ERROR-CORRECTION CODING
IN
DATA STORAGE SYSTEMS

Thesis by

Kar-Ming Cheung

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1987

(submitted May 14, 1987)

ACKNOWLEDGENEMTS

I am grateful to Dr. Wayne Stark who first introduced me to the theory of error-correction coding while I was a senior in the University of Michigan at Ann Arbor. I thank Dr. Henk Van Tiborg who taught me the first course in coding theory while he was on sabbatical at Caltech. Other professors that had a strong influence in my mathematical and engineering trainings are: J. Beck, J. Franklin, E. Posner, P. Vaidyanathan and R. Wilson.

Special thanks are due to Dr. Laif Swanson and Dr. Fabrizio Pollara in JPL for their stimulating criticism and sound advice. I also thank Dr. Mario Blaum and Dr. Paul Siegel in IBM for their constructive criticism and discussion.

I am indebted to Joanne Clark who has been very helpful and patient with me while teaching me TEX. My thanks are also due to Khaled, Eric, Chi-Chao, Lin Zhang and other colleagues in Caltech who have been very helpful in my thesis research.

My most sincere thanks, however, go to my thesis advisor Prof. Bob McEliece for his guidance and help. I thank him for his invaluable advice and encouragement throughout the course of my thesis research.

ABSTRACT

This thesis is divided into two parts. The first part is a study of the decoder error probability of linear *maximum distance separable* (MDS) codes. An exact formula for the decoder error probability of linear MDS codes is derived. The random characteristic of this class of codes is analyzed, and a lower bound for the decoder error probability is given. The second part is a study of error-correction coding in data storage systems, particularly in tape machines. The helical interleaving scheme is generalized from single channel to n parallel channels. A new code, which is specially designed for tape machines, is introduced. This code corrects more error patterns than the AXP code, and it possesses a simple hardware structure. Lastly, a class of error-correcting DC free trellis code, and a class of error-correcting RLL code are introduced.

CONTENTS

PART TWO: ERROR-CORRECTION CODING IN DATA STORAGE SYSTEMS

CHAPTER

To my wife and my parents

PART ONE

ON THE DECODER ERROR PROBABILITY OF MAXIMUM DISTANCE SEPARABLE CODES

CHAPTER I

WEIGHT DISTRIBUTION FORMULA FOR DECODABLE WORDS IN A LINEAR MDS CODE

1. Introduction

We begin with the following definitions. Let q be a positive power of a prime. Let C be a linear code of length n, dimension k, and minimum distance d. A (n, k, d) linear code C over $GF(q)$ is *maximum distance separable* (MDS) if the Singleton bound is achieved; that is, $d = n - k + 1$. An MDS code is t-error correcting if for some integer t, $2t \leq d - 1$.

The class of *Reed-Solomon* (RS) codes is a subclass of the MDS code. Reed-Solomon codes are the most widely used block codes today. Some examples are the $(255, 223)$ 16-error correcting RS code (the NASA code) in deep space communications, the $(31, 15)$ 8-error correcting RS code (the JTIDS code) in military communications, and the Cyclic Interleaving RS Code (CIRC) in compact disc industry. A detailed treatment of MDS codes, their properties and open questions about them is given in [1]. The weight distribution of a linear MDS code with the parameters n, k, d, t, and q was independently found by three groups of researchers: Assmus, Mattson and Turyn[2], Forney[3] and Kasami, Lin and Peterson[4].

In this chapter, we rederive the weight distribution formula for a linear MDS code by using the principle of inclusion and exclusion, and then extend this method to obtain the exact weight distribution formula for "decodable words" in any linear

MDS code. By decodable words, we mean all the words that are lying within distance t from a codeword. If we assume the decoder to be a bounded distance decoder, then the weight distribution formula for the decodable words can be used to find the undetected error probability for linear MDS codes. This will be discussed in detail in Chapters II and III.

This chapter is divided into 5 sections. Section 1 is a brief introduction. In Section 2, we review some basic mathematical tools that are needed to derive the formulae. In Section 3, we first derive the weight distribution formula for the number of codewords in a linear MDS code, and then we derive the weight distribution formula for the number of decodable words in a linear MDS code. In Section 4, we give some numerical examples and finally, in Section 5, we end this chapter with some concluding remarks.

2. Some Basic Tools

In this section, we review the basic tools that are required to derive the weight distribution formulae for the number of codewords in a linear MDS code and for the number of decodable words in a linear MDS code.

Let C be an (n, k) code over $GF(q)$, not necessarily linear. If we examine any set of k-1 components of the codewords, we find that there are only q^{k-1} possibilities for the q^k codewords. Thus, there must be a pair of codewords that agree on these k-1 components, and so the minimum distance d of the code must satisfy $d \leq n - k + 1$. This upper bound on d is known as the Singleton bound, and a code for which

$d = n - k + 1$ is called an MDS code. RS codes and cosets of RS codes are examples of MDS codes.

One important tool that we need is the basic combinatoric property of the MDS code. Let K be a subset of k coordinate positions of an MDS code. If two codewords were equal on K, the distance between them would be at most $n - k$. This contradicts the fact that $d = n - k + 1$. Thus, all q^k codewords are different in K. Let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ be a k-tuple of elements from $GF(q)$. From the above argument, there exists a unique codeword whose k coordinates in K equal the k components in α. We call this important fact the basic combinatorial property of MDS codes.

Another important tool that we need is the principle of inclusion and exclusion [5]. Suppose we have N objects and a number of properties $P(1), \ldots, P(n)$. Let N_i be the number of objects with property $P(i)$, and $N_{i_1,i_2,\ldots,i_r}$ be the number of objects with properties $P(i_1)$, $P(i_2)$, ..., $P(i_r)$. The number of objects $N(0)$ with none of the properties is given by the following formula :

$$(1)\quad N(0) = N - \sum_{i} N_i + \sum_{i_1<i_2} N_{i_1 i_2} + \ldots + (-1)^r \sum_{i_1<i_2\ldots<i_r} N_{i_1 i_2 \ldots i_r} + \ldots + (-1)^n N_{123\ldots n}.$$

The proof can be found in [5].

The basic combinatorial property of MDS codes and the principle of inclusion and exclusion will be referred to in the proofs in later sections.

3. Derivation of Formulae

This section is divided into three parts. In Part (a), we derive the formula for the number of codewords of weight u in a linear MDS code, using the principle of inclusion and exclusion. In Part (b), we extend this idea by deriving a general formula for the number of decodable words of weight u. Last of all, in Part (c), we simplify the key formula by using some combinatoric identities.

Part a :

Let $\bar{c}$ be some codeword of C. Let $\bar{c}$ have a Hamming weight u, $u \geq d$. Let the coordinates of codeword $\bar{c}$ be indexed by $\{0, 1, 2, \ldots, n-1\}$. Define $v = n - u$. Then $\bar{c}$ has v zeros. We now want to find the number of codewords of weight u in C having exactly v zeros at some particular v coordinates where $v = n (u = 0)$ or $v \leq n - d = k - 1 (u \geq d)$. Obviously the number of codewords of weight zero $(u = 0)$ is one — the all zero codeword. The following discussion applies only to codewords with weight $u \geq d$.

Let V be a set of v coordinates, $|V| = v$. Let $\{i_1, i_2, \ldots, i_j\} \subset \{1, 2, ..., n\} - V$ be a set of j coordinates. Define $S(i_1, i_2, \ldots, i_j) = \{\bar{c} : \bar{c} \in C$ and $\bar{c}$ has zeros in V and $\{i_1, i_2, \ldots, i_j\}\}$. For $j \leq k - v$, the number of zeros in a codeword in $S(i_1, i_2, \ldots, i_j)$ is at least $j + v \leq k$ $(j + v \leq k)$. By using the basic combinatorial property of MDS code, for each particular choice of $\{i_1, i_2, \ldots, i_j\}$ we can specify q^{k-v-j} codewords having zeros at V and $\{i_1, i_2, \ldots, i_j\}$. So

$$|S(i_1, i_2, \ldots, i_j)| = q^{k-v-j} \qquad 0 \leq j \leq k - v. \tag{2}$$

For $j \geq k-v+1$, the number of zeros in a codeword is $j+v \geq k+1$. This implies that the weight of the codeword is less than d, so $S(i_1, i_2, \ldots, i_j) = \{\bar{0}\}$. That is,

$$|S(i_1, i_2, \ldots, i_j)| = 1 \qquad k-v+1 \leq j \leq u. \tag{3}$$

Note that we choose $i_1, i_2, \ldots, i_j$ from a set of $u = n-v$ coordinates so that for every choice of j, we have $\binom{u}{j}$ $S(i_1, i_2, \ldots, i_j)$'s.

By the principle of inclusion and exclusion, the number of codewords with exactly v zeros at V equals

$$\begin{aligned}
&|S(0)| - \sum_{i_1} |S(i_1)| + \ldots + (-1)^u |S(i_1, i_2, \ldots, i_u)| \\
&= \sum_{j=0}^{k-v-1} (-1)^j \binom{u}{j} q^{k-v-j} + \sum_{j=k-v}^{u} (-1)^j \binom{u}{j} \\
&= \sum_{j=0}^{u-d} (-1)^j \binom{u}{j} q^{u-d-j+1} - \sum_{j=0}^{u-d} (-1)^j \binom{u}{j} \\
&= \sum_{j=0}^{u-d} (-1)^j \binom{u}{j} (q^{u-d-j+1} - 1).
\end{aligned}$$

We have $\binom{n}{v} = \binom{n}{u}$ ways to choose v zeros from $\{0, 1, 2, \ldots, n-1\}$. Thus, the number of codewords of weight u, which is denoted by A_u, is given by the following expression :

$$A_u = \binom{n}{u} \sum_{j=0}^{u-d} (-1)^j \binom{u}{j} (q^{u-d-j+1} - 1) \qquad d \leq u \leq n. \tag{4}$$

After deriving this relatively simple formula for the number of codewords of weight u in a linear MDS code, we proceed to derive the more complicated formula for the number of decodable words of weight u in a linear MDS code.

Part b:

Let D be the set of decodable words in an MDS code. Let V be a set of v coordinates, $|V| = v$. Let $\{i_1, i_2, \ldots, i_j\}$ be a set of j coordinates, where $\{i_1, i_2, \ldots, i_j\} \subset \{0, 1, 2, \ldots, n-1\} - V$. Define $S(i_1, i_2, \ldots, i_j) = \{\bar{d} : \bar{d} \in D$ and $\bar{d}$ has zeros in V and $\{i_1, i_2, \ldots, i_j\}\}$. We proceed to derive the weight distribution formula for the number of decodable words of weight u in a linear MDS code by using the principle of inclusion and exclusion. Our problem is now reduced to finding the cardinality of $S(i_1, i_2, \ldots, i_j)$ for all j subjected to a given V. This problem is solved with the help of the following theorems.

Theorem 1.1:

$$|S(i_1, i_2, \ldots, i_j)| = q^{u-d+1-j} V_n(t) \qquad 0 \leq j \leq u - d, \tag{5}$$

where $V_n(t) = \sum_{i=0}^{t} \binom{n}{i} (q-1)^i$.

Proof:

The argument here is similar to the derivation given in *Part a.* We note that each coset of a linear MDS code is also an MDS code. Also, since all words lying within the Hamming spheres (with volume $V_n(t)$) that surround codewords are decodable words, we have $V_n(t)$ disjoint cosets that contain decodable words. From the basic combinatorial property of the MDS code we can, for each particular choice of $\{i_1, i_2, \ldots, i_j\}$, specify $q^{k-v-j} = q^{u-d+1-j}$ decodable words to each of these cosets. Thus, we have altogether $q^{u-d+1-j} V_n(t)$ decodable words having zeros at V and $\{i_1, i_2, \ldots, i_j\}$. This completes the proof. ■

Theorem 1.2

$$(6) \qquad |S(i_1, i_2, \ldots, i_j)| = \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} \sum_{i=0}^{w-d+u-j} (-1)^i \binom{w}{i} (q^{w-d+u-j-i+1} - 1)$$
$$\times \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} + \sum_{i=0}^{t} \binom{u-j}{i} (q-1)^i$$

for $u - d + 1 \leq j \leq u - d + t$

Proof :

For $u - d + 1 = k - v \leq j$, the number of zeros in a decodable word is equal to $v + j \geq k$. Since $\bar{d}$ is a decodable word, $\bar{d}$ can be uniquely decomposed into a codeword $\bar{c}$ and an error pattern $\bar{e}$ with weight that is less than or equal to t. If we "project" $\bar{c}$ onto $V \cup \{i_1, i_2, \ldots, i_j\}$, then the result will be a certain $(v + j, k)$ code. Since the parent code has a minimum distance $d = n - k + 1$, the new code must have a minimum distance $d' \geq d - (n - v - j) = (v + j) - k + 1$. Since it is impossible for d' of the $(v + j, k)$ code to be greater than $(v + j) - k + 1$ (because of the Singleton bound), d' must be equal to $d - (n - v - j) = (v + j) - k + 1$.

If $\bar{c} + \bar{e}$ vanishes on $V \cup \{i_1, i_2, \ldots, i_j\}$, then $\bar{c}$ must have weight that is less than or equal to t on $V \cup \{i_1, i_2, \ldots, i_j\}$. Let w be the weight of $\bar{c}$ on $V \cup \{i_1, i_2, \ldots, i_j\}$. From the above argument we also know that C, when restricted to $V \cup \{i_1, i_2, \ldots, i_j\}$, is a linear $(v+j, k)$ MDS code with a minimum distance $d-(n-v-j) = (v+j)-k+1$. Thus, w is either 0 (in the case of the all-zero codeword) or is between $d - u + j$ and t. So the number of codewords of weight w in the $(v + j, k)$ MDS code is (by using Equation (4))

$$\binom{n-u+j}{w} \sum_{i=0}^{w-(d-(u-j))} (-1)^i \binom{w}{i} (q^{w-(d-(u-j))-i+1} - 1)$$

for $d-u+j \leq w \leq t$ and 1 for w=0. For each codeword $\bar{c}$ with weight w in $V \cup \{i_1, i_2, \ldots, i_j\}$, where $d' \leq w \leq t$ ($d' = v+j-k+1$), we must count the number of $\bar{e}$'s such that $\bar{c}+\bar{e}$ vanishes on $V \cup \{i_1, i_2, \ldots, i_j\}$. Suppose that $\bar{e}$ has weight $s \geq w$. $\bar{e}$ must match $\bar{c}$ exactly on $V \cup \{i_1, i_2, \ldots, i_j\}$, but the $s-w$ other nonzero components can be arbitrarily placed outside $V \cup \{i_1, i_2, \ldots, i_j\}$. Then the total number of $\bar{e}$'s for a given $\bar{c}$ of weight w on $V \cup \{i_1, i_2, \ldots, i_j\}$ is $\sum_{s=w}^{t} \binom{u-j}{s-w}(q-1)^{s-w}$. When w=0, all components of $\bar{e}$ must lie outside the set $V \cup \{i_1, i_2, \ldots, i_j\}$. So there are $\sum_{i=0}^{t} \binom{u-j}{i}(q-1)^i$ $\bar{e}$'s for the case w=0. Combining the above results, we obtain the theorem. ■

Theorem 1.3:

$$(7) \quad |S(i_1, i_2, \ldots, i_j)| = \sum_{i=0}^{t} \binom{u-j}{i}(q-1)^i \qquad \text{for} \quad u-d+t+1 \leq j \leq u-t-1$$

Proof :

For $k-v+t \leq j \leq u-t-1$, the number of zeros in a decodable word is greater than or equal to $k+t$ but less than or equal to $n-t-1$. Thus any decodable words in $S(i_1, i_2, \ldots, i_j)$ have weight that is less than or equal to $d-t-1$. It is not hard to see that the element of $S(i_1, i_2, \ldots, i_j)$ cannot be decoded into a codeword of weight other than $\bar{0}$. Therefore, $S(i_1, i_2, \ldots, i_j)$ contains all words of weight that is less than or equal to t in the coordinates $\{0, 1, \ldots n-1\}$ - $(V \cup \{i_1, i_2, \ldots, i_j\})$. This completes the proof. ■

Theorem 1.4:

(8) $$|S(i_1, i_2, \ldots, i_j)| = q^{u-j} \qquad u - t \leq j \leq u$$

Proof :

Since j is greater than or equal to $u - t$, the number of zeros is equal to $v + j$ and is greater than or equal to $n - t$. Therefore, the number of nonzero components is less than or equal to t. Thus, all words with zeros on $V \cup \{i_1, i_2, \ldots, i_j\}$ are decodable and this completes the proof. ■

As in *Part a*, we choose $i_1, i_2, \ldots, i_j$ from $v = n - u$ coordinates. Thus, for every choice of j, we have $\binom{u}{j}$ $S(i_1, i_2, \ldots, i_j)$'s. Denote $N_j = \binom{u}{j}|S(i_1, i_2, \ldots, i_j)|$. Again, by the principle of inclusion and exclusion, we see that the number of decodable words which have exactly $v = n - u$ zeros at V equals $\sum_{j=0}^{u}(-1)^j N_j$. However, we have $\binom{n}{u} = \binom{n}{v}$ ways to choose v zeros from $0, 1, \ldots, n-1$. Thus, the number of decodable words of weight u is given by

(9) $$D_u = \binom{n}{u} \sum_{j=0}^{u}(-1)^j N_j \qquad \text{for} \quad d - t \leq u \leq n$$

Part c:

The weight enumerator formula that we have just derived is complicated and clumsy. There are four different expressions for N_j's, and these expressions are combined together by the inclusion and exclusion formula. The following theorem will show that the weight distribution formula for the number of decodable words in a linear MDS code can be simplified, and there are only two expressions for the N_j's.

Theorem 1.5:

$$(10) \qquad A = \sum_{j=0}^{u-t-1} (-1)^j \binom{u}{j} \sum_{i=0}^{t} \binom{u-j}{i} (q-1)^i + \sum_{j=u-t}^{u} (-1)^j \binom{u}{j} q^{u-j} = 0$$

Proof:

$$\begin{aligned} A &= \sum_{j=0}^{u-t-1} (-1)^j \binom{u}{j} [q^{u-j} - \sum_{i=t+1}^{u-j} \binom{u-j}{i} (q-1)^i] + \sum_{j=u-t}^{u} (-1)^j \binom{u}{j} q^{u-j} \\ &= \sum_{j=0}^{u} (-1)^j \binom{u}{j} q^{u-j} - \sum_{j=0}^{u-t-1} (-1)^j \binom{u}{j} \sum_{i=t+1}^{u-j} \binom{u-j}{i} (q-1)^i \\ &= (q-1)^u - \sum_{i=t+1}^{u} (q-1)^i \sum_{j=0}^{u-i} \binom{u-j}{i} \binom{u}{j} (-1)^j \\ &= (q-1)^u - (q-1)^u - \sum_{i=t+1}^{u-1} (q-1)^i \sum_{j=0}^{u-i} \binom{u-j}{i} \binom{u}{j} (-1)^j \end{aligned}$$

Notice that $\binom{u}{j}\binom{u-j}{i} = \binom{u}{i}\binom{u-i}{j}$ and $\sum_{j=0}^{u-i} \binom{u-i}{j}(-1)^j = 0$, then

$$\sum_{j=0}^{u-i} \binom{u-j}{i} \binom{u}{j} (-1)^j = \binom{u}{i} \sum_{j=0}^{u-i} \binom{u-i}{j} (-1)^j = 0 \qquad t+1 \le i \le u-1$$

Thus, $A = 0$ and the theorem is proved. ∎

With *Theorem 1.5* and equation (5), (6), (7), (8) and (9), the weight enumerator formula can be simplified as follows :

$$(11)\quad D_u = \binom{n}{u} \sum_{j=0}^{u-d+t} (-1)^j N_j \qquad \text{for} \quad d-t \le u \le n$$

$$(12)\quad N_j = \binom{u}{j} [q^{u-d+1-j} V_n(t) - \sum_{i=0}^{t} \binom{u-j}{i} (q-1)^i] \qquad \text{for} \quad 0 \le j \le u-d$$

$$(13)\quad N_j = \binom{u}{j} [\sum_{w=d-u+j}^{t} \binom{n-u+j}{w} \sum_{i=0}^{w-d+u-j} (-1)^i \binom{w}{i} (q^{w-d+u-j-i+1} - 1)$$
$$\sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w}] \qquad \text{for} \quad u-d+1 \le j \le u-d+t.$$

4. Examples

Example 1 (4,2) MDS code over $GF(5)$ with t=1.

Weight	# of decodable words	Upper bound[6]
0	1	-
1	16	-
2	48	48
3	192	272
4	168	272

Total number of decodable words = $q^k V_n(t) = 425$.

Table 1

Example 2 (6,3) MDS code over $GF(4)$ with t=1.

Weight	# of decodable words	Upper bound[6]
0	1	-
1	18	-
2	0	-
3	180	180
4	405	855
5	378	1026
6	234	513

Total number of decodable words = $q^k V_n(t) = 1216$.

Table 2

5. Remarks

The formula for the number of decodable words of weight u, where $d-t \leq u \leq n$, is derived in the previous sections. If we set t=0, then we get back the weight enumerator for linear MDS code — Equation (4). In the case of $u = d - t$, for example, we have

$$D_{d-t} = \binom{n}{d}\binom{d}{t}(q-1),$$

and the answer is consistent with the result derived in [6].

The formula is a bit clumsy, but can be easily implemented by computer program. We include in the appendix a source listing of the computer program that calculates the number of decodable words and the decoder error probability for a linear MDS code.

CHAPTER II

DECODER ERROR PROBABILITY OF A LINEAR MDS CODE

1. Number of Decodable Words vs. Decoder Error Probability

Let C be an (n, k, d) linear code capable of correcting t errors. When a codeword $\bar{c} \in C$ is transmitted over a communication channel, channel noise may corrupt the transmitted signals. As a result, the receiver receives the corrupted version of the transmitted codeword $\bar{c} + \bar{e}$, where $\bar{e}$ is an error pattern of weight u. If $u \leq t$, then the decoder on the receiver's end detects and corrects the error $\bar{e}$ and recovers $\bar{c}$. If $u > t$, then the decoder fails and it either

i) detects the presence of the error pattern but is unable to correct it, or

ii) misinterprets (miscorrects) the received pattern $\bar{c} + \bar{e}$ for some other codeword $\bar{c}'$ if the received pattern falls into the Hamming sphere of $\bar{c}'$.

Case (ii) is, in most cases, more castastrophic than case (i). This can occur (with a nonzero probability) when an error pattern $\bar{e}$ is of weight $u \geq d - t$. Let us further assume that all error patterns of weight u are equally probable, and let us denote the decoder error probability given that an error pattern of weight u occurs by $P_E(u)$[7]. It is not hard to see that $P_E(u)$ is given by the following expression:

$$P_E(u) = \frac{D_u}{\binom{n}{u}(q-1)^u} \qquad d - t \leq u \leq n. \tag{1}$$

That is, $P_E(u)$ is the ratio of the number of decodable words of weight u to the number of words of weight u in the whole vector space. Thus, the problem of finding the $P_E(u)$'s is essentially the same as the problem of finding the weight distribution of the set of decodable words. Equations (11), (12) and (13) of Chapter I and Equation (1) of this chapter together enable us to find the exact decoder error probability of a linear MDS code.

Let the probability that a completely random error pattern will cause decoder error be denoted by Q. It is the ratio of the number of decodable words to the cardinality of the whole vector space. That is,

$$Q = \frac{(q^k - 1)V_n(t)}{q^n} \simeq q^{-r}V_n(t), \tag{2}$$

where $r = n - k$ is the code's redundancy and $V_n(t) = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$ is the volume of a Hamming sphere of radius t. It is shown in the next section that if $q \geq n$, which is generally true, then $P_E(u)$ approaches Q very rapidly as u increases.

