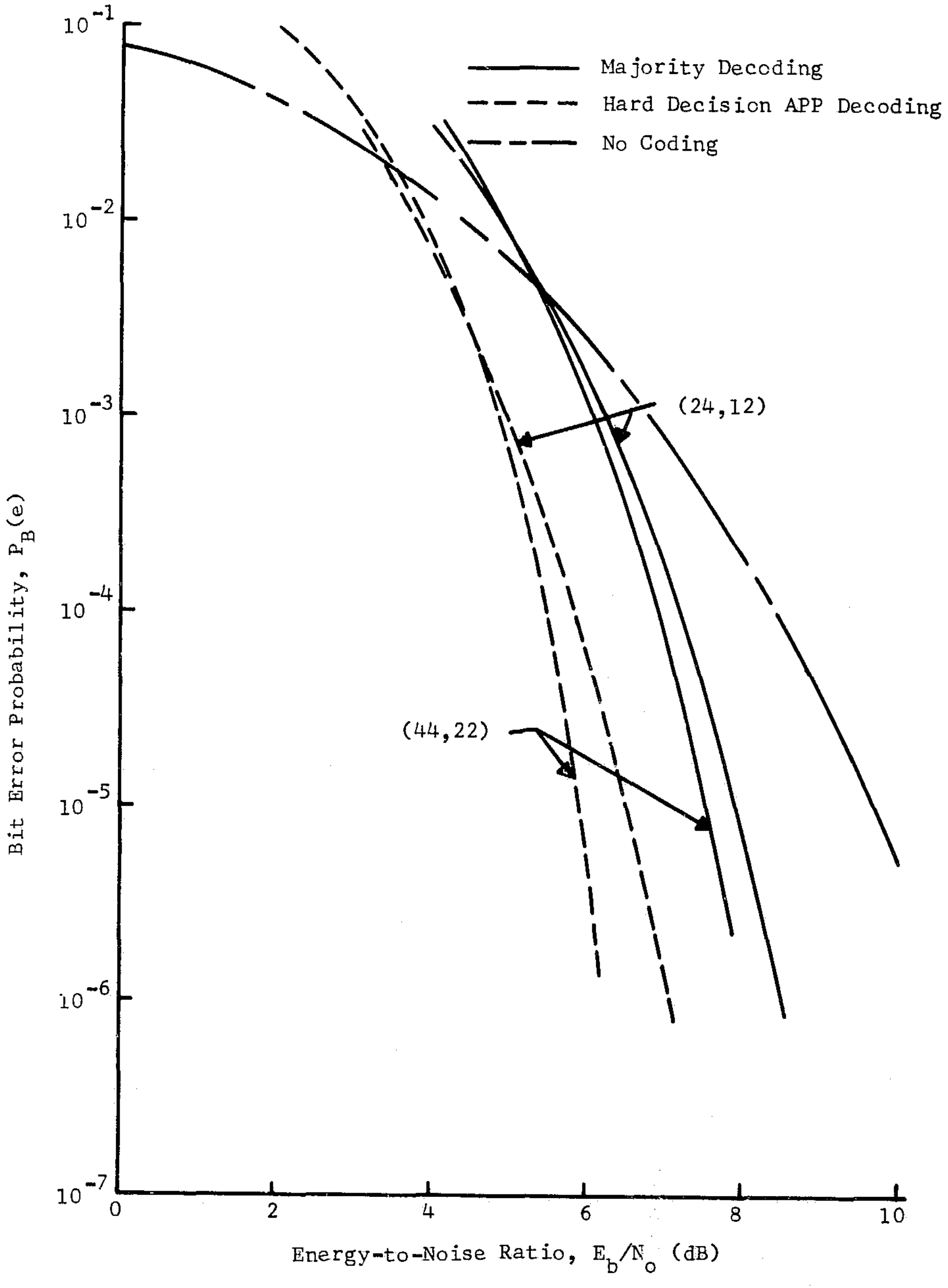

Fig. 7.2. Performance of Threshold Decoding of Convolutional Codes.

transmitted sequences according to a distance measure generally referred to as a metric. For binary quantization of the matched-filter output, the Hamming distance is used. For close-grained quantization of the matched filter output a generalized metric is used. The sequential decoding algorithm most often used is due to Fano.[51]

Wozencraft and Reiffen have shown that convolutional tree codes can be decoded by sequential decoding with an average probability of error which decreases exponentially with the constraint length of the code for all rates $R_N = k_o/n_o$ less than the computational limit R_{comp}.[52] This quantity is less than the channel capacity. For binary antipodal signaling over a white-Gaussian-noise channel and binary quantization of the matched-filter output,

$$R_{comp} = 1 - \log_2 \left[1 + 2\sqrt{p(1-p)}\right] , \qquad (7.3)$$

where

$$p = \frac{1}{\sqrt{2\pi}} \int_{\sqrt{2R_N E_b/N_o}}^{\infty} e^{-\alpha^2/2} d\alpha \ . \qquad (7.4)$$

For close-grained quantization of the matched-filter output,

$$R_{comp} = 1 - \log_2 \left[1 + \exp(-R_N E_b/N_o)\right] \qquad (7.5)$$

is obtained.[53] By using close-grained quantization rather than binary quantization, the same performance can be obtained with 1 to 2 dB less energy per information bit.

The principal advantage of sequential decoding is that for R_N sufficiently less than R_{comp}, the average number of computations per decoded digit is small. In general, sequential decoding requires a smaller average number of computations per decoded bit than other methods exhibiting comparable performance such as BCH codes. However, sequential decoding has the disadvantage that the computational demand on the decoder is extremely variable. This creates a need for input buffering, and since the buffer must be of finite capacity, there is a finite probability of buffer overflow. It has been shown that the probability of buffer overflow decreases only algebraically with the size of the buffer.[54] This is a distinct disadvantage because it is desirable to have a probability of error that is an exponentially decreasing function of decoder complexity.

Two types of errors occur with sequential decoding. An undetected error occurs when the sequential decoder decodes incorrectly, but the buffer does not overflow. The probability of undetected error decreases exponentially with the code constraint length. The other type of error occurs when the buffer overflows. In designing a practical decoder, the constraint length of the code is made long enough so that the probability of undetected error is negligible compared with the probability of buffer overflow. The reason for this is that decreases in the probability of undetected error can be obtained rather cheaply compared with the cost of obtaining similar decreases in the probability of buffer overflow.

A sequential decoder is very complex. The design of the decoder must be preceded by extensive simulation on a digital computer. This must be done to determine the code constraint length, the decoder speed advantage, and the decoder buffer size which are required in order to achieve the desired probability of error. The buffer requirement is generally 1000 to 10,000 tree branches. Also, the decoder must have a replica of the encoder for generating the code tree and all the arithmetic and control circuitry necessary for implementing the complex Fano algorithm.

Several simulations have been performed to evaluate the probability of error of various sequential decoders. The results of a simulation done by Wadden, Jones, and Bussgang are shown in Fig. 7.3.[55] The constraint lengths of the codes were 72 and 108, and both codes had a rate equal to one-third. The simulation was done for the binary symmetric channel, but the results were converted to the Gaussian channel by assuming that each received digit was coherently detected by a matched filter with a two-level output. Two runs were made for each code to evaluate the performance of the decoder when tuned to different values of the BSC transition probability p. However, this is not an entirely accurate measure of the performance of the decoder because the decoder was not allowed to overflow.

Much better performance with sequential decoding can be obtained by using fine quantization of the matched-filter output. A simulation of a sequential decoder utilizing 3-bit quantization (8 levels) of the matched-filter output has been performed by Lumb.[56] The results of this simulation of a (50,25) code are shown in Fig. 7.4. As predicted, this leads to much better performance than two-level quantization. A simulation was also done for the same code concatenated with a (7,6) parity check code. In this case the Fano algorithm was used to force the parity check bit to be correct. The encoder was resynchronized every 224 bits, and frame synchronization words were also included. The effect of a finite-length buffer was simulated by placing a constraint of 12,000 on the number of node trials in any one frame.

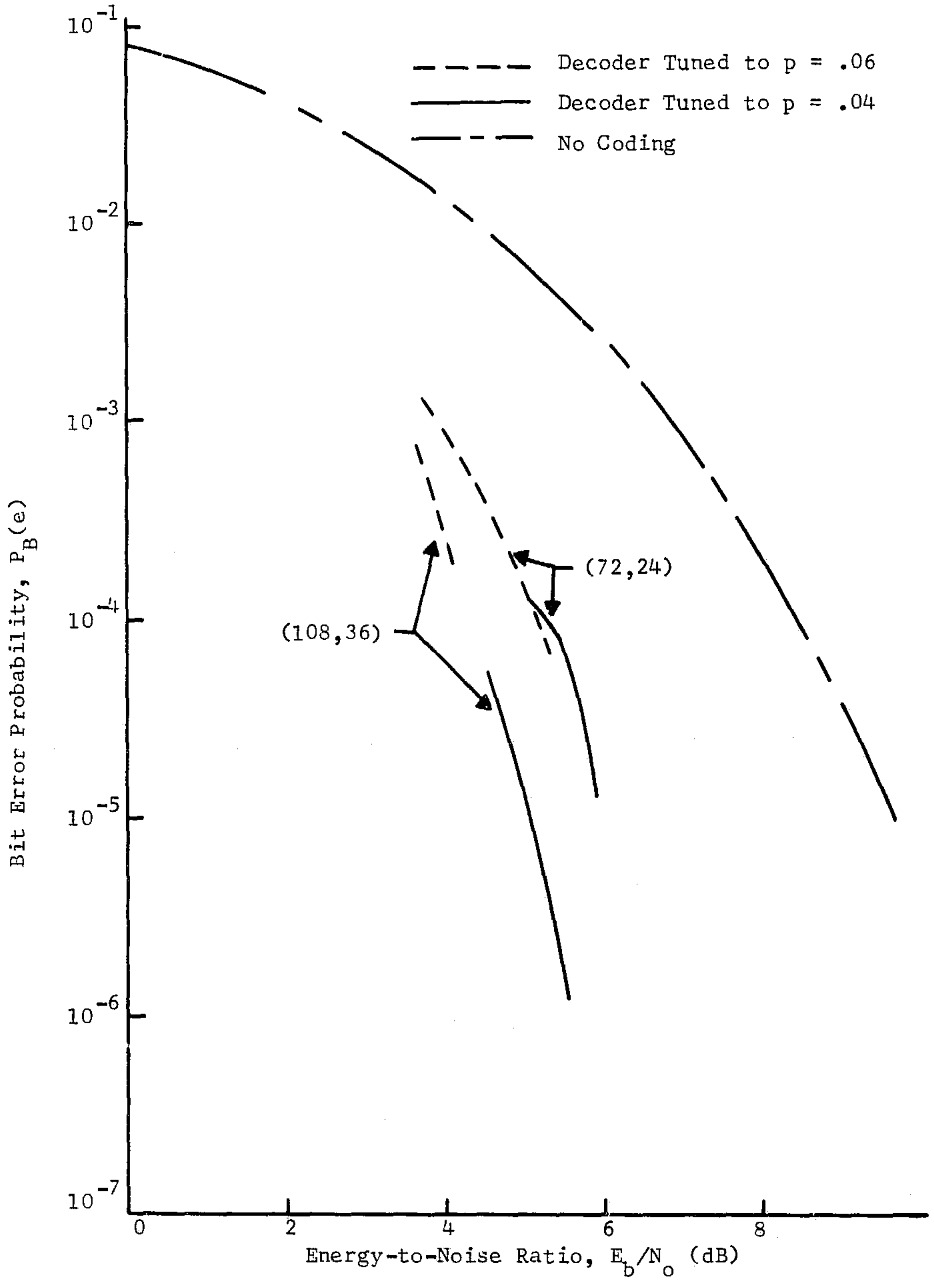

Fig. 7.3. Performance of Sequential Decoding of Convolutional Codes Using Two-Level Quantization of Matched-Filter Output.

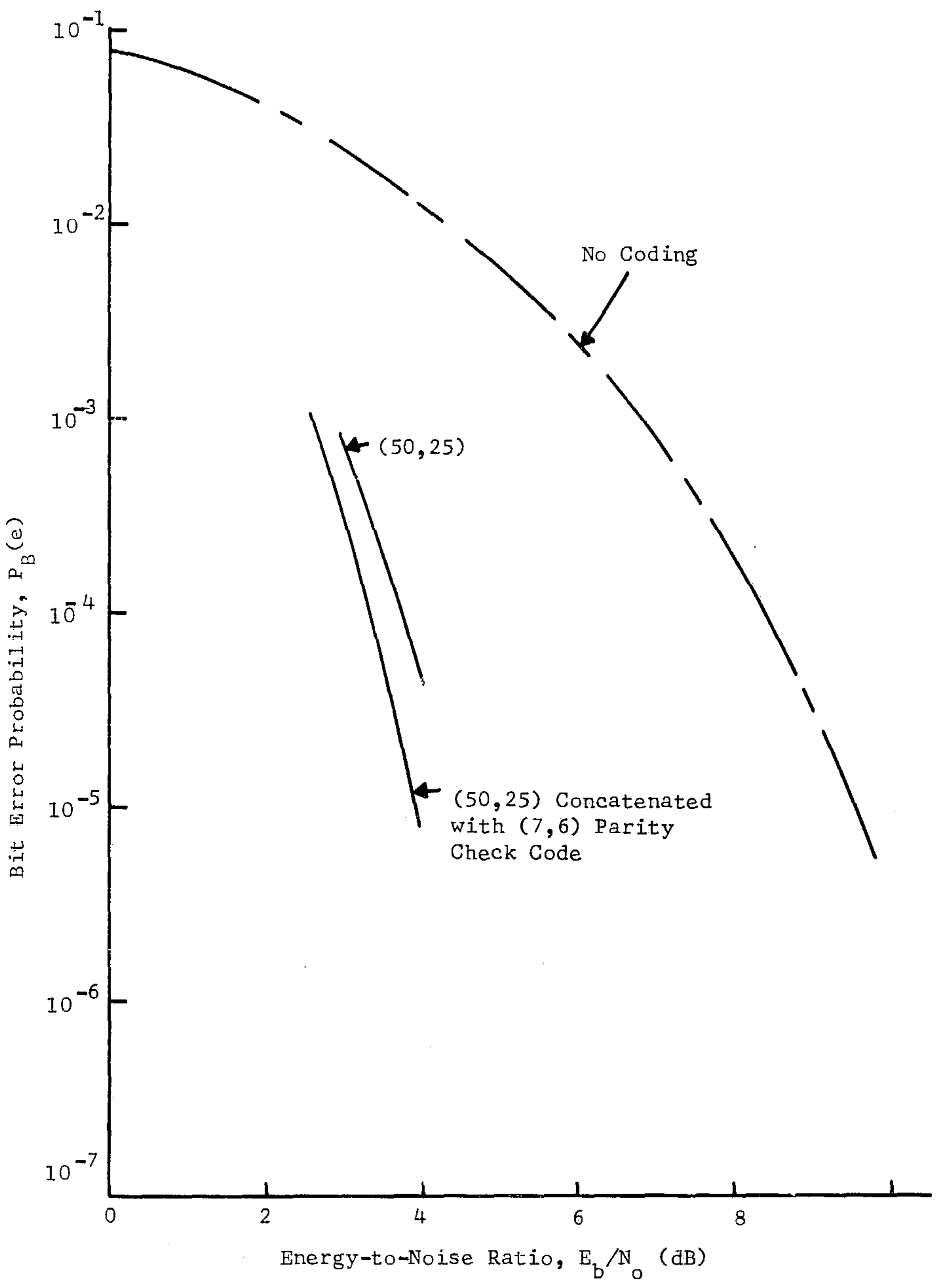

Fig. 7.4. Performance of Sequential Decoding of Convolutional Codes Using Eight-Level Quantization of Matched-Filter Output.

By comparing Figs. 7.2, 7.3, and 7.4, the better performance of sequential decoding is evident particularly if close-grained quantitization of the matched-filter output is used. The (44,22) code and the (50,25) code have comparable constraint lengths, but at a bit error probability of 10^{-4} with sequential decoding, 2.0 dB less energy per information bit is needed than with hard-decision APP decoding. Of course, this better performance is paid for with a large increase in decoder complexity.

CHAPTER VIII

FEEDBACK METHODS

All methods previously considered have utilized error-correcting codes over forward transmission channels. Many times a feedback link is available, and it seems reasonable that this link could be used to improve communication over the forward channel. Thus, Shannon's result that the channel capacity of a memoryless noisy channel is not increased by noiseless feedback is somewhat surprising.[57] However, the feedback channel can be used to decrease the complexity of encoding and decoding over the forward channel for a specified performance.

This chapter is primarily a survey of feedback methods to introduce to the reader the advantages to be gained when a feedback channel is available. Also, a modification of concatenated coding which allows the use of a feedback channel will be analyzed. The complexity of the schemes which are discussed will not be evaluated in detail, as has been done in previous chapters, since the cost of the feedback channel is highly dependent upon the particular application. The discussions will be limited to schemes utilizing a noiseless feedback channel.

A. Feedback Schemes with a Single Level of Coding

In this section four schemes which utilize a noiseless feedback channel will be discussed. These include a simple detection-retransmission scheme, a scheme analyzed by Viterbi[58] called sequential decision feedback, a scheme analyzed by Wyner[59] which is a modification of the Schalkwijk-Kailath[60] scheme, and a scheme analyzed by Kramer[61].

Detection-retransmission. One of the simplest feedback schemes to implement would be to use an (n,k) cyclic code for error detection and to request a retransmission whenever an error is detected. Transmission over a binary-symmetric forward channel with a noiseless, delayless, feedback channel will be assumed. If P(m,n) denotes the probability of m errors occurring in an n-bit block,

$$P(m,n) = \binom{n}{m} p^m (1-p)^{n-m} \tag{8.1}$$

for the BSC. The weight distribution, W(m), of a code is the number of code words of weight m. If all error patterns of a given weight occur with equal probability as is the case on the BSC, the probability of erroneous decoding is

$$P_e = \sum_{m=d}^{n} \frac{W(m)}{\binom{n}{m}} P(m,n) \quad , \tag{8.2}$$

where d is the minimum distance of the code. It has been observed by Peterson[62] that the weight distribution of many random-error-correcting codes, such as the BCH codes, is approximated by

$$W(m) \approx \frac{\binom{n}{m}}{2^{n-k}} \quad , \tag{8.3}$$

and thus P_e is approximated by

$$P_e \approx \frac{1}{2^{n-k}} \sum_{m=d}^{n} P(m,n) \quad . \tag{8.4}$$

The probability of correct decoding is

$$P_c = P(0,n) \quad . \tag{8.5}$$

The probability of detecting an error, P_d, is therefore

$$P_d = 1 - P_c - P_e \quad . \tag{8.6}$$

This system has a probability of word error equal to

$$\begin{aligned} P_W(e) &= (1-P_d)P_e + (1-P_d)P_dP_e + (1-P_d)P_d^{\,2}P_e + \cdots \\ &= (1-P_d)P_e \sum_{i=0}^{\infty} P_d^{\,i} \\ &= P_e \quad . \end{aligned} \tag{8.7}$$

The average number of transmissions per decoded word for this system is

$$\begin{aligned} T &= (1-P_d) + 2P_d(1-P_d) + 3P_d^{\,2}(1-P_d) + \cdots \\ &= \frac{(1-P_d)}{P_d} \sum_{n=1}^{\infty} nP_d^{\,n} \\ &= \frac{1}{1-P_d} \quad . \end{aligned} \tag{8.8}$$

If the BSC is constructed from a white-Gaussian-noise channel by using antipodal signaling with matched-filter detection, the transition probability, p, as a function of E_s/N_o, the ratio of the received signal energy per symbol to the noise spectral density, is given by (3.50). The received energy per information symbol for this system is

$$E_b = \frac{n}{k} T \, E_s \quad . \tag{8.9}$$

With (3.50) and (8.1) through (8.9) curves of $P_W(e)$ vs. E_b/N_o can be determined for the detection-retransmission system with a noiseless feedback link.

A comparatively small amount of equipment is required to implement a simple detection-retransmission scheme. The encoder is implemented with either k or n-k stages of shift register. The decoder will consist of a syndrome register of n-k stages and two data registers of k stages each. Since the number of transmissions per code word is variable, a buffer will be needed at the encoder. The size of the buffer will be determined by the code and p.

For values of n and p large enough the average number of transmissions per decoded word, T, becomes excessive so that the system is not efficient. This situation can be remedied by performing some error correction. The decoder examines the received word and determines if it lies within distance $t < d/2$ of a code word. If it does, the decoder performs the necessary error correction. If the received word does not lie within distance t of a code word, the decoder requests a retransmission. The number of errors, t, which are corrected can be any integer in the range $0 \leq t \leq d/2$, and for a given code and p the value of t which results in the best performance must be determined by trial-and-error. For this system the probability of correct decoding is

$$P_c = \sum_{m=0}^{t} P(m,n) \quad . \tag{8.10}$$

The probability of erroneous decoding is approximated by

$$P_e \approx \frac{\sum_{i=0}^{t} \binom{n}{i}}{2^{n-k}} \sum_{m=d-t}^{n} P(m,n) \quad , \tag{8.11}$$

and the probability of detecting an error is again given by (8.6).[63] For this system (8.7) and (8.8) still hold if the values of P_c and P_e given in (8.10) and (8.11) are used. Again by using the appropriate equations, curves of $P_W(e)$ vs. E_b/N_o can be determined.

A decoder for this system need not be complex. If BCH codes are used, and t is small, the Peterson algorithm may be solved before the decoder is built making the Gauss-Jordan matrix reduction unnecessary. A buffer will also be needed, the size of which is determined by the code, t, and p.

The two systems analyzed above utilize a noiseless, delayless, feedback link. When there is propagation delay a loss in efficiency of communication occurs which is not too serious. However, a feedback error will cause a loss of synchronization which is indeed serious. If the feedback channel is noisy, the feedback data must be protected by coding or by some type of retransmission logic so that a loss of synchronization does not occur.

Sequential decision feedback . Viterbi has analyzed a postdecision feedback strategy called sequential decision feedback.[64] The feedback channel is used only to inform the transmitter that a decision has been made. The set of code words are M orthogonal signals $s_i(t)$, $i = 0, 1, \cdots, M-1$. Correlation detection is used. Letting the received waveform be denoted by r(t), the correlation detector computes the M quantities

$$Z_i(t) = \int_0^t r(u)s_i(u)du, \quad i = 0,1,\cdots, M-1 \quad . \tag{8.12}$$

The M statistics are compared with a threshold level L. As soon as $Z_j(t) \geq L$ for some j, the receiver decides that the transmitted signal was $s_j(t)$, and it signals the transmitter to terminate transmission. Of course, the threshold level L is determined by the specified probability of error. By applying the units of normalized length and rate for the Gaussian channel the error probability for sequential decision feedback becomes

$$P_W(e) = \exp\left[-n^* \left[e_1(r^*) + \varepsilon_1(n^*)\right]\right] \quad , \tag{8.13}$$

where $\varepsilon_1(n^*) \to 0$ as $n^* \to \infty$ and where

$$e_1(r^*) = \begin{cases} 2(\sqrt{2}-1) - r^* \ , & 0 \leq r^* \leq 1/\sqrt{2} \ , \\ \\ (\sqrt{1/r^*} - \sqrt{r^*})^2, & 1/\sqrt{2} < r^* < 1 \quad . \end{cases} \tag{8.14}$$

Using (8.13) and (8.14), $P_W(e)$ can be determined as a function of E_b/N_o for any value of k, the number of information bits per code word. Fig. 8.1 shows $P_W(e)$ vs. E_b/N_o for integer values of k in the range $2 \leq k \leq 8$. Comparing the curves of Fig. 8.1 with similar curves in Fig. 4.1, which show $P_W(e)$ vs. E_b/N_o for bi-orthogonal codes, at a $P_W(e) = 10^{-5}$ for k = 8 the system with feedback requires 1.4 dB less energy per information bit. The advantage to be gained by using feedback decreases as k decreases until at k = 2 the system with feedback requires 1.0 dB less energy per information bit. Thus, the use of sequential decision feedback offers significantly better performance than the use of bi-orthogonal coding.

With sequential decision feedback the complexity of the decoder grows exponentially with the number of information bits per code word since correlation detection is used. However, the channel capacity requirements of the feedback channel are small because only decision information is fed back. There is a buffering problem at the transmitter since the transmission time per code word is variable. The effect of propagation delay is a decrease in actual information rate.

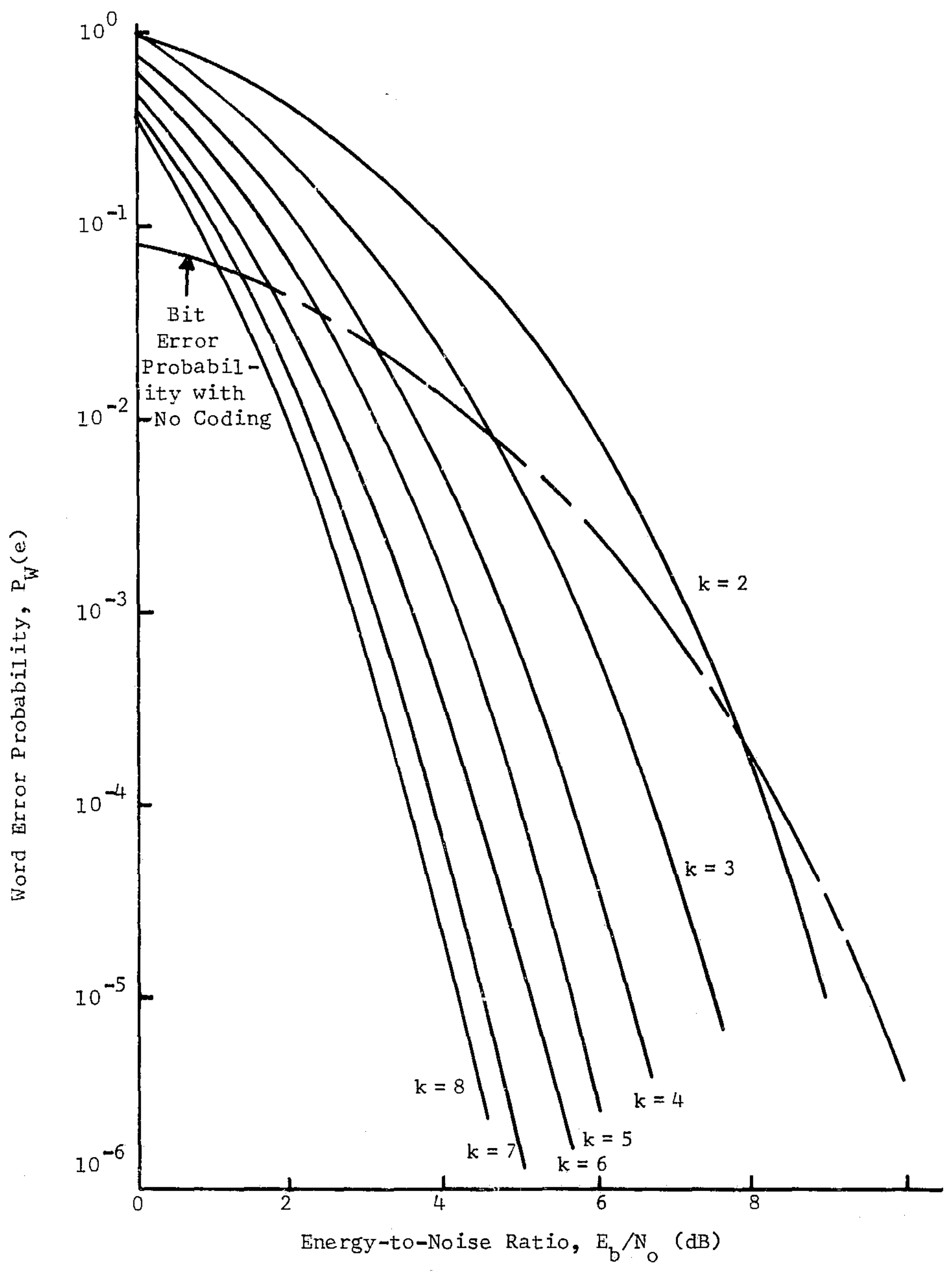

Fig. 8.1. Performance of Sequential Decision Feedback.

