CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r. 82/08)

Canada

NON-BOUNDED-DISTANCE DECODING WITH ONE-STEP MAJORITY LOGIC

by

Zaki A.H. Muscati

Thesis submitted to the School of Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the Ph.D. degree in Electrical Engineering

Ottawa, Ontario 1981

ACKNOWLEDGEMENT

The author is deeply indebted to the invaluable advice and continuous encouragement received from his research supervisor Professor S.G.S. Shiva, without whom the work reported in this thesis , would not have been possible. The author is also grateful to the University of Ottawa and the National Science and Engineering Research Council of Canada, for their financial assistance while carrying out this work.

ABSTRACT

In this thesis we study the capability of non-bounded distance decoding with 1-step majority-logic using feedback error-correction. We give two decoding algorithms to correct certain errors of weight $t+1$, one for even minimum distance $d = 2t+2$ and another one for odd $d = 2t+1$. Both algorithms undergo up to two decoding rounds of majority-logic estimation. We analyze the conditions under which a $(t+1)$-error is correctable by either algorithm and then derive lower bounds for the number Q_{t+1} of such correctable errors. With respect to the two classes of difference-set codes and M-sequence codes, we use certain properties of the orthogonal check-sums in order to estimate the exact values of Q_{t+1} for the cases of $t = 2$, 3 and 4. On the basis of such properties we also show that by undergoing up to two rounds of estimation, we can correct every error of weight $t+1$ which is uniquely decodable in the sense of not forming part of a codeword of weight $d = 2t+2$. We then generalize our decoding algorithms to correct certain errors of weight $t+s$, $s \leq t-1$, by undergoing $s+1$ or fewer rounds. Finally we demonstrate the improvement achieved by the correction of uniquely-decodable errors of weight $t+1$, after analyzing the coding performance with respect to the trade-offs among error rate, transmitted power and bandwidth.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES, FLOWCHARTS AND TABLES

INTRODUCTION

Error-Control Coding

Error-control coding is finding an increasing number of applications in the transmission and storage of digital information [1-8] . This can be mainly attributed to one or more of the following developments.

(a) Rapid advances in large-scale integrated circuits and microprocessors which made possible the economical realization of complex coding and decoding techniques.

(b) An increasing emphasis on high speed digital transmission with low error rates in modern communication systems.

(c) A growing interest in very large solid state memory systems which became easy to fabricate but would require error correction to maintain low failure rates in a total memory system.

(d) Development of highly efficient schemes for the digital encoding of analog signals such as voice, video, graphics and facsimile, requiring very low error rates to maintain good signal fidelity.

The simplest error-control technique is that of error detection and retransmission, commonly known as Automatic Repeat on Request (ARQ) [5,6,9]. This technique, however, is only possible with a communication system having a feedback path, or with a duplicated memory system where the same information can be retrieved from an alternative memory, in case an error is detected. Moreover, ARQ is not readily applicable to the transmission of digital information pertaining to continuous analog signals such as voice and video signals, where the retransmission delay may not be tolerable. ARQ is also not generally suitable for transmission systems with excessive delay similar to that experienced in satellite communications. Even in the presence of a feedback path and absence of any significant transmission delay, the usefulness of ARQ will usually be limited to cases of sufficiently low error rates that will not require excessive retransmission[10].

Forward Error Correction (FEC), on the other hand, provides a more suitable alternative for error-control coding in applications where ARQ is not suitable. FEC is also attractive when used in conjunction with ARQ to reduce the frequency of retransmission to acceptably low levels [5,10] .

The techniques for error detection using parity checks in ARQ systems are simple and well-known. In this thesis we

are only concerned with FEC coding techniques, and more particularly with those using 1-step majority-logic codes.

Some examples of practical FEC applications are as in the following:

i) Power-limited satellite communication systems [11,12].

ii) Interference-prone mobile radio systems [13].

iii) Videotex and broadcast Teletext systems [14,15].

iv) Computer communications [16].

v) Memory devices and microprocessor-based systems [3,17-19].

In the case of memory systems, FEC performance will simply depend on how many errors can be corrected in a memory unit of a given size. In the case of communication systems, on the other hand, the analysis of FEC prformance is somewhat more involved and requires a knowledge of the channel characteristics, as we shall see from the following discussion.

Coding Trade-offs in Binary Channels

In most existing systems, digital information is stored and transmitted in the form of binary signals. For this reason we shall confine our discussion throughout this thesis to the case of binary systems and use the Hamming metric in the decoding procedure.