2. Examples and Observations

Two well-known examples of linear MDS codes — the NASA code and the JTIDS code — are tabulated in Table 1 and Table 2, respectively. From the above two examples, we observe that $P_E(u)$ approaches the constant Q as u increases. In fact, $P_E(u)$ approaches Q rapidly for $u \ll n$. In the case of large q and $q \geq n$, $P_E(u)$ approaches Q even for $u < d$. The $P_E(u)$ and Q of the NASA code agree to eight significant digits for $u \geq 26 < d = 33$. If $P_E(u)$ and Q are interpreted combinatorically as ratios, then we have the following relationship:

$$\frac{\#\text{ of decodable words of weight u}}{\#\text{ of vectors of weight u}} \rightarrow \frac{\#\text{ of decodable words}}{\#\text{ of words in vector space}}$$

This astonishing relationship cited above infers that a linear MDS code, which possesses rigid algebraic and combinatoric structures, behaves (in some sense) like a random code with no structure at all. Some laws of large number somehow come into play.

In order to describe analytically how fast $P_E(u)$ approaches Q when u is large, an upper bound of the expression $|\frac{P_E(u)}{Q} - 1|$ is derived in the following paragraphs. This upper bound is denoted by $\mathbf{U}(u)$, where $u \geq d$. It will be shown that $\mathbf{U}(u)$ approaches ϵ as u increases, where ϵ is a very small number close to 0.

As in Chapter I, let D_u denote the exact number of decodable words of weight u. Let N_j's be the corresponding terms in the inclusion and exclusion formula of D_u as expressed in Equations (11), (12) and (13) of Chapter I. Let $\hat{D}_u$ denote the estimated number of decodable words of weight u. Let $\hat{N}_j$'s be the corresponding terms in the inclusion and exclusion formula of $\hat{D}_u$. The expression of $\hat{N}_j$, $0 \leq j \leq u$, is constructed by extrapolating the first term on the right-hand side of Equation

(12) of Chapter I from $0 \le j \le u-d$ to $0 \le j \le u$. We now have the following equations for $\hat{D}_u$ and $\hat{N}_j$:

$$\hat{D}_u = \binom{n}{u} \sum_{j=0}^{u} (-1)^j \hat{N}_j \qquad d-t \le u \le n \tag{3}$$

$$\hat{N}_j = \binom{u}{j} q^{u-d+1-j} V_n(t) \qquad 0 \le j \le u. \tag{4}$$

Now we want to find an upper bound, denoted by U_j, for N_j in Equation (13) of Chapter I for $u-d+1 \le j \le u-d+t$.

$$\begin{aligned}
N_j &= \binom{u}{j} \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} \sum_{i=0}^{w-d+u-j} (-1)^i \binom{w}{i} (q^{w-d+u-j-i+1} - 1) \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} \\
&= \binom{u}{j} \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} (q-1) [\sum_{i=0}^{w-d+u-j} (-1)^i \binom{w-1}{i} q^{w-d+u-j-i}] \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} \\
&\le \binom{u}{j} \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} (q-1) q^{-d+u-j} q^w \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} \\
&= \binom{u}{j} \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} q^{u-j-d+1} (\frac{q}{q-1})^{w-1} (q-1)^w \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} \\
&\le q^{u-j-d+1} (\frac{q}{q-1})^{t-1} \binom{u}{j} \sum_{w=d-u+j}^{t} \binom{n-u+j}{w} (q-1)^w \sum_{s=w}^{t} \binom{u-j}{s-w} (q-1)^{s-w} \\
&= q^{u-j-d+1} (\frac{q}{q-1})^{t-1} \binom{u}{j} \sum_{s=d-u+j}^{t} (q-1)^s \sum_{w=d-u+j}^{s} \binom{n-u+j}{w} \binom{u-j}{s-w} \\
&\le q^{u-j-d+1} (\frac{q}{q-1})^{t-1} \binom{u}{j} V_n(t) \\
&= (\frac{q}{q-1})^{t-1} \hat{N}_j \stackrel{\text{def}}{=} U_j
\end{aligned}$$

Note that $\binom{u}{j} q^{u-d+1-j} V_n(t) = \hat{N}_j < U_j$, and so $U_j \ge \max\{N_j, \hat{N}_j\}$. Also, with the additional assumption that $q \ge n$, which is generally true, U_j is a descending function of j.

NASA Code ($(255, 223)$ *RS code.* $q = 256$ $t = 16$)

$$P_E(17) = 9.4641648 \times 10^{-15}$$

$$P_E(18) = 1.9130119 \times 10^{-14}$$

$$P_E(19) = 2.4010995 \times 10^{-14}$$

$$P_E(20) = 2.6598044 \times 10^{-14}$$

$$P_E(21) = 2.6017177 \times 10^{-14}$$

$$P_E(22) = 2.6076401 \times 10^{-14}$$

$$P_E(23) = 2.6087596 \times 10^{-14}$$

$$P_E(24) = 2.6088773 \times 10^{-14}$$

$$P_E(25) = 2.6088880 \times 10^{-14}$$

$$P_E(26) = 2.6088888 \times 10^{-14}$$

$$P_E(27) = 2.6088888 \times 10^{-14}$$

$$P_E(28) = 2.6088888 \times 10^{-14}$$

$$P_E(29) = 2.6088888 \times 10^{-14}$$

$$P_E(30) = 2.6088888 \times 10^{-14}$$

. . .

. . .

etc.

Table 1

JTIDS Code ($(31,15)$ *RS code.* $q = 32$ $t = 8$)

$$P_E(\ 9) = 3.7493431 \times 10^{-7}$$

$$P_E(10) = 1.4392257 \times 10^{-6}$$

$$P_E(11) = 2.9507015 \times 10^{-6}$$

$$P_E(12) = 4.3287703 \times 10^{-6}$$

$$P_E(13) = 5.1888955 \times 10^{-6}$$

$$P_E(14) = 5.5466000 \times 10^{-6}$$

$$P_E(15) = 5.6291887 \times 10^{-6}$$

$$P_E(16) = 5.6296979 \times 10^{-6}$$

$$P_E(17) = 5.6255686 \times 10^{-6}$$

$$P_E(18) = 5.6256673 \times 10^{-6}$$

$$P_E(19) = 5.6259065 \times 10^{-6}$$

$$P_E(20) = 5.6258313 \times 10^{-6}$$

$$P_E(21) = 5.6258455 \times 10^{-6}$$

$$P_E(22) = 5.6258434 \times 10^{-6}$$

$$P_E(23) = 5.6258437 \times 10^{-6}$$

$$P_E(24) = 5.6258437 \times 10^{-6}$$

.

etc.

Table 2

Now let us consider the second term on the right-hand side of Equation (12) of Chapter I, and denote it by $\Theta(u)$. We want to find an upper bound for $\Theta(u)$.

$$\begin{aligned}
\Theta(u) &= \sum_{j=0}^{u-d}(-1)^j\binom{u}{j}\sum_{i=0}^{t}\binom{u-j}{i}(q-1)^i \\
&= \sum_{j=0}^{u-d}(-1)^j\sum_{i=0}^{t}\binom{u}{i}\binom{u-i}{j}(q-1)^i \\
&= \sum_{i=0}^{t}\binom{u}{i}(q-1)^i\sum_{j=0}^{u-d}(-1)^j\binom{u-i}{j} \\
&\le \sum_{i=0}^{t}\binom{u}{i}(q-1)^i\sum_{j=0}^{u-d}\binom{u-i}{j} \\
&\le \sum_{i=0}^{t}\binom{u}{i}(q-1)^i\sum_{j=0}^{u-i}\binom{u-i}{j} \\
&= \sum_{i=0}^{t}\binom{u}{i}(q-1)^i 2^{u-i} \\
&= 2^u\sum_{i=0}^{t}\binom{u}{i}(\frac{q-1}{2})^i \\
&= 2^u V_u^*(t),
\end{aligned}$$

where $V_u^*(t) = \sum_{i=0}^{t}\binom{u}{i}(\frac{q-1}{2})^i$.

We then want to find an upper bound of $|D_u - \hat{D}_u|$, where $d \le u \le n$. We have

$$\begin{aligned}
|D_u - \hat{D}_u| &= |\binom{n}{u}[\sum_{j=0}^{u-d+t}(-1)^j N_j - \sum_{j=0}^{u}(-1)^j\hat{N}_j]| \\
&\le \binom{n}{u}[|\sum_{j=u-d+1}^{u-d+t}(-1)^j N_j - \sum_{j=u-d+1}^{u}(-1)^j\hat{N}_j| + \Theta(t)] \\
&= \binom{n}{u}[|\sum_{j=u-d+1}^{u}(-1)^j(N_j - \hat{N}_j)| + \Theta(u)] \qquad \text{(set } N_j\text{=0 for } u-d+t+1 \le j \le u\text{)} \\
&\le \binom{n}{u}[\sum_{j=u-d+1}^{u}|N_j - \hat{N}_j| + \Theta(u)] \\
&\le \binom{n}{u}[\sum_{j=u-d+1}^{u}\max\{N_j, \hat{N}_j\} + \Theta(u)] \\
&\le \binom{n}{u}[\sum_{j=u-d+1}^{u}U_j + \Theta(u)]
\end{aligned}$$

$$\leq \binom{n}{u}[dU_{u-d+1} + \Theta(u)] \qquad (U_j \text{ is a descending function})$$
$$= \binom{n}{u}[d(\frac{q}{q-1})^{t-1}\binom{u}{d-1}V_n(t) + 2^u V_u^*(t)].$$

We are finally ready to derive an upper bound for $|\frac{P_E(u)}{Q} - 1|$. By the definition of $\hat{D}_u$ in Equations (3) and (4), it is not hard to see that

$$\hat{D}_u = \binom{n}{u}\sum_{j=0}^{u}(-1)^j\binom{u}{j}q^{u-d+1-j}V_n(t)$$
$$= \binom{n}{u}V_n(t)q^{-d+1}(q-1)^u.$$

Now for $d \leq u \leq n$,

$$|\frac{P_E(u)}{Q} - 1| = |\frac{q^n D_u}{\binom{n}{u}(q-1)^u q^k V_n(t)} - 1|$$
$$= |\frac{q^n(D_u - \hat{D}_u)}{\binom{n}{u}(q-1)^u q^k V_n(t)}|$$
$$= \frac{q^n|D_u - \hat{D}_u|}{\binom{n}{u}(q-1)^u q^k V_n(t)}$$
$$\leq (\frac{q}{q-1})^{t-1}\frac{q^{d-1}\binom{u}{d-1}d}{(q-1)^u} + \frac{q^{d-1}2^u}{(q-1)^u}\frac{V_u^*(t)}{V_n(t)} \stackrel{\text{def}}{=} \mathbf{U}(u),$$

where $V_n(t) = \sum_{i=0}^{t}\binom{n}{i}(q-1)^i$ and $V_u^*(t) = \sum_{i=0}^{t}\binom{u}{i}(\frac{q-1}{2})^i$.

Thus, the upper bound $\mathbf{U}(u)$ of $|\frac{P_E(u)}{Q} - 1|$, which is a function of u for $d \leq u \leq n$, is given by the following equation:

$$(5) \qquad |\frac{P_E(u)}{Q} - 1| \leq (\frac{q}{q-1})^{t-1}\frac{q^{d-1}\binom{u}{d-1}d}{(q-1)^u} + \frac{q^{d-1}2^u}{(q-1)^u}\frac{V_u^*(t)}{V_n(t)} = \mathbf{U}(u).$$

The upper bounds of $|\frac{P_E(u)}{Q} - 1|$ of the NASA code and the JTIDS code are tabulated in Table 3 and Table 4 respectively.

NASA Code $(255, 223)$ *RS Code.* $q = 256$ $t = 16$

u:	33	34	35	36	37	...
$\mathbf{U}(u)$:	5.133	0.3422	0.0157	5.526×10^{-4}	1.512×10^{-5}	...

Table 3

JTIDS Code $(31, 15)$ *RS Code.* $q = 32$ $t = 8$

u:	17	18	19	20	21	...
$\mathbf{U}(u)$:	19.35	5.618	1.148	0.1851	0.02508	...

Table 4

3. Remarks

With the assumptions that q is greater than or equal to n and u is considerably large, Equation (5) shows that the upper bound of $|\frac{P_E(u)}{Q} - 1|$ is dominated by the denominator term $(q-1)^u$. Thus, the upper bound of $|\frac{P_E(u)}{Q} - 1|$ decays nearly exponentially as a function of u. This upper bound is not a very tight bound, but it is sufficient to illustrate the point that $P_E(u)$ approaches Q very rapidly as u increases.

CHAPTER III

A LOWER BOUND FOR DECODER ERROR PROBABILITY OF THE LINEAR MDS CODE

1. Introduction

In Chapter I, by repeated use of the inclusion and exclusion principle, we derive an exact expression for D_u. In Chapter II, by using the results in Chapter I, we evaluate the exact decoding error probability $P_E(u)$ of a linear MDS code. However, the formulae derived in Chapters I and II are complicated and clumsy, and offer no mathematical insight. In this chapter, by assuming that $q \geq n$, we derive the lower bound of $P_E(u)$ (and $D(u)$) from a completely different approach — simply by counting the dominant types of decodable words around codewords. In Sections 2 and 3 we show that the lower bound derived in this paper is similar in form, and close numerically to the upper bound derived in [7]. In Section 4 we show that with the assumption that $q \geq n$, the lower bound of $P_E(u)$ as a function of u achieves its minimum value at $u = d - t$. Thus, the lower bound for $u = d - t$ is the overall lower bound of $P_E(u)$. For $q < n$, this may not be true. The (6,4) MDS code over $GF(4)$ with $t = 1$ provides a counterexample.

2. Lower Bound of the Number of Codewords of Weight w

Let A_w denote the number of codewords of weight w. A lower bound of A_w is given by the following theorem.

Theorem 3.1:

$$A_w \geq C\binom{n}{w}q^{-d+1}(q-1)^w \qquad d \leq w \leq n, \tag{1}$$

where $C = 1 - \frac{\binom{d}{2}q^{d-2}}{(q-1)^d}$.

Proof:

From Chapter I, A_w is given by the following expression:

$$\begin{aligned} A_w &= \binom{n}{w}(q-1)\sum_{i=0}^{w-d}(-1)^i\binom{w-1}{i}q^{w-d-i} \\ &= \binom{n}{w}(q-1)q^{-d+1}\sum_{i=0}^{w-d}\binom{w-1}{i}q^{w-1-i} \\ &= \binom{n}{w}(q-1)q^{-d+1}[(q-1)^{w-1} - \sum_{i=w-d+1}^{w-1}(-1)^i\binom{w-1}{i}q^{w-1-i}]. \end{aligned}$$

Consider the second term of the above expression. Since $q \geq n$, $\binom{w-1}{i}q^{w-1-i} \geq \binom{w-1}{i+1}q^{w-1-i-1}$ for $d \leq w \leq n$ and $w-d+1 \leq i \leq w-1$. It is not hard to see that we can get the following inequalities.

$$A_w \geq \binom{n}{w}q^{-d+1}(q-1)^w \tag{2}$$

$$A_w \leq \binom{n}{w}q^{-d+1}(q-1)^w[1 + \frac{\binom{w-1}{w-d+1}q^{d-2}}{(q-1)^{w-1}}] \qquad w = d, d+2, d+4... \tag{3}$$

and

$$A_w \leq \binom{n}{w}q^{-d+1}(q-1)^w \tag{4}$$

$$A_w \geq \binom{n}{w}q^{-d+1}(q-1)^w[1 - \frac{\binom{w-1}{w-d+1}q^{d-2}}{(q-1)^{w-1}}] \qquad w = d+1, d+3, d+5... \tag{5}$$

Consider the bracketed term in Equation (5). Since $q \geq n$, it is an ascending function of w. So if we denote

$$C = [1 - \frac{\binom{w-1}{w-d+1} q^{d-2}}{(q-1)^{w-1}}]_{w=d+1}$$
$$= 1 - \frac{\binom{d}{2} q^{d-2}}{(q-1)^d},$$

we have

$$A_w \geq C \binom{n}{w} q^{-d+1} (q-1)^w \qquad d \leq w \leq n,$$

where C is a scaling factor very close to 1. ■

3. Derivation of Lower Bound of $P_E(u)$

Let $\bar{d}$ be a decodable word. Then $\bar{d}$ can be expressed uniquely as a sum $\bar{c} + \bar{e}$, where $\bar{c}$ is a codeword and $\bar{e}$ is an error pattern of weight less than t. Let $\bar{d}$ have weight u and $\bar{e}$ have weight s. The weight of $\bar{c}$ is then confined within a certain set of values, depending on the value of u and s. The main idea of deriving the lower bound of the number of decodable words of weight u is to count a certain "dominant" subset of codewords such that when added to appropriate error patterns, codewords in this subset give rise to decodable words of weight u. Let us define

$$B_{u,s} = \{w : w \text{ is the weight of a codeword that is at a distance } s \ (s \leq t) \text{ from a decodable word of weight } u\}$$

We then have the following expression for $B_{u,s}$, depending on the value of u and s:

i) $d - t \leq u \leq d - 1$

$B_{u,s} = \{w : d \leq w \leq u + s\}$

ii) $d \leq u \leq d + t - 1$

$B_{u,s} = \{w : d \leq w \leq u + s\}$

iii) $d + t \leq u \leq n - t$

$B_{u,s} = \{w : u - s \leq w \leq u + s\}$

iv) $n - t + 1 \leq u \leq n$

$B_{u,s} = \{w : u - s \leq w \leq u + s\}$ if $u + s \leq n$

$B_{u,s} = \{w : u - s \leq w \leq n\}$ if $u + s > n$.

We can then express D_u as follows:

(6) $$D_u = \sum_s \sum_{w \in B_{u,s}} A_w \times \{\#\text{ of error patterns of weight } s \text{ that give rise to a decodable word of weight } u \text{ from a codeword of weight } w\}.$$

We see that in the case $d - t \leq u \leq d - 1$, an allowable error pattern must be of weight $s \in \{d - u, ...t\} \subset \{0, 1, ...t\}$. In the case $d \leq u \leq n$, an allowable error pattern must be of weight $s \in \{0, 1, ...t\}$.

We also observe that for a linear MDS code, if $q \geq n$ and q is large, then

$$\frac{A_w}{A_{w-1}} \gg 1 \qquad \text{for most} \quad d \leq w \leq n.$$

Thus, for the sole purpose of finding a lower bound of D_u, we do not need to consider all $w \in B_{u,s}$. We need only to count those w's that give rise to most decodable words of weight u. It is then logical to consider only those $w \in B'_{u,s} \subseteq B_{u,s}$, where $B'_{u,s}$ is a subset of $B_{u,s}$ ($B'_{u,s}$ consists of the larger numbers in $B_{u,s}$), instead of all $w \in B_{u,s}$. We now define $B'_{u,s}$ as follows:

i) $d - t \leq u \leq d - 1$

$B'_{u,s} = \{w : d \leq w \leq u + s\}$

ii) $d \leq u \leq d + t - 1$

$B'_{u,s} = \{w : u \leq w \leq u + s\}$

iii) $d + t \leq u \leq n - t$

$B'_{u,s} = \{w : u \leq w \leq u + s\}$

iv) $n - t + 1 \leq u \leq n$

$B'_{u,s} = \{w : u \leq w \leq u + s\}$ if $u + s \leq n$

$B'_{u,s} = \{w : u \leq w \leq n\}$ if $u + s > n$.

Before we proceed, we want to categorize the decodable words according to the following definition.

Definition 3.1:

Let $\bar{d}$ be a decodable word such that it can be expressed in the form $\bar{d} = \bar{c} + \bar{e}$. Let $T_{\bar{c}}$ denote the set of nonzero coordinates of $\bar{c}$ and $T_{\bar{e}}$ denote the set of nonzero coordinates of $\bar{e}$.

a) $\bar{d}$ is defined to be of type-A iff $T_{\bar{e}} \subset T_{\bar{c}}$.

b) $\bar{d}$ is defined to be of type-B iff it is not of type-A.

It can be shown that for a given u, the number of type-A decodable words of weight u is usually much greater than the number of type-B decodable words of weight u for most u. However, an explanation of the above claim is complicated and clumsy, and it is very hard to present a formal proof. A crude and oversimplified explanation is that type-A decodable words lie within Hamming spheres of codewords of weights up to $u+t$, whereas type-B decodable words lie in the Hamming sphere of codewords of weights only up to $u+t-2$. As was mentioned before, $A_w \gg A_{w-1}$ for most w. This explains partly why the number of type-A decodable words is much greater than the number of type-B decodable words of weight u.

Summing up the above results, a lower bound of the number of decodable words of weight u is given by the following expression.

$$(7) \qquad D_u \geq \sum_{s} \sum_{w \in B'_{u,s}} A_w \times \{\#\text{ of error patterns of weight } s \text{ that give rise to a type-A decodable word of weight } u \text{ from a codeword of weight } w\ \}$$

We have four cases to consider, depending on the value of u.

i) $d-t \leq u \leq d-1$

In this case, $s \in \{d-u, ...t\}$ and $w \in B'_{u,s} = \{d, d+1, ...u+s\}$. There are $\binom{w}{s}$ ways of choosing s coordinates that give rise to type-A decodable words. But in order to have a type-A decodable word of weight u, the $w-u$ nonzero coordinates in $\bar{c}$ must match with the corresponding $w-u$ nonzero coordinates in $\bar{e}$ to give $w-u$ zeros in these coordinates. The remaining $s-(w-u)$ coordinates of $\bar{e}$ must also match the corresponding $s-(w-u)$ coordinates of $\bar{c}$ to give a nonzero value in each of the $s-(n-w)$ coordinates. There are $(q-2)^{s-(u-w)}$ ways to do so.

Thus, the number of decodable words of weight u, where $d-t \leq u \leq d-1$, is lower bounded as follows:

$$D_u \geq \sum_{s=d-u}^{t} \sum_{w \in B'_{u,s}} A_w \binom{w}{s} \binom{s}{w-u} (q-2)^{s-(w-u)}.$$

We then substitute the lower bound of A_w in Equation (1) for the above expression, and we have a lower bound of D_u as follows:

$$D_u \geq \sum_{s=d-u}^{t} \sum_{w=d}^{u+s} C \binom{n}{w} q^{-d+1} (q-1)^w \binom{w}{s} \binom{s}{w-u} (q-2)^{s-(w-u)}.$$

We see that $\binom{n}{w}\binom{w}{s}\binom{s}{w-u}$ can be expressed as $\binom{n}{u}\binom{n-u}{w-u}\binom{u}{s-(w-u)}$. Let $\lambda = w-u$. The above expression can be rewritten as

$$D_u \geq \sum_{s=d-u}^{t} \sum_{\lambda=d-u}^{s} C (\frac{q-1}{q})^{d-1} (\frac{q-2}{q-1})^{s-w+u} \binom{n}{u} \binom{n-u}{\lambda} \binom{u}{s-\lambda} (q-1)^{u+s-d+1}.$$

Next, it is not hard to see that for the given ranges of u, s and w, $(\frac{q-2}{q-1})^{s-w+u} > (\frac{q-2}{q-1})^t$. Also, for the purpose of consistency with the equations that follow, the lower

limit of the first summation on RHS of the above expression can be replaced with 0 and thus our final expression is

$$D_u \geq C(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}\sum_{s=0}^{t}\sum_{\lambda=d-u}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s,$$

where $C = 1 - \frac{\binom{d}{2}q^{d-2}}{(q-1)^d}$.

ii) $d \leq u \leq d+t$

In this case $s \in \{0,1,...t\}$ and $w \in \{u, u+1, ...u+s\}$. The derivation of lower bound of the number of decodable words of weight u is very similar to case (i), and the details of derivation are omitted. Since the smallest value of the codeword weights that are involved in counting is u, the scaling factor of the lower bound is now $C' = 1 - \frac{\binom{u-1}{u-d+1}q^{d-2}}{(q-1)^{u-1}}$ which is closer to 1 than C. The lower bound of D_u is then given by

$$D_u \geq C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}\sum_{s=0}^{t}\sum_{\lambda=0}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s$$

The lower bound can again be simplified by recalling the famous combinatoric identity

$$\sum_{\lambda=0}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda} = \binom{n}{s},$$

and the final expression for this case is

$$\begin{aligned} D_u &\geq C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}\sum_{s=0}^{t}\binom{n}{s}(q-1)^s \\ &= C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}V_n(t) \quad d \leq u \leq d+t. \end{aligned}$$

iii) $d+t+1 \leq u \leq n-t$

In this case, $s \in \{0,...t\}$ and $w \in \{u,...u+s\}$. The derivation is exactly the same as in case (ii), and the lower bound is given by

$$D_u \geq C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}V_n(t) \quad d+t+1 \leq u \leq n-t.$$

iv) $n-t+1 \leq u \leq n$

In this case, if $u+s \leq n$ then $w \in \{u, ...u+s\}$, and if $u+s > n$ then $w \in \{u, ...n\}$. The derivation of the lower bound is slightly different from those of cases (ii) and (iii), but the final expression turns out to be the same. That is,

$$D_u \geq C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}V_n(t) \quad n-t+1 \leq u \leq n.$$

In summary, the lower bound of the number of decodable words is given by the following equations:

$$(8)\ D_u \geq C(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}\sum_{s=0}^{t}\sum_{\lambda=d-u}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s \qquad d-t \leq u \leq d-1$$

$$(9)\ D_u \geq C'(\frac{q-1}{q})^{d-1}(\frac{q-2}{q-1})^t \binom{n}{u}(q-1)^{u-d+1}V_n(t) \quad n-t+1 \leq u \leq n,$$

where $C = 1 - \frac{\binom{d}{2}q^{d-2}}{(q-1)^d}$ and $C' = 1 - \frac{\binom{u-1}{u-d+1}q^{d-2}}{(q-1)^{u-1}}$.