Feedback errors will cause serious synchronization errors which must be combated by coding the feedback information or by some other means.

Wyner's modification of the Schalkwijk-Kailath feedback scheme. A predecision feedback strategy has been described by Schalkwijk and Kailath.[65] With this scheme the number of feedback iterations is fixed at a number, say (N-1), to give a specified probability of error. The forward channel is then used N times per message. After the j'th use of the forward channel, the receiver obtains an estimate of the transmitted message, which typically becomes better as j increases. An advantage of this scheme is that the signals are formed by amplitude modulating a single basic waveform. The signaling scheme is constructed so that as the receiver estimate becomes better, the energy required in the next transmission is decreased, and the expected transmitter power per iteration decreases as j increases. The probability of error for this scheme has a double exponential behavior, i.e.,

$$P_W(e) = \exp\left[-\exp T\left[2(C-R) + \varepsilon_2(T)\right]\right] , \qquad (8.15)$$

where $\varepsilon_2(T) \to 0$ as $T \to \infty$. Thus, the error probability has very good behavior, but there is a disadvantage since the number of uses of the forward channel per message grows exponentially with T causing very large peak-to-average power ratios to be obtained.

Wyner[66] considers a more realistic system which is identical to that of Schalkwijk and Kailath except that a peak energy constraint is imposed. For this scheme the average power (a random variable) is denoted by P, and the corresponding energy is denoted by E = PT. The expectations of these quantities are $\overline{P}$ and $\overline{E}$. Wyner's modification consists of terminating transmission whenever the scheme requires energy $E > a\bar{E}(a > 1)$ and declaring an error. In this case he showed that

$$P_W(e) = \exp\left[-n^* \left[e_2(a) + \varepsilon_3(n^*)\right]\right] , \qquad (8.16)$$

where $\varepsilon_3(n^*) \to 0$ as $n^* \to \infty$ and

$$e_2(a) = \frac{(a-1)^2}{4a} , \; 0 \leq r^* \leq 1 . \qquad (8.17)$$

This exponent is a positive constant for all rates in the range $0 \leq r^* \leq 1$ and thus is positive at $r^* = 1$, an unusual property.

As shown in (8.17) any value of the exponent, $e_2(a)$, can be obtained by choosing the appropriate value of a. For large values of a this system has much better performance than sequential decision feedback. The signals used in this scheme are easy to generate and to demodulate, since they are constructed from a single basic waveform which is amplitude modulated. The influence of propagation delay becomes negligible for large N. Use of the Schalkwijk-Kailath scheme also involves several disadvantages. Keeping a non-zero rate as T is increased involves letting N grow exponentially with T, so that a very large number of feedback iterations is generally required. Even though this scheme operates under an average power constraint and a peak-energy-per-code-word constraint, the ratio of instantaneous power during any given feedback iteration to average power gets very large as N gets large. Finally, if there is feedback noise, either a vanishing probability of error can be obtained with a signaling rate approaching zero, or for a specified non-zero rate there is a minimum achievable probability of error different from zero.

Kramer's feedback scheme. Kramer[67] has analyzed a predecision feedback scheme which is a modification of Schalkwijk's[68] center-of-gravity feedback scheme. With Kramer's scheme the number of feedback iterations is a fixed number, say (N-1). After each use of the forward channel, the receiver informs the transmitter via the feedback channel which message has the largest a posteriori probability. The transmitter then subtracts the signal corresponding to the correct message and transmits the result. At the receiver the signal which was subtracted is added in again. By doing this the probability of error is not affected, but the average energy the transmitter uses can be decreased. Orthogonal signals with correlation detection are used in this scheme. If it is required to send one of M messages every T seconds, the signals are required to be orthogonal over the time interval $\tau = T/N$. For this system if the peak power, P_{pk}, is defined as the largest average power over a subinterval τ,

$$\frac{P_{pk}}{P_{av}} \to 2N \text{ as } \tau \to \infty \quad . \tag{8.18}$$

The error probability for large T is

$$P_W(e) = \exp\left[-n^*\left[e_3(r^*) + \varepsilon_4(n^*)\right]\right] \quad , \tag{8.19}$$

where $\varepsilon_4(n^*) \to 0$ as $n^* \to \infty$ and

$$e_3(r^*) = \begin{cases} N/2 - r^* \ , & 0 \leq r^* \leq \min(1, N/4) \quad , \\ \left[\sqrt{N} - \sqrt{r^*}\right]^2, & \min(1, N/4) \leq r^* \leq 1 \quad . \end{cases} \tag{8.20}$$

This scheme has several advantages. It has an error exponent which, for a fixed rate, increases with increasing values of N. For large values of N this system has much better performance than sequential decision feedback. The number of feedback iterations per code word can be fixed independent of the desired error probability, and the ratio of the transmitter peak power to average power can be fixed and still have arbitrarily small error probability for non-zero information rates. If the feedback channel is noisy but the average power is above a fixed (finite) rate, the error probability has the same asymptotic behavior as for noiseless feedback. Also, if there is propagation delay, modifications can be made so that performance is not altered much.

The disadvantages of this scheme are that for large N the ratio of peak power to average power is large, and since orthogonal signaling with correlation detection is used, the complexity of the receiver grows exponentially with the number of information bits per code word.

B. Concatenation Schemes for the Gaussian Channel

A block diagram of a concatenated coding system with noiseless, delayless, feedback is shown in Fig. 8.2. This system differs from the one proposed by Forney shown in Fig. 6.1 by the addition of the feedback channel associated with the inner code. Schemes utilizing outer-code feedback will not be investigated because extensive buffering would be required in any such scheme. In this section concatenation schemes will be analyzed which utilize inner-code feedback. The inner coding schemes to be used are those analyzed in the previous section: sequential decision feedback, Wyner's modification of the Schalkwijk-Kailath feedback scheme, and Kramer's feedback scheme. The concatenation error exponents for these schemes are calculated in Appendix D and presented in (D.8), (D.12), (D.14), (D.18), (D.20, (D.23), (D.24), (D.29), and (D.30).

Three sets of error exponents have been obtained which when substituted into the Coding Theorem of Chapter VI merely shows the existence

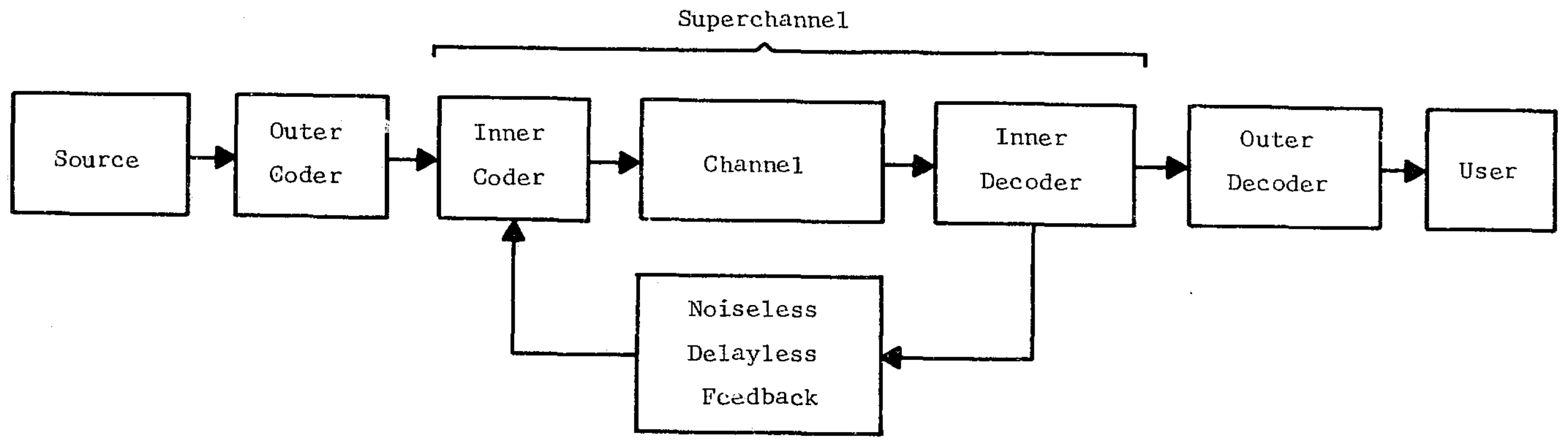

Fig. 8.2. A Concatenated Coding System with Noiseless, Delayless, Feedback.

of an outer coding scheme which achieves the predicted performance. Forney shows that by using a Reed-Solomon outer code with errors-only decoding an error exponent equal to one-half the upper-bound concatenation exponent can be achieved. The Reed-Solomon decoder is readily implemented especially if the error-correction capability of the code is small, implying a high rate. Thus, if a Reed-Solomon outer code implemented with an errors-only decoder is concatenated with sequential decision feedback, Wyner's modification of the Schalkwijk-Kailath feedback scheme, and Kramer's feedback scheme, the error exponents $e_{C1}(r_1)/2$, $e_{C2}(r_1)/2$, and $e_{C3}(r_1)/2$ can be achieved, respectively.

With sequential decision feedback used as the inner code, only one set of error exponents is obtained, and this set is shown in Fig. 8.3. Three members of the family of error exponents obtained when Wyner's modification of the Schalkwijk-Kailath feedback scheme is used as the inner code are plotted in Fig. 8.4. The upper-bound exponents are indicated by solid lines, and the lower-bound exponents are indicated by dotted lines, except for a = 2 where they coincide. From the form of the exponents and the fact that "a" may assume any value greater than one, it is obvious that for any over-all rate in the range $0 \leq r_1 < 1$, any finite positive value for the error exponent can be obtained. Two members of the family of error exponents obtained when Kramer's scheme is used as the inner code are plotted in Fig. 8.5. Since N is equal to the number of transmissions per code word, it may take on integer values only. Thus, not all values of the error exponents can be obtained as with Wyner's modification of the Schalkwijk-Kailath feedback scheme, but an exponent as large as desired can be obtained by taking N sufficiently large.

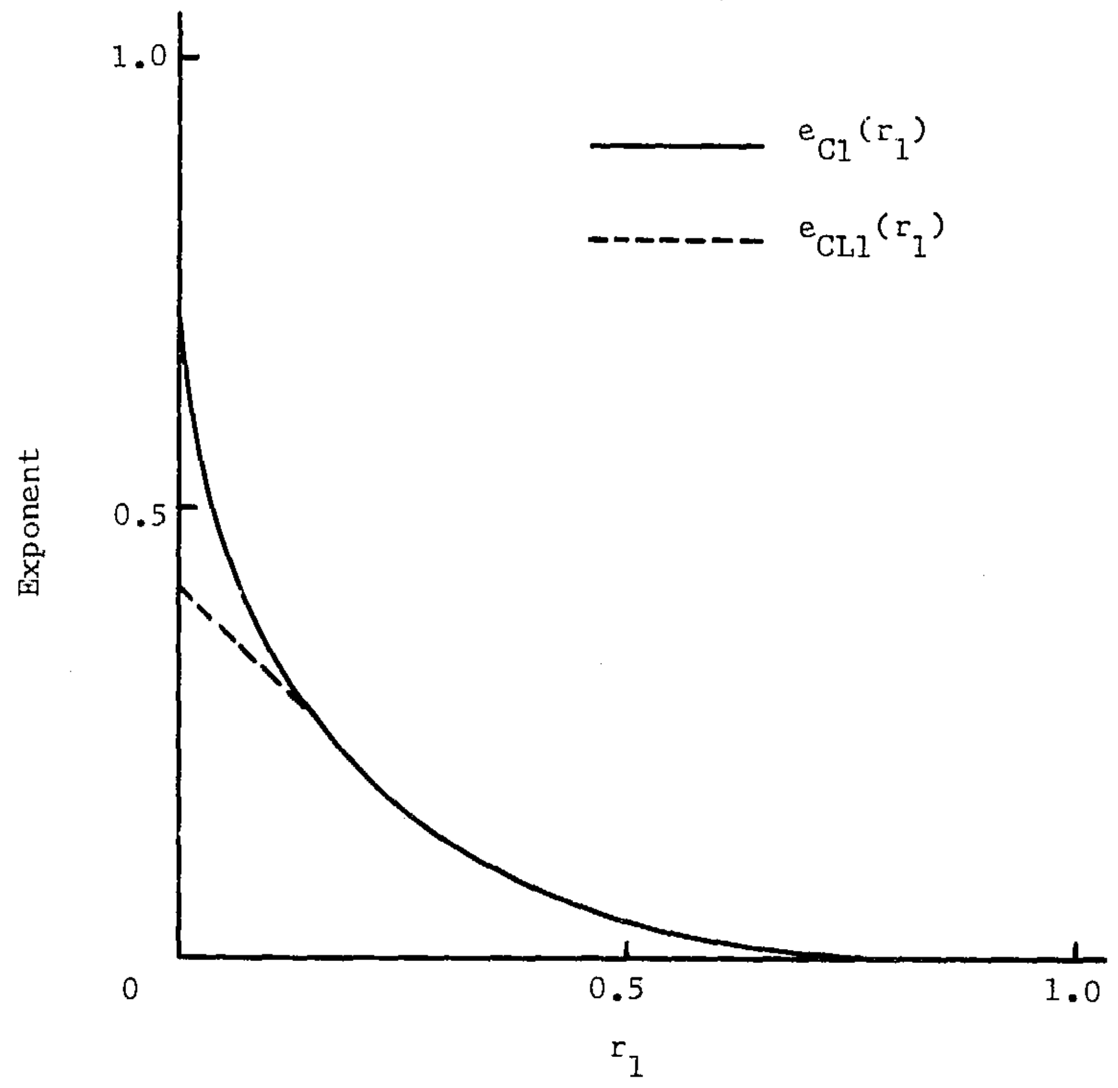

Fig. 8.3. Concatenation Error Exponents for Sequential Decision Feedback.

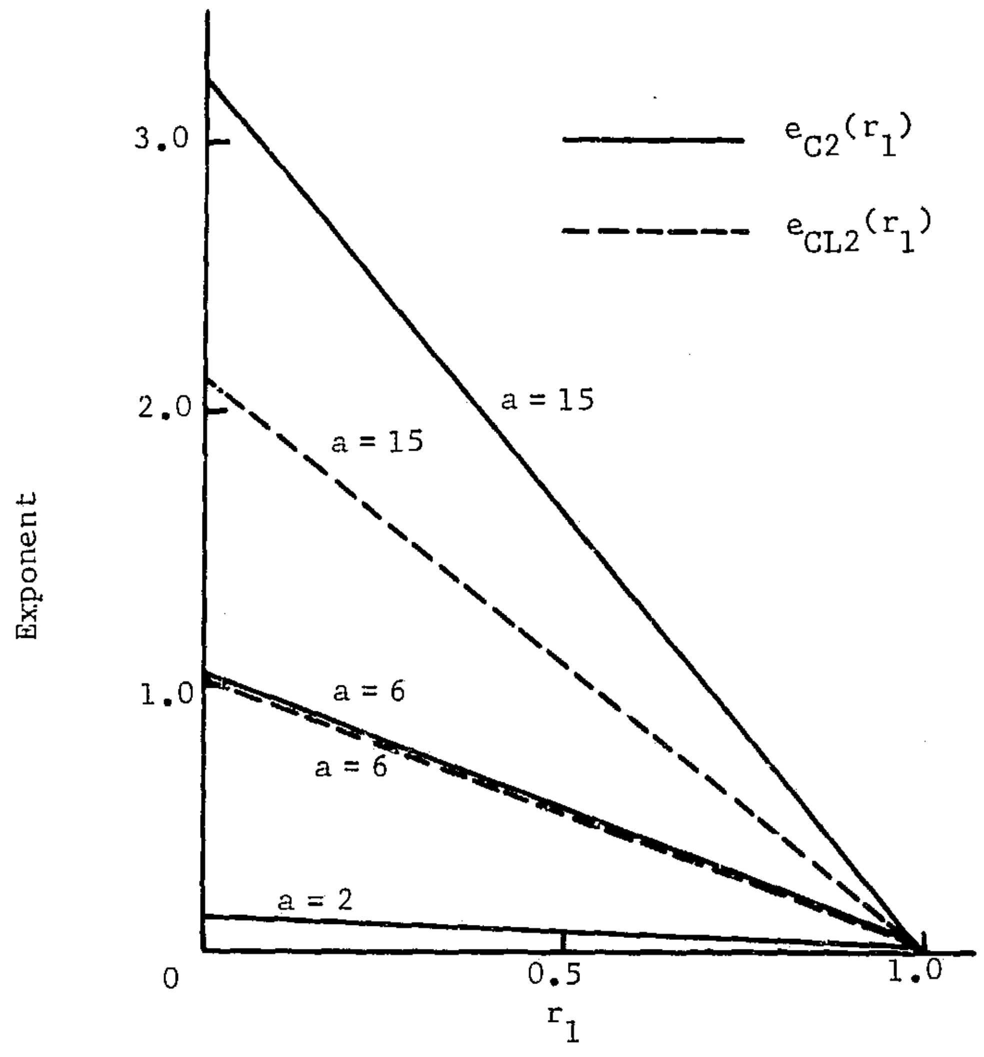

Fig. 8.4. Concatenation Error Exponents for Wyner's Modification of the Schalkwijk-Kailath Feedback Scheme.

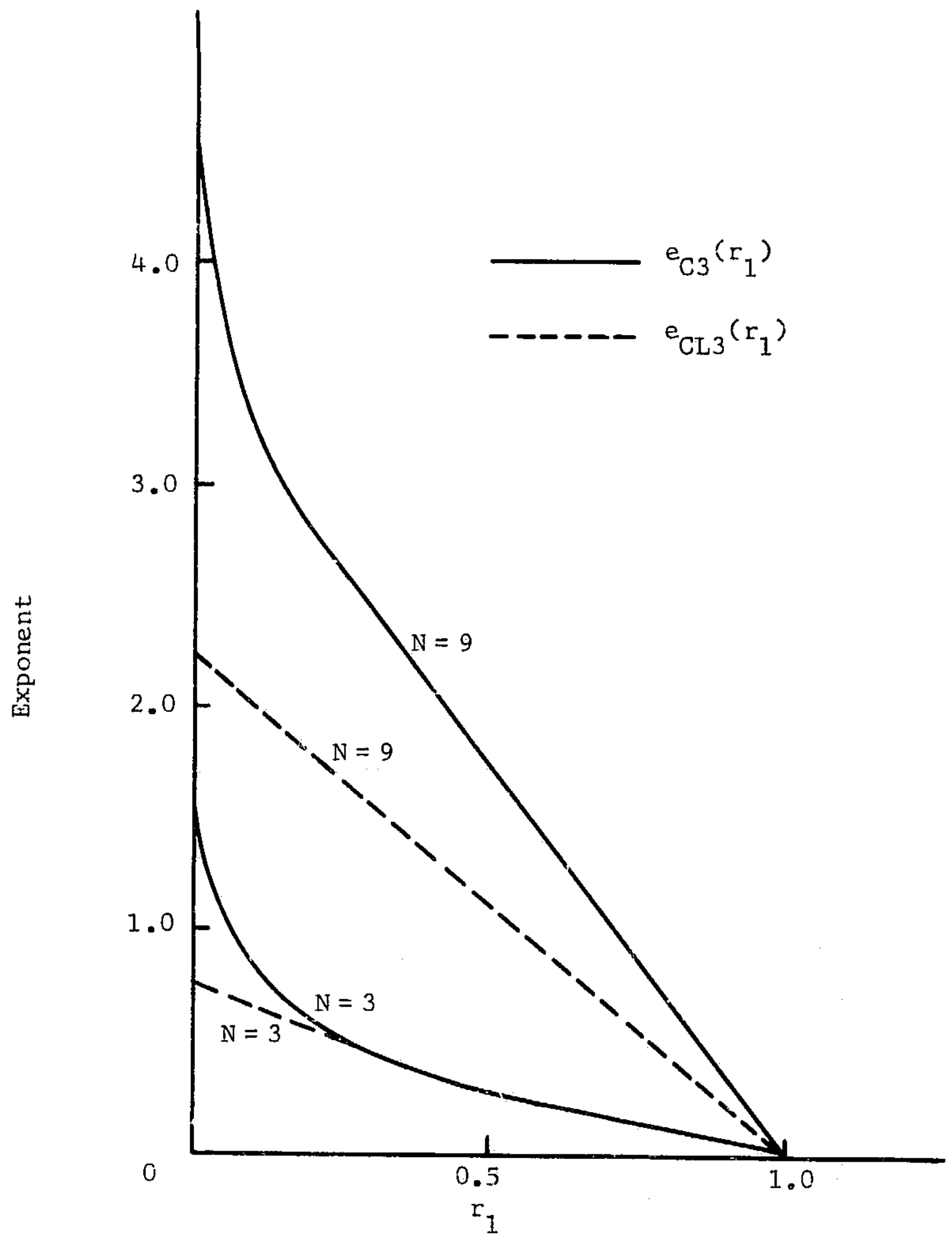

Fig. 8.5. Concatenation Error Exponents for Kramer's Feedback Scheme.

Concatenation introduces a degradation of the error exponent which can be measured by the efficiency of concatenation defined by (6.10). For one of the schemes, Wyner's modification of the Schalkwijk-Kailath feedback scheme, the efficiency is given by the very simple expression

$$\eta(r_1) = 1 - r_1 \quad , \tag{8.21}$$

which is obtained from (8.17), (D.14), and (6.10). This efficiency and the efficiencies of concatenation for sequential decision feedback and for the Gaussian channel with no feedback are shown in Fig. 8.6. There is not much difference between the efficiency of sequential decision feedback and that for the Gaussian channel. However, the efficiency of Wyner's modification of the Schalkwijk-Kailath feedback scheme is considerably larger than either of the other two. The efficiency for Kramer's feedback scheme coincides with that for the white-Gaussian-noise channel for N = 1 and increases with N. For large N the efficiency of Kramer's feedback scheme is

$$\begin{aligned} \eta(r_1) &= \frac{(N/2 - 1)(1 - r_1)}{N/2 - r_1} \\ &\approx 1 - r_1 \quad , \end{aligned} \tag{8.22}$$

which is the efficiency of Wyner's modification of the Schalkwijk-Kailath feedback scheme.

The concatenation error exponents obtained with the feedback schemes can be readily compared with the concatenation error exponent for the Gaussian channel with no feedback by forming the functions,

$$A_{C1}(r_1) = \frac{e_{C1}(r_1)}{e_C(r_1)} \quad , \tag{8.23}$$

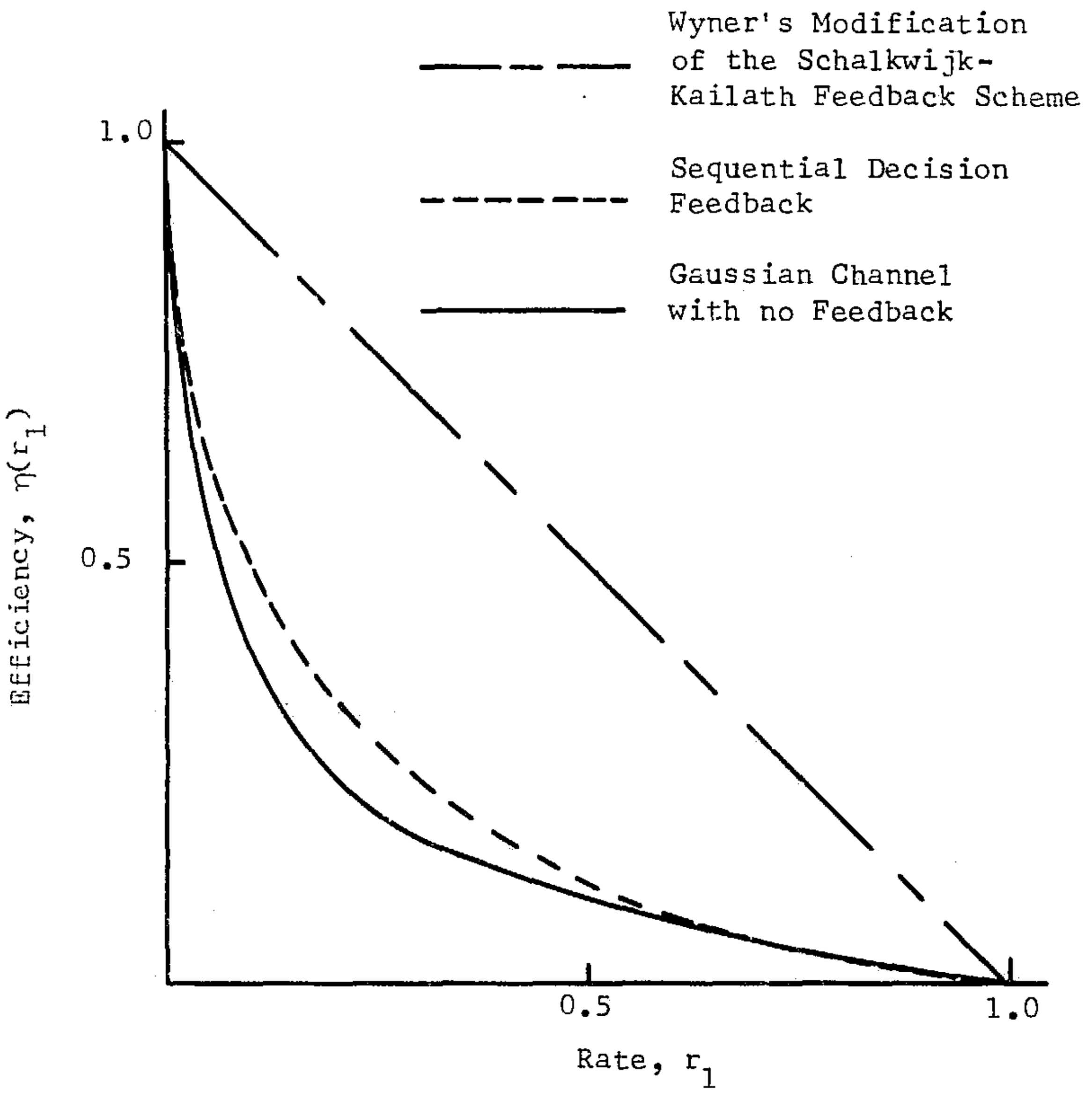

Fig. 8.6. Efficiency of Concatenation for Wyner's Modification of the Schalkwijk-Kailath Feedback Scheme, Sequential Decision Feedback, and the Gaussian Channel with No Feedback.

$$A_{C2}(r_1) = \frac{e_{C2}(r_1)}{e_C(r_1)} \quad , \tag{8.24}$$

and

$$A_{C3}(r_1) = \frac{e_{C3}(r_1)}{e_C(r_1)} \quad . \tag{8.25}$$

The function $A_{C1}(r_1)$ is shown in Fig. 8.7. It has a value greater than 4.0 for all rates in the range $0.36 \leq r_1 \leq 1$, which is a large improvement over using no feedback. This improvement is gained at virtually no loss in efficiency of concatenation. The functions $A_{C2}(r_1)$ and $A_{C3}(r_1)$ are shown in Figs. 8.8 and 8.9 for several values of a and N. The fact that these functions approach infinity as r_1 approaches one is somewhat surprising. This happens because the error exponents $e_{C2}(r_1)$ and $e_{C3}(r_1)$ approach zero linearly as r_1 approaches one, while the error exponent $e_C(r_1)$ approaches zero somewhat faster. Thus, for high rates Wyner's modification of the Schalkwijk-Kailath feedback scheme and Kramer's feedback scheme can both offer an extremely large improvement in error exponent over that obtained when no feedback is used. Along with this improvement in error exponent an increase in the efficiency of concatenation is also obtained.