When cosidering the use of a given FEC scheme in a given binary channel, we need to take into account the error characteristics of that channel. In the case of memoryless channels, such as those with Additive White Gaussian Noise (AWGN), random errors are produced which occur independantly of each other. For the case of channels with memory, on the other hand, errors may occur in bursts or clusters. Different FEC schemes are available for dealing with either type of channels [20 – 22] .

The work of this thesis is confined to the question of correcting random errors. When studying FEC peformance in the case of random errors, the idealized model of Binary Symmetric Channel (BSC) has

proven to render a relatively simple analysis when dealing with random errors. The BSC model combines the effects of a modulator at the transmitting end, channel noise, and a demodulator at the receiving end with the assumption of a hard decision detection. When a string of binary data is fed at the BSC input, a corresponding binary output will be produced with each bit having an error probability $p < 1/2$. The word symmetric implies that p is the same for "zero" bits and "one" bits.

The application of FEC coding with a BSC is depicted in Figure 1 . As shown, the encoder partitions the sequence of information bits into k-bit segments. It then operates on the information bits within a segment according to the particular code to be employed, producing an n-bit codeword containing $n-k$ parity (redundant) bits. The purpose of parity bits is to structure the encoded data in such a way that it will be possible for the FEC decoder to remove the effect of certain types of errors which are more likely to occur than others. Thus the function of an FEC decoder is to provide the best estimate of the original information blocks of k bits, such that the probability p_o of an erroneously decoded bit will be made as low as possible. The decoder is said to be optimum if it selects among the set of possible code sequences, the one that is nearest possible to the received sequence. This process is called maximum likelihood decoding.

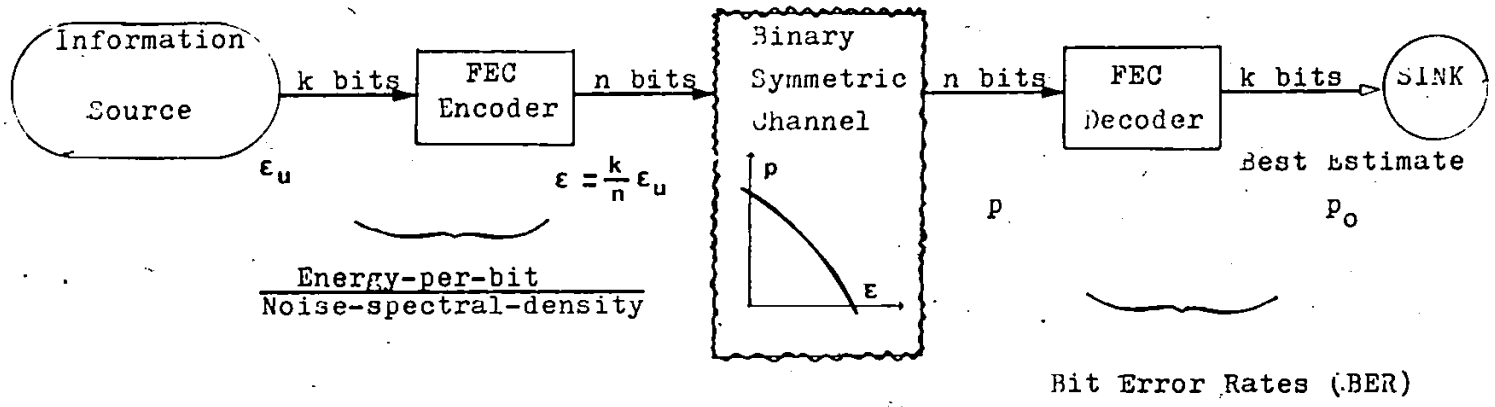


Figure 1 : Forward Error Correction with a Binary Symmetric Channel.

For the BSC model considered here, the extent by which code sequences differ form each other and from the received sequence, is measured by Hamming distance which is defined as the number of dissimilar bit positions. Associated with this is the definition of Hamming weight, which is the number of "one" bits in a given sequence. On this basis the Hamming distance between the transmitted sequence and the received sequence equals the Hamming weight of the error.

Figure 1 shows that the encoder reduces the energy-per-bit available for transmission by a factor of k/n, if we assume that the total transmitted power and the rate of information flow will remain the same with and without coding. This means that we get with coding a higher Bit Error Rate (BER) in the channel than that without coding. Therefore, in order that FEC coding will produce a net reduction in BER at the decoder output, the effect of decoding must outweigh the increase in channel BER due to the redundancy added by the encoder. The net advantage of coding [1,23,24] may then be based on either one of the following two definitions.