We have shown in Chapter II that the decoder error probability is related to the number of decodable words via Equation (2) and thus the decoder error probability $P_E(u)$ is lower bounded as follows:

$$(10)\qquad P_E(u) \geq Cq^{-d+1}(\frac{q-2}{q-1})^t\sum_{s=0}^{t}\sum_{\lambda=d-u}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s \qquad d-t \leq u \leq d-1$$

$$(11)\qquad P_E(u) \geq C'q^{-d+1}(\frac{q-2}{q-1})^t V_n(t) \qquad d \leq u \leq n,$$

where $C = 1 - \frac{\binom{d}{2}q^{d-2}}{(q-1)^d}$ and $C' = 1 - \frac{\binom{u-1}{u-d+1}q^{d-2}}{(q-1)^{u-1}}$.

4. Overall Lower Bound of $P_E(u)$

In this section, an overall lower bound of $P_E(u)$ for all u is given by the following theorem and corollary.

Theorem 3.2:

The lower bound of $P_E(u)$ in Equations (10) and (11) is smallest for $u = d - t$.

Proof:

First of all, it is not hard to see that the lower bound in Equation (10) is always smaller than the lower bound in Equation (11) because $\binom{n}{s}$ is always greater than the incomplete Vandermonde convolution $\sum_{\lambda=d-u}^{t} \binom{n-u}{\lambda}\binom{u}{t-\lambda}$. Also, the scaling factor C' in Equation (11) is always greater than the scaling factor C in Equation (10). Thus, to prove the theorem, we need only to consider the lower bound of $P_E(u)$ for $d - t \leq u \leq d - 1$. It is not hard to see that a sufficient condition is to show that

$$\sum_{s=0}^{t}\sum_{\lambda=d-u}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s \geq \binom{n-d+t}{t}(q-1)^t \qquad d-t \leq u \leq d-1.$$

It is obvious that

$$\sum_{s=0}^{t}\sum_{\lambda=d-u}^{s}\binom{n-u}{\lambda}\binom{u}{s-\lambda}(q-1)^s \geq \sum_{\lambda=d-u}^{t}\binom{n-u}{\lambda}\binom{u}{t-\lambda}(q-1)^t.$$

We now proceed to show that $\sum_{\lambda=d-u}^{t} \binom{n-u}{\lambda}\binom{u}{t-\lambda}(q-1)^t \geq \binom{n-d+t}{t}(q-1)^t$. Let $l = t - d + u$ and $m = t - \lambda$; we have

$$\sum_{\lambda=d-u}^{t}\binom{n-u}{\lambda}\binom{u}{t-\lambda}(q-1)^t = \sum_{m=0}^{l}\binom{n-d+t-l}{t-m}\binom{d-t+l}{m}(q-1)^t.$$

Since $d \geq 2t+1$ and $0 \leq l \leq t-1$,

$$\binom{d-t+l}{m} \geq \binom{l}{m}.$$

Thus,

$$\sum_{m=0}^{l} \binom{n-d+t-l}{t-m}\binom{d-t+l}{m}(q-1)^t \geq \sum_{m=0}^{l} \binom{n-d+t-l}{t-m}\binom{l}{m}(q-1)^t$$
$$= \binom{n-d+t}{t}(q-1)^t,$$

and the theorem is proved. ■

Corollary

An overall lower bound of $P_E(u)$ for all u is

$$P_E(u) \leq C(\frac{q-2}{q-1})^t(\frac{q-1}{q})^{d-1}P_E(d-t)$$
$$= C(\frac{q-2}{q-1})^t q^{-d+1}\binom{n-d+t}{t}(q-1)^t,$$

where $C = 1 - \frac{\binom{d}{2}q^{d-2}}{(q-1)^d}$.

Proof:

A direct result from *Theorem 3.2.* ■

5. Remarks

For $q \geq n$, the upper bound and lower bound of $P_E(u)$ give a good estimation of $P_E(u)$. The upper bounds[7], lower bounds and exact values of the $P_E(u)$'s of the NASA code and the JTIDS code are tabulated in Table 1 and Table 2, respectively. We observe that the estimated values (upper bound and lower bound) are more or less of the same order of magnitude as the exact value in each case.

Also, we have shown that with the assumption that $q \geq n$, an overall lower bound of $P_E(u)$ (for all u) is given by $C(\frac{q-2}{q-1})^t P_E(d-t)$. For $q < n$, this may not be true. The MDS code in example 2 of I.4 gives a counterexample.

NASA Code $(255, 223)$ *RS Code.* $q = 256 \quad t = 16$

Weight	Lower bound	Actual Value	Upper bound
17	7.769×10^{-15}	9.464×10^{-14}	2.956×10^{-14}
18	1.665×10^{-14}	1.913×10^{-14}	2.957×10^{-14}
19	2.171×10^{-14}	2.401×10^{-14}	2.957×10^{-14}
20	2.361×10^{-14}	2.660×10^{-14}	2.957×10^{-14}
21	2.414×10^{-14}	2.602×10^{-14}	2.957×10^{-14}
22	2.425×10^{-14}	2.608×10^{-14}	2.957×10^{-14}
.	.	.	.
.	.	.	.
.	.	.	.
37	2.450×10^{-14}	2.609×10^{-14}	2.957×10^{-14}
.	.	.	.
.	.	.	.

Table 1

JTIDS Code $(31, 15)$ *RS Code.* $q = 32 \quad t = 8$

Weight	Lower bound	Actual Value	Upper bound
9	1.340×10^{-6}	3.750×10^{-6}	9.250×10^{-6}
10	5.741×10^{-6}	1.439×10^{-6}	9.349×10^{-6}
11	1.310×10^{-6}	2.951×10^{-6}	9.350×10^{-6}
12	2.123×10^{-6}	4.329×10^{-6}	9.350×10^{-6}
13	2.767×10^{-6}	5.189×10^{-6}	9.350×10^{-6}
14	3.140×10^{-6}	5.547×10^{-6}	9.350×10^{-6}
.	.	.	.
.	.	.	.
.	.	.	.
25	4.328×10^{-6}	5.626×10^{-6}	9.350×10^{-6}
.	.	.	.
.	.	.	.

Table 2

REFERENCES FOR PART ONE

[1] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes.* Amsterdam, The Netherlands: North-Holland, 1983.

[2] Assmus,E. F., H. F. Matton, Jr., and R. J. Turyn, *Cyclic Codes*, AFCRL-65-332, Air Force Cambridge Research Labs, Bedford, Mass., 1965.

[3] Forney, G. D., Jr., *Concatenated Codes,* The M.I.T. Press, Cambridge, Mass., 1966.

[4] Kasami, T., S. Lin, and W. W. Peterson, "Some Results on Weight Distributuons of BCH codes," *IEEE Trans. Inform. Theory*, IT-12 (1966): 274.

[5] Marshall Hall, *Combinatorial Theory*, Blaisdell, 1967.

[6] E. R. Berlekamp and J. L. Ramsey, 1978 "Readable erasures improve the performance of Reed-Solomon codes," *IEEE Tran. Inform. Theory*, vol. IT-24, 632-633.

[7] R. J. McEliece and L. Swanson, 1986 "On the Decoder Error Probability for Reed-Solomon Codes," *IEEE Tran. Inform. Theory*, vol. IT-32, 701-703.

PART TWO

ERROR-CORRECTION CODING IN DATA STORAGE SYSTEMS

CHAPTER IV

GENERALIZATION OF HELICAL INTERLEAVING TO N PARALLEL CHANNELS

1. Introduction

Noise bursts are common in many different channels. They arise from numerous causes, including multipath, interference, jamming, and fading. The duration of noise bursts ranges from a few bits to billions of bits, depending on the characteristics of channels. Interleaving is the common strategy for enhancing the performance of error-correcting codes in the presence of noise bursts.

Errors and erasures in magnetic tape recording are primarily caused by defects on the magnetic media or by variations in head-media separation in the presence of dust particles. These errors and erasures often come in bursts, and they can affect as many as 100 bits at a time.

A block interleaving scheme is now being used in many tape machines. For example, in the IBM 3850 Mass Storage System (MSS), the codewords of the $(15, 13)$ BCH code over $GF(2^8)$ are block-interleaved to a depth of 16. The block interleaving scheme has a rather straightforward and simple hardware implementation, and it imposes no constraint on the interleaving depth. However, block interleaving requires substantial memory and causes a long interleaving delay in the transmission of data.

In an unpublished paper by Berlekamp and Tong[1], a novel interleaving scheme called helical interleaving was presented and was shown to have several advantages over block interleaving. For the same code length and interleaving depth, memory required in helical interleaving is one-half that of block interleaving, and interleaving delay in helical interleaving is one-half that of block interleaving. Also, helical interleaving allows burst forecasting strategy, which enhances the error-correcting capability of the code. Moreover, Berlekamp has shown that helical interleaving offers a one RAM implementation as shown in Figure 1. The transmitter's interleaver and the receiver's deinterleaver are identical, and the addressing sequence has a period that is about twice the RAM's memory size.

A One-RAM Implementation

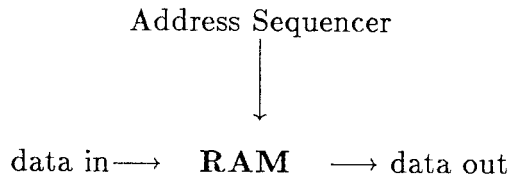


Figure 1

In Berlekamp's paper, only a single channel was considered. In a magnetic tape environment, data are written on parallel tracks along the length of the tape. If a helical interleaving scheme is to be used in a magnetic tape with n tracks, n sets of interleaver/deinterleaver hardware will be required. However, we found that with only slight modification on the interleaving scheme, we could apply helical interleaving to n parallel tracks (n is a number which divides the length of the codeword), using only a single RAM chip (with memory size the same as that

required of a single channel) but preserving the same interleaving depth in each channel.

The following discussion is mainly a generalization of the idea of helical interleaving, together with some modifications and comments on Berlekamp's paper. In Section 2, we describe the general idea of interleaving. In Section 3, we present simple algorithms to construct the addressing sequences of the one RAM implementation and the optimal one RAM implementation of helical interleaving. In Section 4, we give the advantages of helical interleaving over block interleaving. In Section 5, we discuss another type of interleaver called deep-staggered interleaver, which is formed by concatenating block and helical interleavers. In Section 6, we give modifications on Berlekamp's helical interleaving scheme, which generalizes the scheme from one channel to n channels. Last of all in Section 7, we give a conclusion to this chapter.

2. Interleaving

Interleaving is a common strategy for enhancing the performance of an error-correcting code in the presence of bursty noise. The logical position of an interleaver in a typical communication system is shown in Figure 2. The transmitter's interleaver lies between the encoder and the channel; the receiver's deinterleaver lies between the channel and the decoder.

An interleaver is nothing more than a buffer that accepts incoming data from the encoder, stores the data, and reads the data out to the channel at appropriate instants. The characters in the data stream, when read out to the channel, are

shuffled in such a way that any two characters from the same codeword are separated from each other by at least i characters from other codewords. We call i the interleaving depth. When a noise burst occurs, it is hoped that the burst will corrupt no more than it characters in the data stream where t is the number of errors the code can correct. The decoder will thus be able to recover the data from the corrupted data stream.

Logical Position of Interleavers in a Communication System

TRANSMITTER:

Source $\longrightarrow$ Encoder $\longrightarrow$ Interleaver $\longrightarrow$ Channel

RECEIVER:

Channel $\longrightarrow$ Deinterleaver $\longrightarrow$ Decoder $\longrightarrow$ User

Figure 2

The operation of an interleaver is best depicted by the following example in Figure 3. It is a schematic diagram of a conventional block interleaver. It shows a code of length 4 block-interleaved to depth 3. The codewords occupy positions labeled ABCD, EFGH, IJKL, MNOP,...,etc. In this figure, codeword characters are read into the interleaver column by column from top to bottom; the characters are transmitted across the channel row by row from left to right.

Block Interleaver of Length 4 and Depth 3

A	**E**	**I**
B	**F**	**J**
C	**G**	**K**
D	**H**	**L**
M	**Q**	**U**
N	**R**	**V**
O	**S**	**W**
P	**T**	**X**
Y	δ	
Z	γ	
α	ε	
β		

Figure 3

At any instant of time, exactly two of the characters shown in Figure 3 are visible at the interfaces between the boxes in Figure 2. One character is entering the interleaver from the encoder; another is coming out of the interleaver across the channel and into the deinterleaver. For example, there may be an instant of time at which the character labelled Z in Figure 3 is entering the interleaver, while the character labelled Q is going across the channel.

The operation of a deinterleaver is similar to that of an interleaver. A deinterleaver receives the scrambled and corrupted data from the channel, unscrambles the data, and sends the data to the decoder.

Since "variable delays" is the essence of interleaving, an interleaver can be viewed as a collection of delay lines of various lengths with multiplexers shuffling

data through successively different routes. This is the viewpoint expounded in the classical papers by Ramsey (1970) and Forney (1971). The work of Clark and Cain (1981) also contains an excellent discussion on the present state of art of the subject.

In an unpublished paper by Berlekamp and Tong[1], a novel interleaving scheme, called helical interleaving, was introduced. A schematic diagram of a helical interleaver is shown in Figure 4. It shows a code of length 4 helically interleaved to depth 3. The codewords are read into the interleaver to occupy positions labelled ABfa, CDbc, EFde,...,etc. The characters are read out horizontally from left to right from positions labelled abFA, cdBC, efDE,...,etc.

Helical Interleaver of Length 4 and Depth 3

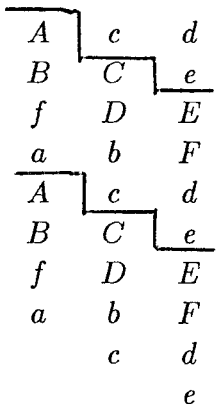


Figure 4

In a block interleaver, interleaving depth is arbitary and is not constrained by codeword length. However, it was shown in [1] that interleaving depth of a helical interleaver must be equal to the codeword length minus one. In the case of a deep-staggered interleaver (Section 5), the interleaving depth must be a multiple of codeword length minus one.

3. One RAM Implementation

It was pointed out in [1] that a helical interleaving scheme allows one RAM implementation, which is composed of one RAM chip and an address sequencer. A schematic diagram of a one-RAM implementation is shown in Figure 1. In Berlekamp's paper[1] some examples of optimal one-RAM implementation are constructed, but the methodology is complicated and hard to comprehend. Here a simple algorithm for the construction of an addressing sequence is presented. Let l be the codeword length and $l-1$ be the interleaving depth.

One RAM Implementation:

1. Construct a $l-1 \times l-1$ *symmetric matrix* M such that the entries along the diagonal and those on the left (or right) side of the diagonals are all different.
2. "Duplicate" the diagonal to obtain an $l \times l-1$ matrix.
3. Read out the addressing sequence from the matrix column by column from column 1 to column $l-1$. At column i, $1 \leq i \leq l-1$, read out the character at

position $(i+1,i)$ first, then $(i+2,i)$,...,(l,i),$(1,i)$,...,and finally (i,i).

The above algorithm is illustrated by an example in Figure 5 in which a code of length 4 is helically interleaved to depth 3. The addressing sequence is periodic modulo 12:

ABFACDBCEFDEABFA...

The memory size required in a one-RAM implementation is $\binom{l}{2}$, and the addressing sequence is periodic modulo $l(l-1)$.

Algorithm for One-RAM Implementation

Figure 5

Optimal One-RAM Implementation:

1. Construct an $l-1 \times l-1$ *symmetric matrix* M such that the entries along the diagonal are the same and those on the left (or right) side of the diagonals are all different from each other and also different from the diagonal entries.
2. "Duplicate" the diagonal to obtain an $l \times l-1$ matrix.
3. Read out the addressing sequence from the matrix column by column from

column 1 to column $l-1$. At column i, $1 \leq i \leq l-1$, read out the character at position (i,i) first, then $(i+1,i)$,...,(l,i),$(1,i)$,..., until all column entries are exhausted.

The above algorithm is illustrated by an example in Figure 6 in which a code of length 4 is helically interleaved to depth 3. The addressing sequence is periodic modulo 12 :

AABCAADBAACDAABC...

The memory size required in an optimal one-RAM implementation is $\binom{l}{2} - l + 2$ and the addressing sequence is periodic modulo $l(l-1)$.

Algorithm for Optimal One-RAM Implementation

$$\begin{pmatrix} A & B & C \\ B & A & D \\ C & D & A \end{pmatrix} \quad \longrightarrow \quad \begin{pmatrix} A & B & C \\ A & A & D \\ B & A & A \\ C & D & A \end{pmatrix}$$

Figure 6

4. Advantages

It was pointed out in [1] that for a given codeword length l and an interleaving depth i, helical interleaving is better than block interleaving in several aspects :

1. *Memory*—memory required by helical interleaving is less than $\binom{l}{2}$, which is equal to one-half of li, whereas for block interleaving the memory size is li.

2. *Interleaving Delay*—interleaving delay in helical interleaving is li, whereas for block interleaving the delay is $2li$.

3. *Burst Forecasting Strategy*— In [1] Berlekamp suggested putting a synchronization character at the end of each codeward. This character, besides being used for the purpose of sychronization, can be used as an erasure indicator, which indicates the beginning of an error burst. The burst forecasting strategy is not applicable in block interleaving. The difficulty is to get started.

5. Other Types of Interleavers

In [1] different types of interleaver like deep-staggered interleaver, shallow-staggered interleaver, and helical-helical interleaver were constructed by concatenating block and helical interleavers. In the magnetic tape environment, only helical and deep-staggered interleavers are of interest. The construction of helical interleaver is given in above sections. Deep-staggered interleaver is constructed by concatenating a block interleaver and a helical interleaver as shown in figure 7. Each row of output of the block interleaver is treated as a single "super character" by the helical interleaver. If the code length is l characters and the block interleaver has depth i, then the depth of the deep-staggered interleaver is $(l-1)i$ where i can be any positive integer. This provides more freedom in the choice of interleaving depth. Figure 8 shows the overall effect of the deep-staggered interleaving system $(l=4, i=3)$ on the codewords.

Deep-staggered Interleavers in a Communication System

TRANSMITTER:

Source ⟶ Encoder ⟶ Block Interleaver ⟶ Helical Interleaver ⟶ Channel

RECEIVER:

Channel ⟶ Helical Deinterleaver ⟶ Block Deinterleaver ⟶ Decoder ⟶ User

Figure 7

Overall Effect of Deep-Staggered Interleaver

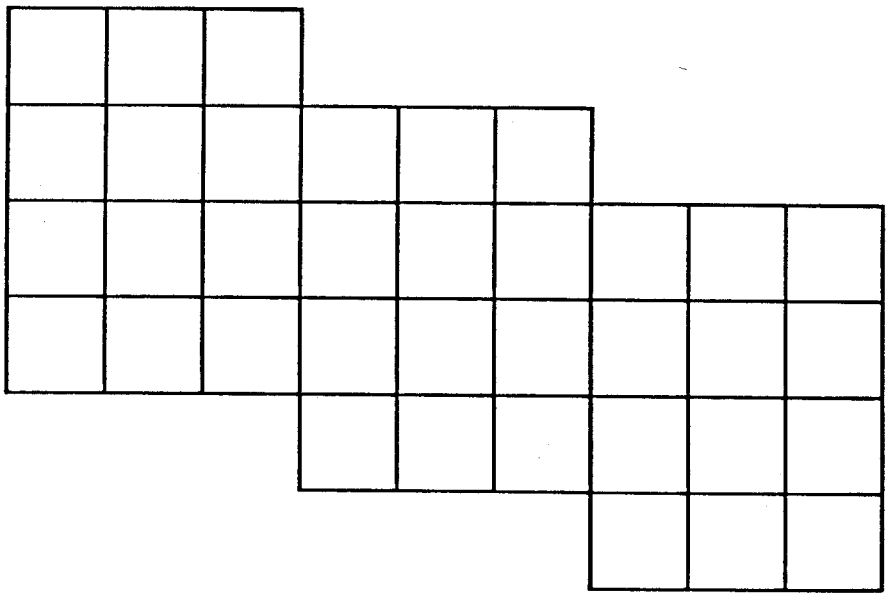


Figure 8

6. Generalization

In Berlekamp's paper[1] on interleaving, only the single channel is being considered. However, in many tape machines, parallel tracks of data are written across the length of the tape. Each track is a bursty channel of the kind mentioned in [1].

In order to apply helical interleaving to this parallel bursty channel environment (say n tracks), it seems that n sets of interleaver/deinterleaver would be required. It would mean a substantial increase in memory size and hardware complexity, particularly when n is large. In this section, we want to present a modified scheme on helical interleaving, which generalizes the scheme from one channel to n parallel channels. It can be shown that by imposing a constraint on the codeword length—number of tracks relatively prime to interleaving depth, we can apply helical interleaving or deep-staggered interleaving to n parallel tracks, using only a single RAM chip (with memory size the same as that required for a single channel) but preserving the same interleaving depth in each track. The modified scheme requires only a multiplexer (1 to n) on the *write* side and a demultiplexer (n to 1) on the *read* side in addition to a single set of helical interleaver/deinterleaver hardware. The modified scheme is depicted in Figure 9. However, with the above constraint on codeword length alone, although interleaving depth is preserved, characters of same codeword are often distributed on the tape in an unorganized manner, making sychronization rather difficult. This difficulty can be overcome if we put further restriction on codeword length, namely,

a) the number of tracks n divides the length of codeword l in the case of helical interleaving, and

b) the number of tracks n divides the length of codeword l, and n is relatively prime to i (depth of block interleaver in deep-staggered interleaving system) in the case of deep-staggered interleaving.

Theorem 4.1 will prove the above assertion.

Block Diagram of Generalized Helical Interleaving Scheme

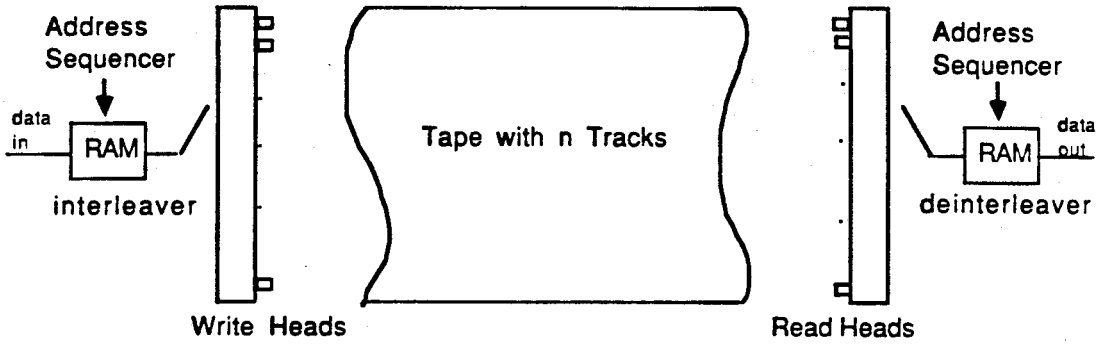


Figure 9

By imposing the above constraint on codeword length, characters of the same codeword are then written on the tape in regular "zigzag" patterns (as shown in Figures 10, 11, 12, 13) such that the beginnings and ends of codewords can be easily tracked. This makes sychronization much easier to obtain.