By using the error exponents $e_{C1}(r_1)/2$, $e_{C2}(r_1)/2$, and $e_{C3}(r_1)/2$ for calculating the probability of error for these concatenation schemes, $P_W(e)$ vs. E_b/N_o curves can be obtained. These curves for several combinations of inner and outer code parameters are shown in Figs. 8.10, 8.11, and 8.12 for sequential decision feedback, Wyner's modification of the Schalwijk-Kailath feedback scheme (a = 4), and Kramer's feedback scheme (N = 4), respectively. The label for each curve is of the form (n,k,m), where n is the block length of the outer code, k is the number

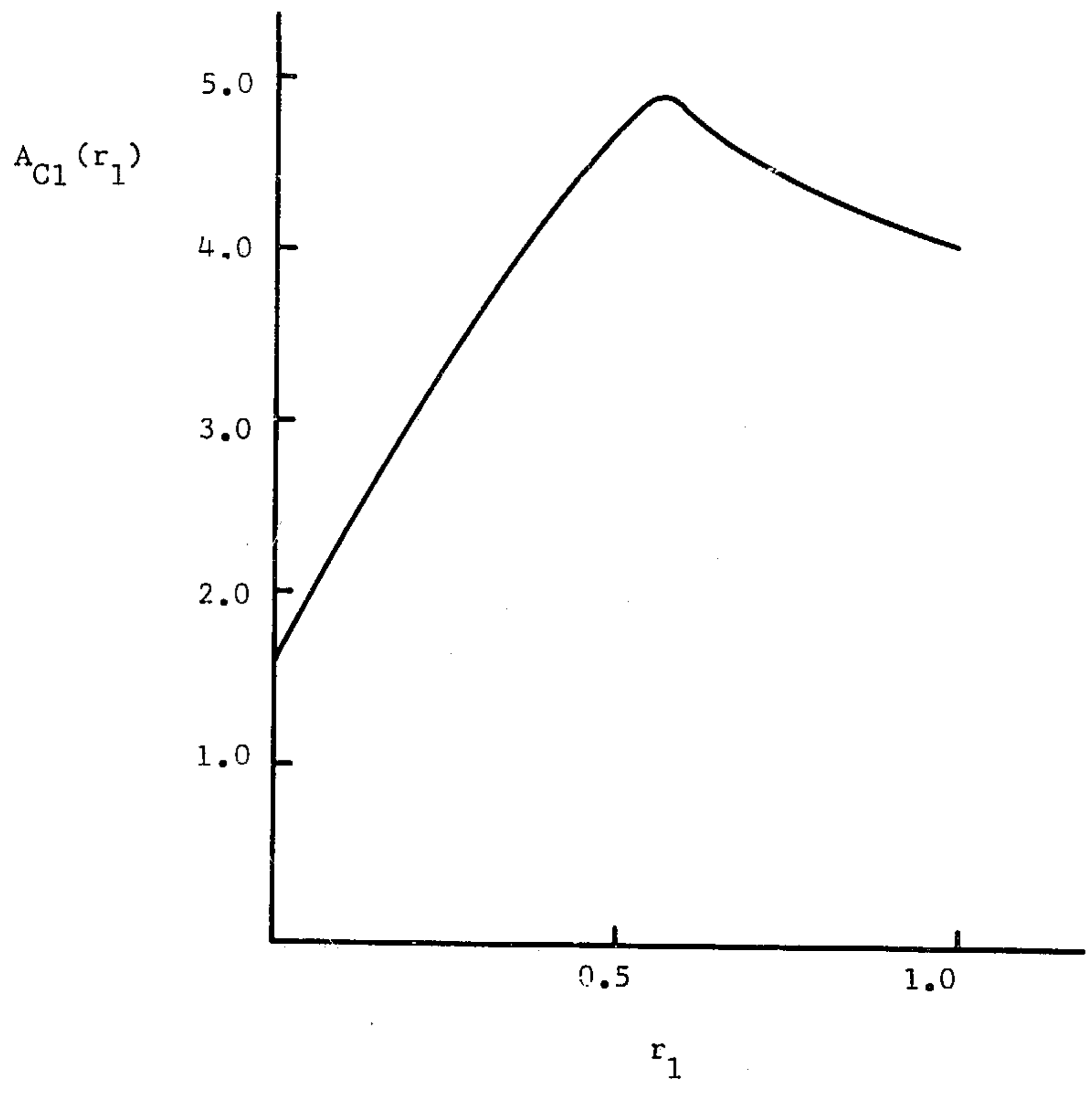

Fig. 8.7. A Comparison of the Upper-Bound Concatenation Exponent for Sequential Decision Feedback, $e_{C1}(r_1)$, to that for the Gaussian Channel with No Feedback, $e_C(r_1)$, where $A_{C1}(r_1) = e_{C1}(r_1)/e_C(r_1)$.

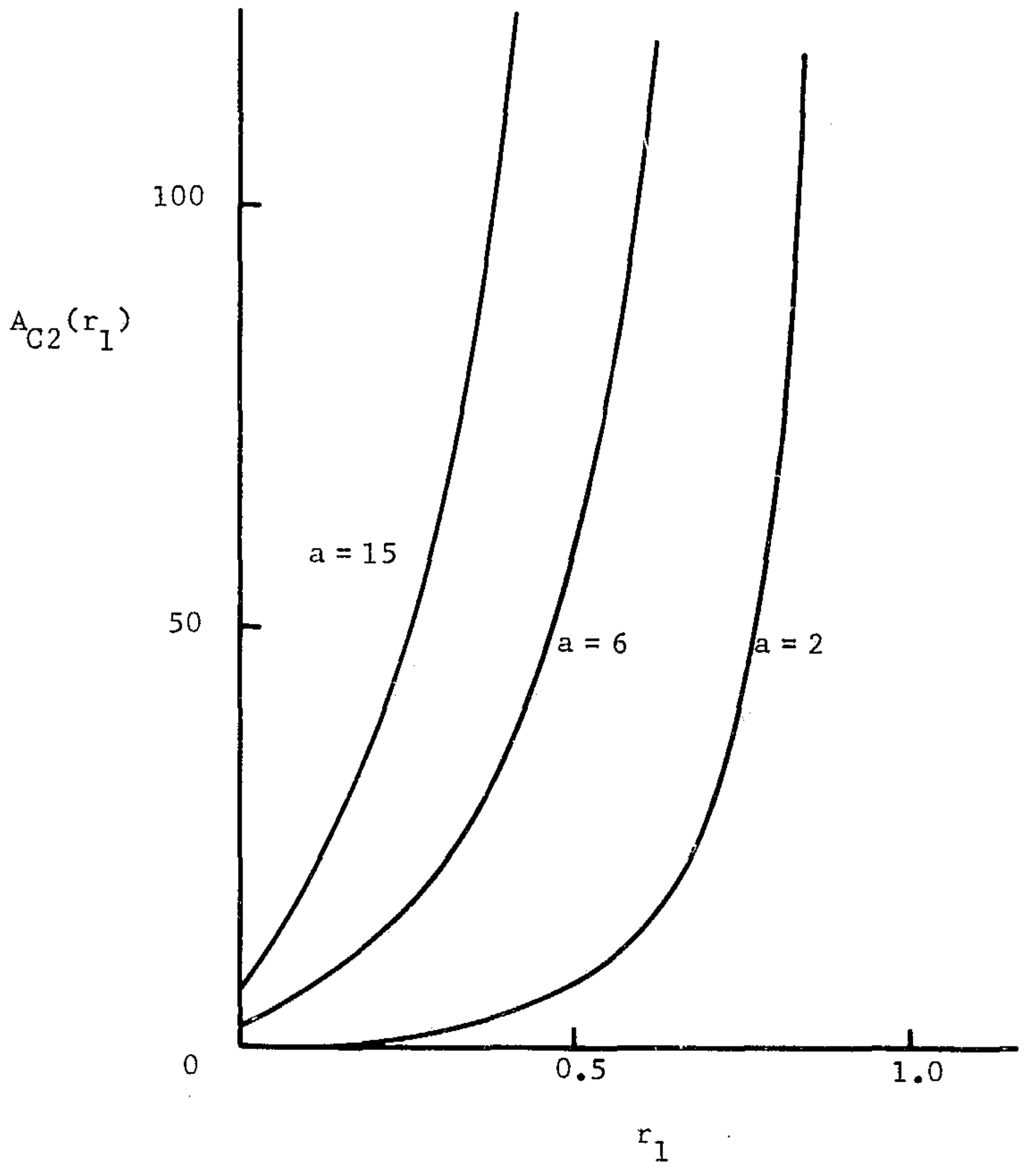

Fig. 8.8. A Comparison of the Upper-Bound Concatenation Exponent for Wyner's Modification of the Schalkwijk-Kailath Feedback Scheme, $e_{C2}(r_1)$, to that for the Gaussian Channel with No Feedback, $e_C(r_1)$, where $A_{C2}(r_1) = e_{C2}(r_1)/e_C(r_1)$.

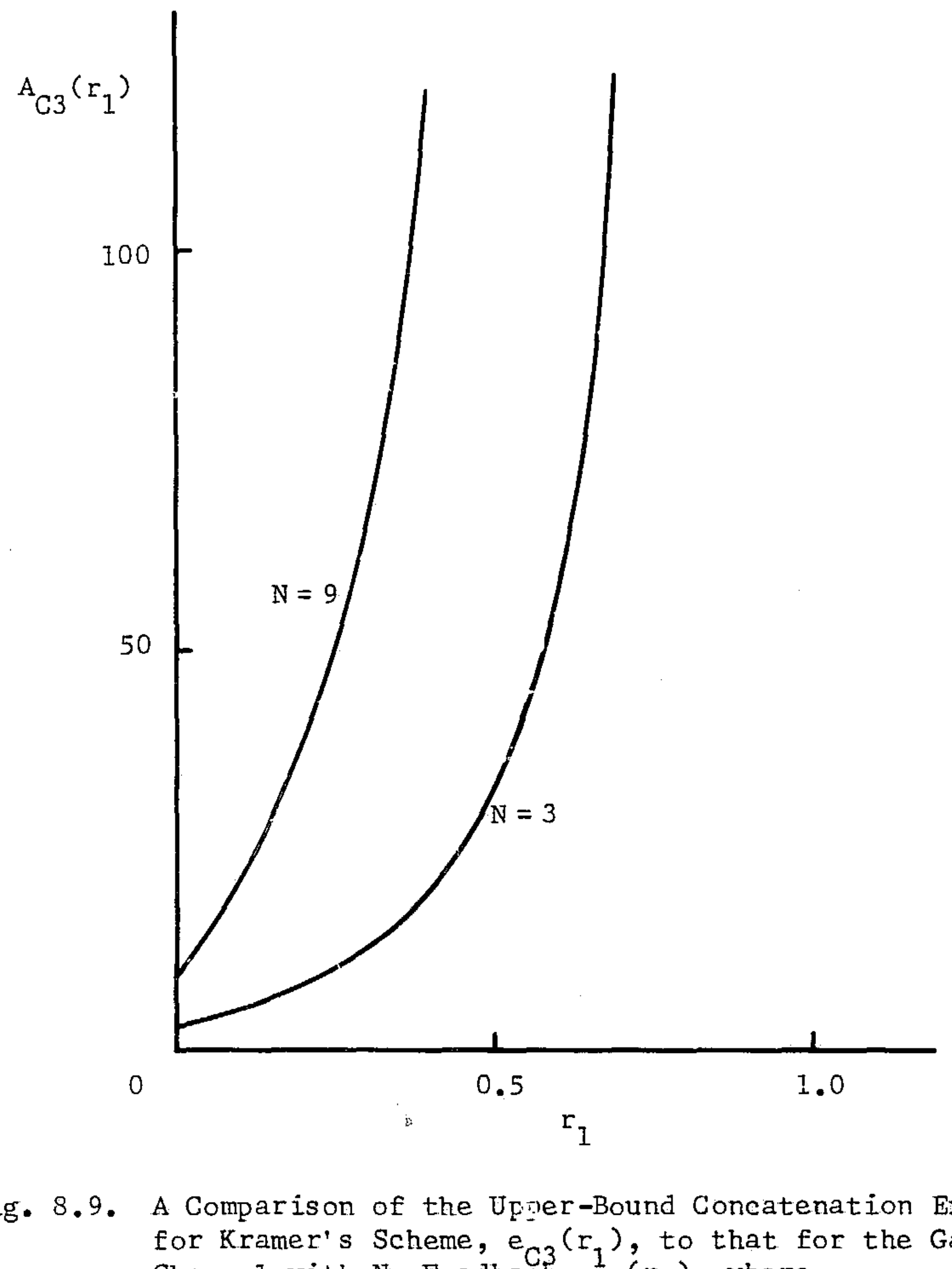

Fig. 8.9. A Comparison of the Upper-Bound Concatenation Exponent for Kramer's Scheme, $e_{C3}(r_1)$, to that for the Gaussian Channel with No Feedback, $e_C(r_1)$, where $A_{C3}(r_1) = e_{C3}(r_1)/e_C(r_1)$.

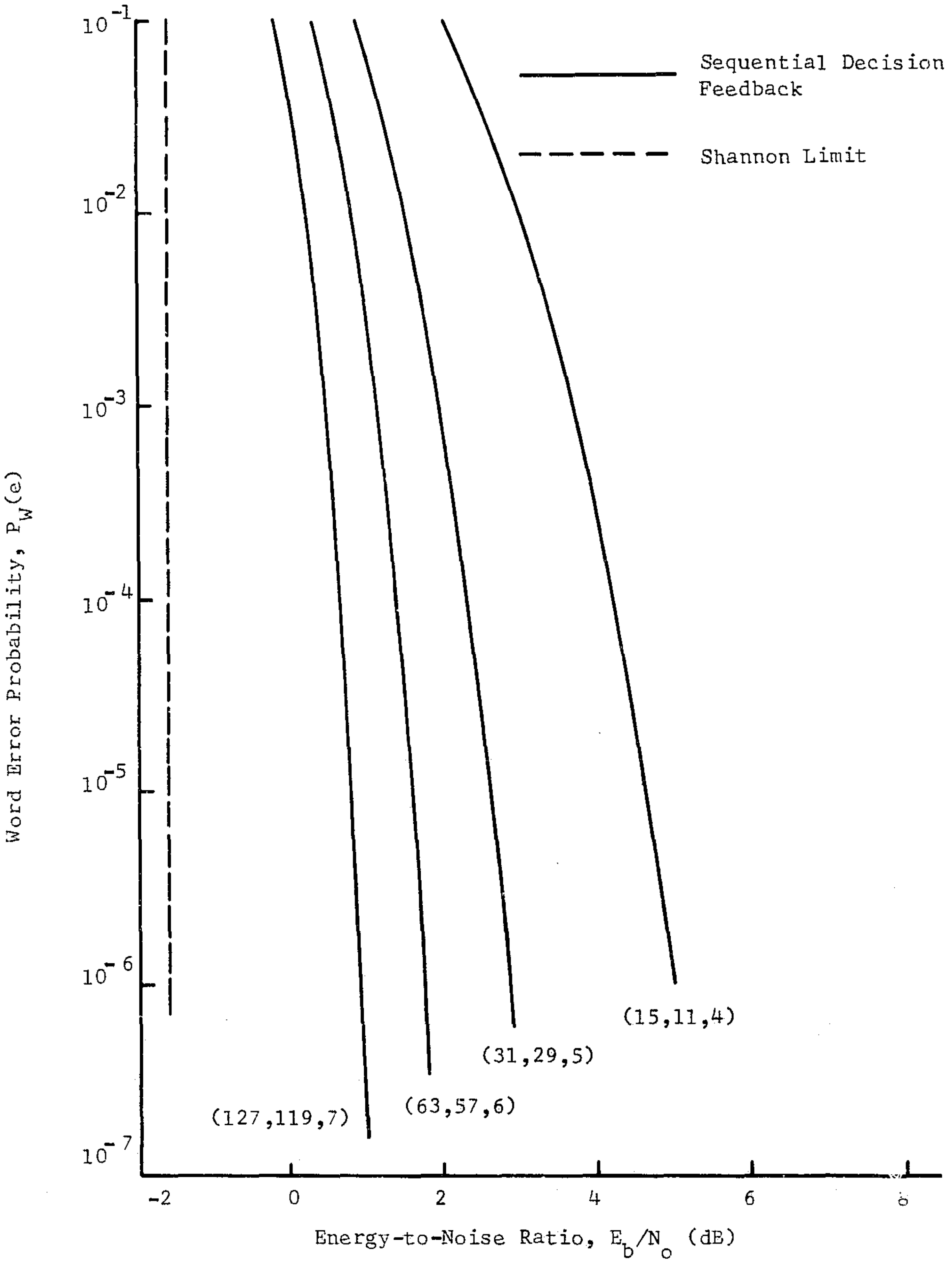

Fig. 8.10. Performance of Several Concatenation Schemes Utilizing Sequential Decision Feedback as the Inner Code.

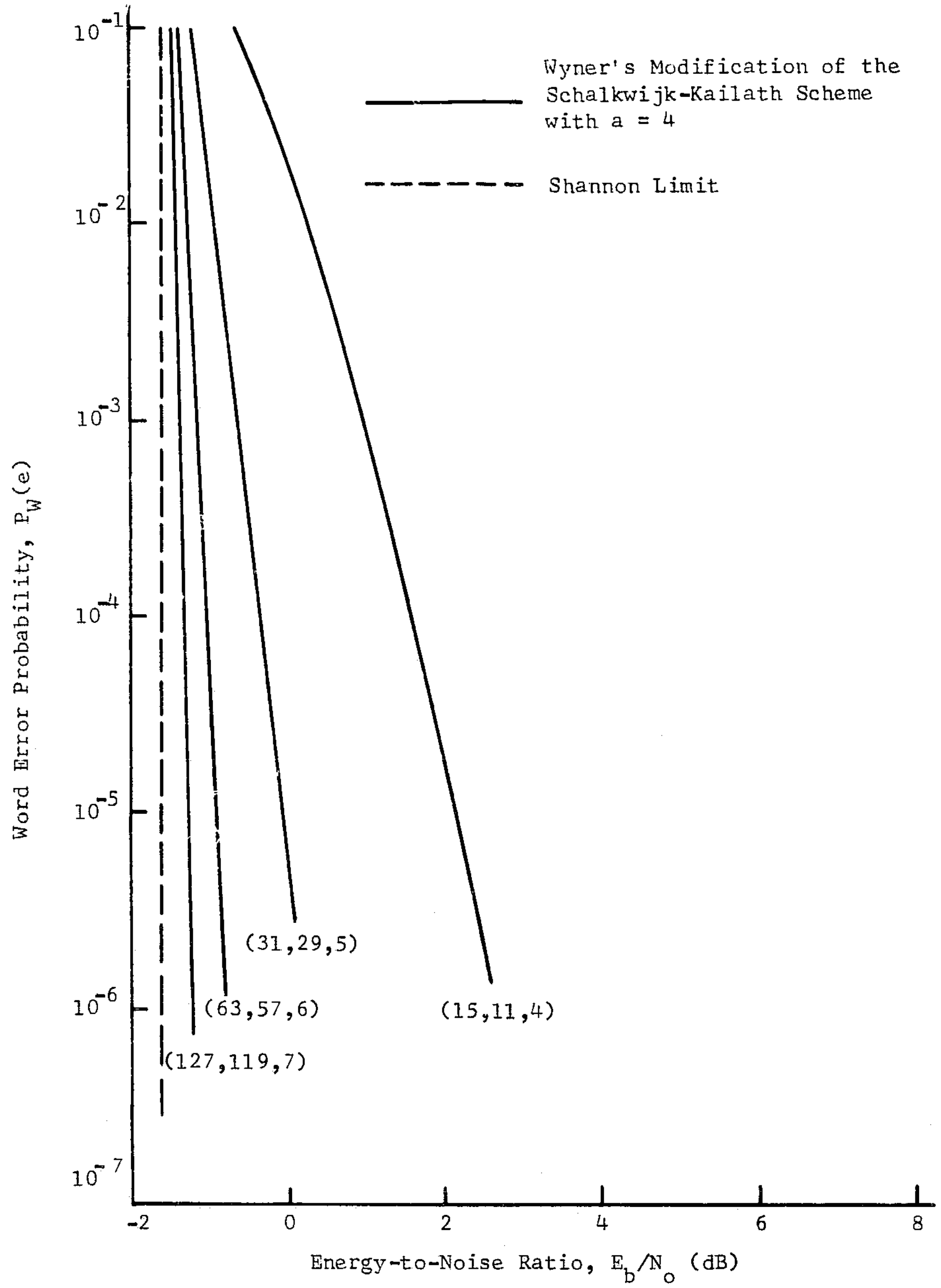

Fig. 8.11. Performance of Several Concatenation Schemes Utilizing Wyner's Modification of the Schalkwijk-Kailath Scheme (a = 4) as the Inner Code.

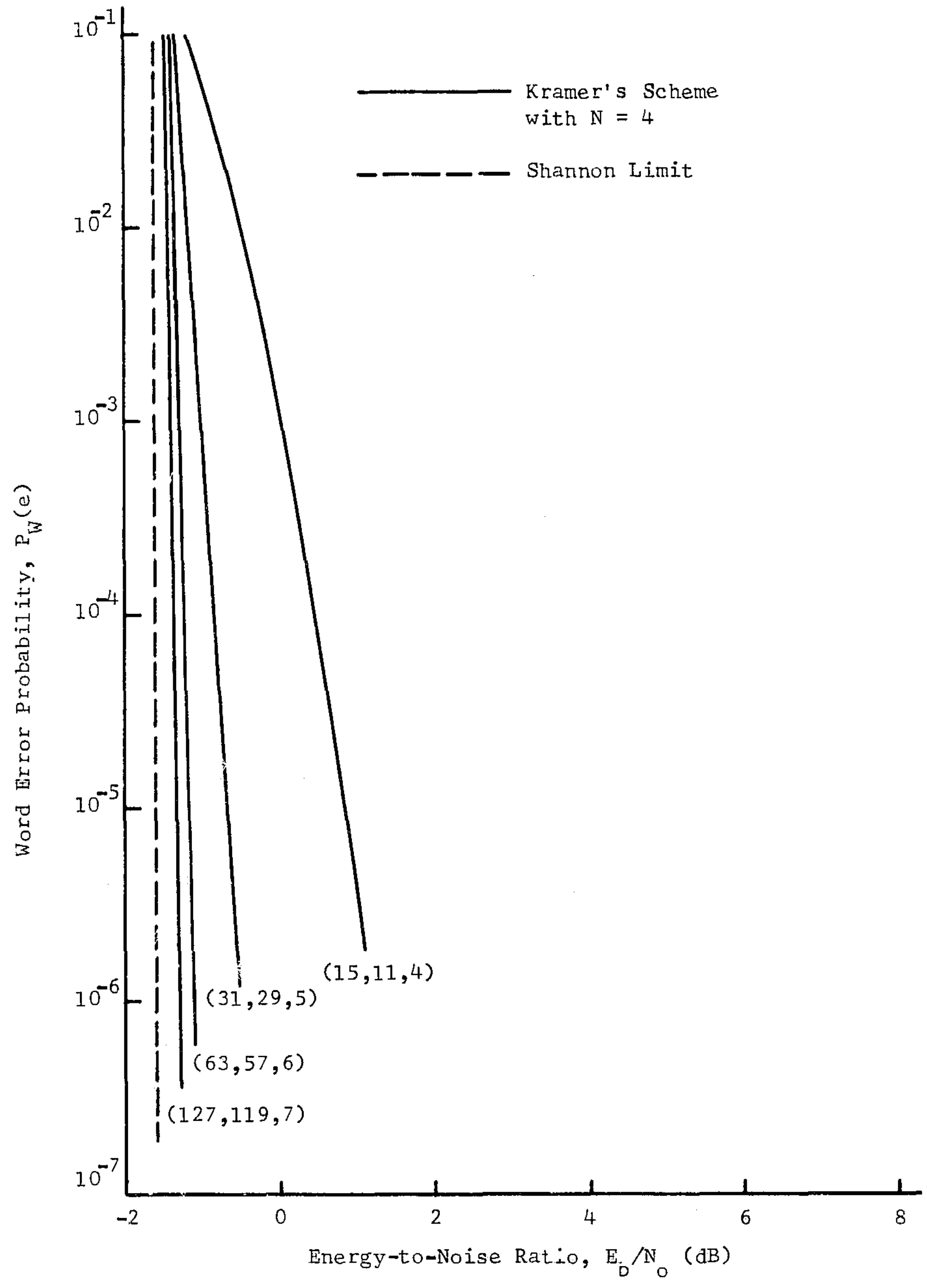

Fig. 8.12. Performance of Several Concatenation Schemes Utilizing Kramer's Feedback Scheme (N = 4) as the Inner Code.

of information bits per outer code word, and m is the number of information bits per inner code word. It should be noted that the two predecision feedback schemes, Wyner's modification of the Schalkwijk-Kailath scheme and Kramer's scheme, have considerably better performance than the postdecision feedback scheme, sequential decision feedback. The most important observation to make in examining Figs. 8.10, 8.11, and 8.12 is that by using concatenation with inner-code feedback, communication at values of E_b/N_c very near Shannon's limit ($E_b/N_o = 1.6$ dB) with a low probability of error is possible. The performance curves shown here are far better than any which have previously been reported in the literature.

Of course, before selecting a scheme for a practical application, the advantages and disadvantages of various schemes should be considered. For two of the feedback schemes presented the decoder complexity increases exponentially with the number of information bits per inner code word, and for the other scheme N increases exponentially with T. However, the complexity of the outer decoder increases only as a small power of the outer code length, n. Thus, in a given situation several tradeoffs will arise between the inner and outer code parameters of the various schemes. Many factors will influence the final choice of inner and outer code parameters including the nature and quality of the feedback channel, the maximum tolerable decoding delay, the limitations on peak-to-average power ratios, etc. Although only the Gaussian channel case has been analyzed, the concept of a concatenation scheme with inner-code feedback should be useful with other channels including channels with and without memory.

CHAPTER IX

BLOCK CODING FOR CHANNELS WITH MEMORY

On many channels errors tend to occur in bursts and hence are not independent. Such a channel is said to have memory. Typical channels which suffer from burst errors are telephone links, high-frequency channels, ionospheric-scatter channels, and tropospheric-scatter channels. Burst-error behavior is caused by such phenomena as fading, multipath, impulse noise, and pulse dropout. The nature of the causes of these errors is such that increasing signal power is not an economical way to reduce the error rate. However, with a moderate decrease in data rate, error control can be used to obtain very low error rates.

A burst-error channel may be thought of as having two states of operation, one which is very good and one which is very bad. A channel based upon this idea has been proposed by Gilbert[69], and a generalization of this channel has been suggested by Elliott[70]. It has been shown that this channel is a good model for certain burst-noise binary channels. In this chapter the performance of several block coding schemes on this channel will be evaluated and compared. The implementation complexity of these schemes will also be discussed. The schemes to be considered include detection-retransmission schemes, burst-error-correcting codes, concatenation schemes, and interleaving schemes.