(a) The Net Coding Gain (NCG) defined as the relative reduction in the transmission power requirement for a specified BER.

(b) The Net Error Improvement (NEI) defined as the reduction in BER at the decoder output for a given transmitted power.

The price being paid in both situations is an expansion of the transmission bandwidth by a factor of n/k . Figure 2 illustrates a graphical method of quantifying these two measures of coding performance.

[NOTE: The trade-off between transmitted power and bit error rate as discussed above is only valid for the case where all the transmitted information is coded using one particular error-correcting code. On the other hand, when only a small portion of the data is coded, the penalty in terms of bandwidth expansion and the adverse effect of reduced energy-per-bit may be insignificant. In such a case the advantage of coding will be more pronounced than that depicted in Figure 2 .]

In view of the above-mentioned trade-offs, the coding problem is one of finding FEC codes which give a large Hamming distance between the code sequences and have as high a coding rate k/n as possible. At the same time, encoding and decoding schemes must be economically implementable.

Various classes of FEC codes are known today which aim at solving the coding problem mentioned above. These classes comprise two fundamentally different types of codes; namely block codes and convolutional codes. In the remainder of this section we shall briefly discuss and compare these two types of codes.

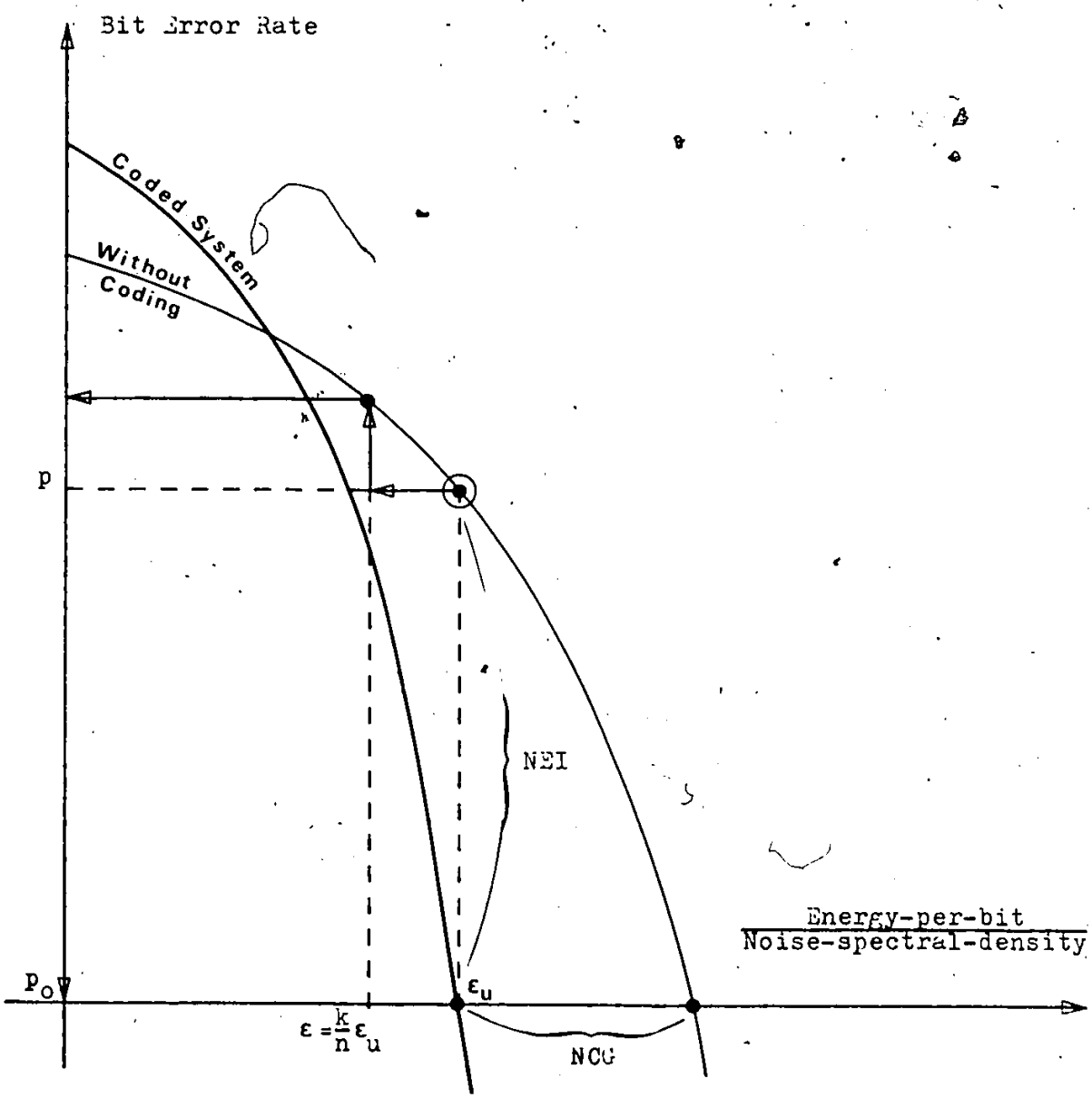


Figure 2 : Graphical Illustration of Coding Performance

Binary Block Codes

An encoder for a binary block code operates on individual blocks of k information bits separately, and converts them into corresponding n-bit codewords according to the code implemented. Here k is referred to as the dimension of the code, and n as its length. The resulting codeword is transmitted through the channel and will be received by the decoder after a possible corruption by noise. The decoder will then examine each corresponding block of n received bits (received word) separately in order to give its best estimate of the original k information bits.

The error-correcting capability of a given code is largely determined by the minimum pair-wise Hamming distance d between any two codewords. In theory, a knowledge by the decoder of all the 2^k possible codewords enables it to compare the received word with all such codewords. in the BSC case, the best decoding decision is to select the nearest codeword having the smallest Hamming distance from the received word. On this basis, the decoder guarantees a correction of any error of weight $t = \lfloor \frac{d-1}{2} \rfloor$ or less, where $\lfloor\ \rfloor$ denotes the integer portion of the enclosed expression. It may also be possible to correct some -but not all- errors of weight greater than t. This capability is referred to as non-bounded-distance decoding, which is the main topic of this thesis. We shall review further the subject of block codes in chapter 1 .

Binary Convolutional Codes