The modified interleaving scheme works as follows: During the writing process, data are first helically interleaved by the interleaver, and are then distributed by the multiplexer to the n *write* heads, which write the data along the length of the tape in n parallel tracks. During the reading process, data are first concentrated to a single data stream by the demultiplexer, and are then deinterleaved by the helical deinterleaver.

Theorem 4.1:

a.) Given that the number of tracks n divides the length of codeword l, the modified helical interleaving scheme gives the same interleaving depth on each track as the unmodified scheme of the same memory size.

b.) Given that the number of tracks n divides length of codeword l and is relatively prime to i (depth of block interleaver in deep-staggered interleaver), the modified deep-staggered interleaving scheme gives the same interleaving depth on each track as the unmodified scheme of same memory size.

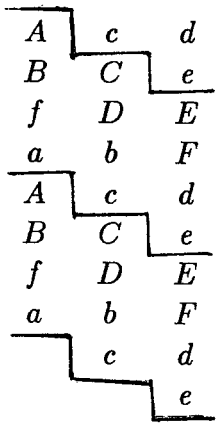
Proof:

a.) The proof of the helical interleaving case is as follows. The interleaving depth in helical interleaving is $l-1$. Given that n divides l, let $l = nk$, where k is an integer greater than or equal to one. Let the tape be labelled vertically and horizontally as shown in Figure 9. The multiplexer distributes the characters from the helical interleaver to the n *write* heads in the order 1, 2,...,n , 1, 2,...,n ,...etc. The demultiplexer, on the other end, concentrates the data from the n *read* heads to a single data stream in the order 1, 2,...,n ,1 ,...,n ,...etc. Consider a codeword $\bar{c} = (c_1, c_2, ..., ...c_l)$. Without loss of generality, let c_1 be written on $(n, 1)$. Since the interleaving depth is $l - 1 = nk - 1 = -1 \bmod n$, by following the writing procedure of the modified scheme, c_2 is then written on the $n-1$ row and on the $k+1$ column (on coordinate $(n-1, k+1)$). This process continues for $c_3, c_4, ...,$ and c_n is written on $(1, (n-1)k+1)$. The next character c_{n+1} is then written on the nth row and the nkth column. This is the first time that a character of the same codeword is written on the nth row again. The separation between these two characters on the nth row is $nk - 1 = l - 1$, which is the interleaving depth. A similar argument holds for characters in other rows, and

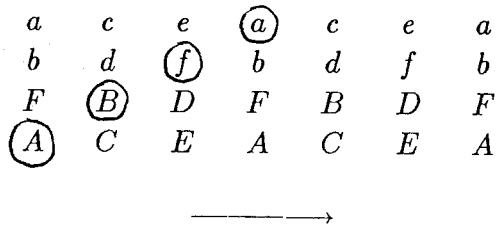
the theorem is proved in the case of helical interleaving. Figure 10 and Figure 11 are examples of generalized helical interleaving applying to a tape with 4 tracks. ■

b.) The proof of part (b) is similar to that of part (a). Given that n divides l, let $l = nk$, where k is a positive integer. Consider a codeword $\bar{c} = (c_1, c_2, \ldots, c_l)$. We note that c_j and c_{j+1}, $1 \leq j \leq l$, are separated from each other by $i(nk-1)$ positions in the data stream. Without loss of generality, let c_1 be written on $(1,1)$. Since n is relatively prime to i and $nk-1$, n is relative prime to the interleaving depth $i(nk-1)$. Thus, the characters c_2, $c_3, \ldots, c_n$ would not be written on row 1. The next character c_{n+1}, which is separated from c_1 by $ni(nk-1)$ positions in the data stream, is then written onto row 1. This is the first time that a character of the same codeword is written on row one again. The separation between c_1 and c_{n+1} on row 1 is $i(nk-1)$, which is the interleaving depth. A similar argument holds for characters in other rows and the theorem is proved. Figure 12 and Figure 13 are examples of deep-staggered interleaving, applying to a 4-track tape. ■

Data Format along a Tape with Four tracks



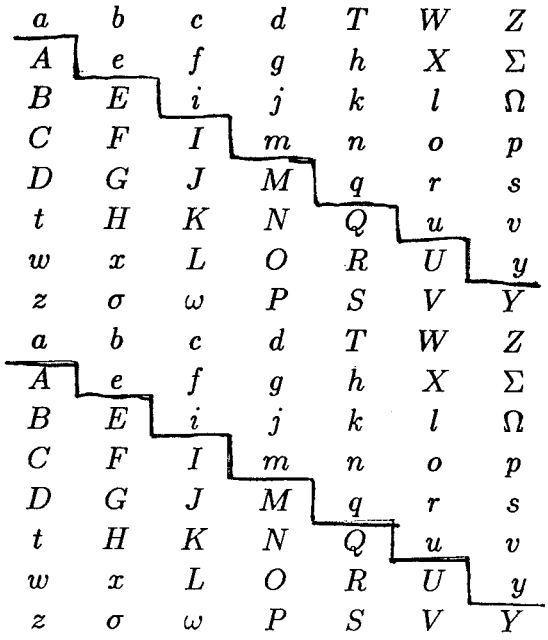
(a) Helical Interleaving Scheme
codeword length = 4
interleaving depth = 3



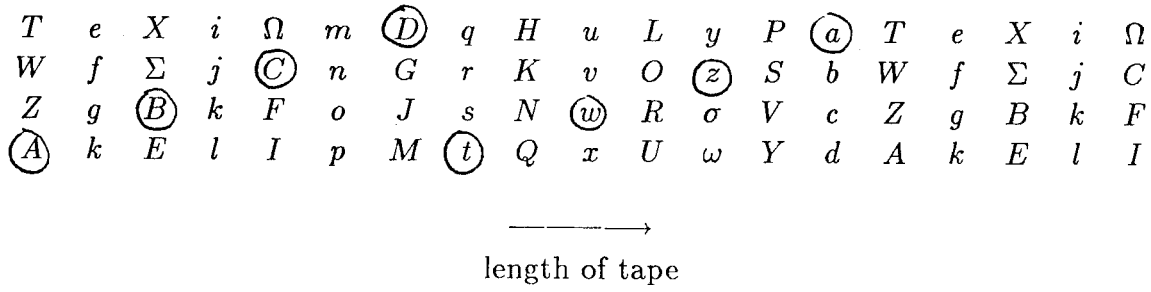
(b) Data Format

Figure 10

Helical Interleaving Scheme on a Tape with Four tracks



(a) Helical Interleaving Scheme
codeword length = 8
interleaving depth = 7



(b) Data Format on tape

Figure 11

Deep-staggered Interleaving Scheme on a Tape with Four tracks

A	*B*	*C*						
A	*B*	*C*	*D*	*E*	*F*			
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
1	2	3	*D*	*E*	*F*	*G*	*H*	*I*
1	2	3	4	5	6	*G*	*H*	*I*
1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
A'	*B'*	*C'*	4	5	6	7	8	9
A'	*B'*	*C'*	*D'*	*E'*	*F'*	7	8	9
A'	*B'*	*C'*	*D'*	*E'*	*F'*	*G'*	*H'*	*I'*
A'	*B'*	*C'*	*D'*	*E'*	*F'*	*G'*	*H'*	*I'*
			D'	*E'*	*F'*	*G'*	*H'*	*I'*
						G'	*H'*	*I'*

(a) Deep-staggered Interleaving Scheme
codeword length = 4
interleaving depth = 9

(1)	*E*	*I*	4	*H*	3	7	2	6	*A'*	5	9	*D'*	8	*C'*
2	*F*	(1)	5	*I*	4	8	3	7	*B'*	6	*A'*	*E'*	9	*D'*
3	*G*	2	6	(1)	5	9	4	8	*C'*	7	*B'*	*F'*	*A'*	*E'*
D	*H*	3	*G*	2	6	(1)	5	9	4	8	*C'*	7	*B'*	*F'*

⟶

length of tape

(b) Data Format on tape

Figure 12

Deep-staggered Interleaving Scheme on a Tape with Four tracks

A	*B*	*C*																	
A	*B*	*C*	*D*	*E*	*F*														
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*											
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*								
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*					
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*		
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	4	5	6	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	4	5	6	7	8	9	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	4	5	6	7	8	9	10	11	12	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	*P*	*Q*	*R*	*S*	*T*	*U*
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	*S*	*T*	*U*
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A'	*B'*	*C'*	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A'	*B'*	*C'*	*D'*	*E'*	*F'*	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A'	*B'*	*C'*	*D'*	*E'*	*F'*	*G'*	*H'*	*I'*	10	11	12	13	14	15	16	17	18	19	20	21
A'	*B'*	*C'*	*D'*	*E'*	*F'*	*G'*	*H'*	*I'*	*J'*	*K'*	*L'*	13	14	15	16	17	18	19	20	21
															16	17	18	19	20	21
																		19	20	21

(a) Deep-staggered Interleaving Scheme
codeword length = 8
interleaving depth = 21

(1) *E I M Q U* 4 *H L P T* 3 7 *K O S* 2 6 10 *N R* (1) 5 9 13 *Q U* 4 8 12 16 *T* 3 7 11 15
2 *F J N R* (1) 5 *I M Q U* 4 8 *L P T* 3 7 11 *O S* 2 6 10 14 *R* (1) 5 9 13 17 *U* 4 8 12 16
3 *G K O S* 2 6 *J N R* (1) 5 9 *M Q U* 4 8 12 *P T* 3 7 11 15 *S* 2 6 10 14 18 (1) 5 9 13 17
D H L P T 3 *G K O S* 2 6 *J N R* (1) 5 9 *M Q U* 4 8 12 *P T* 3 7 11 15 *S* 2 6 10 14 18 (1)

⟶

length of tape

(b) Data Format on tape

Figure 13

7. Conclusion

Berlekamp has introduced a new interleaving scheme called helical interleaving, which can be applied to a single bursty channel. In this paper we generalize this idea to n parallel channels. Apparently, this scheme is particularly applicable to the bursty parallel tracks environment of magnetic tape.

In addition to the advantages mentioned in the previous sections, helical interleaving also serves to interleave a codeword in different tracks as well as along the length of tracks. The characters of each codeword in adjacent tracks are offset by at least one position (k positions in the case of helical interleaving and ki positions in the case of deep-staggered interleaving), and this can help to prevent intersymbol interference of characters of the same codeword in adjacent tracks.

CHAPTER V

HYBRID CODE

1. Introduction

Magnetic tape is today's most popular means of computer data storage. In order to increase data storage capacity and data delivery rate, data density on the tape has to be made very high. This makes the magnetic tape environment very vulnerable to different types of noises. The most common kinds of errors on magnetic tape are random bit errors caused by electric noise, two-consecutive-bit errors caused by a bit shift phenomenon, and long tracks of errors caused by defects on magnetic media or variations in head-media separation due to dust particles.

Error correction coding has long been used in the magnetic tape environment to guarantee reliable delivery of data. The IBM 3420 tape machine uses an error correction scheme[2], which provides on-the-fly correction of two erased tracks out of nine recorded tracks. The IBM 3850 Mass Storage System uses a $(15, 13)$ 1-error correcting BCH code interleaved to depth 16 to correct up to two 16-byte sections in each segment of 240 bytes of serial data[3]. Recently, the IBM 4850 Magnetic Tape Subsystem introduced a tape cartridge with an 18-track data format that uses a new coding scheme called adaptive cross-parity (AXP) code[4]. In the IBM 4850 system, the 18 tracks are divided into two interleaved sets of nine tracks. The AXP code is convolutional in nature. By adaptive use of the checks in the two interleaved sets, the new scheme corrects up to three erased tracks in any one set

of nine tracks and up to four erased tracks in the two sets together. The code is simple in hardware implementation, but it cannot correct all patterns of two track errors and all patterns of four track erasures.

In this paper a new code called the hybrid code, which is specially designed for the parallel track magnetic tape environment, is introduced. This code is a hybrid of block and convolutional code. The choice of the block code is arbitary, but the simple one-error correcting Reed-Solomon (RS) code or BCH code is more preferable. The data are encoded along different directions (except along the length of the tracks) with the chosen block code. The codewords of this block code are grouped together by a convolutional structure that integrates the error correcting capabilities of the codewords. By following a proper decoding algorithm, multiple track errors can be corrected. This code preserves the advantage of simple hardware implementation as in the AXP code, but it can correct more error patterns than the AXP code.

The following discussion of the code is based on a specific example of a 17-track tape. The data are encoded on the tape in two directions—diagonal (lower left to upper right) and vertical. The block code used is a $(15, 13)$ 1-error correcting BCH code (the same code used in the IBM 3850 Mass Storage System [2]). The data format is shown in Figure 1. Tracks 15,16 are the parity tracks of the vertical code, and tracks 0,1 are the parity tracks of the diagonal code. It is shown in later sections that by following a simple decoding algorithm, this code can correct any combination of :

1) two track errors or less,

2) one track error and two track erasures or less, or

3) four track erasures or less.

The block code chosen in this hybrid code makes the code highly effective against random bit errors and 2-consecutive bit errors, which are the most common kinds of random errors in a magnetic tape environment. The code is also very efficient in preventing miscorrection.

We discuss the block code in this hybrid code (the (15,13) BCH code) in great detail in Section 2. We describe the encoding process in Section 3 and the decoding process in Section 4. Last of all, we give the conclusion and generalization in Section 5.

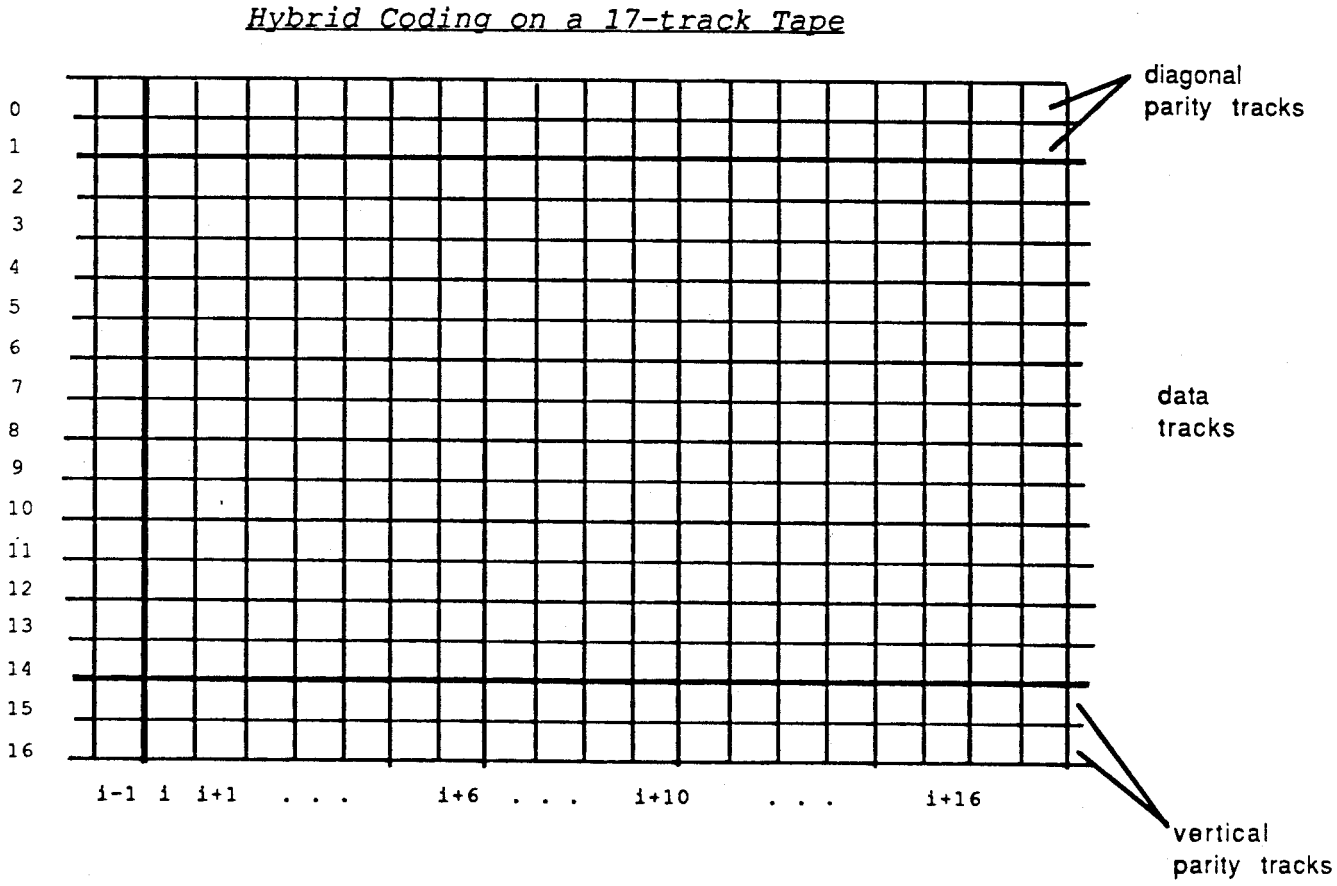


Figure 1

2. The Block Code

As shown in Figure 1, there are 17 parallel tracks recorded along the length of the tape. The data are encoded in diagonal and vertical directions. The block code used is the (15,13) 1-error correcting BCH code over $GF(2^8)[2]$. This code is a *maximum distance separable* (MDS) code. Each character is 8 bits wide. It was shown in [2] that the code is very simple in hardware implementation. It was also shown in [2] that the code can correct one error or two erasures, and it can prevent miscorrection of two 1-bit errors in the same codeword. However, in [2] the choice of a primitive polynomial for generating elements in $GF(2^8)$ is arbitary. We found that with the proper choice of a primitive polynomial, the code can prevent miscorrections of one 1-bit error and one 2-consecutive-bit error in the same codeword, and can also prevent miscorrection of two 2-consecutive-bit errors in the same codeword. The above assertion is proved by the following theorem :

Theorem 5.1:

Let λ be a primitive element of $GF(2^8)$. Let $\alpha = \lambda^{17} \in GF(2^4)$.

a) The code can prevent miscorrection of two 2-consecutive-bit errors in the same codeword for any choice of primitive polynomial.

b) If the primitive polynomial is chosen such that $1 + \lambda = \lambda^z$ and $z \in \Delta = \{8, 9, 10, 25, 26, 27, 42, 43, 44, 59, 60, 61, 79, 80, 81, 93, 94, 95, 110, 111, 112, 127, 128, 129, 144, 145, 146, 161, 162, 163, 178, 179, 180, 195, 196, 197, 212, 213, 214, 229, 230, 231, 246, 247, 248 \}$, then the code can prevent miscorrection of one 1-bit error and one 2-consecutive-bit error in the same codeword.

The proof of the above theorem is given in the appendix. From the above theorem we found that the primitive polynomial used in the 3850 system[2] is not a "good" polynomial ($z = 96$). If the polynomial $1 + x + x^3 + x^5 + x^8$ is used ($z = 25$), additional error correcting capabilities depicted in *theorem 5.1* can be obtained. Let us denote the decoder error probability by $P_E(u)$, where u is the weight of the error pattern. In general, even if the error pattern is completely random the $P_E(u)$ of the $(15, 13)$ BCH code over $GF(2^8)$ does not exceed 0.06 as shown in [5]. The decoder error probability $P_E(u)$'s of the $(15, 13)$ BCH code over $GF(2^m)$ are tabulated in Figure 2.

Another obvious fact about the $(15, 13)$ BCH code, which is essential in later discussion, is given in the following theorem :

Theorem 5.2:

The $(15, 13)$ BCH code does not miscorrect any combination of one error character and one erasure character in a codeword.

Proof:

Let $\bar{r}=(r_0, r_1, \ldots, r_{14})$ be the received pattern. Suppose that an error character e occurs at position x (unknown) and an erasure character e' occurs at position y (known), $x \neq y$. We have the following syndrome equations:

$$\begin{aligned} S_0 &= r_0 + r_1 + \ldots + r_{14} \\ &= e + e' \end{aligned} \tag{1}$$

$$\begin{aligned} S_1 &= r_0 + \alpha r_1 + \ldots + \alpha^{14} r_{14} \\ &= \alpha^x e + \alpha^y e'. \end{aligned} \tag{2}$$

Since the code is 1-error-correcting, it assumes only the existence of an erasure character at position y. If miscorrection does occur, the sydromes S_0 and S_1 must satisfy the equation:

(3) $$\alpha^y S_0 = S_1.$$

But if we substitute the expressions S_0 and S_1 of Equations (1) and (2) into (3), we get $x = y$. This contradicts the fact that the error character e and the erasure character e' are in different positions. ■

3. Encoding Process

The data are encoded in two directions—diagonal and vertical—as shown in Figure 1. Tracks 0 and 1 are parity tracks for the diagonal code and tracks 15 and 16 are parity tracks for the vertical code. Let $C_i(m)$ be a byte on the ith column and on the mth track. At the beginning of each encoding process, zeros are appended to the first 14 columns in a manner as shown in Figure 4a. Similarly, at the end of each encoding process, zeros are appended to the last 14 columns as shown in Figure 4b. The introduction of these redundant bytes serves to indicate the beginning and the end of a data block, and this facilitates the encoding and decoding processes.

The block diagram of the encoding process is depicted as shown in Figure 3. Encoders A and B are block encoders for the $(15, 13)$ BCH code as described in [2]. The data are first encoded by encoder A, and are then fed into the buffer. The buffer introduces variable delays on different tracks so that characters of each codeword can be written diagonally across the tape (diagonal encoding) as shown in Figure 1. Encoder B carries out vertical encoding, and the parity bytes are written on tracks 15 and 16.

Encoder A produces diagonal codeword $\bar{C}^d=(C_{i+14}(0), C_{i+13}(1), \ldots, C_i(14))$ in diagonal encoding such that

$$C_{i+14}(0) + C_{i+13}(1) + \ldots + C_i(14) = 0 \tag{4}$$

$$C_{i+14}(0) + \alpha C_{i+13}(1) + \ldots \alpha^{14} C_i(14) = 0, \tag{5}$$

where α is a primitive element of a subfield $GF(2^4)$ of $GF(2^8)$.

Similarly encoder B produces the vertical codeword $\bar{C}^v=(C_i(16), C_i(15), ...C_i(2))$ in vertical encoding such that

(6) $$C_i(16) + C_i(15) + ... + C_i(2) = 0$$

(7) $$C_i(16) + \alpha C_i(15) + ...\alpha^{14} C_i(2) = 0,$$

where α is a primitive element of a subfield $GF(2^4)$ of $GF(2^8)$.