A. Generalized Gilbert Channel

Gilbert's model of a burst-noise binary channel uses a two-state Markov chain. The generalized Gilbert channel model illustrated in Fig. 9.1 is used to generate noise digits denoted by z_i. The channel

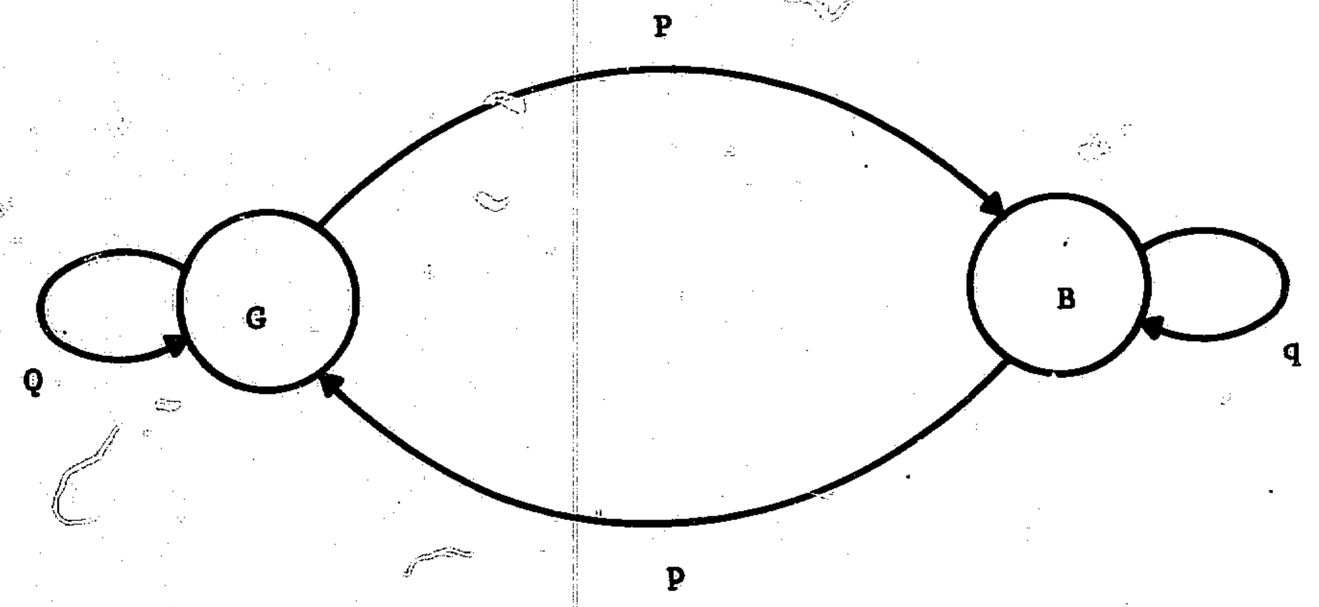

Fig. 9.1. The Generalized Gilbert Channel Model.

has two states called G (for good) and B (for bad). In G the noise digit is $z_i = 0$ with probability k and $z_i = 1$ with probability (1-k). In B the noise digit is $z_i = 0$ with probability h and is $z_i = 1$ with probability (1-h). Naturally, for the good state to be significantly better than the bad state, the relationship $(1-k) << (1-h)$ must hold. The only difference between the generalized model and the original model introduced by Gilbert is that Gilbert did not allow errors to occur in G. It is indicated in Fig. 9.1 that the transition probabilities between states are $P = P(G \rightarrow B)$ and $p = P(B \rightarrow G)$. Of course $Q = 1-P$ and $q = 1-p$ are the probabilities of remaining in G and B, respectively.

A burst of length m in a sequence of length n means that all non-zero bits in the sequence of n bits are contained in a sequence of length m bits having first and last bits nonzero, i.e.,

$$z = (0^K 1 \cdots 1 0^J), \qquad (9.1)$$

where 0^L represents a sequence of L consecutive zeros, $K = 0,1,\cdots, n-m$, and $J = n-m-K$. In evaluating the performance of block codes on this channel, the probability, $P_b(m,n)$, of a burst of length m in a sequence of length n will be useful. This probability is determined in Appendix E for the generalized Gilbert channel.

Experimental noise statistics for an actual channel may be matched to a Gilbert channel model by using the conditional probability, u(K), of a run of K or more zeros following a one, i.e.,

$$u(K) = P(0^K \mid 1) . \qquad (9.2)$$

The technique of matching experimental noise statistics to Gilbert channel parameters is discussed in Gilbert[71] and will not be covered here.

In order to effectively use coding on a burst-error channel, one must have considerable knowledge of the error behavior of the channel. For example, if a code capable of correcting all bursts of length c or less is used on a channel which has a significant percentage of its bursts of length greater than c, then very little improvement can be obtained by using this code. In many cases a feedback channel will be necessary, particularly on channels which commonly have very long error bursts. The probabilities, $P_b(m,n)$, $0 \leq m \leq n$, will be helpful in selecting good codes.

B. Detection-Retransmission Schemes

Cyclic codes are well suited for error detection because they can be designed for a wide range of rates and error detection capabilities. The burst-error-detecting performance of cyclic codes is readily evaluated by means of a theorem due to Brown and Peterson which will be stated without proof.[72]

Theorem 9.1. Every (n,k) cyclic code can detect any burst of length (n-k) or less, and the fraction of bursts of length $m > n-k$ that are undetected is $2^{-(n-k-1)}$ if $m = n-k+1$ and $2^{-(n-k)}$ if $m > n-k+1$.

If $\alpha(m,n,k)$ is defined as the fraction of bursts of length m which are undetected by an (n,k) cyclic code, then

$$\alpha(m,n,k) = \begin{cases} 0 & , \ 0 \leq m \leq n-k \ , \\ 2^{-(n-k-1)} & , \ m = n-k+1 \ , \\ 2^{-(n-k)} & , \ n-k+1 \leq m \leq n \ . \end{cases} \tag{9.3}$$

The first system to be considered utilizes a cyclic (n,k) code for error detection in communication over the Gilbert channel. If all bursts of the same length are assumed equally likely, the probability of erroneous decoding is

$$P_e = \sum_{m=0}^{n} \alpha(m,n,k)P_b(m,n) \ , \qquad (9.4)$$

where $\alpha(m,n,k)$ is given by (9.3) and $P_b(m,n)$ is given by (E.38) and (E.39). The probability of correct decoding is

$$P_c = P_b(0,n) \ , \qquad (9.5)$$

and the probability of detecting an error P_d is therefore

$$P_d = 1 - P_c - P_e \ . \qquad (9.6)$$

If this system is modified to utilize a noiseless, delayless, feedback channel, the situation is more complicated than that analyzed in Chapter VIII because the channel has memory. The system under consideration uses a cyclic (n,k) code for error detection and requests a retransmission via the feedback channel when errors are detected. When a code word is transmitted, the probability of an undetected error and the probability of correct decoding are given by (9.4) and (9.5), respectively. The probability of a retransmission is simply the probability of detecting an error which is given by (9.6). If a retransmission is required, the noise digits from the channel will be dependent upon the state of the channel at the beginning of the retransmission. The worst case is for the channel to be in B and since there were errors in the previous code, this is very likely. The probability of error will be upper bounded by assuming that the channel is in B at the beginning of all retransmissions.

The probability $u_1(K) = P(0^K \mid B)$ satisfies the recurrence relation (E.18) with initial values

$$u_1(0) = 1 \tag{9.7}$$

and

$$u_1(1) = gh + pk \quad . \tag{9.8}$$

Defining $v_1(K) = P(0^K 1 \mid B)$, $v_1(K)$ can be determined from $u_1(K)$ and $u_1(K+1)$ with the aid of (E.23). The probability distribution, $P_{b1}(m,n)$, of the burst lengths given that the channel is originally in B is then

$$P_{b1}(m,n) = \sum_{K=0}^{n-m} v_1(K) \frac{r(m-1)}{P(1)} u(n-m-k) \tag{9.9}$$

for m = 1, 2, $\cdots$, n, and

$$P_{b1}(0,n) = u_1(n) \quad . \tag{9.10}$$

Thus, the probability of undetected error for a retransmission is bounded by

$$P_{e1} \leq \sum_{m=0}^{n} \alpha(m,n,k) P_{b1}(m,n) \quad , \tag{9.11}$$

and the probability of detecting an error on a retransmission is

$$P_{d1} = 1 - P_{b1}(0,n) - P_{e1} \quad . \tag{9.12}$$

The over-all probability of word error for the system is bounded by

$$\begin{aligned} P_W(e) &\leq (1-P_d)P_e + P_d(1-P_{d1})P_{e1} + P_d P_{d1}(1-P_{d1})P_{e1} + \cdots \\ &= (1-P_d)P_e + P_d P_{e1}(1-P_{d1}) \sum_{i=0}^{\infty} P_{d1}{}^i \\ &= (1-P_d)P_e + P_d P_{e1} \quad . \end{aligned} \tag{9.13}$$

The probability, P_e, given by (9.4) is a lower bound to $P_W(e)$. Computation of (9.4) and (9.13) for a number of codes and channel parameters indicates that the bounds are close. In most cases (9.4) and (9.13) differed by about an order by magnitude.

The average number of retransmissions per decoded word for this system is bounded by

$$\begin{aligned} T &\leq (1-P_d) + 2P_d(1-P_{d1}) + 3P_dP_{d1}(1-P_{d1}) + \cdots \\ &= (1-P_d) + \frac{P_d(1-P_{d1})}{P_{d1}^2} \sum_{i=2}^{\infty} i\, P_{d1}^{\,i} \\ &= 1 + \frac{P_d}{1-P_{d1}} \; . \end{aligned} \tag{9.14}$$

A lower bound to T is simply

$$T \geq \frac{1}{1-P_d} \, . \tag{9.15}$$

Since retransmissions are required, the actual information rate will be less than the rate of the (n,k) code. A measure of over-all efficiency of communication is the "throughput" which is the ratio of the number of information bits delivered to the total number of bits transmitted. Then the throughput of a detection-retransmission system utilizing an (n,k) code is

$$T'PUT = \frac{k}{Tn} \; . \tag{9.16}$$

Upper and lower bounds to the throughput may be obtained by applying (9.14) and (9.15) to (9.16).

Estimation of the complexity of implementing this detection-retransmission system is straightforward. The encoder could be implemented with either a k-stage or an (n-k)-stage shift register. Since generally high-rate codes are chosen, the (n-k)-stage encoder would be used. The decoder is implemented with an (n-k)-stage syndrome register and two data registers each consisting of k stages. A total of 2n binary storage elements would therefore be needed. Also, a buffer will be needed at the encoder since data is received by the encoder at a constant rate, and sometimes retransmissions will be needed. The size of this buffer is dependent upon the code and the channel. If the channel frequently has very long bursts, a large buffer will be required.

C. Burst-Error-Correcting Codes

In the preceding chapters error-correcting coding schemes have been discussed which were constructed primarily for correcting random error patterns. However, codes can also be constructed for the correction of errors which occur in bursts. A few good burst-error correcting codes have been found analytically, but the most useful codes have been generated by computer search techniques. This is in contrast with the analytical development of the best random-error-correcting codes.

The capability of linear codes for burst-error correction is stated in the following theorem.[73]

Theorem 9.2. In order to correct all burst errors of length c or less, a linear code must have at least 2c parity check symbols, i.e.,

$$c \leq \frac{n-k}{2} . \qquad (9.17)$$

The most useful burst-error-correcting codes are cyclic because they are so easily implemented. One of the most well-known classes of analytically-constructed burst-error-correcting codes is the class of

Fire codes.[74]

Theorem 9.3. A Fire code is generated by the polynomial

$$g(X) = p(X)(X^{2c-1}+1) \ , \tag{9.18}$$

where p(X) is primitive, has degree $m \geq c$, and has length

$$n = \mathrm{LCM}\left[2^m-1, 2c-1\right] \ . \tag{9.19}$$

This code is capable of correcting any single burst of length c or less.

Let σ be defined by

$$\sigma = n-k-2c \ . \tag{9.20}$$

From Theorem 9.2,

$$\sigma \geq 0 \ . \tag{9.21}$$

The best burst-error-correcting codes have $\sigma = 0$, and a large value of σ implies an inefficient code. For the well-known Fire codes

$$n-k = m-2c-1 \ , \tag{9.22}$$

and hence

$$\sigma = m-1 \ , \tag{9.23}$$

where $m \geq c$. Thus, the Fire codes are inefficient, particularly for large values of burst-correction capability c. This is also true for most analytically-constructed burst-error-correcting codes.

The best burst-error-correcting codes have been found by trial-and-error search procedures. A number of good codes have been given by Elspas[75] and Kasami[76]. In these cases generator polynomials were found which generated good burst-error-correcting codes.

Implementation of a burst-error-correcting cyclic code is straightforward. For encoding, an (n-k)-stage feedback shift register will generally be used, since high-rate codes are desirable. A block diagram of a decoder for a burst-error-correcting cyclic code is shown in Fig. 9.2. The syndrome is calculated by shifting the received vector, R(X), into a device identical to the encoder. At the same time the received vector is stored in the buffer. The logic unit tests the contents of the shift register for a correctable error pattern. If the code is capable of correcting all bursts of length c or less, a correctable error pattern is recognized when the leftmost n-k-c stages of the register are all zero. As long as no correctable error pattern is recognized, the register is shifted with no input, and symbols are read simultaneously out of the buffer. When a correctable pattern is recognized by the logic unit, the gate is opened. Symbols coming out of the shift register are then added to the symbols coming out of the buffer. The rest of the message is then read out of the buffer. If the syndrome is non-zero, and if no correctable error pattern occurs by the time the entire received vector is read out of the buffer, an uncorrectable error has been detected. This decoder is very simple and requires very little control circuitry. After the transmitted vector is received, the only operations that are required are n shifts of a shift register. At most 2n binary storage elements are required.

In some cases because bursts occur too often or for other reasons, it is desirable to shorten a burst-error-correcting cyclic code. A code may be shortened by setting some of the higher-order information symbols to zero and thus omit them. Although the encoding and parity-check

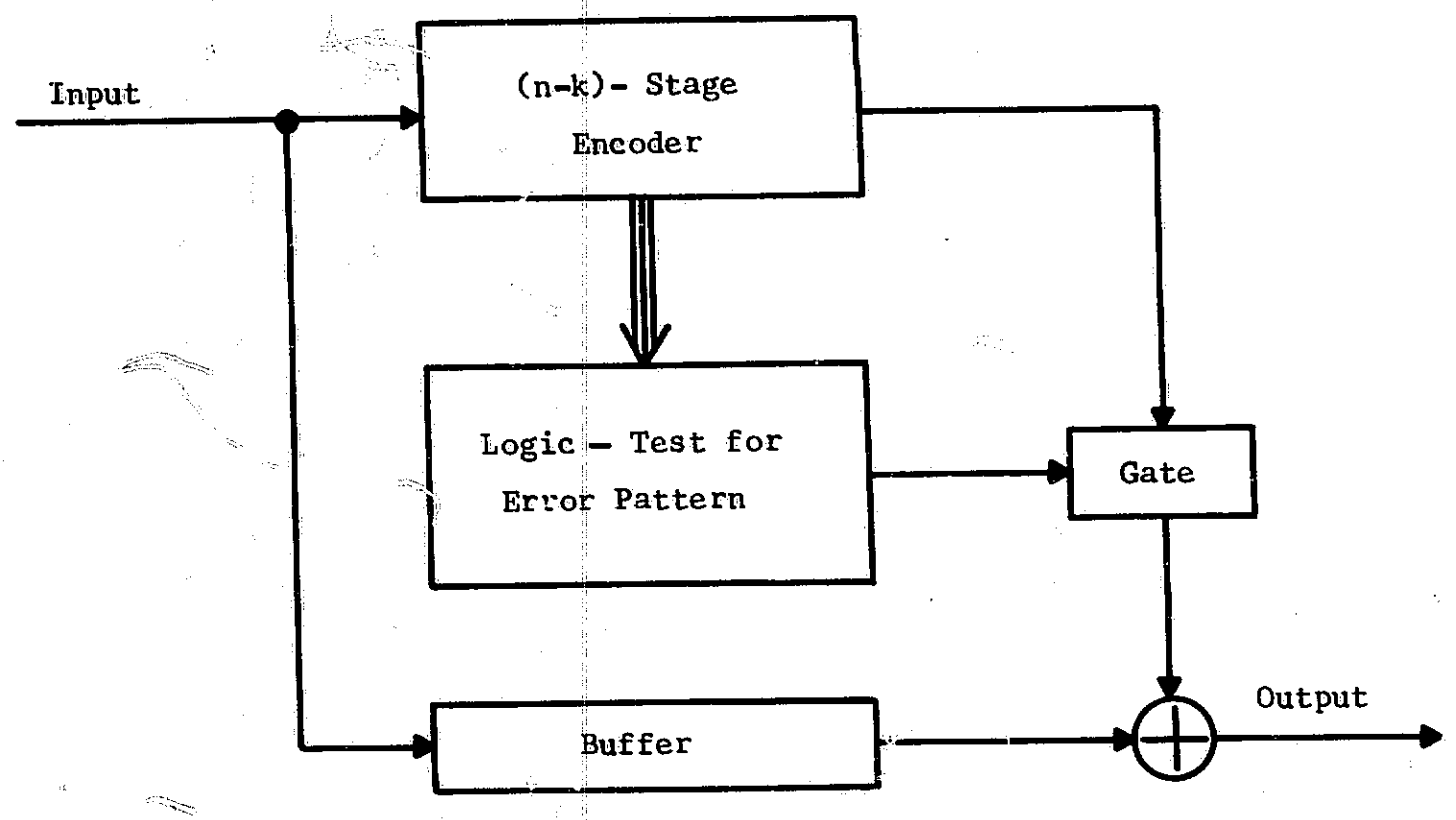

Fig. 9.2. A Decoder for a Burst-Error-Correcting Cyclic Code.

calculations are not affected by the leading zeros, the decoding procedure is affected. Suppose it is desired to use a code with natural length n but with r of the information symbols omitted. In decoding, r shifts corresponding to the omitted information symbols would be required before the actual received vector could be shifted out of the buffer. This procedure could be simplified by an automatic premultiplication by X^r modulo g(X). For the full-length code the parity-check calculation is the residue of $X^{n-k}R(X)$ modulo g(X). Thus, for the shortened code the parity check calculation is the residue of $X^{n-k+r}R(X)$ modulo g(X). Letting n(X) be the remainder after dividing X^{n-k+r} by g(X),

$$n(X) = X^{n-k-r} \mathrm{mod}\ g(X) \ . \tag{9.24}$$

The parity check calculation for the shortened code can be implemented with a circuit of the form illustrated in Fig. 2.6 for multiplying R(X) by n(X) and dividing by g(X). The error correction is performed in the same manner as that for a full-length code. The circuit for decoding a shortened cyclic code is similar to that of Fig. 9.2 with the (n-k)-stage encoder replaced by a circuit for multiplying R(X) by n(X) and dividing by g(X).

The performance of these codes on the Gilbert channel is readily calculated using the distribution of burst lengths derived in Appendix E. If an (n,k) code is capable of correcting all bursts of length c or less, then the probability of word error after decoding is

$$P_W(e) = \sum_{m=c+1}^{n} P_b(m,n) \quad , \tag{9.25}$$

where $P_b(m,n)$ is given by (E.38) and (E.39). When using these codes, the throughput is simply

$$\text{T'PUT} = \frac{k}{n} \ . \tag{9.26}$$

D. Concatenated Codes

On channels where very long bursts frequently occur a code with a very long block length must be used in order to obtain a significant improvement in the probability of error. The technique of concatenation was introduced in Chapter VI as a means of implementing codes with very long over-all block lengths with reasonable complexity. This technique shows much promise for burst-error correction.

Since a burst will generally cause many errors in each inner code word in which it occurs, the inner code will be used only for error detection. Cyclic codes such as BCH codes are well suited for error detection and will be used as inner codes. The error-detecting properties of these codes are given in Theorem 9.1. Reed-Solomon codes will be used as outer codes. The inner decoder will be a replica of the inner coder for calculating the syndrome. If errors are detected, that symbol will be considered an erasure. If no errors are detected, the symbol is assumed correct. Thus, the outer decoder must be capable of correcting erasures as well as errors.

With certain modifications either Berlekamp's of Peterson's method could be used in correcting erasures and errors. The complexity of such a decoder has been analyzed in the same manner as were the BCH and RS decoders of Chapters III and VI. The inner-code parameters are $(N,K) = (N,m)$, where $GF(2^m)$ is the locator field for the outer code, and the outer-code parameters are (n,k). An RS decoder employing the Berlekamp algorithm for decoding erasures and errors requires a number of computations equal to

$$N_C = 2n + 3d^2m + 2d^2 + 4dm + 2mt + 2m^2 + 11d + 6t - 9m \quad , \qquad (9.27)$$

where the minimum distance of the outer code is $d = n-k+1$. The total number of binary storage elements required in such a decoder is

$$N_{SD1} = 2nm + m^2t + 3md + 9mt + 18m + 6 \quad . \tag{9.28}$$

Since only error detection is performed by the inner decoder, all that is required is a syndrome register with

$$N_{SD2} = N - m \tag{9.29}$$

binary storage elements.

A useful upper bound to the probability of error of this system on a Gilbert channel using the distribution of burst lengths of Appendix E has not yet been found. However, this probability can be determined by simulation. Some simulation results will be presented later. The throughput of this scheme is

$$T'PUT = \frac{Kk}{Nn} \quad . \tag{9.30}$$

The performance of this concatenation scheme can be improved considerably if a feedback channel is available. Since the use of outer-code feedback would require a very large buffer, only systems utilizing inner-code feedback will be considered. The inner code to be used is the simple detection-retransmission scheme described in Part B of this chapter. Since a retransmission is requested every time the inner decoder detects an error, the outer decoder performs error correction only. The complexity of the inner decoder is discussed in Part B of this chapter, and the complexity of the outer decoder is discussed in Chapter VI.

Bounds on the performance of concatenation with inner-code feedback may be calculated from the results of Part B of this chapter. A cyclic inner code is assumed. The effect of channel memory is removed by assuming the channel is in B prior to the transmission of each inner code word. If an (n_1,k_1) Reed-Solomon outer code capable of correcting t errors is implemented with an errors-only decoder, the probability of word error is upper bounded by

$$P_W(e) \leq \sum_{i=t+1}^{n_1} \binom{n_1}{i} P_{e1}^{\ i} \left[1 - P_{e1}\right]^{n_1 - i} \ , \tag{9.31}$$

where P_{e1} is given by (9.11). The probability of word error may be lower bounded by replacing P_{e1} in this equation with P_e of (9.4). Upper and lower bounds to the throughput may be obtained by multiplying the upper and lower bounds to the throughput of Part B of this chapter by k_1/n_1.

E. Interleaved BCH Codes

If a relatively short random-error-correcting BCH code is used on a burst-error channel, there would be no significant improvement in performance since when a burst occurs the number of errors would exceed the capability of the code. However, if interleaving is used, random-error-correcting codes can offer an improvement in performance. By definition, with an amount of interleaving, I, the successive bits in each code word are separated in the channel bit stream by I-1 bits from I-1 other code words. By using a large amount of interleaving, the errors which occur with each code word become randomized, i.e., the effects of channel memory are less pronounced. With the proper choice of BCH code and amount of interleaving, I, much improvement in performance on burst-error channels can be obtained. If an (n,k), t-error-correcting, BCH code is used with an amount of interleaving I, the interleaved code has an over-all constraint length of nI and is capable of correcting all bursts of lenth up to tI in addition to many other error patterns.

Encoders and decoders for binary BCH codes have been discussed in Chapter III. The only other implementation problem is that of interleaving.

Studies have shown that interleaving is relatively easy to implement and that the decoder complexity is strongly dependent upon the complexity of the BCH decoder.[77] It requires considerable storage to implement interleaving, but very little control circuitry is required. For this reason, the shortest code is generally selected that can (with a reasonable amount of interleaving) give the desired performance.

No method has yet been found for analytically computing the performance of interleaved codes on the Gilbert channel. However, if an (n,k) t-error-correcting BCH code is interleaved by an amount large enough to effectively randomize the errors, the probability of word error can be approximated by

$$P_W(e) \approx \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} , \qquad (9.32)$$

where p is average bit error rate of the channel. This approximation represents an infinite amount of interleaving, i.e., all effects of channel memory have been removed. The closeness of this approximation can be determined for a given channel, code, and amount of interleaving only by extensive study.

F. Comparisons

For detection-retransmission, burst-error correction, and concatenation with inner-code feedback, estimates of performance on the Gilbert channel can be obtained by using the distribution of burst lengths derived in Appendix E. In this section the performance of these schemes on the Gilbert channel will be compared. Cohn and others have made extensive studies of performance of interleaved BCH codes and concatenation with inner-code error detection using raw error data from a fading, high-frequency channel.[78] The results of these simulations will also be discussed in this section. The Gilbert channel to be considered is specified by the parameters $P = 10^{-6}$, $p = .005$, $h = .70$, and $k = 1$. From (E.13) the probability of bit error on this channel for uncoded transmission is $P_B(e) = 6.0 \times 10^{-4}$. An upper bound on the performance of a detection-retransmission scheme for this channel is given by (9.13). In Table 9.1 an upper bound on the performance and a lower bound on the throughput of several cyclic codes, used in a detection-retransmission scheme are shown. Most of these codes are BCH codes.

Table 9.1. An Upper Bound on the Performance and a Lower Bound on the Throughput of Several Cyclic Codes Used in a Detection-Retransmission Scheme for the Gilbert Channel Specified by $P = 10^{-6}$, $p = .005$, $h = .70$, and $k = 1$.

n	k	T'PUT	$P_W(e)$
7	4	.571	2.71×10^{-5}
12	6	.497	4.61×10^{-6}
14	8	.568	5.27×10^{-6}
15	11	.731	2.29×10^{-5}
15	7	.463	1.25×10^{-6}
15	5	.330	2.59×10^{-7}
31	26	.835	1.31×10^{-5}
31	21	.669	4.00×10^{-7}
31	16	.509	1.20×10^{-8}
63	51	.797	1.13×10^{-7}
63	45	.703	1.70×10^{-9}
63	39	.610	2.54×10^{-11}
127	113	.873	3.50×10^{-8}
127	106	.819	2.61×10^{-10}

The improvement in error probability obtained by the use of these schemes is from one to seven orders of magnitude.

The performance and throughput of several burst-error-correcting codes on the Gilbert channel are shown in Table 9.2.

Table 9.2. Performance and Throughput of Several Burst-Error-Correcting Codes on the Gilbert Channel Specified by $P = 10^{-6}$, $p = .005$, $h = .70$, and $k = 1$.

n	k	c	T'PUT	$P_{r}(e)$
15	5	5	.333	$1.$[illegible] $\times 10^{-4}$
21	12	4	.571	2.00×10^{-4}
21	9	6	.428	1.94×10^{-4}
21	5	8	.238	1.86×10^{-4}
21	3	9	.143	1.82×10^{-4}
27	17	5	.630	2.06×10^{-4}
31	20	5	.645	2.10×10^{-4}
38	24	7	.632	2.13×10^{-4}
43	28	5	.651	2.21×10^{-4}
46	24	10	.522	2.14×10^{-4}
50	30	10	.600	2.18×10^{-4}
55	35	9	.636	2.24×10^{-4}
63	43	9	.683	2.32×10^{-4}
75	55	5	.734	2.52×10^{-4}
92	48	20	.522	2.35×10^{-4}

The codes in this table have been given by Elspas[79] and Kasami[80]. Although an improvement in performance is obtained with each code, the improvement is less than an order of magnitude.

An upper bound on the performance and a lower bound on the throughput for several concatenation schemes with inner-code feedback are given in Table 9.3.

Table 9.3. An Upper Bound on the Performance and a Lower Bound on the Throughput of Several Concatenation Schemes with Inner-Code Feedback for the Gilbert Channel Specified by $P = 10^{-6}$, $p = .005$, $h = .70$, and $k = 1$.

n	k	N	K	T'PUT	$P_W(e)$
31	25	15	5	.268	2.14×10^{-10}
31	23	15	5	.245	2.97×10^{-13}
31	21	15	5	.224	3.04×10^{-16}
63	51	12	6	.403	1.85×10^{-11}
127	117	15	7	.427	3.73×10^{-12}
85	71	14	8	.474	2.10×10^{-12}
51	37	14	8	.412	2.10×10^{-12}
51	33	14	8	.368	1.07×10^{-16}
511	491	14	9	.615	1.75×10^{-15}
73	53	14	9	.465	1.75×10^{-15}

The improvement in error probability obtained by the use of these schemes is from six to twelve orders of magnitude.