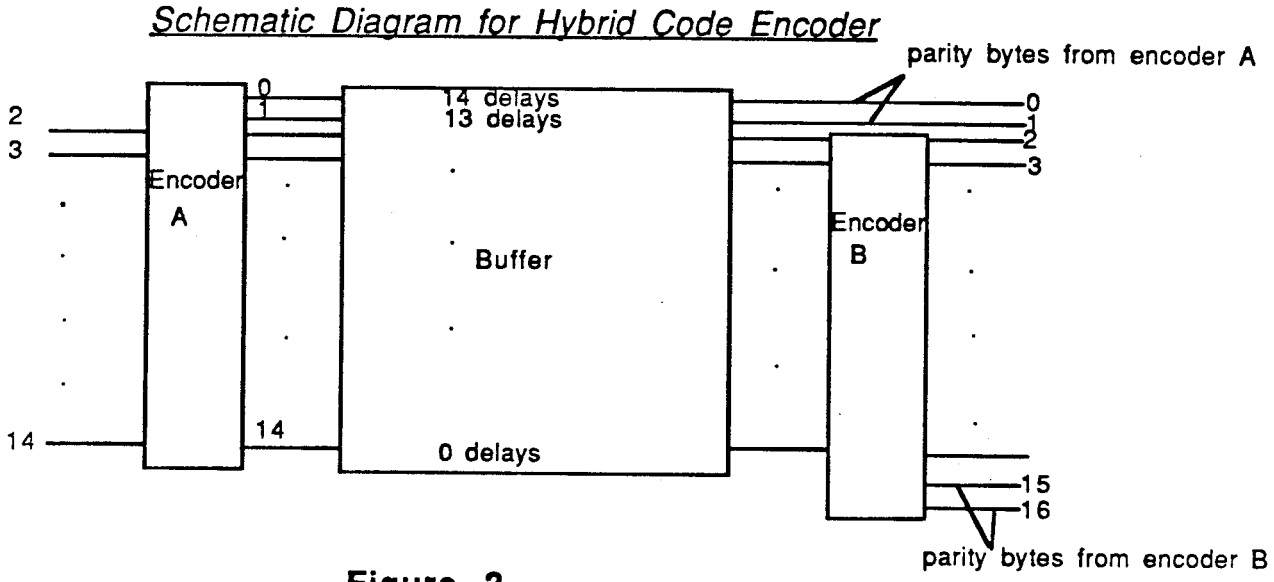
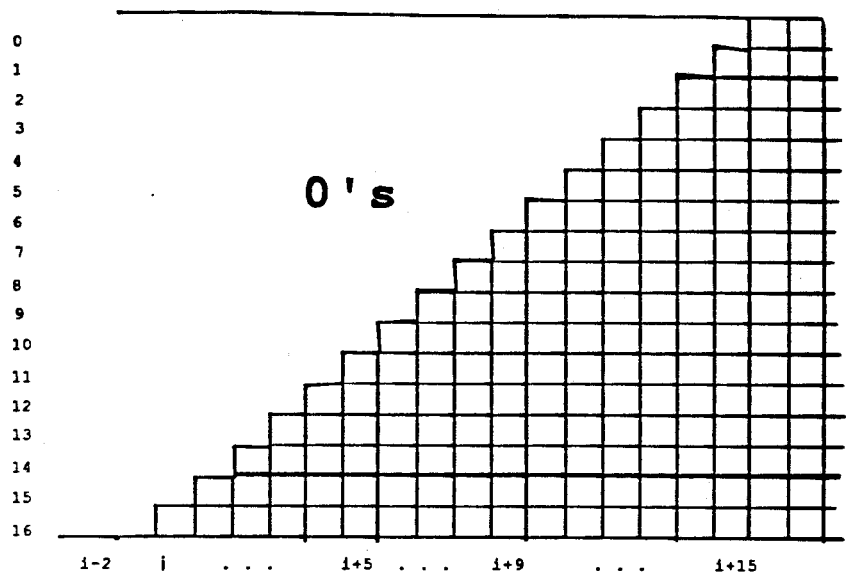
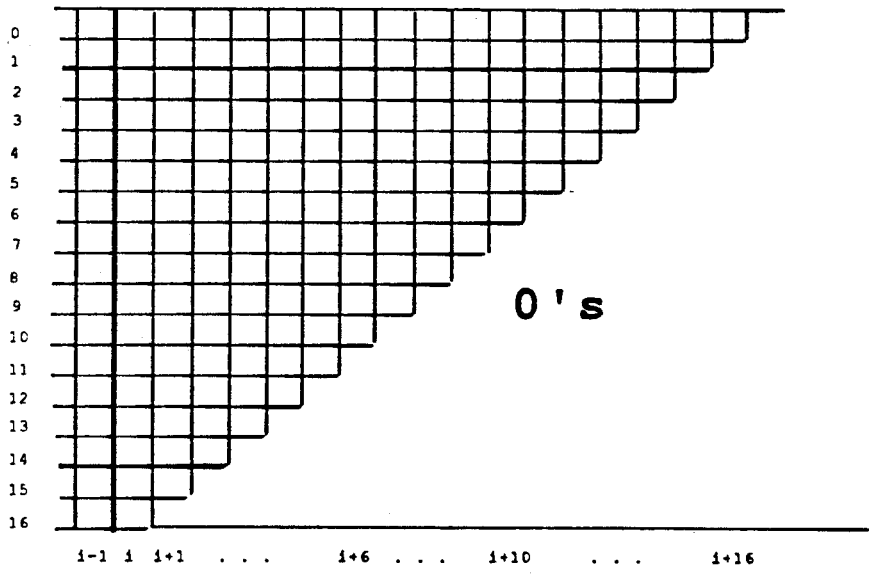


Figure 3

Data Format at Ends of Data Blocks



(a) Beginning of a Data Block



(b) End of a Data Blocks

Figure 4

4. Decoding Process

When data are read out from the tape, the read data are checked for errors by means of the coding Equations (4) and (5) and Equations (6) and (7). Let the received character be denoted by $\hat{C}_i(m) = C_i(m) + E_i(m)$, where $E_i(m)$ is an 8-bit error byte or erasure byte on the ith column and on the mth track of the tape. Let α be a primitive element of $GF(2^4)$, where $GF(2^8)$ is a subfield of $GF(2^8)$. There are two types of syndromes.

a) *Diagonal syndromes :*

$$S^d_{i,0} = \hat{C}_{i+14}(0) + \hat{C}_{i+13}(1) + ... + \hat{C}_i(14) \tag{8}$$

$$S^d_{i,1} = \hat{C}_{i+14}(0) + \alpha\hat{C}_{i+13}(1) + ... + \alpha^{14}\hat{C}_i(14) \tag{9}$$

b) *Vertical syndromes :*

$$S^v_{i,0} = \hat{C}_i(16) + \hat{C}_i(15) + ... + \hat{C}_i(2) \tag{10}$$

$$S^v_{i,1} = \hat{C}_i(16) + \alpha\hat{C}_i(15) + ... + \alpha^{14}\hat{C}_i(2) \tag{11}$$

By comparing (8),(9) with (4), (5) we get

$$S^d_{i,0} = E_{i+14}(0) + E_{i+13}(1) + ... + E_i(14) \tag{12}$$

$$S^d_{i,1} = E_{i+14}(0) + \alpha E_{i+13}(1) + ... + \alpha^{14}E_i(14) \tag{13}$$

Also by comparing (10),(11) with (6), (7) we get

(14) $$S^v_{i,0} = E_i(16) + E_i(15) + ... + E_i(2)$$

(15) $$S^v_{i,1} = E_i(16) + \alpha E_i(15) + ... + \alpha^{14} E_i(2).$$

The decoding algorithm is a set of rules that integrates the error-correcting capabilities of the diagonal and vertical codes, which are both 1-error correcting codes, to form a two-dimensional code that can correct multi-track errors and/or erasures along the length of the tape. The decoding proceeds column by column, with the assumption that all data bytes in previous columns are correctly decoded. In some cases parity bytes along tracks 0,1,16 and 17 are left uncorrected. It is shown later in this section that under normal circumstances (the number of erroneous tracks does not exceed the code's capability), these uncorrected parity bytes would not affect the code's error-correcting capability on the data tracks.

Suppose that the decoder starts to decode column i on the tape. The decoder first examines (in a specific order) the 15 diagonal syndrome pairs associated with the 15 diagonal received patterns, each of which shares a character with the vertical received pattern in column i. When a nonzero diagonal syndrome pair is detected, diagonal decoding corrects the first error or the first two erasures in that column. In the case of erasure decoding (during diagonal decoding), erasures in later columns are sometimes decoded. The detailed diagonal decoding process is described later in this section. After diagonal decoding, the decoder switches to vertical decoding, during which the decoder clears up the remaining error or erasures in that column. This switching from diagonal decoding to vertical decoding takes place when one of the following conditions is met:

1) A received pattern with a single error is detected and corrected in diagonal decoding.

2) A received pattern with two erasures is detected and corrected in diagonal decoding.

3) A received pattern with uncorrectable error(s) and/or erasure(s) is detected in diagonal decoding.

If none of the above conditions is satisfied during diagonal decoding of the 15 diagonal received patterns associated with column i, the decoder will skip vertical decoding of column i and proceed to decode the next column. Another important point about this hybrid code is that each error/erasure decoding by the block code is followed by syndrome updating. The decoding algorithm is not a rigid set of rules. Some minor variations are allowed, depending on the tradeoff between decoding complexity and error-correcting capability.

As in the AXP code, error forecasting strategy[6] may be employed. Once an internal or external track-error pointer is generated, it may be kept on for the entire remaining length of the record. Similarly, any track-error pointer may be turned off at an appropriate byte position in a record, if the error patterns corresponding to the indicated track turn out to be zero consistently for a significant length of the record—thus confirming that the track is error-free.

The following is a detailed discussion on a simple decoding algorithm that can correct the following error/erasure configurations :

1) two track errors or less,

2) four track erasures or less, or

3) one track error and two track erasures or less.

Note that in each block decoding (diagonal or vertical) of the $(15, 13)$ BCH code, the decoder encounters one of three possible states :

a) no erasure state,

b) one erasure state, and

c) two erasures state.

Assuming that the ith column is being decoded. The decoder is first informed by an external indicator on the current erasure track distribution. The diagonal syndromes $S^d_{i,0}$ and $S^d_{i,1}$ are evaluated as in Equations (8) and (9). The diagonal syndrome pairs $(S^d_{i-14,0}, S^d_{i-14,1})$, $(S^d_{i-13,0}, S^d_{i-13,1})$ ... are then examined in this order until a nonzero pair is detected. With the information supplied by the external indicator, the decoder proceeds to decode—diagonal then vertical—according to the decoding algorithm as shown in the flow chart in Figure 5. The decoding algorithm is simple but it can correct error configurations (1), (2) and (3) mentioned above.

a) Two unknown error tracks or less

The decoder corrects the error(s) in column i only. We have three different cases to consider.

i. *Tracks 0,1 are not both error tracks and tracks 15,16 are not both error tracks.*

The error configuration is shown in Figure 6. Assume that track j and track k are error tracks, $j < k$. The presence of the first error byte $E_i(j)$ in column i is indicated by the nonzero diagonal syndrome pair $(S^d_{i-14+j,0}, S^d_{i-14+j,1})$. The decoder first checks to see if the first error is on column i by the following equation:

$$S^d_{i-14+j,1} = \alpha^j S^d_{i-14+j,0}. \tag{16}$$

Flow Chart of the Decoding Algorithm

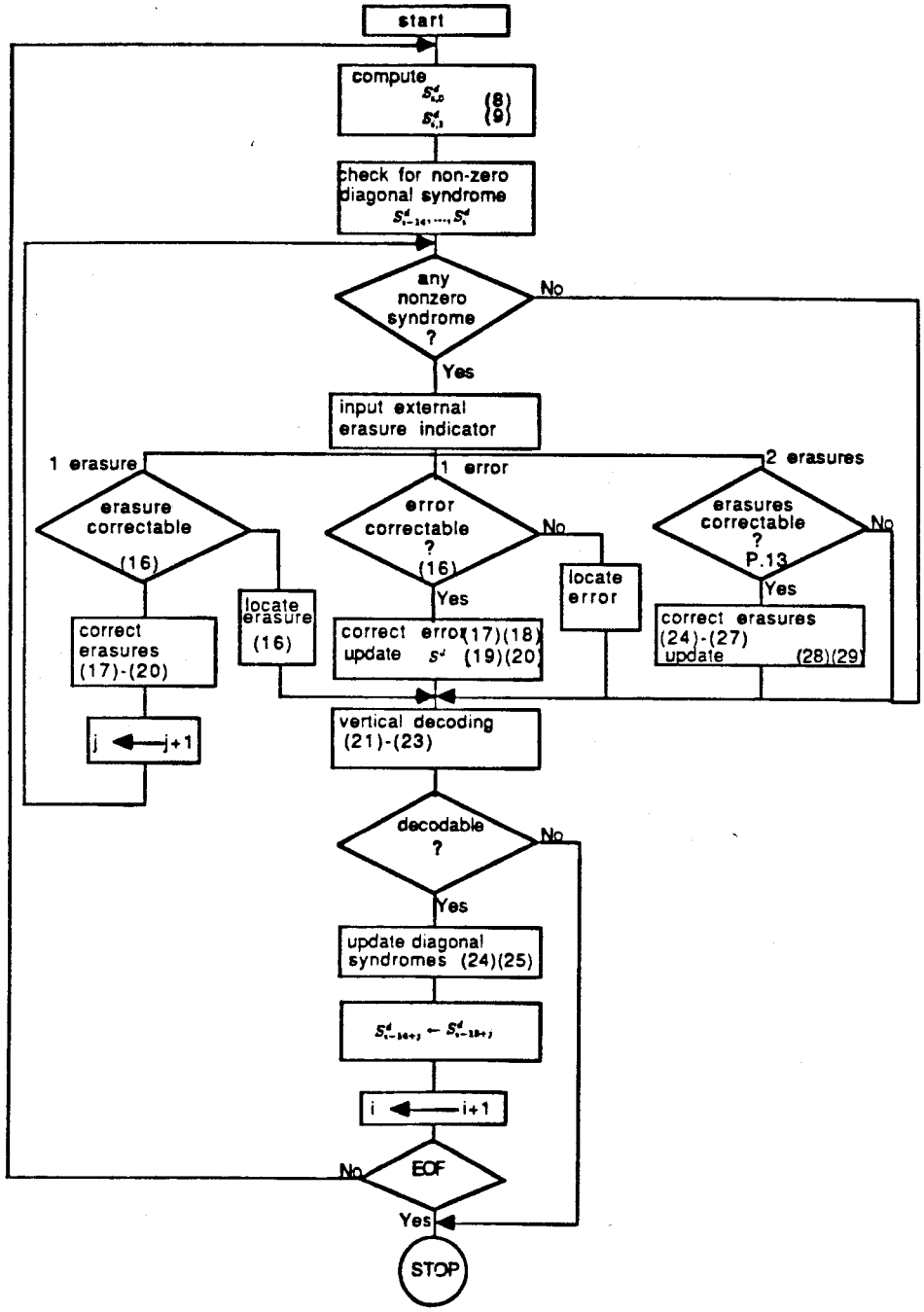


Figure 5

If (11) is satisfied, the decoder corrects the error by the following equations:

(17) $$E_i(j) = S^d_{i-14+j,0}$$

(18) $$C_i(j) = \hat{C}_i(j) + E_i(j).$$

The syndromes $S^d_{i-14+j,0}$ and $S^d_{i-14+j,1}$ are updated as follows:

(19) $$S^d_{i-14+j,0} = S^d_{i-14+j,0} + E_i(j)$$

(20) $$S^d_{i-14+j,1} = S^d_{i-14+j,1} + \alpha^j E_i(j).$$

The decoder then switches to vertical decoding. The vertical syndromes $S^v_{i,0}$, $S^v_{i,1}$ are evaluated as in Equations (10) and (11). The location of the second error $E_i(k)$ — the index k — is found by the following equation (k is unknown):

(21) $$S^v_{i,1} = \alpha^k S^v_{i,0}.$$

This involves at most 15 shift-and-compare operations[2]. The error is then corrected as follows:

(22) $$E_i(k) = S^v_{i,0}$$

(23) $$C_i(k) = \hat{C}_i(k) + E_i(k).$$

The diagonal syndromes $S^d_{i-14+k,0}$ and $S^d_{i-14+k,1}$ are updated by the following equations:

(24) $$S^d_{i-14+k,0} = S^d_{i-14+k,0} + E_i(k)$$

(25) $$S^d_{i-14+k,1} = S^d_{i-14+k,1} + \alpha^k E_i(k)$$

ii. *Track 0 and track 1 are error tracks*

The error configuration is shown in Figure 7. During diagonal decoding of column i, the decoder first detects the nonzero syndrome pair $(S^d_{i-14,0}, S^d_{i-14,1})$. It then finds that Equation (16) is not satisfied ($S^d_{i-14,1} \neq S^d_{i-14,0}$). The decoder marks the character in the diagonal codeword (corresponding to the syndromes $S^d_{i-14,0}$ and $S^d_{i-14,1}$) in column i as an erasure before it switches to vertical decoding.

iii. *Track 16 and track 17 are error tracks*

The error configuration is shown in Figure 8. The decoder first detects that all 15 diagonal syndrome pairs are zeros. The decoder then skips vertical decoding and moves to decode the next column.

b) Four erasure tracks or less

The error configuration is shown in Figure 9. The diagonal decoding involves the corrections of single erasures in a number of diagonal codewords (erasures are not required to be in column i), and the correction of double erasures in a diagonal codeword (one of the erasures is required to be in column i). Without lost of generality, assume that the erasures are on tracks j, k, l and m, where $j < k < l < m$. The presence of first erasure $E_i(j)$ is indicated by the nonzero syndrome pair $(S^d_{i-14+j,0}, S^d_{i-14+j,1})$ and the external erasure indicator. The decoder first checks the decodability of the diagonal codeword containing this erasure by Equation (16) in Part (a). That is, the decoder checks to see if the right-hand-side and the left-hand-side of Equation (16) are equal. Notice that in this case $S^d_{i-14+j,0}$, $S^d_{i-14+j,1}$ and j (erasure location given by external erasure indicator) are known quantities.

If Equation (16) is satisfied the decoder proceeds to correct $E_i(j)$ by Equations (17) and (18), and the syndromes $S^d_{i-14+j,0}$ and $S^d_{i-14+j,1}$ are updated as in Equations (19) and (20) of Part (a). The decoder continues to correct erasures $E_{i+1}(j)$, $E_{i+2}(j)$,...,$E_{i+k-j-1}(j)$ and updates the diagonal syndrome pairs $(S^d_{i-13+j,0}, S^d_{i-13+j,1})$, $(S^d_{i-12+j,0}, S^d_{i-12+j,1})$,...,$(S^d_{i-15+k,0}, S^d_{i-15+k,1})$ in a similar manner as above. It then encounters a diagonal received pattern with two erasures—$E_{i+k-j}(j)$ and $E_i(k)$. The decoder first checks to see if one of the erasures is on column i (with information supplied by the external erasurer indicator). If the check is positive, the decoder corrects erasures $E_{i+k-j}(j)$ and $E_i(k)$ by the following equations:

(24) $$E_i(k) = [1 + \alpha^{k-j}]^{-1}[S^d_{i-14+k,0} + \alpha^{-j} S^d_{i-14+k,1}]$$

(25) $$E_{i+k-j}(j) = E_i(k) + S^d_{i-14+k.0}$$

(26) $$C_i(k) = \hat{C}_i(k) + E_i(k)$$

(27) $$C_{i+k-j}(j) = \hat{C}_{i+k-j}(j) + E_{i+k-j}(j).$$

The diagonal syndromes are then updated as in the following equations:

(28) $$S^d_{i-14+k,0} = S^d_{i-14+k,0} + E_{i+k-j}(j) + E_i(k)$$

(29) $$S^d_{i-14+k,1} = S^d_{i-14+k,1} + \alpha^j E_{i+k-j}(j) + \alpha^k E_i(k).$$

The decoder then switches to vertical decoding. The remaining two erasure bytes ($E_i(l)$ and $E_i(m)$) on column i are then corrected. The diagonal syndromes affected by $E_i(l)$ and $E_i(m)$ are updated as follows:

(30) $$S^d_{i-14+l,0} = S^d_{i-14+l,0} + E_i(l)$$

(31) $$S^d_{i-14+l,1} = S^d_{i-14+l,1} + \alpha^l E_i(l)$$

(32) $$S^d_{i-14+m,0} = S^d_{i-14+m,0} + E_i(m)$$

(33) $$S^d_{i-14+m,1} = S^d_{i-14+m,1} + \alpha^m E_i(m).$$

c) One error track and two erasure tracks or less

We have five cases to consider:

i. *One error track and one erasure track in diagonal parity tracks*

The error configuration is shown in Figure 10. Let an error track be on track j, $2 \leq j \leq 16$, in addition to the two erroneous tracks in the diagonal parity tracks. During diagonal decoding of column i, the decoder finds that $(S^d_{i-14,0}, S^d_{i-14,1})$ is a nonzero syndrome pair, but the diagonal codeword associated with this syndrome pair is uncorrectable. The decoder then switches to vertical decoding to correct the remaining erasure $E_i(j)$ in column i. The diagonal syndromes affected by $E_i(j)$ are then updated and the decoder proceeds to decode the next column.

ii. *An error track and an erasure track in the vertical parity tracks*

The error configuration is shown in Figure 11. As in Part (i) we assume the existence of an erasure track along track j in addition to an error track and an erasure track in the vertical parity tracks. During diagonal decoding, the decoder first corrects erasures $E_i(j)$, $E_{i+1}(j)$, ..., $E_{i+14-j}(j)$ as in Equations (23) and (24) in Part (b). The syndromes that are affected by these erasures are then updated as in Equations (25) and (26). Since no unknown error, uncorrectable error or double erasures are detected in the process of diagonal decoding, the decoder skips vertical decoding and proceeds to decode the next column.

iii. *An error track between two erasure tracks* (other than case i. and ii.)

The error configuration is shown in Figure 12. During diagonal decoding the decoder corrects erasures $E_i(j)$, $E_{i+1}(j)$, ..., $E_{i+k-j-1}(j)$ as in Equations

(16), (17) and (18) in Part (a). The syndromes affected by these erasures are updated as in Equations (19) and (20). The decoder then proceeds to examine the diagonal syndromes $S^d_{i-14+k,0}$ and $S^d_{i-14+k,1}$, and it would find that the received pattern associated with the syndromes is uncorrectable (Theorem 5.2). The decoder thus detects the presence of error byte in $\hat{C}_i(k)$ and marks $\hat{C}_i(k)$ as an erasure character. It then switches to vertical decoding and corrects erasures $E_i(k)$ and $E_i(l)$. The diagonal syndromes affected by $E_i(k)$ and $E_i(l)$ are then updated.

iv. *One error track above two erasure tracks* (other than case i. and ii.)

The error configuration is shown in Figure 13. During diagonal decoding the decoder corrects the first error $E_i(j)$ and updates the diagonal syndromes affected by $E_i(j)$ as in case (a). The decoder then switches to vertical decoding to correct erasures $E_i(k)$ and $E_i(l)$ and updates the diagonal syndromes affected by $E_i(k)$ and $E_i(l)$.

v. *An error track below two erasure tracks* (other than case i. and ii.)

The error configuration is shown in Figure 14. As in case (b), during diagonal decoding the decoder first corrects erasures $E_i(j), \ldots, E_{i+k-j-1}(j)$ along track j and updates the affected syndromes. The decoder proceeds to correct the diagonal received pattern with two erasures $E_i(k)$ and $E_{i+k-j}(j)$. The syndromes affected by $E_i(k)$ and $E_{i+k-j}(j)$ are then updated. The decoder then switches to vertical decoding to correct the error $E_i(l)$ and updates the diagonal syndromes affected by $E_i(l)$.

If no uncorrectable error is detected after vertical decoding, then the decoder updates the 14 pairs of syndromes as follows :

(34) $$S^d_{i-14+j,0} = S^d_{i-13+j,0}$$

(35) $$S^d_{i-14+j,1} = S^d_{i-13+j,1}$$

for $0 \leq j \leq 13$. i is then incremented by one. If the End-of-File is not detected, then the decoding proceeds for the next column. If the uncorrectable error is detected in vertical decoding, then the decoding process stops. The above algorithm requires only two syndrome calculations (one in vertical decoding and one in diagonal decoding), and requires on the average at most two single-error corrections (for a simple one-error correcting code) in each column decoding. The number of calculations involved in hybrid code decoding is much less than the number of calculations in the case of using a double error correcting code in each column.

In some cases, errors on the parity tracks may not be recovered. This is because each character of the four parity tracks is included only in one single-error correcting code, whereas each character of the data tracks is protected by two block codes—the vertical code and the diagonal code. Thus, the information region of the tape receives more protection than the parity region. A more detailed study of the hybrid code shows that this code can correct some error patterns (in the information region) that cannot be corrected by a simple 2-error correcting code. In other words, this code enhances the error-correcting power in the information region at the expense of the error-correcting power in the parity region.