Cohn and others have performed simulations using raw error data from fading, high-frequency channels with interleaved-binary BCH codes and concatenated codes with inner-code error detection.[81] The error data was such that bursts thousands of bits long were not uncommon. The results show that the performance of the two schemes is similar. Improvements in performance of several orders of magnitude can be obtained by using over-all constraint lengths of 40,000 to 60,000 bits. The long constraint lengths are required since for these schemes no feedback channel is used. It is essential that any burst-error coding scheme which does not utilize a feedback channel have a constraint length sufficiently long to span the duration of the longest expected error burst.

The failure of the burst-error-correcting codes of Table 9.2 to provide much improvement in performance is due to the relatively short block lengths used. A large percentage of the bursts were longer than the error-correction capability of the codes. Calculations of error probability have shown that for values of p near 0.5 these codes do provide significant improvements in performance since the bursts which are generated are relatively short. It is not practical to attempt to find very long, efficient burst-error-correcting codes through trial-and-error techniques. Such codes can be generated by using short, efficient, burst-error-correcting codes with an amount of interleaving necessary to achieve the desired length.

The detection-retransmission schemes and the concatenation schemes with inner-code feedback both offer significant improvements in performance. The improvement in performance due to the use of concatenation over that of the simple detection-retransmission scheme is not as much as one might expect considering the increase in system complexity which is required. This and the fact that the channel was assumed in State B at the beginning of each inner code word in order to obtain the upper bound on performance shown in Table 9.3 suggests that the actual performance of the concatenation scheme is closer to the lower bound. It is possible that further study could verify this conjecture.

A number of coding schemes for correction of burst errors have been suggested in this chapter. Essentially they can be classified into two categories, schemes which utilize a feedback channel and schemes which do not utilize a feedback channel. The basic strategy of the feedback schemes is to use the channel when it is in State G to avoid use of the channel when it is in State B. In this case much improvement in performance can be obtained with little coding effort. If no feedback channel is available there are two problems. First, the channel must be used when it is in State B, and second, the code must have a block length sufficiently long to span the duration of the longest expected error burst. Thus, in a practical application where no feedback channel is available, codes with very long block lengths, say 50,000 bits, may be required.

CHAPTER X
CONCLUSIONS

In this work extensive analyses of performance and implementation complexity of most of the major coding schemes have been performed. In addition, a new coding scheme, concatenation with inner-code feedback, has been proposed, and its performance has been analyzed. The major portion of this work has been devoted to coding techniques for correction of independent errors. The Gaussian channel has been used in evaluating the performance of these techniques. A chapter has also been included in which coding techniques for channels with memory are discussed. In this case the Gilbert channel model was used in performance evaluations.

Generally, in selecting a suitable coding scheme for the Gaussian channel, it is desirable to minimize E_b/N_o for a given word error probability, $P_W(e)$, provided that certain constraints on encoder and decoder complexity are satisfied. The results of the preceding chapters can be applied in a comparison of block coding schemes for the Gaussian channel. As an example, the complexity of several coding schemes will be evaluated as a function of the E_b/N_o required for $P_W(e) = 10^{-5}$. The schemes to be considered are BCH codes, bi-orthogonal codes, threshold-decodable codes, and concatenated codes. For each of these schemes curves of $P_W(e)$ vs. E_b/N_o for several codes have been presented in the preceding chapters. Also, formulas have been given for various measures of implementation complexity for these codes. Each point on the curves to be presented is obtained by determining the value of E_b/N_o required by a particular code to obtain $P_W(e) = 10^{-5}$ and the measure of complexity for that code.

The points on each curve are connected only for clarity.

For the BCH codes, bi-orthogonal codes, and threshold-decodable codes, the $P_W(e)$ vs. E_b/N_o curves are shown in Figs. 3.8, 4.1, and 5.4, respectively. For the concatenated codes, curves of $P_W(e)$ vs. E_b/N_o are shown in Figs. 6.7-6.11. Each figure contains two curves, one for a high-rate outer code and the other for a medium-rate outer code.

Several measures of implementation complexity will be considered here including the number of computations per decoded information bit and the number of binary storage elements required in both the encoder and the decoder. It will generally be observed in comparing the complexity of coding schemes that some schemes will appear more attractive than others using one measure of complexity and appear less attractive using other measures of complexity. The specific application will determine which measure of complexity should be the deciding factor.

It will be assumed that a Berlekamp decoder is used in decoding BCH and Reed-Solomon codes. The total number of computations required in decoding a BCH code word, a bi-orthogonal code word, a threshold-decodable code word, or a concatenated code word is given by (3.22), (4.41), (5.26), or (6.30), respectively. The total number of binary storage elements required by a BCH decoder, a bi-orthogonal decoder, a threshold decoder, or a concatenated decoder is given by (3.23), (4.42), (5.24), or (6.31), respectively. The total number of binary storage elements required by the corresponding encoders is given by (3.55), (4.43), (5.27), and (6.32).

The number of computations per decoded information bit as a function of E_b/N_o for the four coding schemes is shown in Fig. 10.1. In this figure and in the next two figures there are two curves for concatenated codes, one for a high-rate outer code and the other for a medium-rate outer code. From Fig. 10.1 it is obvious that for medium values of E_b/N_o (6 to 9 dB) the threshold-decodable codes are much easier to implement than the other schemes, using the number of computations per decoded information bit as the criterion. Over this range of E_b/N_o implementation of BCH codes requires four to seven times as many computations per decoded information bit. The bi-orthogonal codes require more computations per decoded information bit than the BCH codes, and the difference increases as E_b/N_o decreases. For values of E_b/N_o less than 6 dB the difference is several orders of magnitude. Concatenated codes are useful over a different range of E_b/N_o (2 to 5 dB). Operation at these low values of E_b/N_o is not readily achieved using the other techniques. The high-rate concatenated codes are slightly less efficient than the medium-rate codes, since they require more computations per decoded information bit at a fixed value of E_b/N_o. There is some overlap in the curves in the vicinity of E_b/N_o = 5.5 dB. In this region the concatenated codes appear to be less attractive than the BCH and threshold-decodable codes, but more attractive than the bi-orthogonal codes.

The number of binary storage elements required in the decoder as a function of E_b/N_o for the four coding schemes is shown in Fig. 10.2. In this case the pronounced differences between the various coding schemes that were observed in Fig. 10.1 do not exist. Threshold decoding is attractive for values of E_b/N_o from 7 to 9 dB. In this range of E_b/N_o it would be the most easily implemented scheme since very little control

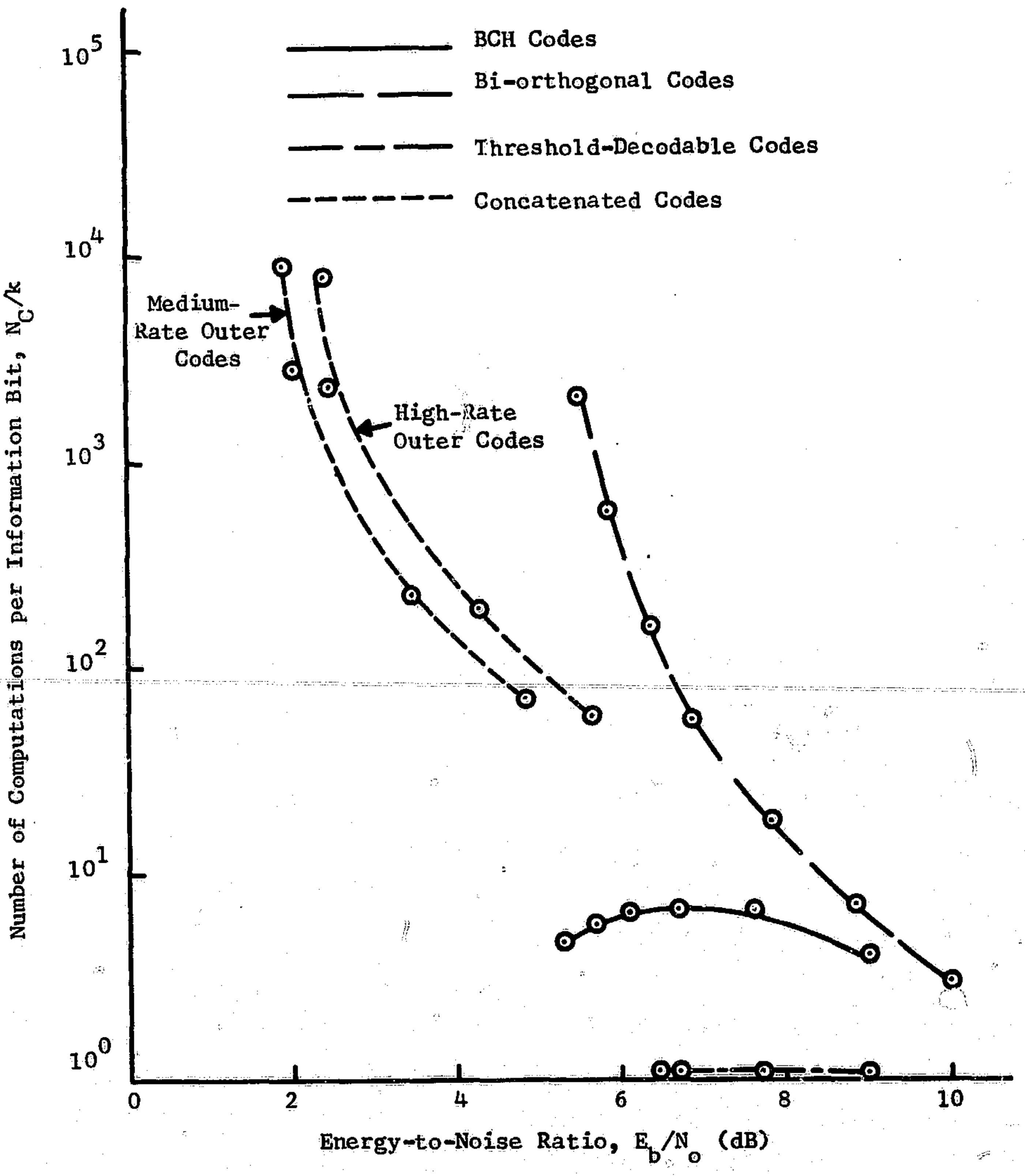

Fig. 10.1. Number of Computations per Information Bit Required in Decoding Several Classes of Codes at a $P_W(e) = 10^{-5}$.

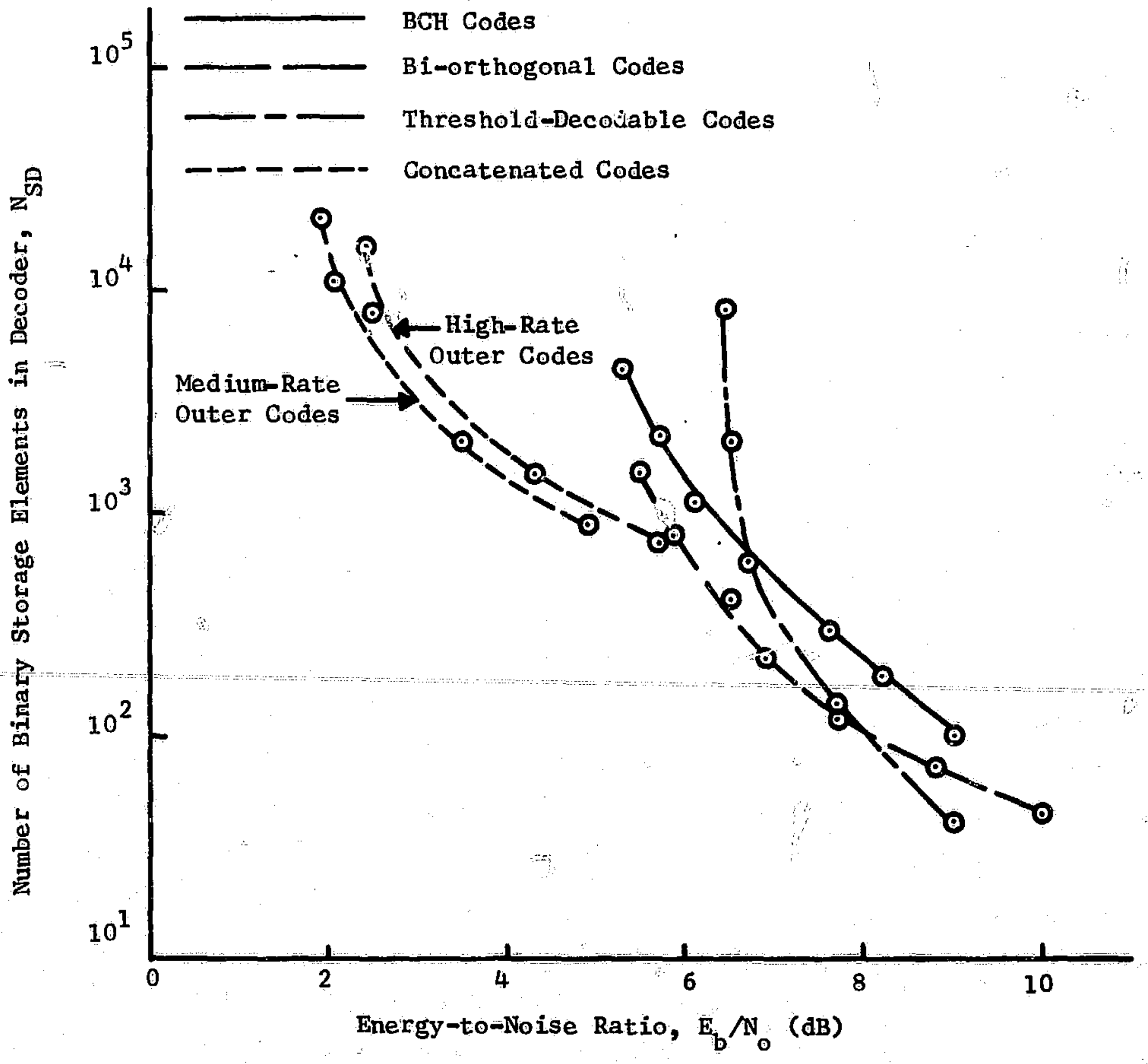

Fig. 10.2. Number of Binary Storage Elements Required in the Decoder for Several Classes of Codes at a $P_W(e) = 10^{-5}$.

circuitry would be required. Throughout the range 5.5-9.0 dB, the bi-orthogonal decoder requires fewer storage elements than the BCH decoder by about a factor of two. Again the concatenated codes are attractive for values of E_b/N_o from 2 to 5 dB.

The number of binary storage elements required in the encoder as a function of E_b/N_o for the four coding schemes is shown in Fig. 10.3. This measure of complexity is particularly important in space communication where there are restrictions on the size of the encoder. For values of E_b/N_o from 5.5-9.0 dB bi-orthogonal coding is by far the easiest to implement. Also in this range the BCH codes are easier to implement than the threshold-decodable codes. Again for low values of E_b/N_o (2 to 5 dB) the concatenated codes become attractive. The high-rate concatenated codes are easier to implement for $E_b/N_o > 2.7$ dB. At $E_b/N_o = 5.7$ dB the concatenated encoder is only slightly more complex than the bi-orthogonal encoder and much less complex than either of the other two schemes.

Sequential decoding of convolutional codes would compare very favorably with the block coding methods just discussed. Since curves of complexity vs. E_b/N_o are not available, this method will be discussed only qualitatively. Sequential decoding has a small average number of computations per decoded information bit so that it would appear to compare favorably with BCH codes and threshold-decodable codes in Fig. 10.1, but since the number of computations is highly variable, a much larger decoder speed advantage is needed to obtain a negligible probability of buffer overflow. Operation of the sequential decoder at values of E_b/N_o from 3 to 4 dB is possible if the decoder speed advantage and buffer size are large enough. Due to the large buffer requirement and complex decoding algorithm,

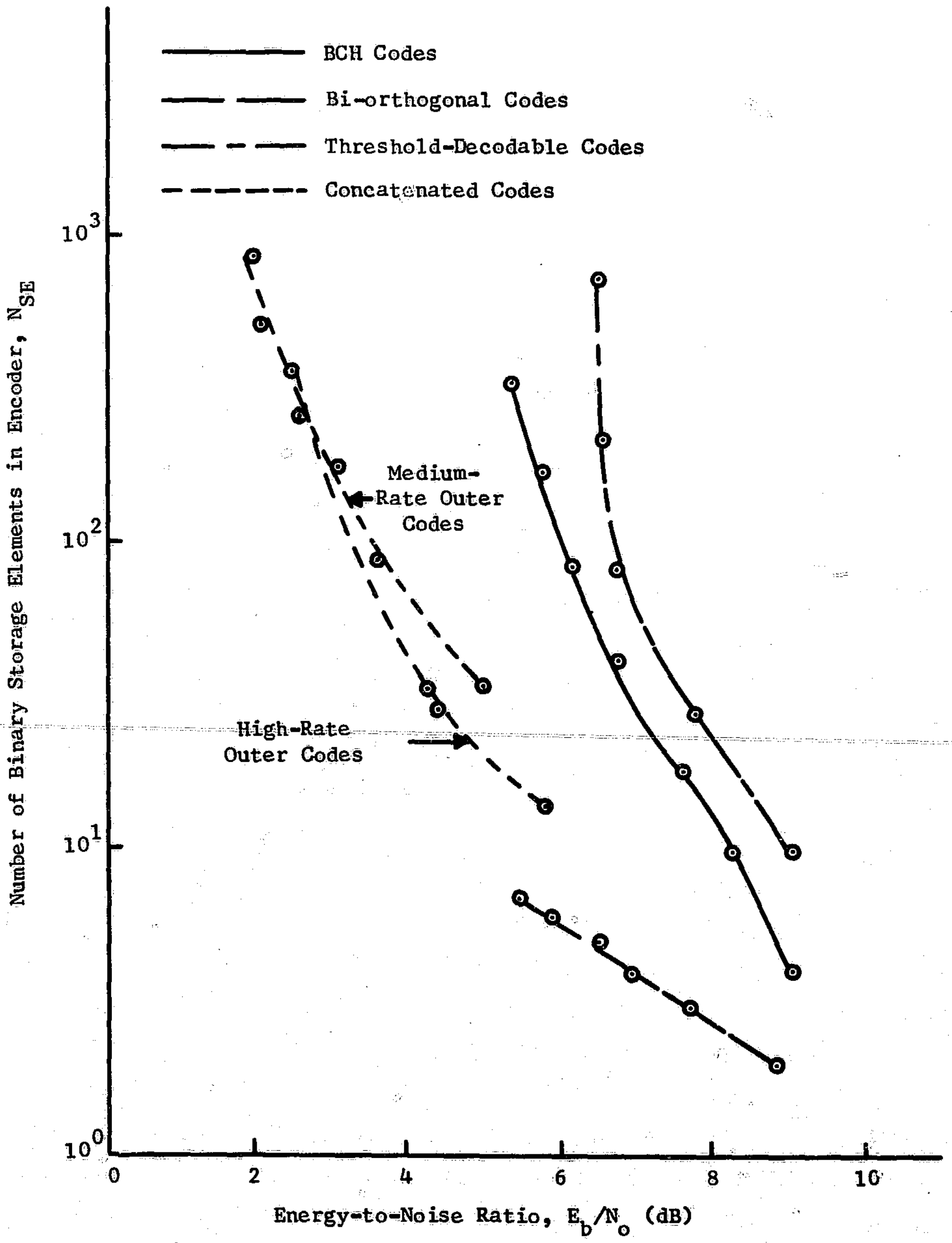

Fig. 10.3. Number of Binary Storage Elements Required in the Encoder for Several Classes of Codes at a $P_W(e) = 10^{-5}$.

the cost of a sequential decoder is comparable to that of a concatenated decoder. One of the principal implementation advantages of sequential decoding is that the convolutional encoder is readily implemented by a shift register which is typically 20-40 stages long. This could be the deciding factor in space communication where the encoder complexity is strictly limited. As shown in Fig. 10.3, the concatenated encoder will be more complex.

Even in using concatenated coding to communicate at $E_b/N_o = 2.0$ dB, Shannon's limit is still 3.6 dB away. At this time communication at values of E_b/N_o below 2.0 dB with practical systems requires the use of a feedback channel. Concatenation schemes utilizing a feedback channel were discussed in Chapter VIII. The curves of $P_W(e)$ vs. E_b/N_o for these schemes were presented in Figs. 8.10, 8.11, and 8.12. Operation very near Shannon's limit is achieved by these schemes.

The coding schemes which have been discussed are attractive to implement for different ranges of E_b/N_o. The ranges of E_b/N_o which these schemes are useful in obtaining a word error probability in the range $10^{-3} < P_W(e) < 10^{-7}$ are tabulated in Table 10.1.

Table 10.1. Coding Techniques for the Gaussian Channel Which Are Useful in Obtaining a Word Error Probability in the Range $10^{-3} < P_W(e) < 10^{-7}$.

Range of E_b/N_o	Techniques
$E_b/N_o > 9$ dB	No coding.
$5 \text{ dB} < E_b/N_o < 9$ dB	There are several schemes which can be implemented with reasonable complexity (BCH codes, bi-orthogonal codes, and threshold-decodable codes).
$2 \text{ dB} < E_b/N_o < 5$ dB	There are two very complex schemes (sequential decoding of convolutional codes and concatenated codes).

$-1.6 \text{ dB} < E_b/N_o < 2 \text{ dB}$	There is a very elaborate coding scheme requiring a feedback channel (concatenation with inner-code feedback).

When coding for channels with memory, a different approach is used. A detailed study of channel error statistics must be undertaken in order to determine the type of coding strategy which should be used. It is important to know the expected length of the error bursts. If a feedback channel is available, it is also important to know the quality of this channel.

If a feedback channel is available, system complexity can be reduced considerably. In this case the strategy essentially is to use the channel when it is in State G and to avoid use of the channel when it is in State B. The simple detection-retransmission scheme works very well for this purpose. Use of the detection-retransmission scheme as the inner coding technique in a concatenation scheme results in extremely low error probabilities.

If no feedback channel is available it is important that the coding scheme selected have a block length sufficiently long to span the longest expected error burst. On many real channels error bursts which are hundreds or thousands of bits long are common. This requires block lengths which are tens of thousands of bits long. The techniques used to achieve these long block lengths are concatenation and interleaving.

The main purpose of this investigation has been to develop suitable and easily understood comparisons among the major coding techniques. Emphasis has been placed on coding techniques for the Gaussian channel. The results of this investigation have been applied in this chapter in a comparison of coding schemes for the Gaussian channel at a $P_W(e) = 10^{-5}$.

The simplicity of the resulting comparison should be evident. Similar comparisons at any other value of $P_W(e)$ can be made using the results of the preceding chapters. Comparisons for other channels can be made by calculating the $P_W(e)$ on these channels. In addition to the comparisons which have been presented, several results of this investigation will find other applications. A new technique, concatenation with inner-code feedback, is analyzed, and it is shown that communication over the Gaussian channel at values of E_b/N_o very near Shannon's limit ($E_b/N_o = 1.6$ dB) is possible with low error probability. In Appendix C a tight upper bound on the average digit error probability of a linear block code is developed. This bound is very useful in evaluating the performance of error-control systems. A detailed analysis of the generalized Gilbert channel is presented in Appendix E. The results of this analysis are used in deriving the probability distribution of the burst lengths on this channel. This distribution is very useful in evaluating the performance of error-control schemes on the Gilbert channel.

There is room for future work to be done on several of the topics of this investigation. Further study of concatenation with inner-code feedback is needed. Possible topics include effects of feedback channel noise, effects of propagation delay, and tradeoffs between inner and outer code parameters. Future work could also include further study of the Gilbert channel. Possible topics include use of the distribution of burst lengths to evaluate the performance of coding schemes and matching real channel error statistics with Gilbert channel parameters.

APPENDIX A

NUMBER OF COMPUTATIONS REQUIRED TO REDUCE A MATRIX TO UPPER TRIANGULAR FORM

In decoding using the Peterson algorithm a system of equations of the form

$$\begin{bmatrix} a_{11} & a_{12} \cdots a_{1t} \\ a_{21} & a_{22} \cdots a_{2t} \\ \cdots & \cdots \\ a_{t1} & a_{t2} \cdots a_{tt} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{2t-1} \end{bmatrix} \qquad (A.1)$$

must be solved, where t denotes the maximum number of errors which the code is capable of correcting. The number of errors which occur in the transmission of a code word is denoted by e. Then these equations have the property that e is the maximum number of successive equations that are linearly independent. This system is solved by reducing the coefficient matrix of (A.1) to upper triangular form.[82]

To begin the reduction of the system (A.1), σ_1 is eliminated from all equations beginning with the second. This is accomplished by adding to the second equation the first multiplied by $-\frac{a_{21}}{a_{11}}$, to the third equation the first multiplied by $-\frac{a_{31}}{a_{11}}$, etc. The result is the system

$$\begin{bmatrix} a_{11} & a_{12} \cdots a_{1t} \\ 0 & a_{22}^{(1)} \cdots a_{2t}^{(1)} \\ \cdots & \cdots \\ 0 & a_{t2}^{(1)} \cdots a_{tt}^{(1)} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3^{(1)} \\ \vdots \\ S_{2t-1}^{(1)} \end{bmatrix}, \qquad (A.2)$$

where

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{ij}, \quad S_{2i-1}^{(1)} - \frac{a_{i1}}{a_{11}} S_1 \quad . \qquad (A.3)$$

In determining the number of computations involved, division will be performed by an inversion and a multiplication, and a subtraction will be considered equal to an addition. The number computations involved in obtaining the system (A.2) is one inversion, (t-1)(t+2) multiplications, and (t-1)(t+1) additions. In the same manner σ_2 can be eliminated from the last (t-2) equations. This requires one inversion, (t-2)(t+1) multiplications, and (t-2)(t) additions. Since the rank of the coefficient matrix is e, after e steps in this procedure the system

$$\begin{bmatrix} a_{11} & a_{12}\cdots a_{1e} & \cdots & a_{1t} \\ 0 & a_{22}^{(1)} \quad a_{2e}^{(1)} & \cdots & a_{2t}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0\cdots a_{ee}^{(e-1)} & \cdots & a_{et}^{(e-1)} \\ 0 & 0\cdots 0 & a_{e+1,e+1}^{(e)}\cdots & a_{e+1,t}^{(e)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0\cdots 0 & a_{t,e+1}^{(e)}\cdots & a_{t,t}^{(e)} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \cdot \\ \cdot \\ \cdot \\ \sigma_t \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2^{(1)} \\ \cdot \\ \cdot \\ \cdot \\ S_{2t-1}^{(e)} \end{bmatrix} \qquad (A.4)$$

is obtained.