Two error tracks in information region

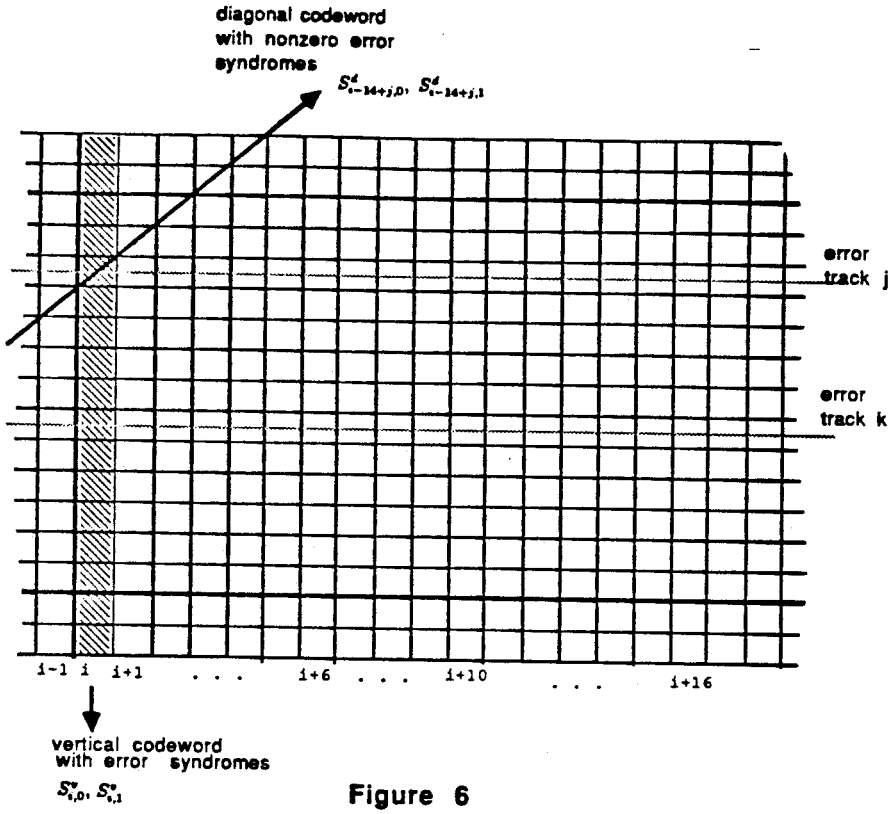


Figure 6

Two Error Tracks in Diagonal Parity Region

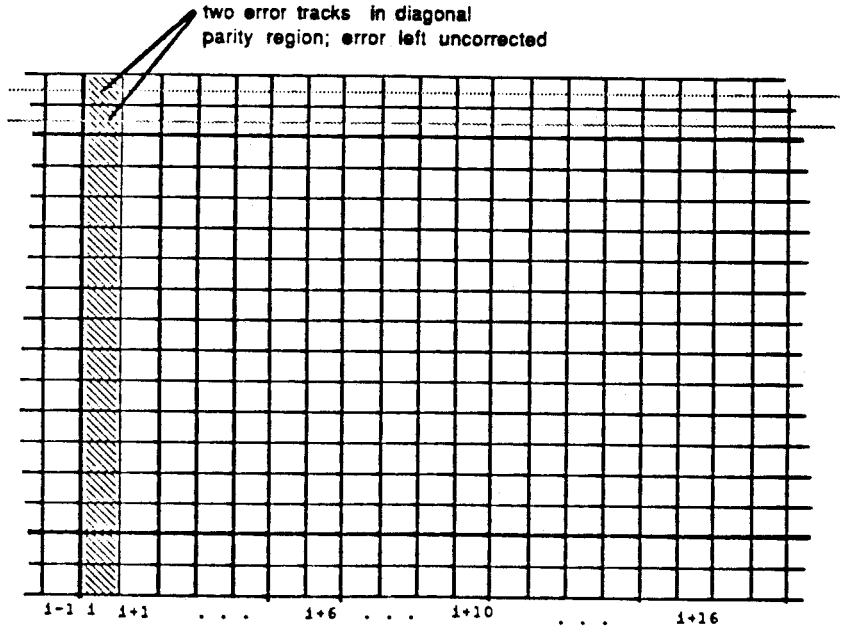


Figure 7

Two error tracks in vertical parity region

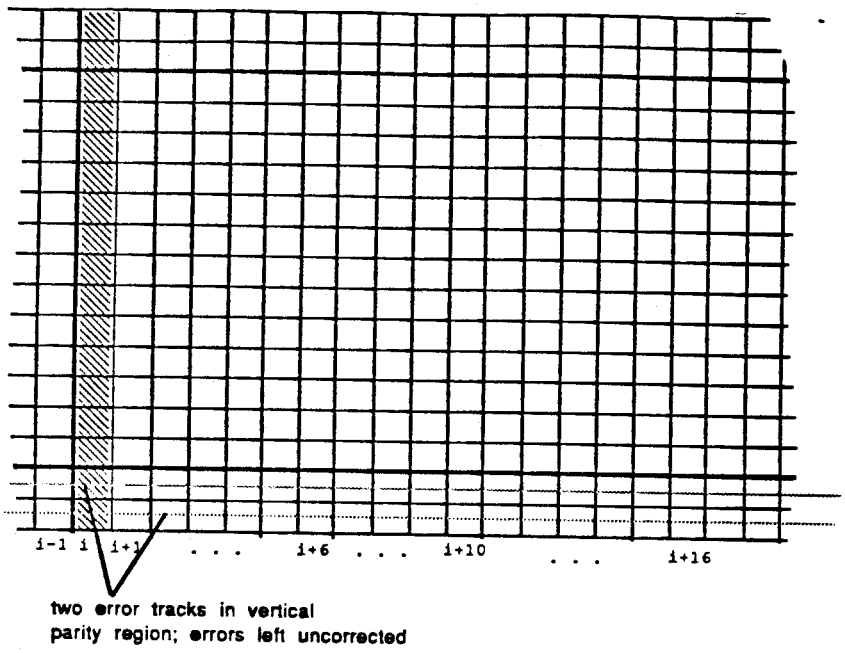


Figure 8

Four erasure tracks

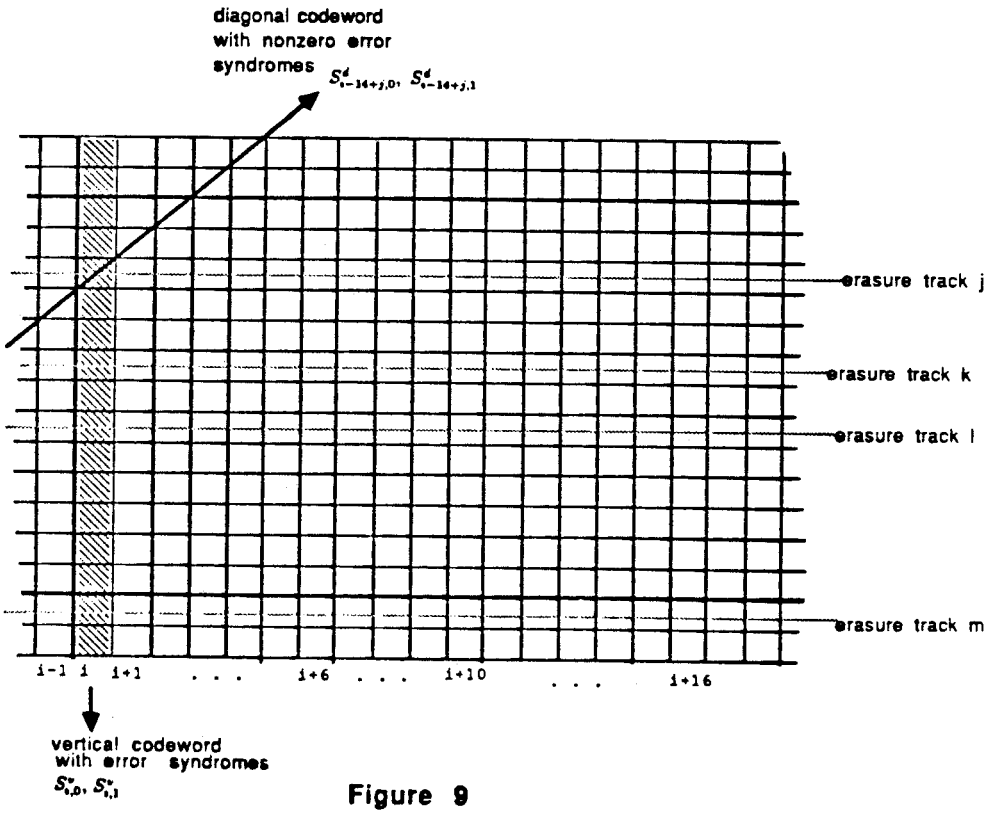


Figure 9

One Error Track and One Erasure Track in Diagonal Parity Region

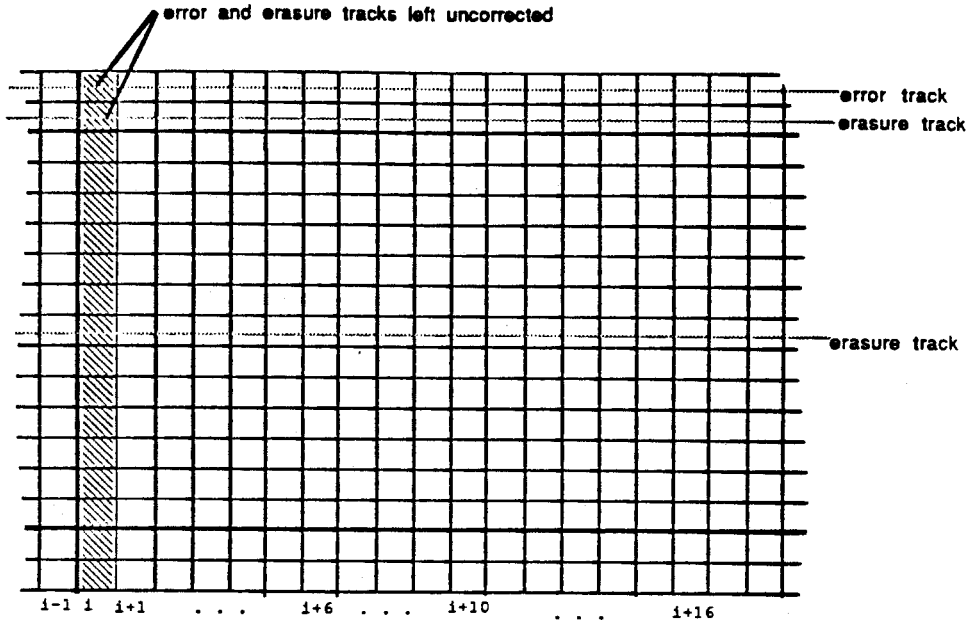


Figure 10

One Error Track and One Erasure Track on the Vertical Parity Region

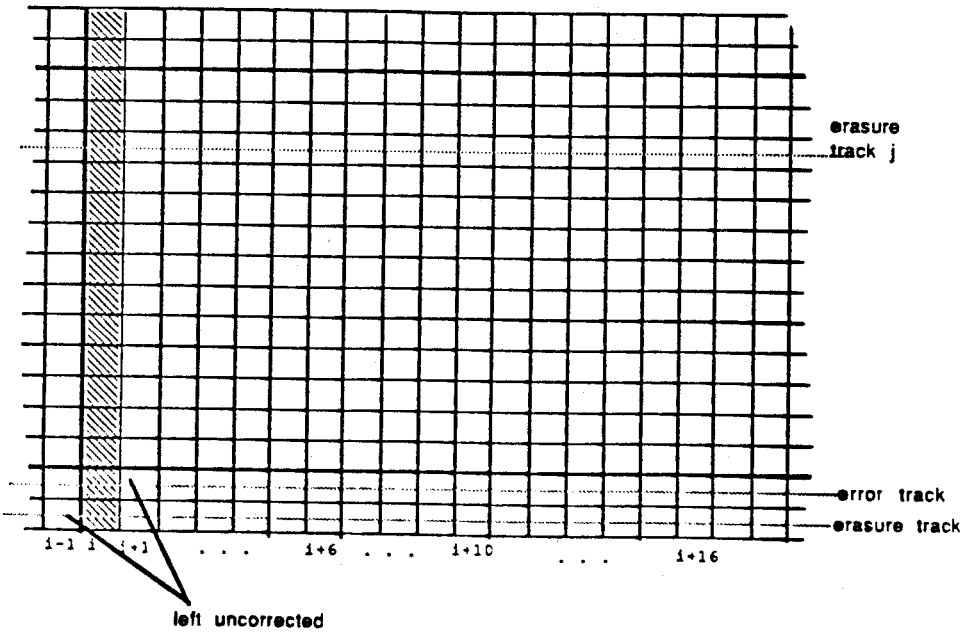


Figure 11

An Error Track between Two Erasure Tracks

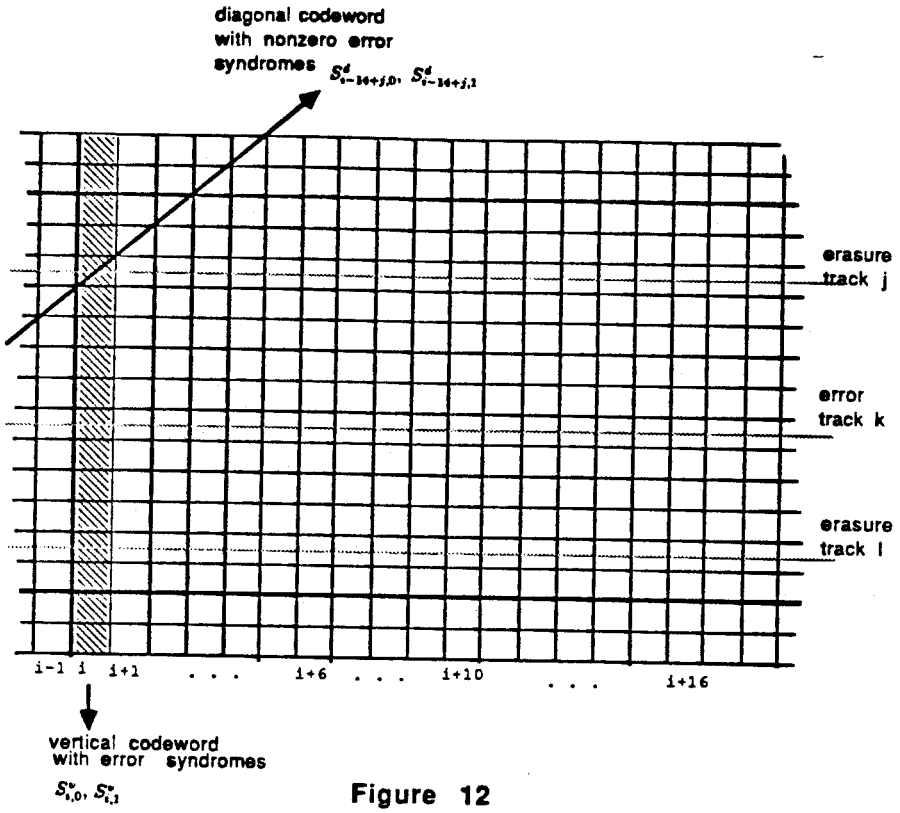


Figure 12

An Error Track above Two Erasure Tracks

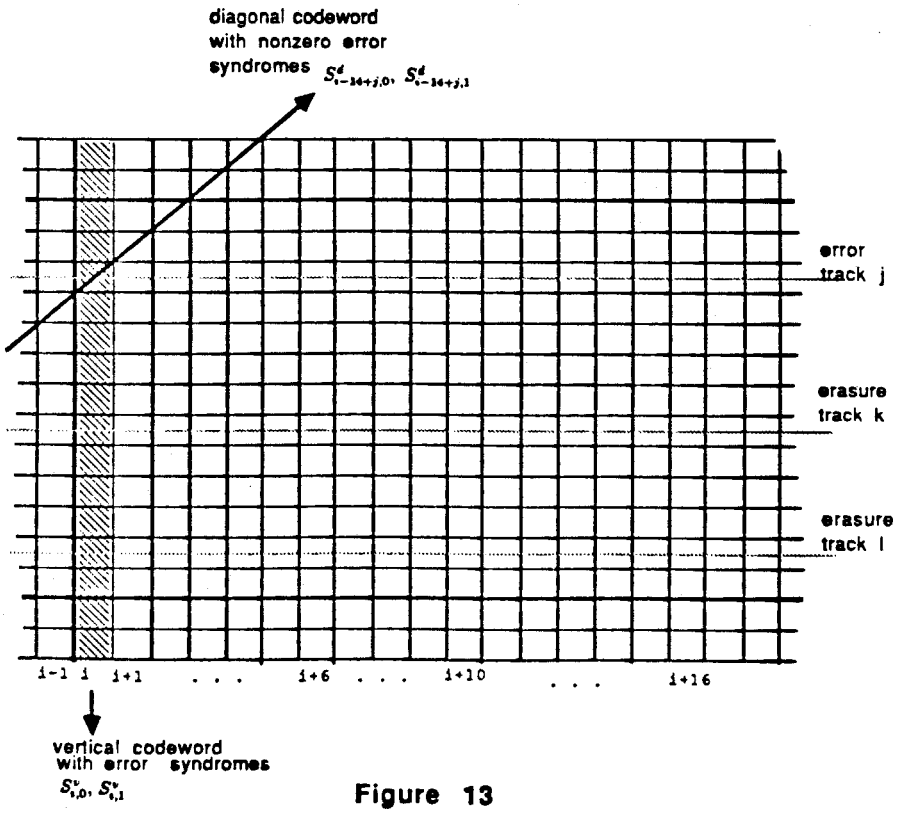


Figure 13

An Error Track below Two Erasure Tracks

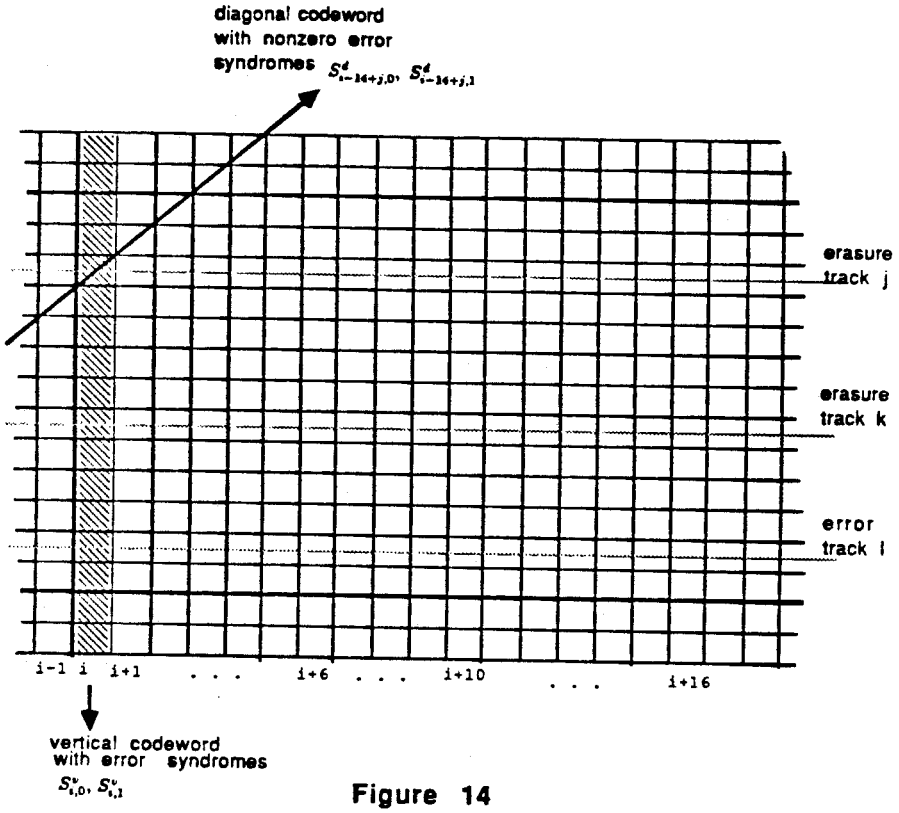


Figure 14

5. Conclusion and Generalization

In this paper a new code, which is a hybrid of the block code and the convolution code, is presented. This hybrid code is particularly designed for error correction in a parallel track magnetic tape environment, in which the common types of errors are long tracks of errors and erasures, together with random bit errors and 2-consecutive-bit errors. This code integrates the error-correcting capabilities of simple block codes in different directions by a convolutional structure. This provides the required error correcting capability without a corresponding increase in redundancy or complexity. The use of $(15, 13)$ BCH code as a block code makes the code very efficient in preventing miscorrection.

The data format of this code is not confined just to the one shown in Figure 1. Instead of orienting the block codes in a vertical direction (with slope $= \infty$) and a diagonal direction (with slope $= 1$) as shown in Figure 1, the two block codes can both be in diagonal directions with slopes 1 and -1. In this case, the geometric configuration of the code offers protection against intersymbol interference between tracks. The concept of integrating the error-correcting capability of simple block codes in directions by a convolutional structure can be easily generalized to multi-error correction schemes.

CHAPTER VI

A NEW LABELLING PROCEDURE FOR TRELLIS CODES

1. Introduction

We start with the following definition:

Definition 6.1:

An (n, k, m) trellis code on a c-connected state diagram is a code with the following properties:

a) The code has rate $\frac{k}{n}$.

b) Its operation can be represented by a state diagram with 2^m states.

c) There are 2^c $(c \leq m)$ branches going into each state, and 2^c branches going out of each state.

d) Each branch of the state diagram is associated with a code (codeword length = n and code size = 2^{k-c}), and any two different codes associated with different branches are disjoint.

It was shown in [7] that an (n, k, m) trellis code [6] on a m-connected state diagram (completely connected) requires at least 2^{m+1} labels. Also, a simple method to define such labels has been suggested in [7]. In this memo a new method to define the labels using shift registers is presented. This new method is particularly suitable

for simple hardware implementation since it simplifies the encoder structure. This method can also be applied to the labelling of a state diagram that is not completely connected to obtain a trellis code with larger free distance.

2. Preliminary

We first review some important results in the theory of convolutional codes. These results would be referred to in the proofs in later sections.

A typical encoder of a (n_1, c, m) convolutional code consists of a linear sequential circuit (with c shift registers) that accepts c input bits and outputs n_1 bits. It is well known that the operation of the encoder can be represented by :

1) a state diagram with 2^m states and 2^c branches going into each state and 2^c branches going out of each state, or

2) a $c \times n_1$ transfer function matrix (denoted by $G(D)$) such that the entries of the matrix are polynomials in D, representing the generator sequences of the code.

In order to avoid catastrophic error propagation, the transfer function matrix must satisfy Massey and Sain's condition [8] (a necessary and sufficient condition) on non-catastrophic codes:

$$(1) \qquad GCD[\Delta_i(D), \qquad i = 1, 2, ... \binom{n_1}{k}] = D^l$$

for some $l \geq 0$. $\Delta_i(D)$, $i = 1, 2, ..., \binom{n_1}{c}$, are the determinants of the $\binom{n_1}{c}$ distinct $c \times c$ submatrices of the transfer function matrix $G(D)$.

3. Generation of Labels by Shift Register

Trellis code encoders have structural properties very similar to those of convolutional encoders, and the operation can be described by a state diagram. In the case of a convolutional code, each branch of the state diagram is labelled by an n_1-bit output sequence, whereas in the case of a trellis code each branch is labelled by a code, not necessarily linear. Any two different codes associated with different branches must be disjoint. Because of the similarities between the convolutional code and the trellis code, we would expect that much of the theory on structural properties of convolutional codes would be applicable to trellis codes.

In order to guarantee a noncatastrophic trellis code with good distance properties, the labelling of the branches of the state diagram must satisfy the following conditions[6][7]:

1) different labels out of each state,

2) different labels into each state, and

3) no paths with identical labels that remain unmerged indefinitely.