In performing the successive steps of the algorithm each of the quantities a_{11}, $a_{22}^{(1)}$, ..., $a_{ee}^{(e-1)}$ had to be different from zero since they were inverted. This requires a test before each inversion is performed. Since the rank of the matrix is e, the quantity $a_{e+1,e+1}^{(e)} = 0$, which is the signal to stop the algorithm. The quantities σ_{e+1}, σ_{e+2}, ..., σ_t, are set to zero, and then the system of equations

$$\begin{bmatrix} a_{11} & a_{12}\cdots & a_{1e} \\ 0 & a_{22}^{(1)}\cdots & a_{2e}^{(1)} \\ \cdots & \cdots & \cdots \\ 0 & 0\cdots & a_{ee}^{(e-1)} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \cdot \\ \cdot \\ \cdot \\ \sigma_e \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2^{(1)} \\ \cdot \\ \cdot \\ \cdot \\ S_{2e-1}^{(e-1)} \end{bmatrix} \qquad (A.5)$$

is solved. Thus

$$\sigma_e = \frac{S_{2e-1}^{(e-1)}}{a_{ee}^{(e-1)}} , \sigma_{e-1} = \frac{S_{2e-3}^{(e-1)} - \sigma_e a_{e-1,e}^{(e-2)}}{a_{e-1,e-1}^{(e-2)}} , \text{ etc.}$$

The total number of computations involved in reducing the coefficient matrix to upper triangular form can now be determined. The number of inversions needed is $I_1 = e$. The number of multiplications needed is

$$\begin{aligned} M_1 &= (t-1)(t+2) + (t-2)(t+1) + \cdots + (t-e)(t-e+3) \\ &= \sum_{k=1}^{e} (t+3-k)(t-k) \\ &= (t^2+3t)e - (2t+3) \sum_{k=1}^{t} k + \sum_{k=1}^{t} k^2 \\ &= (t^2+3t)e - (2t+3) \frac{e(e+1)}{2} + \frac{e(e+1)(2e+1)}{6} . \qquad (A.6) \end{aligned}$$

The total number of additions is

$$\begin{aligned} A_1 &= (t-1)(t+1) + (t-2)(t) + \cdots + (t-e)(t-e+2) \\ &= \sum_{k=1}^{e} (t+2-k)(t-k) \\ &= (t^2+2t)e - (2t+2) \sum_{k=1}^{e} k + \sum_{k=1}^{e} k^2 \\ &= (t^2+2t)e - (2t+2) \frac{e(e+1)}{2} + \frac{e(e+1)(2e+1)}{6} . \qquad (A.7) \end{aligned}$$

The maximum number of computations is needed when the rank of the coefficient matrix is t or (t-1). After the algorithm is performed (t-1) times, the matrix is in upper triangular form. The number of multiplications required is

$$M_1(\max) = (t^2+3t)(t-1) - \frac{(2t+3)(t-1)(t)}{2} + \frac{(t-1)(t)(2t-1)}{6} . \qquad (A.8)$$

The number of additions required is

$$A_1(\max) = (t^2+2t)(t-1) - \frac{(2t+3)(t-1)(t)}{2} + \frac{(t-1)(t)(2t-1)}{6} . \qquad (A.9)$$

To solve for the $\sigma_1, \sigma_2, \cdots, \sigma_t$ requires a number of inversions $I_2 = e$. The number of multiplications required is

$$\begin{aligned} M_2 &= 1 + 2 + \cdots + e , \\ &= \sum_{k=1}^{e} k \\ &= \frac{e(e+1)}{2} , \end{aligned} \qquad (A.10)$$

and the number of additions required is

$$\begin{aligned} A_2 &= 0 + 1 + 2 + \cdots + (e-1) \\ &= \sum_{k=1}^{e-1} k \\ &= \frac{e(e-1)}{2} . \end{aligned} \qquad (A.11)$$

The maximum number of computations is required when the rank of the coefficient matrix is t or (t-1). In this case $I_2(\max) = t$ inversions are needed. The number of multiplications required is

$$M_2(\max) = \frac{t(t+1)}{2} , \qquad (A.12)$$

and the number of additions required is

$$A_2(\max) = \frac{t(t-1)}{2} . \qquad (A.13)$$

An inverse in $GF(2^m)$ can be formed by (2m-2) multiplications as shown in Bartee and Schneider.[83] A total of 2e inversions are needed which can be accomplished by

$$M_3 = (2m - 2)(2e) \tag{A.14}$$

multiplications. The maximum number of inversions needed is (2t - 1) which requires

$$M_3(\max) = (2m - 2)(2t - 1) \tag{A.15}$$

multiplications.

APPENDIX B

OPTIMUM RECEIVER PRINCIPLES

In this section the optimum receiver is derived for a communication system with M equally-likely messages and additive Gaussian noise as the disturbance. The development follows closely that of Wozencraft and Jacobs.[84] Such a receiver is referred to as a maximum-likelihood receiver. A probability is denoted by P( ), and a probability density function is denoted by p( ). The set of messages is denoted by $\{m_i\}$, $i = 0, 1, \cdots, M-1$. When the message is $m = m_i$, the transmitter sends the N-dimensional signal vector,

$$\bar{s}_i = (s_{i1}, s_{i2}, \cdots, s_{iN}), i = 0, 1, \cdots, M-1 \quad , \tag{B.1}$$

over the channel. The M signal vectors $\{\bar{s}_i\}$ can be visualized as M points in an N-dimensional geometric space called the signal space.

A noise vector $\bar{n}$ is added to the signal vector in the channel, and the output of the channel is a random vector $\bar{r}$, i.e.,

$$\bar{r} = (r_1, r_2, \cdots, r_N) \tag{B.2}$$

The channel is defined by the set of conditional probability density functions $\{p(\bar{r} \mid \bar{s}_i)\}$, $i = 0, 1, \cdots, M-1$. Using its knowledge of $\bar{r}$, $\{\bar{s}_i\}$, and $\{p(\bar{r} \mid \bar{s}_i)\}$, the optimum receiver determines which message has maximum a posteriori probability. With this criterion the optimum receiver sets its estimate $\hat{m} = m_k$, whenever

$$P(m_k \mid \bar{r}) > P(m_i \mid \bar{r}), i = 0, 1, \cdots, M-1, i \neq k \quad . \tag{B.3}$$

When the receiver sets $\hat{m} = m_k$, the conditional probability of a correct decision is

$$P(c \mid \bar{r}) = P(m_k \mid \bar{r}) \; . \tag{B.4}$$

Then the unconditional probability of a correct decision is

$$P(c) = \int_{-\infty}^{\infty} P(c \mid \bar{r}) p(\bar{r}) d\bar{r} \; . \tag{B.5}$$

Since $p(\bar{r}) \geq 0$, $P(c)$ is maximized by maximizing $P(c \mid \bar{r})$ which is done in (B.3). Since this maximizes the probability of a correct decision, the maximum-a-posteriori-probability receiver must be optimum. If two or more m_i have equal a posteriori probabilities greater than all the rest, then one of them can be arbitrarily selected without affecting the probability of error.

The criterion of (B.3) can be put in a more convenient form by applying Bayes' rule, i.e.,

$$P(m_i \mid \bar{r}) p(\bar{r}) = P(m_i) p(\bar{r} \mid m_i) \; . \tag{B.6}$$

Thus,

$$P(m_i \mid \bar{r}) = \frac{P(m_i) p(\bar{r} \mid m_i)}{p(\bar{r})} \; . \tag{B.7}$$

When $m = m_i$, the signal vector $\bar{s} = \bar{s}_i$ is transmitted. Thus

$$p(\bar{r} \mid m_i) = p(\bar{r} \mid \bar{s}_i) \; . \tag{B.8}$$

Since $p(\bar{r})$ and $P(m_i)$ are independent of i, the optimum receiver sets $\hat{m} = m_k$ whenever the decision function

$$p(\bar{r} \mid \bar{s}_i), i = 0,1,\cdots,M-1 \tag{B.9}$$

is maximum for $i = k$.

For additive Gaussian noise the received vector $\bar{r}$ and the signal vector $\bar{s}$ are related by

$$\bar{r} = \bar{s} + \bar{n} = (s_1 + n_1, s_2 + n_2, \cdots, s_N + n_N) \quad . \tag{B.10}$$

Thus, if $\bar{r}$ is received when $\bar{s} = \bar{s}_i$ is transmitted then $\bar{n} = \bar{r} - \bar{s}_i$, and

$$p(\bar{r} \mid \bar{s}_i) = p(\bar{n} \mid \bar{s}_i), i = 0,1,\cdots,M-1 \quad , \tag{B.11}$$

with $\bar{n} = \bar{r} - \bar{s}_i$. Now, if $\bar{n}$ and $\bar{s}$ are statistically independent which is usually the case, $p(\bar{n} \mid \bar{s}_i) = p(\bar{n})$. Therefore,

$$p(\bar{n} \mid \bar{s}_i) = p(\bar{n}), \; \bar{n} = \bar{r} - \bar{s}_i \quad . \tag{B.12}$$

The decision function, (B.9), then becomes

$$p(\bar{n}) \text{ with } \bar{n} = \bar{r} - \bar{s}_i \quad . \tag{B.13}$$

A simplification can be made by assuming that the N components of $\bar{n}$ are statistically independent, zero-mean, Gaussian random variables, each with variance σ^2. The probability density function of the noise is then

$$p(\bar{n}) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{j=1}^{N} n_j^2\right) \quad . \tag{B.14}$$

Since

$$\bar{n}^2 = \bar{n} \cdot \bar{n} = \sum_{j=1}^{N} n_j^2 \quad , \tag{B.15}$$

then

$$p(\bar{n}) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(-\frac{|\bar{n}|^2}{2\sigma^2}\right) \quad . \tag{B.16}$$

Now $\bar{n} = \bar{r} - \bar{s}_i$, so the decision function of (B.13) becomes

$$\exp\left(-\frac{|\bar{r} - \bar{s}_i|^2}{2\sigma^2}\right) \quad . \tag{B.17}$$

The factor $(2\pi\sigma^2)^{-N/2}$ was discarded because it is independent of i. The optimum receiver then sets $\hat{m} = m_k$ whenever (B.17) is maximum for $i = k$. Maximizing (B.17) is equivalent to finding the value of i which minimizes

$$|\bar{r} - \bar{s}_i|^2 \quad . \tag{B.18}$$

This decision function can be visualized geometrically. The term $|\bar{r} - \bar{s}_i|^2$ is the square of the Euclidean distance between the points $\bar{r}$ and $\bar{s}_i$ in signal space.

The N-dimensional signal vector may be synthesized from N orthonormal waveforms, $\left\{\phi_j(t)\right\}$, $j = 1, \cdots, N$. By orthonormal is meant

$$\int_{-\infty}^{\infty} \phi_j(t)\phi_l(t)dt = \delta_{jl} = \begin{cases} 1, & j = l \\ 0, & j \neq l \end{cases} \quad . \tag{B.19}$$

Then the transmitted waveforms are

$$s_i(t) = \sum_{j=1}^{N} s_{ij}\phi_j(t), i = 0, 1, \cdots, M-1 \quad . \tag{B.20}$$

This produces the signal vector of (B.1). The signal space is considered to have N mutually perpendicular axes $\phi_1, \phi_2, \cdots, \phi_N$.

Recovery of the vectors $\left\{\bar{s}_i\right\}$ from the signal waveforms $\left\{s_i(t)\right\}$ is a straightforward process. Since the $\left\{\phi_j(t)\right\}$ are orthonormal,

$$\begin{aligned} \int_{-\infty}^{\infty} s_i(t)\phi_k(t)dt &= \int_{-\infty}^{\infty} \left[\sum_{j=1}^{N} s_{ij}\phi_j(t)\right] \phi_k(t)dt \\ &= \sum_{j=1}^{N} s_{ij} \int_{-\infty}^{\infty} \phi_j(t)\phi_k(t)dt \\ &= \sum_{j=1}^{N} s_{ij}\delta_{jk} = s_{ik} \quad . \end{aligned} \tag{B.21}$$

By multiplying and integrating for each $\phi_k(t)$, $1 \leq k \leq N$, all components of the vector,

$$\bar{s}_i = (s_{i1}, s_{i2}, \cdots, s_{iN}) \quad , \tag{B.22}$$

are obtained. This procedure can be implemented with a bank of N multipliers and integrators.

Using this bank of multipliers and integrators to operate on the received random process r(t), the integrator outputs are

$$r_j = \int_{-\infty}^{\infty} r(t)\phi_j(t)dt, \; j = 1, \cdots, N \quad . \tag{B.23}$$

Together these integrator outputs constitute a random vector,

$$\bar{r} = (r_1, r_2, \cdots, r_N) \quad . \tag{B.24}$$

Since $r(t) = s(t) + n(t)$,

$$\bar{r} = \bar{s} + \bar{n} \quad , \tag{B.25}$$

where

$$\bar{n} = (n_1, n_2, \cdots, n_N) \tag{B.26}$$

and

$$n_j = \int_{-\infty}^{\infty} n(t)\phi_j(t)dt, \; j = 1, \cdots, N \quad . \tag{B.27}$$

Intuitively one would suspect that the vector $\bar{r}$ obtained in this way contains all data from r(t) that is relevant to the optimum determination of the transmitted message. This is proved by Wozencraft and Jacobs.[85]

The simplest implementation of the optimum receiver follows by expanding the decision function, $|\bar{r} - \bar{s}_i|^2$, i.e.

$$|\bar{r}-\bar{s}_i|^2 = \sum_{j=1}^{N} (r_j - s_{ij})^2$$

$$= \sum_{j=1}^{N} (r_j^2 - 2r_j s_{ij} + s_{ij}^2)$$

$$= |\bar{r}|^2 - 2\bar{r}\cdot\bar{s}_i + |\bar{s}_i|^2 \quad , \tag{B.28}$$

where

$$\bar{r}\cdot\bar{s}_i \triangleq \sum_{j=1}^{N} r_j s_{ij} \quad . \tag{B.29}$$

Since the term $|\bar{r}|^2$ is independent of i, it may be disregarded. If equal-energy signals are assumed, the term $|\bar{s}_i|^2$ may also be disregarded. In this case the decision rule is simply to maximize

$$\bar{r}\cdot\bar{s}_i, \; i = 0,1\cdots,M-1 \quad . \tag{B.30}$$

Using multiplier-integrator pairs to calculate the components of $\bar{r}$, the optimum receiver is shown in Fig. B.1.

The quantities, $\bar{r}\cdot\bar{s}_i$, $i = 0,1,\cdots,M-1$, can be determined in another manner, which results in a slightly different formulation of the optimum receiver. From (B.20) and (B.23),

$$\int_{-\infty}^{\infty} r(t)s_i(t)dt = \int_{-\infty}^{\infty} r(t)\left[\sum_{j=1}^{N} s_{ij}\phi_j(t)\right] dt$$

$$= \sum_{j=1}^{N} s_{ij} \int_{-\infty}^{\infty} r(t)\phi_j(t)dt$$

$$= \sum_{j=1}^{N} s_{ij} r_j$$

$$= \bar{r}\cdot\bar{s}_i \quad . \tag{B.31}$$

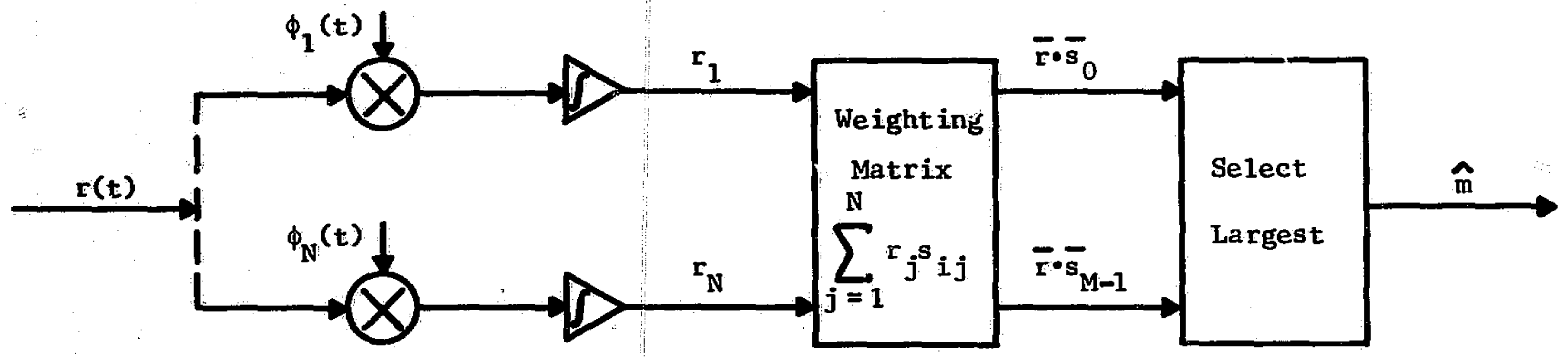
$r(t)$
$\phi_1(t)$
$\phi_N(t)$
r_1
r_N
Weighting Matrix $\sum_{j=1}^{N} r_j s_{ij}$
$\overline{r}\cdot\overline{s}_0$
$\overline{r}\cdot\overline{s}_{M-1}$
Select Largest
$\hat{m}$

Fig. B.1. Optimum Receiver, Form 1.

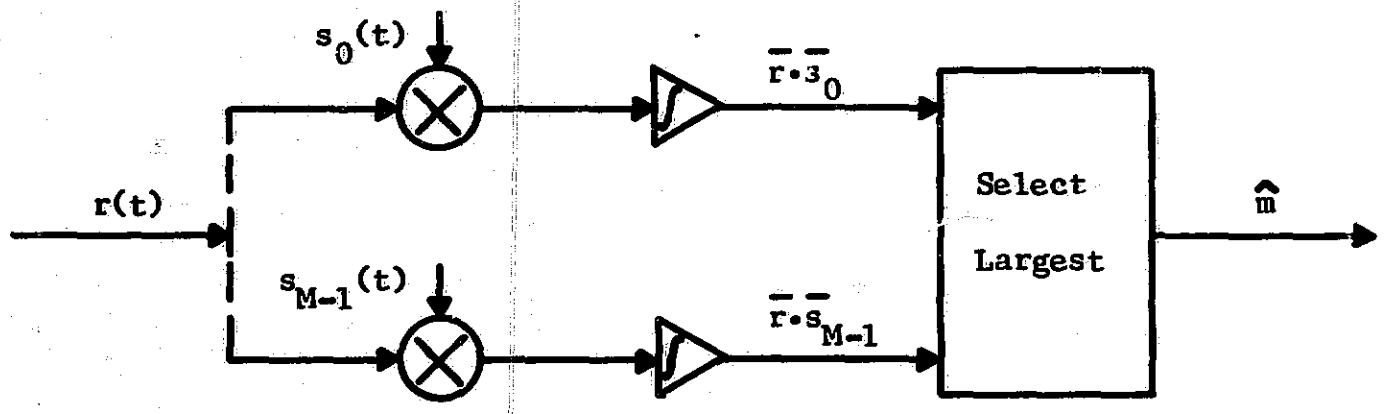
$r(t)$
$s_0(t)$
$s_{M-1}(t)$
$\overline{r}\cdot\overline{s}_0$
$\overline{r}\cdot\overline{s}_{M-1}$
Select Largest
$\hat{m}$

Fig. B.2. Optimum Receiver, Form 2.

Actually, this is just a correlation process. The received signal is correlated with each of the possible transmitted waveforms as shown in Fig. B.2. The decision as to which implementation of the optimum receiver to choose depends on the relationship between M and N. If $N << M$, then the implementation of Fig. B.1 is most desirable. However, if $N \approx M$, then the implementation of Fig. B.2 is simplest.

The decision rule of (B.18) can be used for determining the probability of error for various signal sets. The ones of interest here are the binary antipodal and binary orthogonal signal sets. A set of two equally-likely antipodal signals are shown in Fig. B.3. From (B.18) and Fig. B.3 it is obvious that the optimum decision rule is to set $\hat{m} = m_1$ when $\bar{r}$ lies to the left of the $\bar{\phi}_2$ axis or, in other words, when the noise component $n_1 < d/2$. If $\bar{s}_1$ is transmitted, an error occurs if and only if $n_1 < d/2$. Thus,

$$P(e|m_i) = P(n_1 > d/2) \quad . \tag{B.32}$$

Since n_1 is a zero-mean Gaussian random variable with variance $N_o/2$,

$$P(e|m_1) = \int_{d/2}^{\infty} \frac{1}{\sqrt{\pi N_o}} e^{-\alpha^2/N_o} d\alpha \quad . \tag{B.33}$$

Letting $\beta = \alpha\sqrt{2/N_o}$,

$$P(e|m_1) = \int_{\frac{d/2}{\sqrt{N_o/2}}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\beta^2/2} d\beta \triangleq Q\left(\frac{d}{\sqrt{2N_o}}\right) \quad , \tag{B.34}$$

where

$$Q(\alpha) \triangleq \frac{1}{\sqrt{2\pi}} \int_{\alpha}^{\infty} e^{-\beta^2/2} d\beta \quad . \tag{B.35}$$

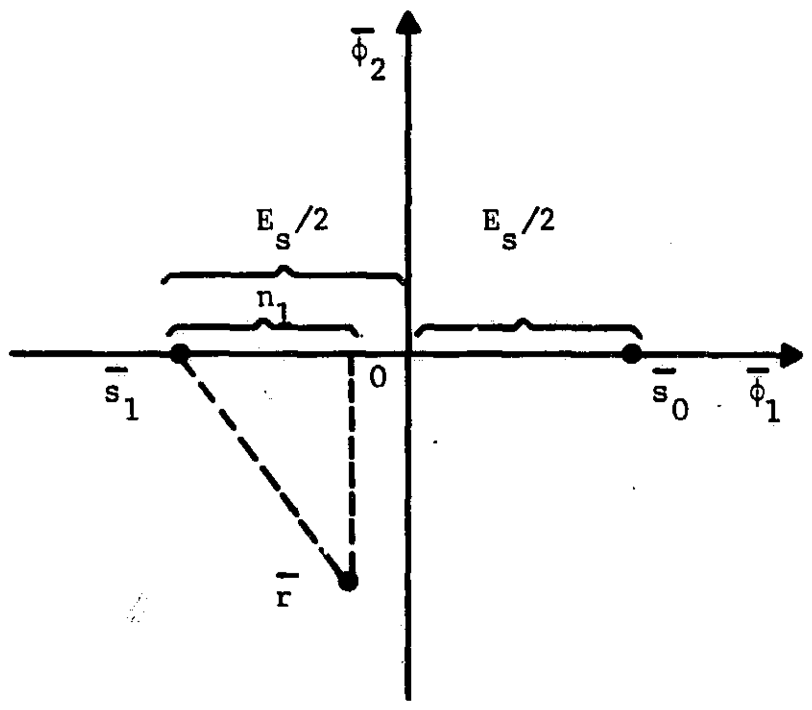

Fig. B.3. Antipodal Signal Set.

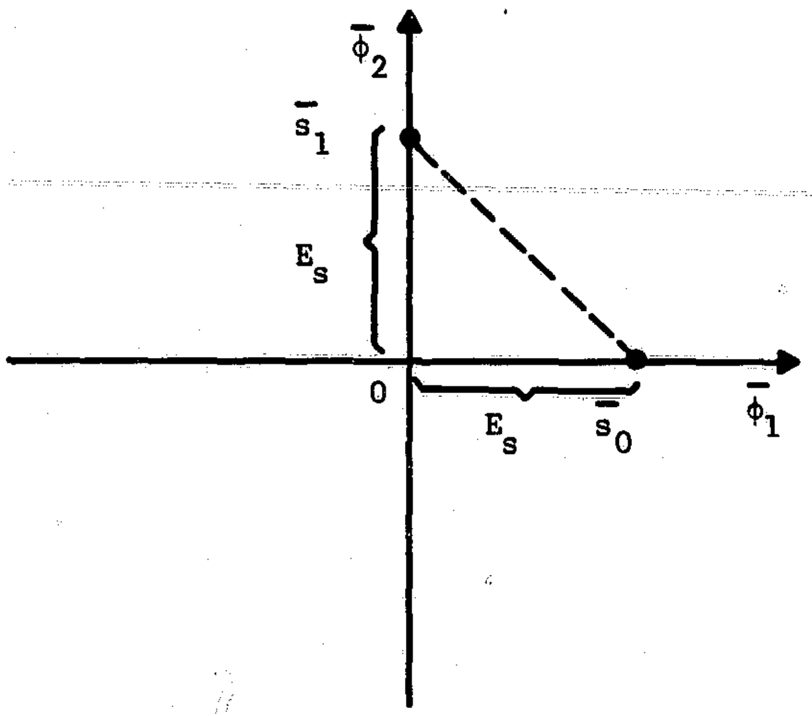

Fig. B.4. Orthogonal Signal Set.

By symmetry, the conditional probability of error is the same for either signal. Thus,

$$P(e) = \sum_{i=0}^{1} P(m_i)P(e|m_i) = P(e|m_1) = Q(\frac{d}{\sqrt{2N_o}}) \ . \qquad (B.36)$$

Actually, this is the probability of error for any two equally-likely signal vectors separated by a distance, regardless of their location in signal space. For equally-likely antipodal signals $d = 2\sqrt{E_s}$, and

$$P(e) = Q(\sqrt{\frac{2E_s}{N_o}}) \ . \qquad (B.37)$$

A set of two equally-likely orthogonal signals is shown in Fig. B.4. It is obvious from Fig. B.4 that $d = \sqrt{2E_s}$ for orthogonal signals. Since (B.36) is applicable to this signal set, the probability of error is simply

$$P(e) = Q(\sqrt{E_s/N_o}) \ . \qquad (B.38)$$

A comparison of (B.37) and (B.38) reveals that an antipodal signal set needs only half the signal power that an orthogonal signal set does to achieve the same P(e). Thus, antipodal signaling has a 3 dB advantage over orthogonal signaling. Because of this, antipodal signaling will be used throughout this work.

The combination of transmitter, Gaussian channel, and optimum receiver just described can be thought of as a binary symmetric channel with cross-over probability p, which is shown in Fig. B.5. With antipodal signaling,

$$p = Q(\sqrt{\frac{2E_s}{N_o}}) \ . \qquad (B.39)$$

This channel is used in comparisons throughout this work.

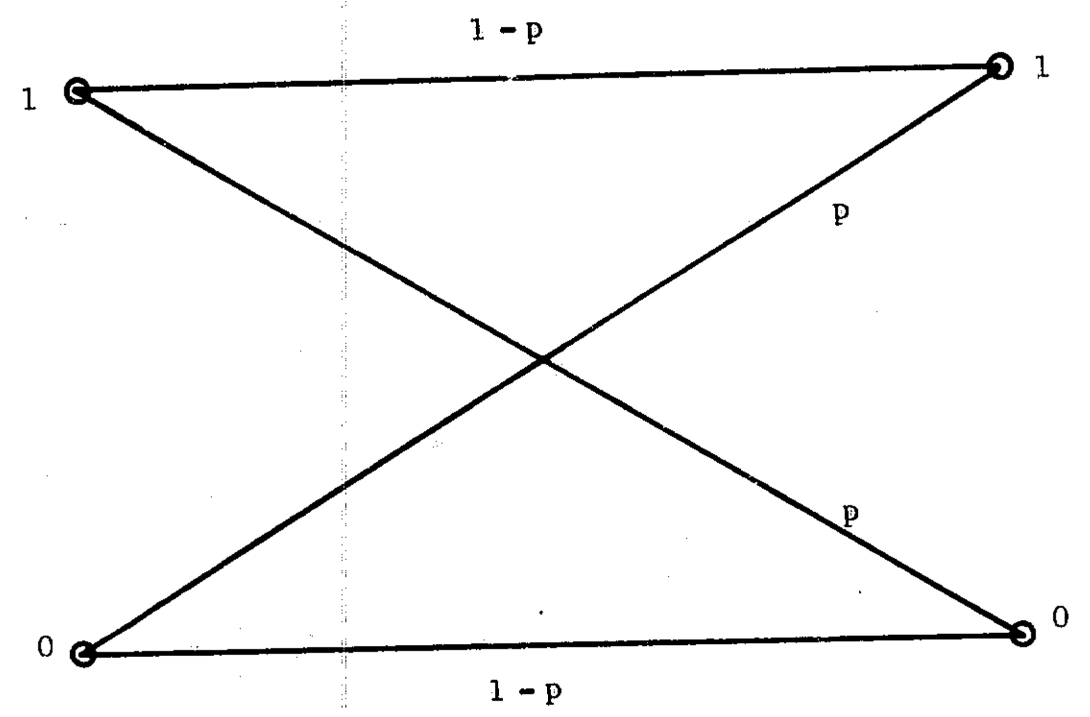

Fig. B.5. Binary Symmetric Channel.

APPENDIX C

AVERAGE DIGIT ERROR PROBABILITY IN DECODING A LINEAR BLOCK CODE

In calculating the performance of a linear block code over a discrete channel the word error probability $P_W(e)$ is generally used and is easily calculated from the error-correcting capability of the code. The probability of digit error denoted by $P_D(e)$ will be defined to be the probability that a given digit at the output of the decoder is in error. This probability may be desired in certain cases, but calculation of it requires knowledge of the detailed structure of the code. In this appendix a method will be derived for calculating an upper bound to $P_D(e)$ which is dependent only on the minimum distance of the code. The tightness of this bound will also be demonstrated, and an example where knowledge of $P_D(e)$ is important will be presented.