Now we want to suggest a method to define the labels of the state diagram of a trellis code by using the linear sequential circuit (with shift registers) of a noncatastrophic (n_1, c, m) convolutional code. Let the c shift registers have lengths $l_1, l_2, ...l_c$ and $l_1 + l_2 ... + l_c = m$. The pth row of the corresponding $c \times n_1$ transfer function matrix thus consists of polynomials in D of degree no greater than l_p for $1 \leq p \leq c$. The state diagram of the convolutional code consists of 2^m states (each state is defined by the shift register content); also, there are 2^c branches going into each state and 2^c branches going out of each state. Each branch in the state diagram is assigned an n_1-bit sequence $b_0, b_1, ...b_{n_1-1}$, which consists of the n_1

output bits of the shift registers. Let's assign to the branches of the state diagram, which are associated with the n_1-bit sequence $b_0, b_1, ...b_{n_1-1}$, the label i such that $i = b_0 + 2b_1 + ... + 2^{n-1}b_{n_1-1}$. Each of these labels represents one of the disjoint codes. There are 2^{n_1} of them. We use this modified state diagram of the convolutional code as the state diagram of an (n, k, m) trellis code on a c-connected state diagram.

Now we are in a position to show the way to construct a shift register circuit that generates the state diagram of a trellis code that satisfies conditions (1), (2) and (3). It is not hard to see that condition (1) is satisfied, if for a fixed shift register content, different inputs to the shift registers produce different outputs. This can be achieved if there exists at least one $c \times c$ submatrix $\Omega_i(D)$ of the transfer function matrix $G(D)$, $i = 1, 2, ...\binom{n_1}{c}$, such that the term "1" appears exactly once in each row and in each column of $\Omega_i(D)$. Similarly, condition (2) is satisfied if, for a fixed input, different shift register contents produce different outputs. This can be achieved if there exists at least one $c \times c$ submatrix $\Omega_j(D)$, $j = 1, 2, ...\binom{n_1}{c}$, such that the term D^{l_p} representing the last shift register stage of the pth shift register appears exactly once in row p for $1 \leq p \leq c$, and each of these $D^{l_1}, D^{l_2}, ...D^{l_c}$ terms appears in different columns of $\Omega_j(D)$.

The following theorem will show that if the (n_1, c, m) convolutional code that generates the state diagram of the trellis code is noncatastrophic, then the state diagram also satisfies condition (3). Thus, we require that the $c \times n_1$ transfer function matrix $G(D)$ of the convolutional code must satisfy Massey and Sain's condition [8] on noncatastrophic convolutional codes. An example of the construction of a trellis code state diagram that satisfies condition (1), (2), and (3) is given in Figure 1 and Figure 2.

State Diagram of the Convolutional Encoder

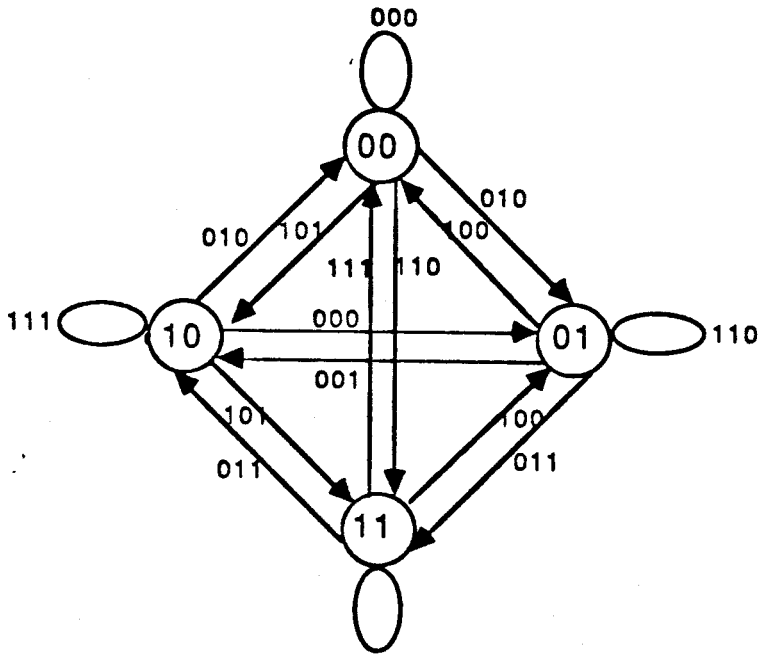
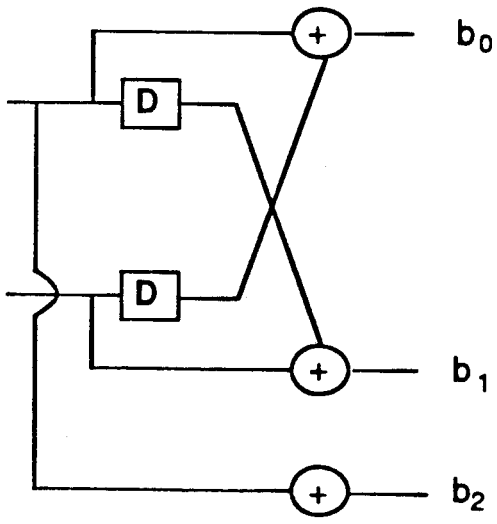


Figure 1

Convolutuonal Encoder That Generates the State Diagram of a Trellis Code



$$\begin{pmatrix} 1 & D & 1 \\ D & 1 & 0 \end{pmatrix}$$

000 --- 0
001 --- 1
010 --- 2
011 --- 3
100 --- 4
101 --- 5
110 --- 6
111 --- 7

Figure 2

Theorem 6.1:

If the (n_1, c, m) convolutional code that generates the state diagram of the trellis code is noncatastrophic, then the state diagram of the corresponding trellis code satisfies condition (3).

Proof:

Suppose that condition (3) is not satisfied. Then there exist two paths p and q with identical labels that remain unmerged indefinitely in the trellis diagram as shown in Figure 3. Let $l_1, l_2, \ldots, l_c$ be the lengths of the c shift registers of the (n_1, c, m) convolutional encoder that gives rise to the state diagram of the trellis code. Let $L = \max_{1 \le j \le c}\{l_j\}$. Let p_i and q_i represent the shift register states (at time i) along paths p and q, respectively, as shown in Figure 3. Let a_i/A_i denote the transition from state p_i to state p_{i+1} along p, where a_i is a block of c input bits to the convolutional encoder and A_i is the corresponding block of n_1 output bits. Similarly, b_i/B_i denotes the transition from state q_i to state q_{i+1} along q, where b_i is a block of c input bits to the encoder and B_i is the corresponding block of n_1 output bits. Notice that A_i and B_i represent the labels associated with the branches of the state diagram of the trellis code. Since by assumption the branch labels along p and q are identical, the corresponding convolutional encoder outputs A_i along p and B_i along q must be the same. Again, by assumption, paths p and q remain unmerged indefinitely, p_i and q_i must be different for all i. Since shift register content is made up of input bits from previous times, it requires that for every L transitions along p and q, there exists at least one a_i that is different from its counterpart b_i. If p is the correct path, a finite number of channel errors that divert the decoding to path q will cause at least one decoder bit error in every L transitions, even when no error is present in the channel bit stream thereafter. This implies that a finite number of channel errors can cause an infinite number of decoding errors and that the (n_1, c, m) convolutional code is catastrophic. This contradicts our hypothesis.

■

Two Disjoint Paths in the Trellis Diagram

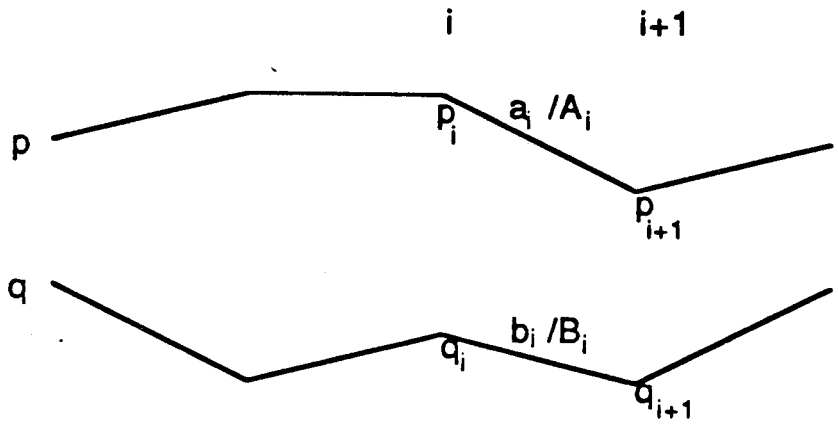


Figure 3

4. Properties

Based on the above labelling procedure by shift register, which is a linear sequential circuit, the trellis code possesses a mathematical structure that facilitates encoding/decoding and simplifies hardware implementation. Also, this labelling procedure is applicable to the construction of trellis codes with incompletely connected state diagrams to obtain larger free distance. Figure 4 and Figure 5 are simulation results of some trellis codes with incompletely connected state diagrams.

Definition 6.2:

Let N be the number of states of a trellis code. A labelling matrix L is defined to be an $N \times N$ matrix, where $L(i,j)$ denotes the label from state i to state j.

Let $\bar{u} = (u_1, u_2, \ldots, u_c)$ represents the c input bits to the convolutional encoder. Let $\bar{D} = (D_1, D_2, \ldots, D_c)$ represents the c last shift register stages of the convolutional encoder. That is, D_p represents the term D^{l_p} in row p for $1 \leq p \leq c$. With the following theorems, we would like to reveal some properties of trellis codes which use the new labelling procedure.

Theorem 6.2:

For a completely connected graph with 2^m states, we need at least 2^{m+1} labels. For an incomplete graph with 2^c branches going into each state and 2^c branches going out of each state, $c < m$, we need at least 2^{c+1} labels.

Proof:

We give the proof for the case of the incomplete graph and the proof for the complete graph will follow by setting $c = m$. Suppose that 2^c labels suffice. The transfer function matrix M of the convolution code that generates the state diagram of the trellis code is then a $c \times c$ matrix. By condition (1), since different labels are coming out of each state, the c output bits can be written as

$$\bar{u}\mathbf{A} + \bar{d},$$

where $\mathbf{A}$ is a $c \times c$ nonsingular matrix and $\bar{d}$ is a constant binary c-tuple which depends upon the shift register contents of the encoder. Thus, $|\mathbf{A}| = \alpha$, where α is a nonzero integer. Thus, we have the term α in the expression of $|\mathbf{M}|$. Similarly by

condition (2), since different labels are going into each state, the c output bits can be written as

$$\bar{D}\mathbf{B} + \bar{e},$$

where $\mathbf{B}$ is a $c \times c$ nonsingular matrix and $\bar{e}$ is a constant binary c-tuple, depending upon the input bits and the shift register contents other than $D_1, \ldots, D_c$. Again, $|\mathbf{B}| = \beta$ for some nonzero integer β . We therefore have $\beta\ D^m = D^{l_1+\ldots+l_c}$ terms in the expression of $|\mathbf{M}|$. Thus, $|\mathbf{M}| = \beta D^m + \ldots + \alpha$ and $|\mathbf{M}|$ is not of the form D^l for some $l \geq 0$. This violates Massey and Sain's condition on the noncatastrophic code. Thus, the convolutional code is catastrophic. By *theorem 6.1* the state diagram generated by this convolutional encoder is catastrophic and thus we need at least $c + 1$ output bits for the convolutional encoder. This implies that we need at least 2^{c+1} labels in the state diagram. ■

In fact, $c + 1$ output bits are sufficient to guarantee noncatastrophe of the convolutional code. In Section 4 we give several examples on the construction of convolution encoders with c input bits and $c + 1$ output bits. It can be shown that the state diagrams of the trellis codes generated by these convolutional encoders also satisfy conditions (1), (2), and (3).

Theorem 6.3:

All entries of a given row (or column) of a labelling matrix L are different.

Proof:

This is a direct consequence of conditions (1) and (2). ■

Theorem 6.4:

Row i and row j (column i and column j), $i \neq j$, of a labelling matrix L have either the same set of labels or a completely different set of labels.

Proof:

The state of the convolutional encoder that generates the required state diagram of the trellis code is defined as the shift register contents of the encoder. For an (n_1, c, m) convolutional code, let the binary m-tuple $[D_1, \ldots, D_m]$ denotes the state that corresponds to the shift register stages $D_1, \ldots, D_m$ of the encoder. Notice that the encoder is constructed in such a way that for a fixed state $[D_1, \ldots, D_m]$, different inputs to the shift registers produce different outputs (condition(1)). If we fixed $[D_1, \ldots, D_m] = [0, \ldots, 0]$, the set of all possible binary n_1-tuples (labels) that represent the output bits of the encoder form a c-dimension subspace K of an n_1-dimension vector space over $GF(2)$ (because the encoder is a linear sequential circuit). This set K is isomorphic to the row of the labelling matrix L that corresponds to the state $[0, \ldots, 0]$. Now, if we fix $[D_1, \ldots, D_m] \neq [0, \ldots, 0]$, then it is not hard to see that the set of all possible output binary n_1-tuples (output bits of the encoder) is of the form $K + \bar{e}$, where $\bar{e}$ is a binary n_1-tuple (constant) determined by $[D_1, \ldots, D_m]$. If $\bar{e} \notin K$, then K and $K + \bar{e}$ are disjoint (since K is a c-dimension subspace in an n_1-dimension vector space). If $\bar{e} \in K$, then $K = K + \bar{e}$. A similar argument holds for the case of $K + \bar{e}_1$ and $K + \bar{e}_2$, where $\bar{e}_1$ and $\bar{e}_2$ are binary n_1-tuples determined by different $[D_1, \ldots, D_m]$'s. That is, if $\bar{e}_1 \notin K + \bar{e}_2$, then $K + \bar{e}_1$ and $K + \bar{e}_2$ are disjoint. If $\bar{e}_1 \in K + \bar{e}_2$ then $K + \bar{e}_1 = K + \bar{e}_2$. This proves that any two rows of a labelling matrix L have either the same set of labels or a completely different sets of labels. The proof for the case of the columns is similar to the one above. ∎

5. Examples

This section gives some examples of constructing shift registers used for labelling the state diagram (completely connected and incompletely connected).

Example 1: complete graph 4 states 8 labels

$$G(D) = \begin{pmatrix} 1 & D & 1 \\ D & 1 & 0 \end{pmatrix} \qquad d_f = 2 \quad \text{branches}$$

Example 2: complete graph 16 states 32 labels

$$G(D) = \begin{pmatrix} 1 & 0 & 0 & D & 1 \\ D & 1 & 0 & 0 & 0 \\ 0 & D & 1 & 0 & 0 \\ 0 & 0 & D & 1 & 0 \end{pmatrix} \qquad d_f = 2 \quad \text{branches}$$

Example 3: incomplete graph 16 states 4 input/output 8 labels

$$G(D) = \begin{pmatrix} 1 & D^2 & 1 \\ D^2 & 1 & 0 \end{pmatrix} \qquad d_f = 2 \quad \text{branches}$$

$$G(D) = \begin{pmatrix} 1+D & D^2 & 1 \\ D^2 & 1+D & 0 \end{pmatrix} \qquad d_f = 3 \quad \text{branches}$$

Example 4: incomplete graph 64 states 4 input/output 8 labels

$$G(D) = \begin{pmatrix} 1+D+D^2 & D^3 & 1 \\ D^3 & 1+D+D^2 & 0 \end{pmatrix} \qquad d_f = 3 \quad \text{branches}$$

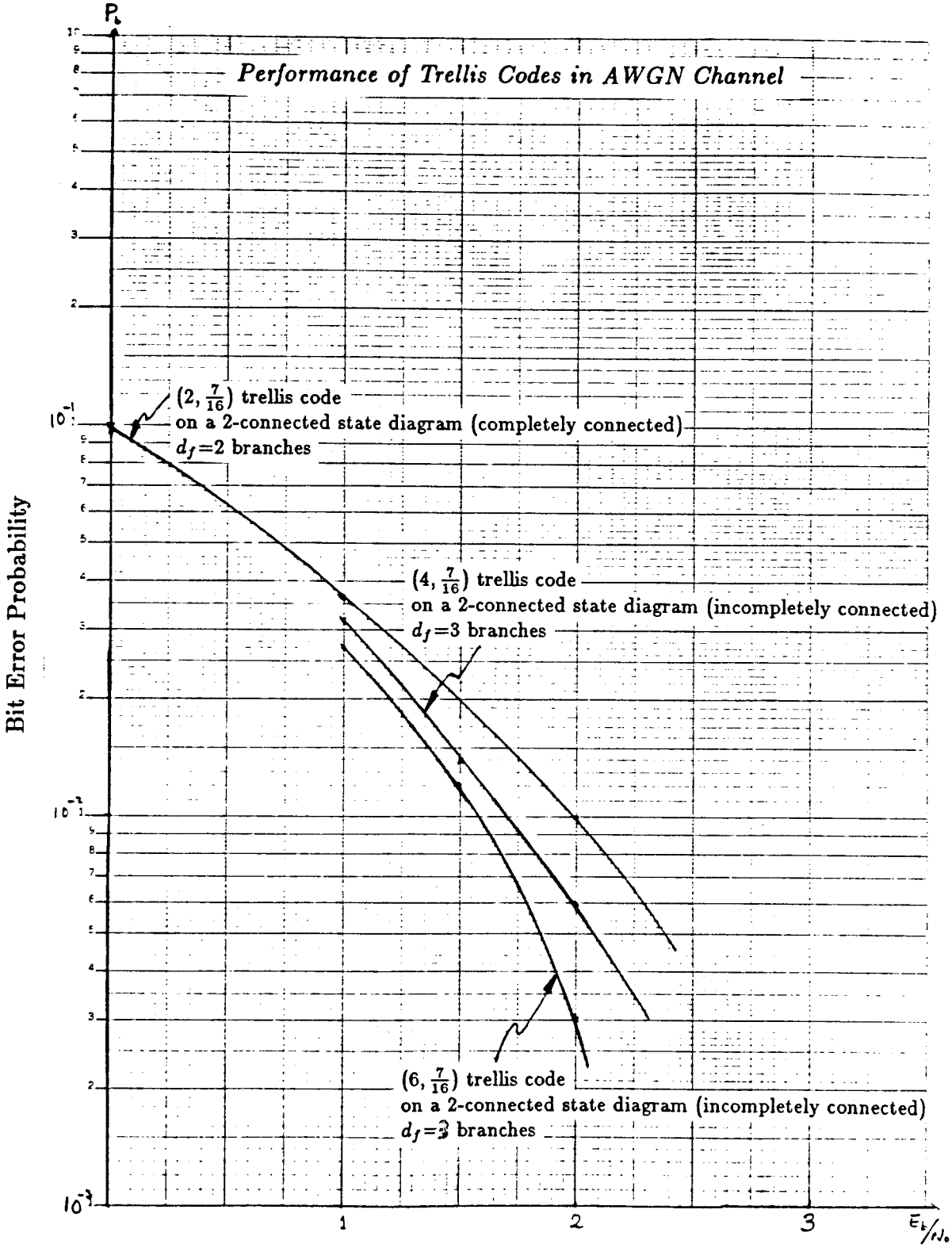

P_b
Performance of Trellis Codes in AWGN Channel
(2, 7/16) trellis code
on a 2-connected state diagram (completely connected)
d_f=2 branches
(4, 7/16) trellis code
on a 2-connected state diagram (incompletely connected)
d_f=3 branches
(6, 7/16) trellis code
on a 2-connected state diagram (incompletely connected)
d_f=3 branches
Bit Error Probability
Signal-to-Noise Ratio
E_b/N_0

Figure 4

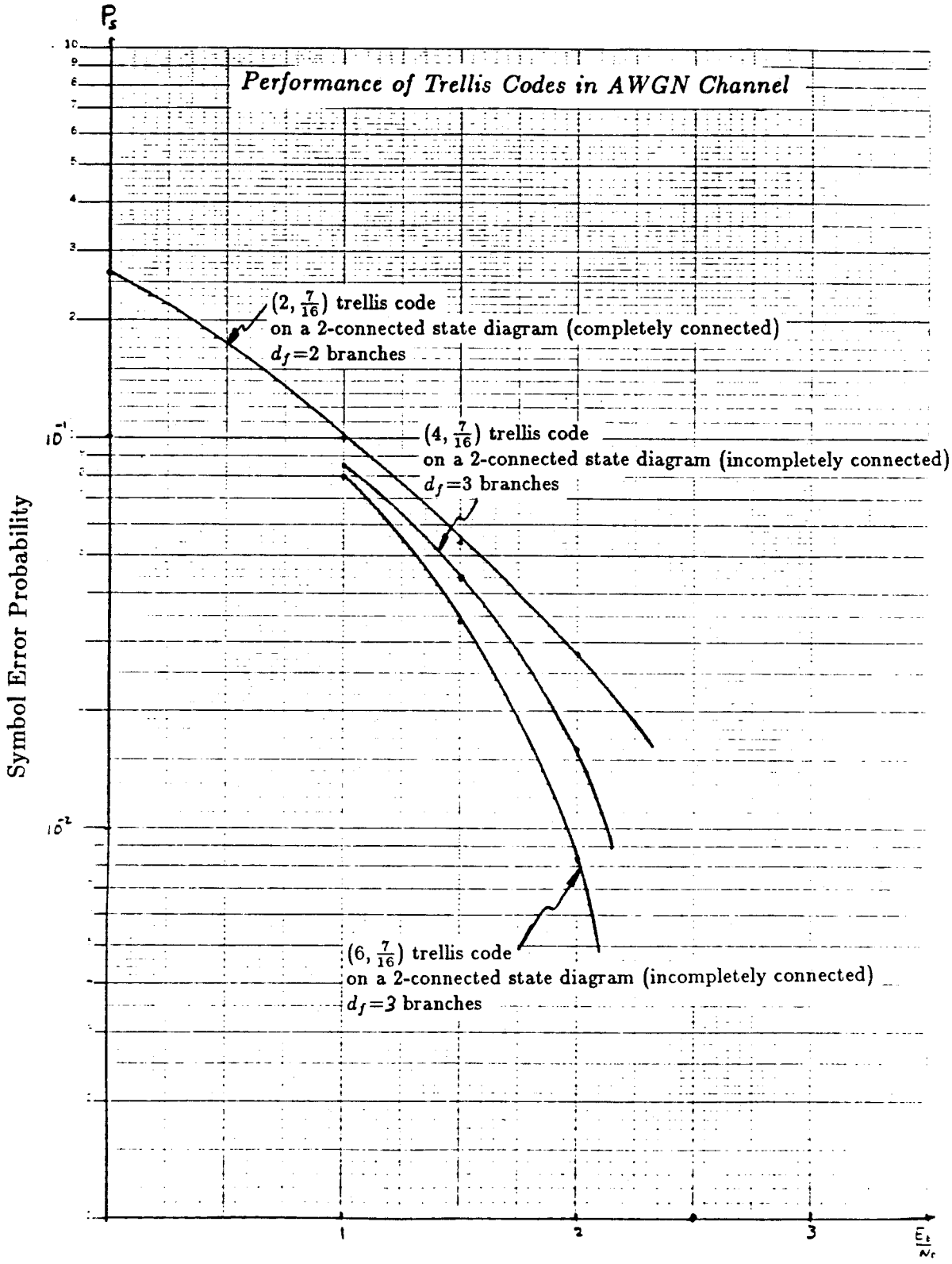

P_s
Performance of Trellis Codes in AWGN Channel
$(2, \frac{7}{16})$ trellis code
on a 2-connected state diagram (completely connected)
d_f=2 branches
$(4, \frac{7}{16})$ trellis code
on a 2-connected state diagram (incompletely connected)
d_f=3 branches
$(6, \frac{7}{16})$ trellis code
on a 2-connected state diagram (incompletely connected)
d_f=3 branches
10^{-1}
10^{-2}
1
2
3
$\frac{E_b}{N_0}$
Symbol Error Probability
Signal-to-Noise Ratio

Figure 5

CHAPTER VII

ON TWO CLASSES OF CODES FOR MAGNETIC RECORDING

1. Introduction

In a magnetic recording channel, a concatenation of an error-correcting code and a DC free code, or a concatenation of an error-correcting code and a runlength limited code is usually used. It would then be desirable to have a single code that possesses the properties of error-correction and DC freeness at the same time, or a code that possesses error-correcting and runlength limited properties at the same time. In this chapter we present two classes of codes that achieve the above goals:

1) an error-correcting DC free charge-constrained trellis code, and

2) an error-correcting runlength limited trellis code.

The two classes of codes mentioned above are trellis codes in structure, and the encoding and decoding operations can be described by means of state diagrams. These codes allow soft-decision decoding and thus possess high error-correcting capability.