The probability of digit error can be written as

$$P_D(e) = P_{D|W}(e)P_W(e) \ , \qquad \text{(C.1)}$$

where $P_{D|W}(e)$ denotes the probability of digit error given that a word error has occurred. A distinction will be made between channel errors which occur at the output of the channel with probability p and digit errors which occur at the output of the decoder with probability $P_D(e)$. It will be assumed that the block length of the code is n and the minimum distance of the code is odd and equal to $d = 2e + 1$, where e is the maximum number of errors which the code can correct. If the channel is memoryless, the number of channel errors in each block of n digits is binomially distributed, i.e.,

$$P_i = \binom{n}{i} p^i (1-p)^{n-i} , \qquad (C.2)$$

where P_i denotes the probability that i channel errors occur in a block of n digits.[86] If the code used over such a channel is capable of correcting e errors, $P_W(e)$ is just the probability that more than e channel errors occur which is

$$P_W(e) = \sum_{i=e+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} . \qquad (C.3)$$

The digit error probability can be written

$$P_D(e) = \sum_{i=0}^{n} P(\text{digit error} \mid i \text{ channel errors}) P_i , \qquad (C.4)$$

where P_i is given by (C.2). An exact expression for the probability of digit error given that i channel errors occur depends upon the detailed structure of the code. However, this quantity can be bounded fairly easily.

In upperbounding $P_D(e)$, it is assumed that the all zeros word is transmitted. No generality is lost by doing this since the code is linear. If the number of channel errors is less than or equal to e, the decoder decodes correctly. Thus,

$$P(\text{digit error} \mid i \text{ channel errors}) = 0, i \leq e . \qquad (C.5)$$

If there are $i > e$ channel errors, the decoder decodes into a word of weight at most $i + e$. Of course, the weight of the decoder output word is never greater than n so for $i \geq n - e$ the decoder decodes into a word of weight at most n. Thus,

$$P(\text{digit error} \mid i \text{ channel errors}) \leq \frac{i+e}{n} , \quad e < i < n-e , \qquad (C.6)$$

and

$$P(\text{digit error} \mid i \text{ channel errors}) \leq 1, n-e \leq i \leq n \ . \tag{C.7}$$

Substituting (C.2), (C.5), (C.6), and (C.7) into (C.4),

$$P_D(e) \leq \sum_{i=e+1}^{n-e-1} \frac{(e+i)}{n} \binom{n}{i} p^i (1-p)^{n-i}$$

$$+ \sum_{i=n-e}^{n} \binom{n}{i} p^i (1-p)^{n-i} \ . \tag{C.8}$$

A similar development for the case d = 2e + 2 leads to the result

$$P_D(e) \leq \sum_{i=e+1}^{n-e-2} \frac{(e+1+i)}{n} \binom{n}{i} p^i (1-p)^{n-i}$$

$$+ \sum_{i=n-e-1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \ . \tag{C.9}$$

The bound which has been presented for $P_D(e)$ is for any of the n digits of the code word. Often the digit error probability of interest is for the k information digits of the code word. The bound presented will hold in this case if the assumption is made that for i channel errors the decoded digit errors are distributed proportionately between the information digits and the parity digits. By writing out the standard array for a number of short codes it can be seen that this condition is very nearly satisfied by most codes and is completely satisfied for some codes such as the (6,3) code generated by

$$G = \begin{bmatrix} 100110 \\ 010011 \\ 001101 \end{bmatrix} . \tag{C.10}$$

Actually, from the law of large numbers it would be expected that the assumption would be better for n large.

To determine how tight the bound in (C.9) is, consider the probability

$$P_{D|W}(e) = \frac{P_D(e)}{P_W(e)} \quad . \tag{C.11}$$

If a word error occurs, the least number of digits which can be in error is d so that

$$P_{D|W}(e) \geq \frac{d}{n} , \tag{C.12}$$

while the greatest number of digits which can be in error is n so that

$$P_{D|W}(e) \leq 1 \quad . \tag{C.13}$$

A much tighter upper bound on $P_{D|W}(e)$ than that of (C.13) can be obtained by applying the upper bound developed in (C.9) for $P_D(e)$ to (C.11). Thus for $d = 2e+1$,

$$P_{D|W}(e) \leq \frac{1}{P_W(e)} \sum_{i=e+1}^{n-e-1} \frac{(e+i)}{n} \binom{n}{i} p^i (1-p)^{n-i}$$

$$+ \frac{1}{P_W(e)} \sum_{i=n-e}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad , \tag{C.14}$$

and for $d = 2e+2$,

$$P_{D|W}(e) \leq \frac{1}{P_W(e)} \sum_{i=e+1}^{n-e-2} \frac{(e+1+i)}{n} \binom{n}{i} p^i (1-p)^{n-i}$$

$$+ \frac{1}{P_W(e)} \sum_{i=n-e-1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \quad . \tag{C.15}$$

Fig. C.1 shows (C.12), (C.13), and (C.14) plotted for $n = 63$ and several values of e with $d = 2e+1$. The upper bound of (C.14) very nearly coincides with the lower bound of (C.12) for $p \leq 0.10$. The same results were

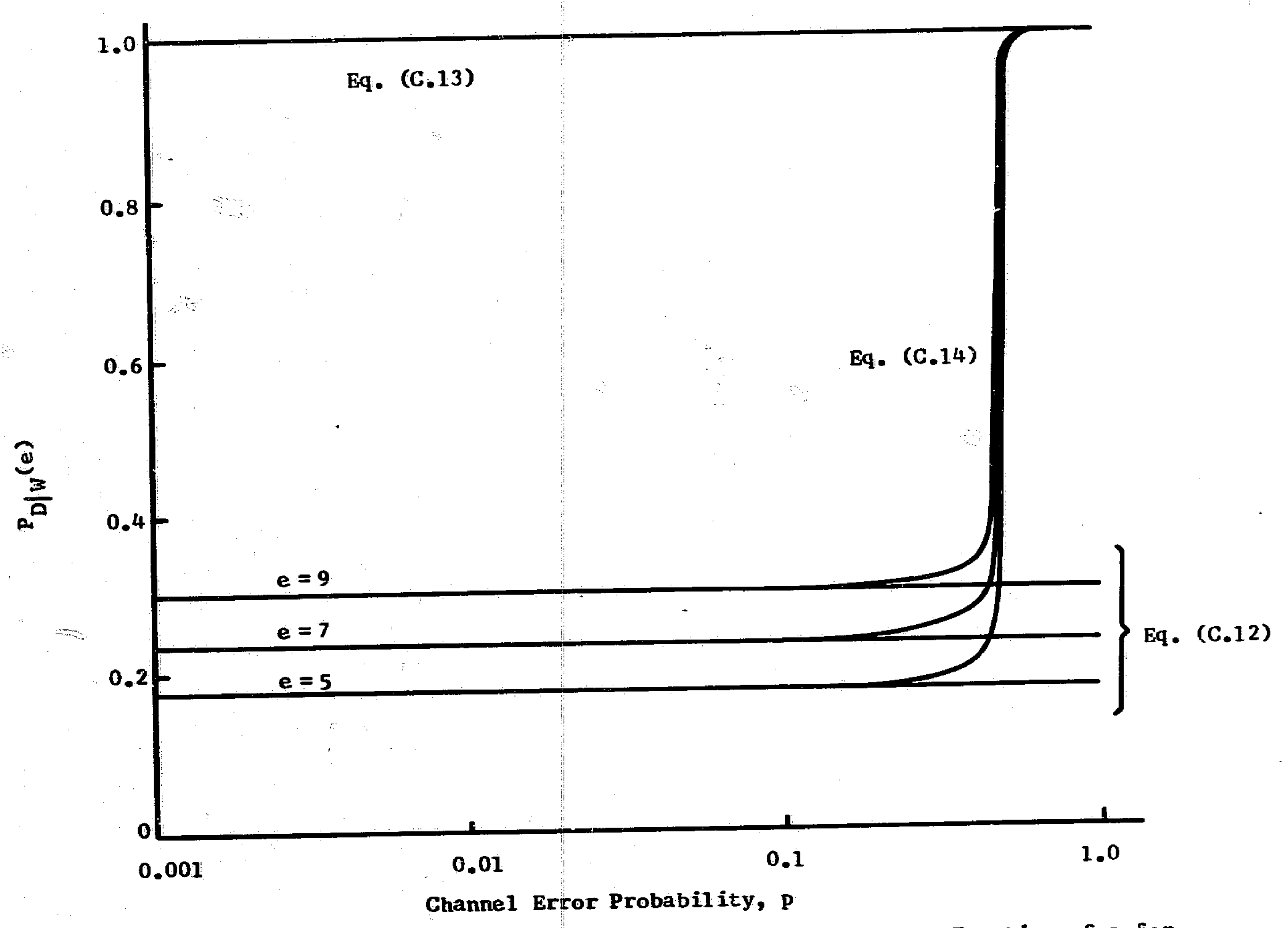

Fig. C.1. Equations (C.12), (C.13), and (C.14) Plotted as a Function of p for Several Codes of Length 63 and $d = 2e + 1$.

obtained when (C.12), (C.13), and (C.15) were plotted for a number of codes having different lengths and minimum distances. This indicates that the bounds given by (C.8) and (C.9) are very tight at least for $p \leq 0.10$ which is the region of most interest.

An example in which knowledge of $P_D(e)$ is important is the concatenated coding system for the Gaussian channel consisting of a bi-orthogonal inner code and a Reed-Solomon outer code. If the information to be transmitted is grouped into characters each of which is transmitted by a single inner code word, the error probability of interest is the $P_D(e)$ for the outer code. For example, for an inner code with $k = 6$ information bits per code word, the combination of inner coder, Gaussian channel, and inner decoder can be thought of as a superchannel with $q = 2^6 = 64$ inputs and 64 outputs. If the word error probability of the bi-orthogonal inner code is denoted by p, this quantity is also the channel error probability for transmission through the superchannel. If a (63,49) outer code with $d = 15$ is used, $P_D(e)$ can be upper bounded by (C.8) and $P_W(e)$ is given by (C.3). The values of p as a function of the energy-to-noise ratio on the Gaussian channel are given in Golomb.[87] In Fig. C.2, $P_W(e)$ and $P_D(e)$ are plotted as a function of E_b/N_o, where E_b represents energy per information bit and N_o represents single-sided noise spectral density. The difference between the two curves is fairly significant. For example, for an error probability of 10^{-4}, using the $P_W(e)$ criterion requires an energy-to-noise ratio of 3.55 dB whereas if the $P_D(e)$ criterion is used the required energy-to-noise ratio is 3.25 dB or a difference of 0.30 dB. Also in this region of the curves, $P_D(e) \approx 0.24\ P_W(e)$.

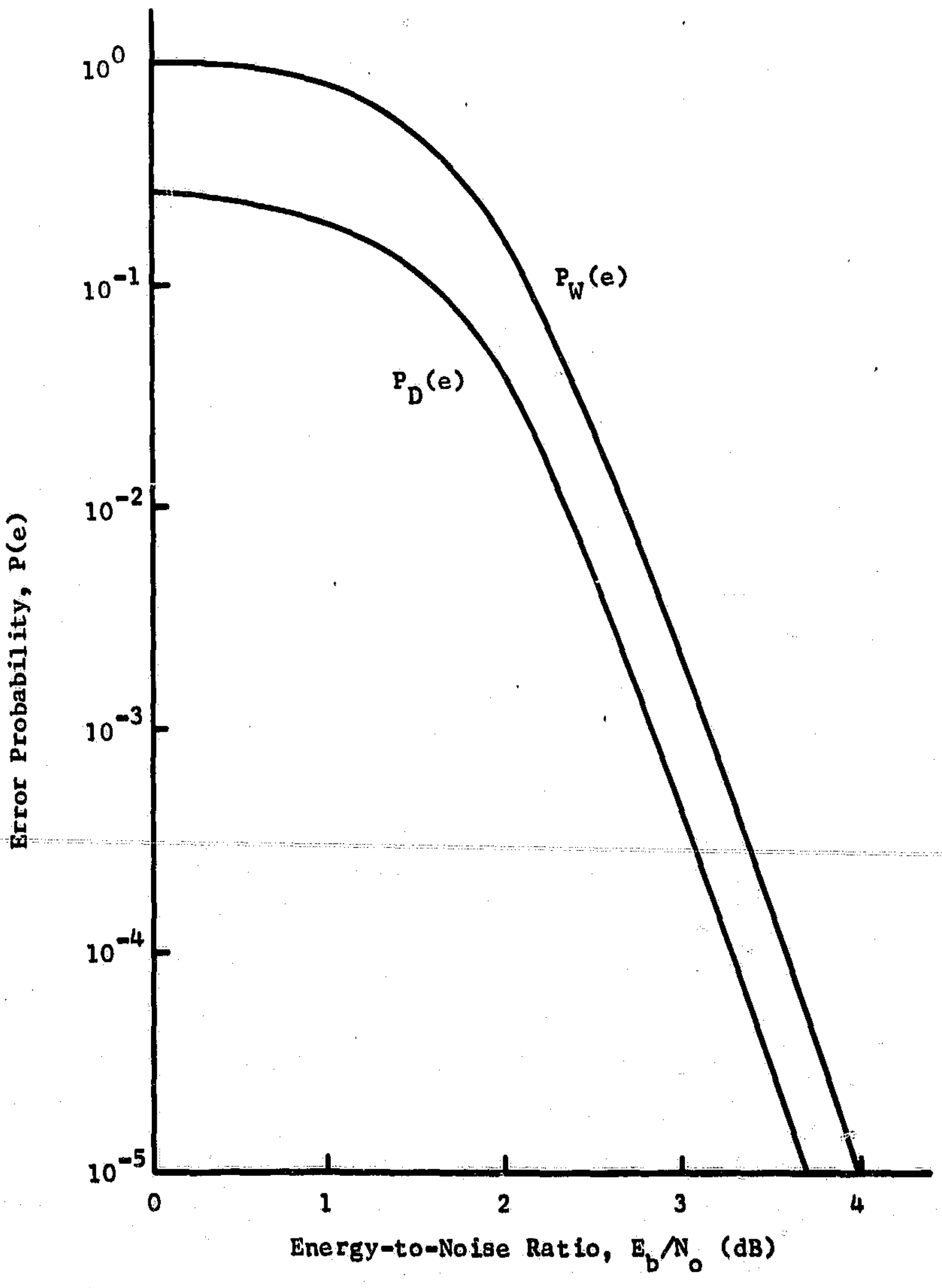

Fig. C.2. Performance of a Concatenated Coding System Using a (32,6) Bi-orthogonal Inner Code and a (63,49) Reed-Solomon Outer Code.

APPENDIX D

CALCULATION OF ERROR EXPONENTS FOR CONCATENATION SCHEMES WITH INNER-CODE FEEDBACK

In this section the error exponents for the three concatenation schemes discussed in Chapter VIII will be computed. These concatenation schemes use inner coding methods which utilize a feedback channel. The inner coding methods are sequential decision feedback, Wyner's modification of the Schakwijk-Kailath feedback scheme, and Kramer's feedback scheme which have error exponents given by (8.14), (8.17), and (8.20), respectively.

For sequential decision feedback the error exponent is given by (8.14). If this exponent is substituted into (6.7), the upper-bound concatenation exponent is

$$e_{C1}(r_1) = \max_{rr^* = r_1} (1-r)e_1(r^*) . \tag{D.1}$$

Denoting the quantity to be maximized by $F_1(r,r^*)$,

$$F_1(r,r^*) = \begin{cases} (1-r)\left[2(\sqrt{2}-1) - r^*\right] , & 0 \le r^* \le 1/\sqrt{2} , \\ (1-r)\left[\sqrt{1/r^*} - \sqrt{r^*}\right]^2 , & 1/\sqrt{2} \le r^* \le 1 . \end{cases} \tag{D.2}$$

The quantity $F_1(r,r^*)$ is to be maximized subject to the constraint

$$g = rr^* = r_1 . \tag{D.3}$$

The maximization is performed in a straightforward manner by using Lagrange multipliers. It is found that the first segment of (D.2) is maximized by taking

$$r = \frac{\sqrt{r_1}}{\sqrt{2}\left[\sqrt{2}-1\right]^{1/2}} \tag{D.4}$$

and

$$r^* = \left[2r_1(\sqrt{2}-1)\right]^{1/2} \quad . \qquad (D.5)$$

The second segment is maximized by taking

$$r = 1/4 + \left[r_1/2 + 1/16\right]^{1/2} \qquad (D.6)$$

and

$$r^* = \left[2r_1 + 1/4\right]^{1/2} - 1/2 \quad . \qquad (D.7)$$

Then the resulting upper-bound error exponent is

$$e_{C1}(r_1) = \begin{cases} \left[(2\sqrt{2}-2)^{1/2} - \sqrt{r_1}\right]^2 & , 0 \le r_1 \le \dfrac{1}{4(\sqrt{2}-1)} , \\ \dfrac{\left[3/2 - (2r_1 + 1/4)^{1/2}\right]^3}{2(2r_1 + 1/4)^{1/2} - 1} & , \dfrac{1}{4(\sqrt{2}-1)} \le r_1 \le 1 . \end{cases} \qquad (D.8)$$

The lower-bound concatenation error exponent determined by substituting (8.14) into (6.6) is

$$e_{CL1}(r_1) = \max_{rr^* = r_1} (1-r)\left[\frac{e_1(r^*) + \min\left[e_1(r^*), r^*\right]}{2}\right] \quad . \qquad (D.9)$$

The function $e_1(r^*)$ is strictly decreasing while r^* is strictly increasing, and since $e_1(r^*) = r^*$ at $r^* = \sqrt{2}-1$,

$$\min\left[e_1(r^*), r^*\right] = \begin{cases} r^* & , 0 \le r^* \le \sqrt{2}-1 , \\ 2(\sqrt{2}-1) - r^* & , \sqrt{2}-1 \le r^* \le 1/\sqrt{2} , \\ \left[\sqrt{1/r^*} - \sqrt{r^*}\right]^2 , & 1/\sqrt{2} \le r^* \le 1 . \end{cases} \qquad (D.10)$$

If the quantity to be maximized in (D.9) is denoted by $F_2(r,r^*)$, then

$$F_2(r,r^*) = \begin{cases} (1-r)(\sqrt{2}-1) & , 0 \le r^* \le \sqrt{2}-1 , \\ (1-r)\left[2(\sqrt{2}-1) - r^*\right] & , \sqrt{2}-1 \le r^* \le 1/\sqrt{2} , \\ (1-r)\left[\sqrt{1/r^*} - \sqrt{r^*}\right]^2 & , 1/\sqrt{2} \le r^* \le 1 . \end{cases} \qquad (D.11)$$

The first segment of $F_2(r,r^*)$ is a strictly decreasing function of r and is independent of r^*. Maximizing this segment amounts to minimizing r with the constraint $rr^* = r_1$. This requires picking the largest value of r^* in the interval $0 \leq r^* \leq \sqrt{2}-1$, which is $r^* = \sqrt{2}-1$. The other two segments of $F_2(r,r^*)$ are the same as the two segments of $F_1(r,r^*)$ which have already been maximized. Then the resulting lower-bound exponent is

$$e_{CL1}(r_1) = \begin{cases} (\sqrt{2}-1) - r_1 & , 0 \leq r_1 \leq \frac{\sqrt{2}-1}{2} , \\ \left[(2\sqrt{2}-2)^{1/2} - \sqrt{r_1}\right]^2 & , \frac{\sqrt{2}-1}{2} \leq r_1 \leq \frac{1}{4(\sqrt{2}-1)} \\ \frac{\left[3/2 - (2r_1 + 1/4)^{1/2}\right]^3}{2(2r_1 + 1/4)^{1/2} - 1} & , \frac{1}{4(\sqrt{2}-1)} \leq r_1 \leq 1 . \end{cases} \quad (D.12)$$

For Wyner's modification of the Schalkwijk-Kailath scheme the error exponent is given by (8.17). If (8.17) is substituted into (6.7), the upper-bound concatenation exponent is

$$e_{C2}(r_1) = \max_{rr^* = r_1} (1-r) \frac{(a-1)^2}{4a} . \quad (D.13)$$

Since the quantity to be maximized in (D.13) is a strictly decreasing function of r and is independent of r^*, it is maximized by letting r^* assume its maximum value, i.e., $r^* = 1$. Thus,

$$e_{C2}(r_1) = \frac{(a-1)^2}{4a} (1 - r_1) . \quad (D.14)$$

It is easily verified that $e_2(a) = 1$ at $a = 3 + 2\sqrt{2}$. Then for $1 < a \leq 3 + 2\sqrt{2}$,

$$\min\left[e_2(a), r^*\right] = \begin{cases} r^* & , 0 \leq r^* \leq \frac{(a-1)^2}{4a} , \\ \frac{(a-1)^2}{4a} , & \frac{(a-1)^2}{4a} \leq r^* \leq 1 , \end{cases} \quad (D.15)$$

and for $a > 3 + 2\sqrt{2}$,

$$\min\left[e_2(a), r^*\right] = r^*, \; 0 \le r^* \le 1 \quad . \tag{D.16}$$

To obtain the lower-bound exponent for $1 < a \le 3 + 2\sqrt{2}$ the quantity

$$F_3(r,r^*) = \begin{cases} \frac{(1-r)}{2}\left[r^* + \frac{(a-1)^2}{4a}\right] , & 0 \le r^* \le \frac{(a-1)^2}{4a} , \\ (1-r)\,\frac{(a-1)^2}{4a} , & \frac{(a-1)^2}{4a} \le r^* \le 1 , \end{cases} \tag{D.17}$$

must be maximized subject to the constraint $rr^* = r_1$. Both segments of (D.17) are strictly decreasing functions of r. The first segment is a strictly increasing function of r*, and the second segment is independent of r*. Both segments are therefore maximized by picking the maximum value of r*. Thus, for $1 \le a \le 3 + 2\sqrt{2}$,

$$e_{CL2}(r_1) = \frac{(a-1)^2}{4a}(1-r_1), \; 0 \le r_1 \le 1 \; . \tag{D.18}$$

Similarly for $a > 3 + 2\sqrt{2}$,

$$e_{CL2}(r_1) = \max_{rr^* = r_1} \frac{(1-r)}{2}\left[r^* + \frac{(a-1)^2}{4a}\right], \; 0 \le r^* \le 1 , \tag{D.19}$$

is maximized by picking $r^* = 1$. Thus, for $a > 3 + 2\sqrt{2}$,

$$e_{CL2}(r_1) = \left[1/2 + \frac{(a-1)^2}{8a}\right](1-r_1), \; 0 \le r_1 \le 1 \quad . \tag{D.20}$$

In this case a family of exponents is obtained with each member determined by the constant "a".

For Kramer's scheme the error exponent is given by (8.20). If this exponent is substituted into (6.7), the quantity to be maximized in (6.7) is

$$F_4(r,r^*) = \begin{cases} (1-r)(N/2-r^*) & , 0 \le r^* \le \min(1,N/4) , \\ (1-r)\left[\sqrt{N}-\sqrt{r^*}\right]^2 , & \min(1,N/4) \le r^* \le 1 . \end{cases} \qquad (D.21)$$

By using Lagrange multipliers,

$$e_{C3}(r_1) = \begin{cases} \left[\sqrt{N/2}-\sqrt{r_1}\right]^2 & , 0 \le r_1 \le \frac{2}{N}\min(1,N^2/16) , \\ \left[N^{1/3}-r_1{}^{1/3}\right]^3 , & \frac{2}{N}\min(1,N^2/16) \le r_1 \le 1/N . \end{cases} \qquad (D.22)$$

For $N > 1$ the Lagrange multiplier maximization does not provide the exponent for all rates in the range $0 \le r_1 \le 1$. The remainder of the $e_{C3}(r_1)$ curve is provided by taking $r^* = 1$ in (D.21). That this provides the correct maximum is clearly seen from Fig. D.1 in which $F_4(r,r^*)$ is plotted for $N = 6$. For a given value of r^*, F_4 is a linear decreasing function of r_1 which equals $e_3(r^*)$ at $r_1 = 0$ and zero at $r_1 = r^*$. If these straight lines are plotted for all r^*, the exponent is the convex curve which is their upper envelope. At the maximum value of r_1 provided by the Lagrange multiplier maximization the straight line for $r^* = 1$ lies above all others, and it has the largest slope. Thus, for $1 \le N \le 3$, the upper-bound exponent is

$$e_{C3}(r_1) = \begin{cases} \left[\sqrt{N/2}-\sqrt{r_1}\right]^2 & , 0 \le r_1 \le N/8 , \\ \left[N^{1/3}-r_1{}^{1/3}\right]^3 , & N/8 \le r_1 \le 1/\sqrt{N} , \\ \left[\sqrt{N}-1\right]^2(1-r_1) , & 1/\sqrt{N} \le r_1 \le 1 , \end{cases} \qquad (D.23)$$

and for $N \ge 4$ it is

$$e_{C3}(r_1) = \begin{cases} \left[\sqrt{N/2}-\sqrt{r_1}\right]^2 & , 0 \le r_1 \le 2/N , \\ (N/2-1)(1-r_1) , & 2/N \le r_1 \le 1 . \end{cases} \qquad (D.24)$$

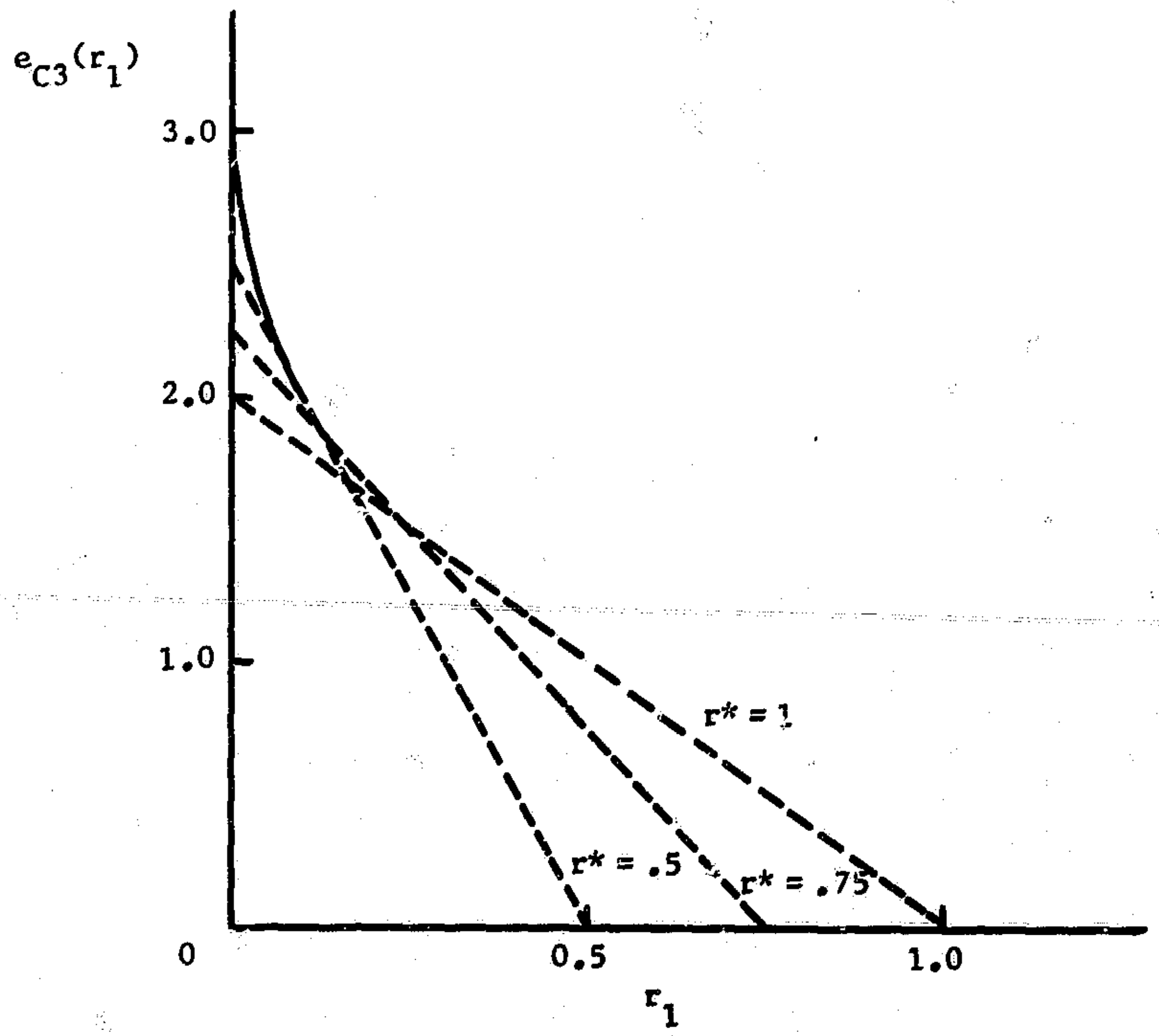

Fig. D.1. Construction of the Upper-Bound Concatenation Error Exponent for Kramer's Feedback Scheme with N = 6.

For $1 \leq N \leq 3$,

$$\min\left[e_3(r^*), r^*\right] = \begin{cases} r^*, & 0 \leq r^* \leq N/4 , \\ (\sqrt{N} - \sqrt{r^*})^2, & N/4 \leq r^* \leq 1 , \end{cases} \tag{D.25}$$

and for $N \geq 4$,

$$\min\left[e_3(r^*), r^*\right] = r^*, \; 0 \leq r^* \leq 1 . \tag{D.26}$$

If (8.20) is substituted into (6.6) and the quantity to be maximized is denoted by $F_5(r,r^*)$, then for $1 \leq N \leq 3$,

$$F_5(r,r^*), = \begin{cases} (1-r)N/4 & , 0 \leq r^* \leq N/4 , \\ (1-r)\left[\sqrt{N} - \sqrt{r^*}\right]^2, & N/4 \leq r^* \leq 1 , \end{cases} \tag{D.27}$$

and for $N \geq 4$,

$$F_5(r,r^*) = (1-r)N/4, \; 0 \leq r^* \leq 1 . \tag{D.28}$$

The first segment of (D.27) and all of (D.28) are independent of r^* and are monotonically decreasing functions of r. Picking the maximum value of r^* maximizes them. The second segment of (D.27) has already been maximized. The comments made in determining the final segments of (D.23) and (D.24) also apply to (D.27). Thus, for $1 \leq N \leq 3$,

$$e_{CL3}(r_1) = \begin{cases} (N/4 - r_1) & , 0 \leq r_1 \leq N/8 , \\ (N^{1/3} - r_1^{1/3})^3 & , N/8 \leq r_1 \leq 1/\sqrt{N} , \\ (\sqrt{N}-1)^2(1-r_1), & 1/\sqrt{N} \leq r_1 \leq 1 , \end{cases} \tag{D.29}$$

and for $N \geq 4$,

$$e_{CL3}(r_1) = (1-r_1)N/4, \; 0 \leq r_1 \leq 1 . \tag{D.30}$$

As with Wyner's modification of the Schalkwijk-Kailath scheme a family of exponents is obtained. Each distinct value of N generates a different set of exponents.

APPENDIX E

THE DISTRIBUTION OF BURST LENGTHS ON A GILBERT CHANNEL

In this section the distribution of burst lengths, $P_b(m,n)$, $m = 0, 1, 2, \cdots, n$, for the generalized Gilbert channel discussed in Chapter IX will be calculated. Computation of $P_b(m,n)$ requires development of the recurrence probabilities and the covariance function of the noise digits using recurrent-events theory (See Feller[88]). These are generalizations of the recurrence probabilities and the covariance function which Gilbert developed.[89]

If f_K denotes the conditional probability given B that the first return to B will happen at step K,

$$f_K = P(G^{K-1}B \mid B) \quad . \tag{E.1}$$

Then $f_1 = q$ and $f_K = pPQ^{K-2}$ for $K \geq 2$. The generating function of these probabilities is

$$F(t) = \sum_{K=1}^{\infty} f_K t^K = qt + \frac{pPt^2}{1-Qt} \quad . \tag{E.2}$$

Gilbert shows that the probability $s(K,m)$ of exactly m returns to B in K steps (but not necessarily a return on step K) has the generating function

$$\sum_{K=1}^{\infty} s(K,m)t^K = \left[1 + \frac{pt}{1-Qt}\right] \left[F(t)\right]^m \quad . \tag{E.3}$$

If $u_1(K)$ denotes the conditional probability of K or more zeros following an excursion into B, i.e.,

$$u_1(K) = P(O^K \mid B) \quad , \tag{E.4}$$

then

$$u_1(K) = \sum_{m=0}^{\infty} s(K,m)h^m k^{K-m} \quad . \tag{E.5}$$

The corresponding generating function is

$$U_1(t) = \sum_{K=0}^{\infty} u_1(K)t^K$$

$$= \left[1 + \frac{ptk}{1-Qtk}\right] \frac{1}{1 - \frac{h}{k} F(tk)} \quad , \tag{E.6}$$

or applying (E.2) to (E.6),

$$U_1(t) = \frac{1-k(p-Q)t}{1-(Qk+qh)t-hk(p-Q)t^2} \quad . \tag{E.7}$$

Letting $u_2(K) = P(O^K \mid G)$,

$$u_2(K) = P(G^K \mid G)k^K + P(G^{K-1} \mid G)\ hk^{K-1}$$

$$+ \sum_{J=1}^{K-1} k^{J-1}h\ P(G^{J-1}B \mid G)\ P(O^{K-J} \mid B)$$

$$= Q^K k^K + Q^{K-1}phk^{K-1}$$

$$+ \sum_{J=1}^{K-1} Q^{J-1}pk^{J-1}hu_1(K-J) \quad . \tag{E.8}$$

The corresponding generating function is

$$U_2(t) = \sum_{K=0}^{\infty} u_2(K)t^K \quad , \tag{E.9}$$

or applying (E.8) to (E.9) yields after some manipulation

$$U_2(t) = \frac{1 + PhtU_1(t)}{1 - Qkt} \quad . \tag{E.10}$$

The states G and B occur with probabilities

$$P(G) = \frac{p}{p + P} \tag{E.11}$$

and

$$P(B) = \frac{P}{p + P} \quad . \tag{E.12}$$

The probability of a one occurring as a noise digit is then

$$P(1) = \frac{p(1-k) + P(1-h)}{p + P} \quad . \tag{E.13}$$

The conditional probabilities of being in G and B given that a one has occurred are

$$P(G \mid 1) = \frac{p(1-k)}{p(1-k) + P(1-h)} \tag{E.14}$$

and

$$P(B \mid 1) = \frac{P(1-h)}{p(1-k) + P(1-h)} \quad . \tag{E.15}$$

By using (E.14) and (E.15) the conditional probability of a run of K or more zeros following a one can be written as

$$\begin{aligned} u(K) &= P(0^K \mid 1) \\ &= P(0^K \mid B)P(B \mid 1) + P(0^K \mid G)P(G \mid 1) \\ &= u_1(K)P(B \mid 1) + u_2(K)P(G \mid 1) \quad . \end{aligned} \tag{E.16}$$

The corresponding generating function is

$$U(t) = U_1(t)P(B \mid 1) + U_2(t)P(G \mid 1)$$

$$= \frac{\left[Pk(1-h)(p-Q) + ph(1-k)(P-q)\right] t + \left[P(1-h) + p(1-k)\right]}{\left[P(1-h) + p(1-k)\right]\left[1 - (Qk+qh)t - hk(p-Q)t^2\right]} \; . \qquad \text{(E.17)}$$

By expanding the left-hand side of (E.17) in powers of t and equating coefficients of t^K on both sides of the equation, the recurrence formula

$$u(K) = (Qk+qh)u(K-1) + hk(p-Q)u(K-2) \qquad \text{(E.18)}$$

is obtained for K = 2, 3, •••. The initial values are

$$u(0) = 1 \qquad \text{(E.19)}$$

and

$$u(1) = \frac{P(1-h)(pk+qh) + p(1-k)(Ph+Qk)}{P(1-h) + p(1-k)} \; . \qquad \text{(E.20)}$$

Another probabiltiy of interest is the conditional probability of the sequence 0^K1 following a one, i.e.,

$$v(K) = P(0^K1 \mid 1) \; . \qquad \text{(E.21)}$$

Since the event 0^{K-1} is the union of $10^{K-1}1$ and 10^K, which are disjoint,

$$P(0^{K-1} \mid 1) = P(0^{K-1}1 \mid 1) + P(0^K \mid 1) \qquad \text{(E.22)}$$

or

$$u(K-1) = v(K-1) + u(K) \; . \qquad \text{(E.23)}$$

Thus the probability v(K) can be determined from u(K) and u(K+1), which can be found from (E.18), (E.19), and (E.20). The generating function for v(K) is

$$V(t) = \sum_{K=0}^{\infty} v(K)t^K \; , \qquad \text{(E.24)}$$

and application of (E.23) to (E.24) yields

$$V(t) = \frac{1}{t} + \frac{t-1}{t} U(t)$$

$$= \frac{At+B}{D(t)D_1} , \qquad (E.25)$$

where

$$A = Pk(1-h)\left[p-Q+Qh-ph\right] + ph(1-k)\left[P-q+Qk-pk\right] ,$$

$$B = P(1-h)\left[1-pk-qh\right] + p(1-k)\left[1-Ph-Qk\right] ,$$

$$D_1 = p(1-k) + P(1-h), \text{ and}$$

$$D(t) = 1-(Qk+qh)t - hk(p-Q)t^2 .$$

The covariance function of the noise digits can be formed with the aid of the generating function (E.25). This covariance function is just the joint probability

$$r(K) = P(z_0 = 1, z_K = 1) . \qquad (E.26)$$

The corresponding generating function is

$$R(t) = \sum_{K=0}^{\infty} r(K)t^K$$

$$= \sum_{m=0}^{\infty} P(1)\left[tV(t)\right]^m$$

$$= \frac{P(1)}{1-tV(t)} , \qquad (E.27)$$

The term $\left[P(1)\left[tV(t)\right]^m\right]$ in the sum generates the probabilities of finding $z_0 = z_K = 1$, with exactly m-1 of the digits $z_1, \cdots, z_{K-1}$ equal to one. Application of (E.25) to (E.27) yields

$$R(t) = \frac{P(1)D_1D(t)}{A_1t^2+B_1t+D_1} \quad , \qquad (E.28)$$

where

$$A_1 = ph(1-k)(Q-p) + Pk(1-h)(Q-p)$$

and

$$B_1 = p(1-k)(ph-Qh-1) + P(1-h)(pk-Qk-1) \quad .$$

By expanding the left-hand side of (E.28) in powers of t and equating coefficients of t^K on both sides of the equation, the recurrence formula

$$r(K) = \frac{p(1-k)(1+Qh-ph) + P(1-h)(1+Qk-pk)}{p(1-k) + P(1-h)} r(K-1)$$

$$+ \frac{ph(1-k)(p-Q) + Pk(1-h)(p-Q)}{p(1-k) + P(1-h)} r(K-2) \qquad (E.29)$$

is obtained for K = 3, 4, ···. The initial values are

$$r(0) = P(1) \quad , \qquad (E.30)$$

$$r(1) = \frac{p(1-k)(1-Qk-Ph) + P(1-h)(1-pk-qh)}{p+P} \qquad (E.31)$$

and

$$r(2) = hk(Q-p)P(1) - \frac{A_1P(1)}{D_1} - \frac{B_1r(1)}{D_1} \quad . \qquad (E.32)$$

the probability

$$w(K) = P(0^K1) \quad , \qquad (E.33)$$

introduced by Elliott, will simplify the calculations which follow.[90] Since the forward and backward state transition probabilities are identical,

$$\begin{aligned} w(K) &= P(10^K) \\ &= P(1)P(0^K \mid 1) \\ &= P(1)u(K) \quad . \end{aligned} \qquad \text{(E.34)}$$

The generating functions which have been derived, and the probabilities u(K), v(K), and r(K), are generalizations of the formulas given by Gilbert for the original channel model which he introduced. By letting k = 1 in these formulas, the corresponding formulas given by Gilbert are obtained.

The probability of the sequence $0^K1\cdots10^J$, $P(0^K1\cdots10^J)$, must be determined in order to determine $P_b(m,n)$. Therefore, as the product of three probabilities,

$$P(0^K1\cdots10^J) = P(0^K1)P(z_{m-1} \mid z_0 = 1)P(0^J \mid 1) \quad . \qquad \text{(E.35)}$$

Using (E.16), (E.26), (E.33), and (E.34) with the relation

$$P(z_0 = 1, z_{m-1} = 1) = P(z_{m-1} \mid z_0 = 1)P(1) \quad , \qquad \text{(E.36)}$$

(E.35) becomes

$$\begin{aligned} P(0^K1\cdots10^J) &= w(K)\frac{r(m-1)}{P(1)}u(J) \\ &= u(K)r(m-1)u(n-m-K) \end{aligned} \qquad \text{(E.37)}$$

for m = 1, 2, ⋯, n. Since K can assume any of the values 0,1,2,⋯, n-m, the probability, $P_b(m,n)$, of a burst of length m in a sequence of length n is

$$P_b(m,n) = \sum_{K=0}^{n-m} u(K)r(m-1)u(n-m-K) \qquad \text{(E.38)}$$

for m = 1, 2, ⋯, n. The probability of a burst of length m = 0 is

$$P_b(0,n) = P(0^n)$$

$$= u_1(n)P(B) + u_2(n)P(G)$$

$$= u_3(n) \quad . \qquad (E.39)$$

The probability $u_3(n)$ satisfies the recurrence relation (E.18) with initial values

$$u_3(0) = 1 \qquad (E.40)$$

and

$$u_3(1) = \frac{P(qh+pk) + p(Qk+Ph)}{p+P} \quad . \qquad (E.41)$$

The envelope of the distribution, $P_b(m,n)$, with $n = 63$ is shown in Fig. E.1 for two generalized Gilbert channels. Both channels have $P = .0001$, $h = .70$, and $k = .999$, but one channel has $p = .01$ while the other has $p = .001$. The channel with the smaller value of $p = P(B \to G)$ has a tendency to persist in B for longer intervals. This tendency is illustrated by Fig. E.1 since for $p = .001$ the longer burst lengths are more probable while the shorter burst lengths are less probable than those for $p = .01$. The hump in the distribution for large m is due to the probability of multiple returns to B after the channel has left B. A decrease in the distribution is noted at $m = 62$ and 63 because a burst of length 62 can have only two positions and a burst length 63 can have only one position in a sequence of length 63.

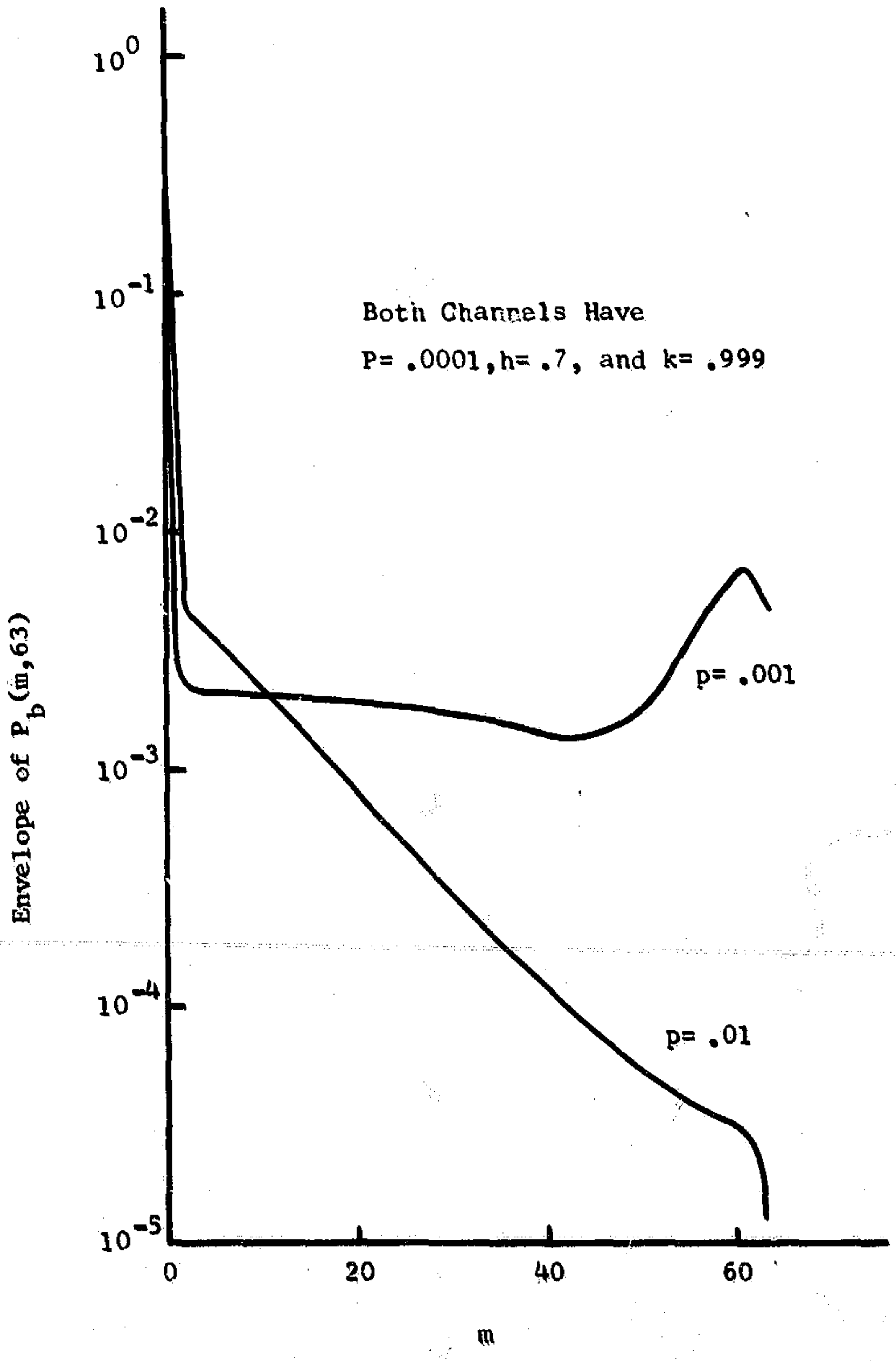

Fig. E.1. Envelope of $P_b(m,63)$ for Two Generalized Gilbert Channels.

REFERENCES

1. C. E. Shannon, "A mathematical theory of communication," Bell System Technical Journal, vol. 27, pp. 379-423, July, 1948.

2. R. G. Gallager, "A simple derivation of the coding theorem and some applications," IEEE Trans. Information Theory, vol. IT-11, pp. 3-18, January, 1965.

3. R. D. Carmichael, Introduction to the Theory of Groups of Finite Order, New York: Dover, 1956, pp. 242-286.

4. W. W. Peterson, Error-Correcting Codes, New York: Wiley, 1961, pp. 11-160.

5. Ibid., p. 90.

6. Ibid., p. 102.

7. Ibid., pp. 118, 119.

8. E. R. Berlekamp, Algebraic Coding Theory, New York: McGraw-Hill, 1968, p. 47.

9. R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error-correcting binary group codes," and "Further results on error-correcting binary group codes," Information and Control, vol. 3, pp. 68-79 and pp. 279-290, 1960.

10. A. Hocquenghem, "Codes correcteurs d'erreurs," Chiffres, vol. 2, pp. 147-156, 1959.

11. I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," J. Soc. Indust. Appl. Math., vol. 8, pp. 300-314, 1960.

12. D. Gorenstein and N. Zierler, "A class of error-correcting codes in p^m symbols," J. Soc. Indust. Appl. Math., vol. 9, pp. 207-214, 1961.

13. Peterson, op. cit., pp. 162-164.

14. G. D. Forney, Concatenated Codes, Cambridge, Mass.: MIT Press, 1966, pp. 17-19.

15. Peterson, op. cit., p. 165.

16. Berlekamp, op. cit., p. 176-199.

17. Peterson, op. cit., pp. 175-177.

18. E. R. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," IEEE Trans. Information Theory, vol. IT-11, pp. 577-579, October, 1965.

19. Berlekamp, Algebraic Coding Theory, pp. 178-188.

20. R. T. Chien, "Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes," IEEE Trans. Information Theory, vol. IT-10, pp. 357-363, October, 1964.

21. T. C. Bartee and D. I. Schneider, "Computation with finite fields," Information and Control, vol. 6, pp. 79-98, 1963.

22. Berlekamp, Algebraic Coding Theory, pp. 36-44.

23. Peterson, loc. cit.

24. Berlekamp, "On decoding binary Bose-Chaudhuri-Hocquenghem codes," loc. cit.

25. Shannon, loc. cit.

26. S. W. Golomb et al., Digital Communications with Space Applications, Englewood Cliffs, New Jersey: Prentice-Hall, 1964, pp. 48-51.

27. Ibid., pp. 17-32.

28. J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering, New York: Wiley, 1965, pp. 229-232.

29. A. J. Viterbi, "On coded phase-coherent communications," IRE Trans. Space Electronics and Telemetry, vol. SET-7, pp. 3-14, March, 1961.

30. Wozencraft and Jacobs, op. cit., pp. 342-346.

31. Forney, op. cit., p. 67.

32. Wozencraft and Jacobs, op. cit., pp. 399-401.

33. J. L. Massey, Threshold Decoding, Cambridge, Mass.: MIT Press, 1963, pp. 3-10.

34. Ibid.

35. Ibid., pp. 82-84.

36. E. J. Weldon, Jr., "Difference-set cyclic codes," Bell System Technical Journal, vol. 45, pp. 1045-1055, September, 1966.

37. L. D. Rudolph, "A class of majority logic decodable codes," IEEE Trans. Information Theory, vol. IT-13, pp. 305-307, April, 1967.

38. E. J. Weldon, Jr., "Non-primitive Reed-Muller codes," Scientific Report, AFCRL-67-0177, Air Force Cambridge Research Laboratories, Bedford, Massachusetts, pp. 7-21, February, 1967.

39. Ibid.

40. F. Neuman and D. R. Lumb, "Performance of several convolutional and block codes with threshold decoding," NASA Technical Note D-4402, pp. 12-18, March, 1968.

41. Forney, op. cit., pp. 5, 6.

42. Ibid., pp. 83-88.

43. Gorenstein and Zierler, loc. cit.

44. Berlekamp, Algebraic Coding Theory, pp. 218-228.

45. Golomb, op. cit., pp. 196-204.

46. P. Elias, "Coding for noisy channels," IRE International Convention Record, pt. 4, pp. 37-46, 1955.

47. S. Lin and H. Lyne, "Some results on binary convolutional code generators," IEEE Trans. Information Theory, vol. IT-13, pp. 134-139, January, 1967.

48. Massey, op. cit., pp. 38-52.

49. J. P. Robinson and A. J. Bernstein, "A class of binary recurrent codes with limited error propagation," IEEE Trans. Information Theory, vol. IT-13, pp. 106-113, January, 1967.

50. Neuman and Lumb, op. cit., pp. 53-59.

51. R. M. Fano, "A heuristic discussion of probabilistic decoding," IEEE Trans. Information Theory, vol. IT-9, pp. 64-74, April, 1963.

52. J. M. Wozencraft and B. Reiffen, Sequential Decoding, Cambridge, Mass.: MIT Press, 1961, pp. 25-46.

53. Wozencraft and Jacobs, loc. cit.

54. J. E. Savage, "The computation problem with sequential decoding," Ph.D. Thesis, Department of Electrical Engineering, MIT, pp. 19-54, February, 1965.

55. W. R. Wadden, D. M. Jones, and J. Bussgang, "Theoretical and computer study of convolutional codes and sequential decoding," Scientific Report AFCRL 62-756, Air Force Cambridge Research Laboratories, Bedford, Massachusetts, pp. 33-57, September, 1962.

56. D. R. Lumb, "A study of codes for deep space telemetry," presented at the IEEE International Convention and Exhibition, March, 1967.

57. C. E. Shannon, "The zero-error capacity of a noisy channel," IRE Trans. Information Theory, vol. IT-2, pp. 8-19, September, 1956.

58. A. J. Viterbi, "The effect of sequential decision feedback on communication over the Gaussian channel," Information and Control, vol. 8, pp. 80-92, 1965.

59. A. D. Wyner, "On the Schalkwijk-Kailath coding scheme with a peak energy constraint," IEEE Trans. Information Theory, vol. IT-14, pp. 129-134, January, 1968.

60. J. P. M. Schalkwijk and T. Kailath, "A coding scheme for additive noise channels with feedback-part I: no bandwidth constraint," IEEE Trans. Information Theory, vol. IT-12, pp. 172-182, April, 1966.

61. A. J. Kramer, "Analysis of communication schemes with an intermittent feedback link," Stanford University Technical Report No. 7050-11, pp. 9-48, March, 1967.

62. W. W. Peterson, "On the weight structure and symmetry of BCH codes," Scientific Reprot AFCRL-65-515, Air Force Cambridge Research Laboratories, Bedford, Massachusetts, pp. 14, 15, July, 1965.

63. R. W. Lucky, J. Salz, and E. J. Weldon, Principles of Data Communication, New York: McGraw-Hill, 1968, pp. 359, 360.

64. Viterbi, loc. cit.

65. Schalkwijk and Kailath, loc. cit.

66. Wyner, loc. cit.

67. Kramer, loc. cit.

68. J. P. Schalkwijk, "Center-of-gravity information feedback," Report No. 501, Sylvania Applied Research Laboratory, Watham, Mass., May, 1966.

69. E. N. Gilbert, "Capacity of a burst-noise channel," Bell System Technical Journal, vol. 39, pp. 1253-1265, September, 1960.

70. E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," Bell System Technical Journal, vol. 42, pp. 1977-1997, September, 1963.

71. Gilbert, loc. cit.

72. D. T. Brown and W. W. Peterson, "Cyclic codes for error detection," Proc. IRE, vol. 49, pp. 228-235, January, 1961.

73. Peterson, Error-Correcting Codes, p. 61.

74. P. Fire, "A class of multiple-error-correcting binary codes for non-independent errors," Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Laboratory, Mountain View, Calif., 1959.

75. B. Elspas, "Design and instrumentation of error-correcting codes," Technical Report RADC-TDR-62-511, Rome Air Development Center, Griffiss Air Force Base, New York, pp. 9-15, October, 1962.

76. T. Kasami, "Optimum shortened cyclic codes for burst-error correction," IEEE Trans. Information Theory, vol. IT-9, pp. 105-109, April, 1963.

77. D. L. Cohn et al., "Performance of selected block and convolutional codes on a fading HF channel," IEEE Trans. Information Theory, vol. IT-14, pp. 627-640, September, 1968.

78. Ibid.

79. Elspas, loc. cit.

80. Kasami, loc. cit.

81. Cohn et al., loc. cit.

82. F. R. Gantmacher, The Theory of Matrices, vol. I, New York: Chelsea, 1959, pp. 23-28.

83. Bartee and Schneider, loc. cit.

84. Wozencraft and Jacobs, op. cit., pp. 209-266.

85. Ibid.

86. Ibid, p. 27.

87. Golomb, loc. cit.

88. W. Feller, An Introduction to Probability Theory and Its Application, vol. I, 3rd Ed., New York: Wiley, 1968, pp. 303-335.

89. Gilbert, loc. cit.

90. Elliott, loc. cit.