We shall use the following definitions on these two classes of codes:

Definition 7.1:

An $(m, s, \frac{b}{n}, l, c, d_f)$ DC free error-correcting trellis code (binary) is a code with the following properties:

a) There are 2^m states in the state diagram of the encoder.

b) There are 2^s branches going into each state, and 2^s branches going out of each state ($s \leq m$).

c) Each branch in the state diagram depicts a transition from one state to another. Each transition represents an input of b information bits and an output of n channel bits.

d) Maximum runlength of a symbol (0 or 1) is l.

e) Maximum accumulation charge is c.

f) Free distance is d_f.

Definition 7.2:

An $(m, s, \frac{b}{n}, d, k, d_f)$ error-correcting runlength limited trellis code (binary) is a code with the following properties:

a) There are 2^m states in the state diagram of the encoder.

b) There are 2^s branches going into each state, and 2^s branches coming out of each state ($s \leq m$).

c) Each branch in the state diagram depicts a transition from one state to another. Each transition represents an input of b information bits and an output of n channel bits.

d) Minimum number of 0 between two 1's is d.

e) Maximum number of 0 between two 1's is k.

f) Free distance is d_f.

2. Error-correcting DC Free Trellis Code

We now present a $(4, 2, \frac{3}{6}, 4, 2, 6)$ error-correcting DC free code as follows:

Consider all binary 6-tuples with an equal number of 0's and 1's. There are $\binom{6}{3} = 20$ of them. Divide them into 10 groups such that each group contains a word and its complement. Label the groups from 0 to 9 as follows.

$$\left\{\begin{matrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{matrix}\right\} \quad \mathbf{0} \qquad \left\{\begin{matrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{matrix}\right\} \quad \mathbf{5}$$

$$\left\{\begin{matrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{matrix}\right\} \quad \mathbf{1} \qquad \left\{\begin{matrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{matrix}\right\} \quad \mathbf{6}$$

$$\left\{\begin{matrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{matrix}\right\} \quad \mathbf{2} \qquad \left\{\begin{matrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}\right\} \quad \mathbf{7}$$

$$\left\{\begin{matrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{matrix}\right\} \quad \mathbf{3} \qquad \left\{\begin{matrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{matrix}\right\} \quad \mathbf{8}$$

$$\left\{\begin{matrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{matrix}\right\} \quad \mathbf{4} \qquad \left\{\begin{matrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{matrix}\right\} \quad \mathbf{9}$$

It was shown in [9] that the state diagram of a trellis code can be generated by a convolutional encoder. Let us consider the state diagram generated by the shift register with a transfer function matrix

$$\begin{pmatrix} 1+D & D^2 & 1 \\ D^2 & 1+D & 0 \end{pmatrix}.$$

The state diagram has 16 states and 4 branches going into each state and 4 branches going out of each state. Map group $i, 0 \leq i \leq 7$, to the branches in the state diagram labelled by b_0, b_1, b_2 (the output bits of the shift register) such that $i = b_0 + 2b_1 + 4b_2$. According to the theory of trellis codes[6][7][9], we can construct a noncatastrophic trellis code for which there are at least 3 branch differences for any two paths to diverge from one state and remerge later. Now the distance between words in the same group, denoted by d_2, is 6 and the distance between words in a different group, denoted by d_1, is 2. Thus, $d_f = \min(d_2, 3d_1) = 6$. DC freeness is achieved because of an equal number of 0's and 1's in each block of 6 output bits. The values of l and c can be easily justified from the above construction.

Now we want to give the encoding and decoding schemes. For each transition from one state to another, we need two bits to decide the destination state and one bit to specify which word to use in each group. We therefore have a $\frac{3}{6}$ code. Soft-decision decoding can be done by using hard-quantized Viterbi decoder, which computes the branch metrics by comparing 8 adversaries per state. Figure 1 and Figure 2 are the bit error probability and symbol error probability of the code when applied to an additive white Gaussian noise (AWGN) channel.

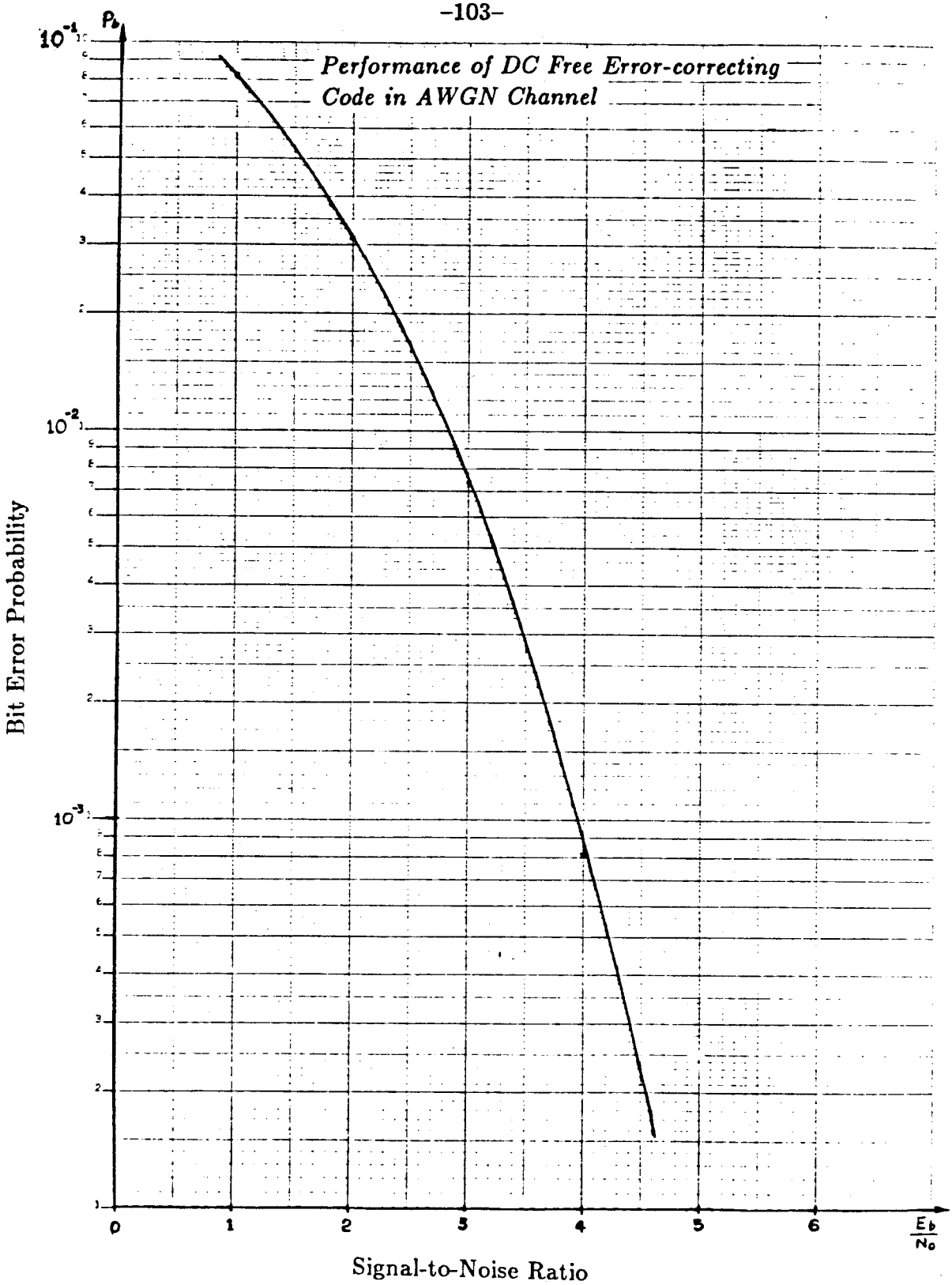

Performance of DC Free Error-correcting Code in AWGN Channel
P_b
10^{-1}
10^{-2}
10^{-3}
Bit Error Probability
0
1
2
3
4
5
6
$\frac{E_b}{N_0}$
Signal-to-Noise Ratio

Figure 1

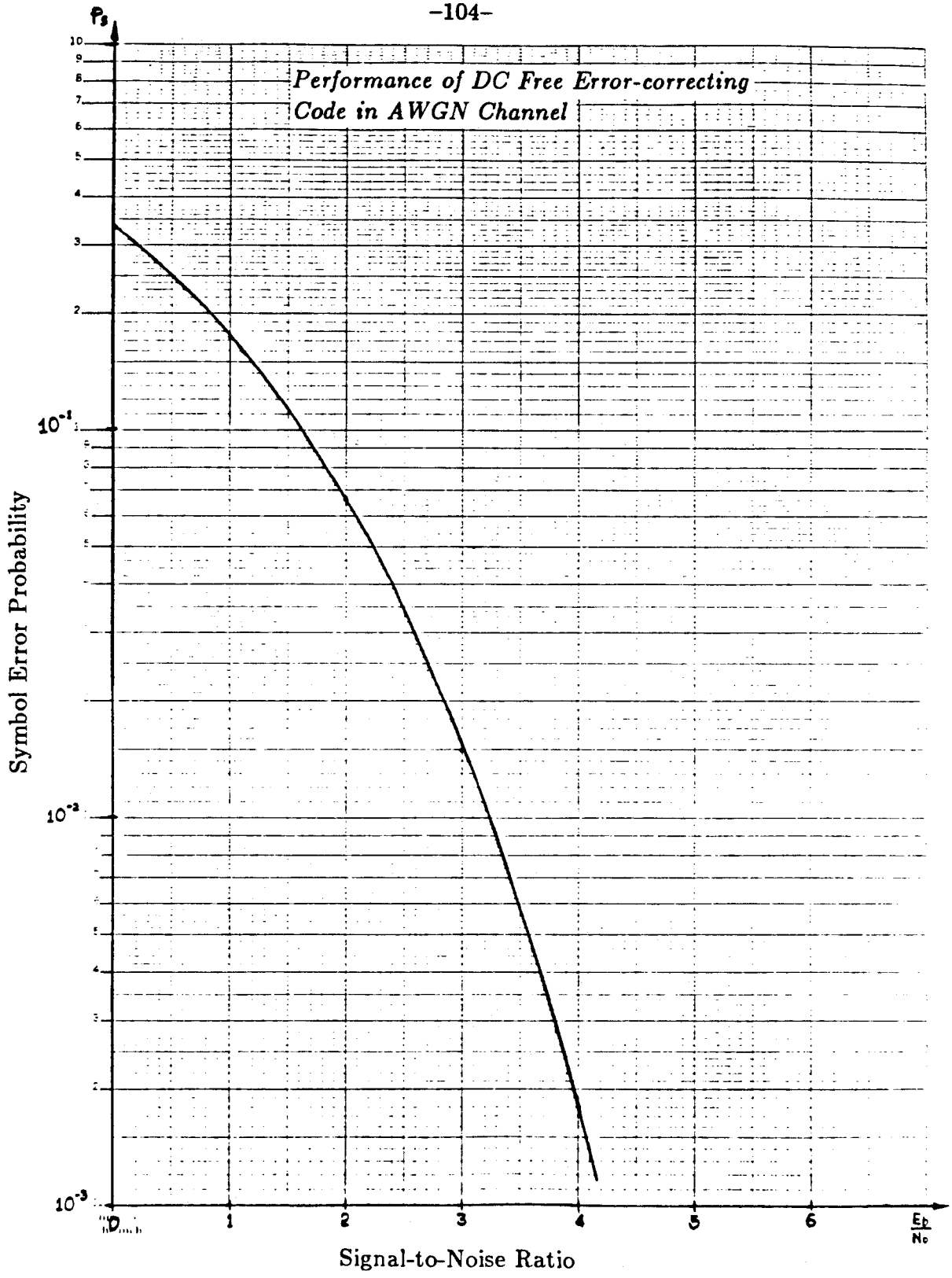

Performance of DC Free Error-correcting Code in AWGN Channel
Symbol Error Probability
Signal-to-Noise Ratio
P_s
10^{-1}
10^{-2}
10^{-3}
0
1
2
3
4
5
6
$\frac{E_b}{N_o}$

Figure 2

3. Error-correcting RLL Trellis Code

We shall now present a $(2, 2, \frac{4}{16}, 3, 18, 4)$ runlength limited trellis code as follows:

Consider all binary 16-tuples with weight 2 such that there are at least 3 0's between the two 1's, and there are at least 3 zeros at the end of the word. There are 45 words altogether. Choose 32 of them and put them into 8 groups as shown below, and label each group by a number from 0 to 7.

$$\left\{\begin{matrix} 1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0 \end{matrix}\right\} \quad \mathbf{0}$$

$$\left\{\begin{matrix} 1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0 \end{matrix}\right\} \quad \mathbf{1}$$

$$\left\{\begin{matrix} 1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0 \end{matrix}\right\} \quad \mathbf{2}$$

$$\left\{\begin{matrix} 1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0 \end{matrix}\right\} \quad \mathbf{3}$$

$$\left\{\begin{matrix} 1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0 \end{matrix}\right\} \quad \mathbf{4}$$

$$\left\{\begin{matrix} 1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0 \end{matrix}\right\} \quad \mathbf{5}$$

$$\left\{\begin{matrix} 0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0 \end{matrix}\right\} \quad \mathbf{6}$$

$$\left\{\begin{matrix} 0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0 \end{matrix}\right\} \quad \mathbf{7}$$

Now consider the state diagram generated by the shift register with the transfer function matrix

$$\begin{pmatrix} 1 & D & 1 \\ D & 1 & 0 \end{pmatrix}.$$

We have a completely connected graph. There are 4 states in the graph and there are 4 branches going into each state and 4 branches going out of each state. As in Part I map group i, $0 \leq i \leq 7$, to the branches in the state diagram labelled b_0, b_1, b_2 (output bits of the shift register) such that $i = b_0 + 2b_1 + 4b_2$.

According to the theory of the trellis code[6][7][9], we now have a noncatastrophic trellis diagram for which there are at least 2 branch differences for any two paths to diverge from one state and remerge later. In this case, d_2 is 4 and d_1 is 2. Thus, $d_f = \min(d_2, 2d_1) = 4$. This code has a density ratio of DR=$\frac{b}{n}(d+1) = 1$.

In the encoding and decoding, we need 2 bits to decide the destination state and 2 bits to specify which word to use in each group. We thus have a $\frac{4}{16}$ code. This code can be decoded using soft-decision viterbi decoding as in the case of the DC free error-correcting trellis code.

4. Conclusion and Generalization

Although we give only two specific code constructions in the above discussion, the main purpose of this chapter is to illustrate the idea of imposing DC free and runlength limited properties in a trellis code structure to obtain an error-correcting code with the desired properties (DC freeness and RLL) for magnetic recording. This paper is just a preliminary stage of this aspect of research. This idea can be easily modified, extended, and generalized to obtain other DC free error-correcting codes and runlength limited error-correcting codes.

APPENDIX A: A PROGRAM LISTING

```
Program MDS1;
var
N,K,Q,T,D,N1,U,J1,J2,J3,J4,J5,J6,J,OPTION,ALPHA,M :integer;
TOTAL,LGNU,LGDN,LGX,A :real;
function lchose(n,r :integer):real;
var
i,p :integer;
lchose1 :real;
begin
if r=0 then lchose1 := 0.0
else
begin
lchose1 := 0.0;
p := r-1;
for i:=0 to p do
lchose1 := lchose1+ln(n-i)-ln(r-i);
end;
lchose := lchose1;
end { lchose };
function power(q,l:integer):real;
var
i :integer;
x :real;
begin
x :=1.0;
if l<>0 then
for i := 1 to l do
x:=x*q;
power := x;
end;
procedure proced1(var u,q,k,n,j,t :integer; var a,lgx :real);
var
lnj,lx,lx1,sum :real;
i :integer;
begin
sum := 0.0;
lnj := (k+u-n-j)*ln(q);
for i:=0 to t do
begin
lx := lnj+lchose(n,i)+i*ln(q-1)+lgx;
if lx<=-85.0 then lx1 := 1.1E-38
else lx1 := exp(lx);
sum := sum+lx1;
end;
a := sum;
end { proced1 };
procedure proced2 (var u,q,n,j,t,d :integer; var a,lgx :real);
const
lcor1: Real = 0.0;
lcor2: Real = 100.0;
var
x,k1,w,k2,i,s,m,p :integer;
```

```
sum,sum2,sum3,lx,lx1,ly,ly1,lz,lz1,lw,lw1,lpw,lgx1 :real;
begin
if u <= 28 then lgx1:=lcor1
else lgx1 := lgx+lcor2;
sum := 0.0;
x := u-j;
k1 := d-x;
for w:=k1 to t do
begin
sum2 := 0.0;
k2 := w-k1;
for i:=0 tc k2 do
begin
sum3 := 0.0;
for s:=w to t do
begin
lx := lchose(u-j,s-w)+(s-w)*ln(q-1)+lgx1;
if lx>=-85.0 then
begin
lx1 := exp(lx);
sum3 := sum3+lx1;
end;
end;
if odd(i) then m := -1
else m :=1;
if (k2-i+1) >= 5 then lpw := (k2-i+1)*ln(q)
else lpw := ln(power(q,k2-i+1)-1);
ly := lchose(w,i)+lpw+ln(sum3);
if ly >= -85.0 then
begin
ly1 := exp(ly);
sum2 := sum2+ly1*m;
end;
end;
lz := lchose(n-x,w)+ln(sum2);
if u <= 28 then lz := lz+lgx-lcor1
else lz := lz-lcor2;
if lz >= -85.0 then
begin
lz1 := exp(lz);
sum := sum+lz1;
end;
end;
for p:=0 to t do
begin
lw := lchose(x,p)+p*ln(q-1)+lgx;
if lw >= -85.0 then
begin
lw1 := exp(lw);
sum := sum+lw1;
end;
end;
a := sum;
end;
```

```
procedure proced3(var u,j,q :integer; var a,lgx :real);
var
i :integer;
sum,lx,lx1 :real;
begin
sum := 0.0;
for i:=0 to t do
begin
lx := lchose(u-j,i)+i*ln(q-1)+lgx;
if lx <= -85.0 then lx1 := 1.1E-38
else lx1 := exp(lx);
sum := sum+lx1;
end;
a:=sum;
end;
procedure proced4(var u,j,q :integer;var a,lgx :real);
var
lx,lx1,sum :real;
begin
lx := (u-j)*ln(q)+lgx;
if lx >= -85.0 then lx1 := exp(lx);
a := lx1;
end;
begin
writeln ('what is n?');
readln (n);
writeln ('what is k?');
readln (k);
writeln ('what is q?');
readln (q);
writeln ('what is t?');
readln (t);
writeln ('want Du or Pe ?  Du = 0    Pe = 1');
readln (option);
writeln;
writeln;
writeln;
writeln (lst);
writeln (lst);
writeln (lst);
writeln ('( ',n:3,',',k:3,' ) code.  q = ',q:3,'  t = ',t:3);
writeln ('-----------------------------------');
writeln (lst,'( ',n:3,',',k:3,' ) code.  q = ',q:3,'  t = ',t:3);
writeln (lst,'-----------------------------------');
writeln;
writeln;
writeln (lst);
writeln (lst);
d := n-k+1;
n1 := d-t;
for u:=n1 to n do
begin
a := 0.0;
lgdn := lchose(n,u)+u*ln(q-1);
```

```
j1 := k+u-n-1;
j2 := j1+1;
j3 := j2+t-1;
j4 := j3+1;
j5 := u-t-1;
j6 := j5+1;
total := 0.0;
for j:=0 to u do
begin
lgnu := lchose(u,j)+lchose(n,u);
if option=0 then lgx := lgnu
else lgx := lgnu-lgdn;
if (j>=0) and (j<=j1) then proced1(u,q,k,n,j,t,a,lgx);
if (j>=j2) and (j<=j3) then proced2(u,q,n,j,t,d,a,lgx);
if (j>=j4) and (j<=j5) then proced3(u,j,q,a,lgx);
if (j>=j6) and (j<=u) then proced4(u,j,q,a,lgx);
if odd(j) then alpha := -1
else alpha :=1;
total := total+a*alpha;
end;
if option=1 then
begin
writeln ('Pe(',u:4,')=',total);
writeln (lst,'Pe(',u:4,')=',total);
end
else
begin
writeln ('D(',u:4,')=',total);
writeln (lst,'D(',u:4,')=',total);
end;
end;
writeln;
writeln;
writeln (lst);
writeln (lst);
end.
```

APPENDIX B: PROOF OF THEOREM 5.1

Let λ be a primitive element of $GF(2^8)$ such that $\lambda^{17} = \alpha$.

a) Let e_{11} and e_{22} be the two 2-consecutive-bit error. Let x and y denote the location of e_{11} and e_{22}, respectively. The syndrome expressions are

$$(A1) \qquad S_0 = e_{11} + e_{22}$$

$$(A2) \qquad S_1 = \alpha^x e_{11} + \alpha^y e_{22} \quad x \neq y \ .$$

The syndrome expressions for a single-byte error are

$$(A3) \qquad S_0 = e$$

$$(A4) \qquad S_1 = \alpha^z e,$$

where e is the byte error and z is the location of this error.

A miscorrection of two 2-consecutive-bit errors implies that $e = e_{11} + e_{22}$. We therefore get the following equation.

$$(A5) \qquad \alpha^z(e_{11} + e_{22}) = \alpha^x e_{11} + \alpha^y e_{22}$$

(A5) can be rewritten as

$$(A6). \qquad [\alpha^z + \alpha^y]^{-1}[\alpha^z + \alpha^x]e_{11} = e_{22}.$$

Now since $\alpha = \lambda^{17}$, and by using the closure property of the subfield elements we have

$$(A7) \qquad \lambda^{17m} e_{11} = e_{22} \quad \text{for some } m \neq 0 \ .$$

However, e_{11} and e_{22} are 2-consecutive-bit errors. When they are expressed as powers of λ, their power indices would not differ from the other by more than 6. So Equation (A7) gives a contradiction. ■

b) Let e_1 be a 1-bit error and e_{11} be a 2-consecutive-bit error. e_1 and e_{11} can be expressed as a power of λ. Let e_1 be denoted by λ^k, where $0 \leq k \leq 7$, and let e_{11} be denoted by λ^l. As in the proof of (a), if a miscorrection occurs, then e_1 and e_{11} are related by the following equation :

$$\lambda^{17m} e_1 = e_{11} \quad \text{for some } m \neq 0 \,. \tag{A8}$$

If the primitive polynomial is chosen such that the element $1 + \lambda$ (11000000) equals λ^z, then l can be expressed as $z + i$, where $0 \leq i \leq 6$. Substituting the expressions of e_1 and e_{11} into (A8), we get

$$\lambda^{17m+k} = \lambda^{z+i}. \tag{A9}$$

That is,

$$z = 17m - i + k, \tag{A10}$$

where $m \neq 0$, $0 \leq i \leq 6$, and $0 \leq k \leq 7$. It is easy to see that if $z \in \Delta$, then Equation (A10) gives a contradiction. ■

REFERENCES FOR PART TWO

[1] E. R. Berlekamp and Po Tong, "Interleaving", unpublished.

[2] A.M. Patel and S.J. Hong, "Optimal Rectangular Code for High Density Magnetic Tapes," IBM J. Res. Develop. **18**, No.6 , 384-389 (January 1980).

[3] Arvind M. Patel, "Error Recovery Scheme for the IBM 3850 Mass Storage System," IBM J. Res. Develop. **24**, No.1, 32-32 (January 1980).

[4] Arvind M. Patel, "Adaptive Cross-parity (AXP) Code for a High-density Magnetic Tape Subsystem," IBM J. Res. Develop. **29**, No.6, 546-562 (November 1985).

[5] Kar-Ming Cheung and Robert J. McEliece,"More on Decoder Error Probability of MDS code," to appear.

[6] F. Pollara, "Construction of Trellis Code from Block Codes," JPL IOM 331-86.2-152.

[7] F. Pollara, "Cosets of R-S Codes," JPL IOM 331-86.2-211.

[8] Massey and Sain,"Inverse of Linear Sequential Circuit," *IEEE Trans. Comput.*, C-17

[9] K. Cheung, "A New Labelling Procedure for Trellis Codes," JPL IOM 331-87.2-232.