A convolutional encoder operates not only on the block of k information bits currently present at its input, but will also use $K-1$ previous k-bit information blocks, in generating a code sequence of n bits. Thus each k-bit information block will affect a code sequence of length $n_e=nK$. The quantity n_e is called the encoding constraint length. A code sequence can basically be represented by an infinite code tree, the nodes of which correspond to time intervals of successive k-bit information blocks. There are 2^k branches coming from each node to denote all possible n-bit outputs for the particular position in the code tree.

Whereas the encoded sequences represented by a code tree can be of indefinite length, the decoder in practice can only examine a received sequence up to a given length at any particular time interval. This length is called the decoding constraint length n_d which can be several times greater than n_e. On this basis, the error-correcting capability of a convolutional decoder is limited by the minimum Hamming distance between two code sequences of length n_d that differ in their intial information bit. Nevertheless, the concept of free minimum distance taken over an arbitrarily long encoded sequence, is used as an indication of the ultimate error-correcting capability of a convolutional code, when making n_d arbitrarily large.

The fact that a single k-bit information block affects only K encoded n-bit blocks, makes it possible to represent the encoded sequence by means of a repetitive trellis. Viterbi [21,25] exploited this property in defining a maximum-likelihood decoding algorithm, which compares a received sequence of finite length Kn , with the 2^{Kk} possible branches of the code tree. The algorithm will then select as "survivors" the $2^{(K-1)k}$ branches nearest to the received sequence , according to a given metric. In the BSC case, the metric is based on the Hamming distance. All branches other than the survivors will be discarded from the decoder memory. The decoder will then proceed in this manner for every forthcoming block of n received bits until a point is reached where all the $2^{(K-1)k}$ survivors share the same first n bits. At that point the decoder will take these n bits as its estimate of the first block of n transmitted bits.

Viterbi algorithm provides a practical means for optimal decoding of codes with relatively small encoding constraint length. For codes of larger constraint length, sequential decoding can be more practical where a decoder complexity is relatively insensitive to the encoding constraint length. A sequential decoder traces only one particular path through the code tree at any particular instant, and monitors the distance from the received sequence. When such a distance exceeds a given threshold, the decoder will abandon the path and start pursuing another one.

Viterbi algorithm and sequential decoding are similar in using the method of comparing the received sequence with alternative code sequences, without making use of the algebraic properties of the code. In contrast to this, there are some known algebraic decoding methods such as threshold decoding, which are simpler to implement than Viterbi algorithm and sequential decoding. Algebraic decoding, however, is only applicable to algebraically constructed codes which are usually not as powerful as non-algebraic codes of similar code rate k/n and comparable constraint length n_e .

Comparison of Block and Convolutional Codes

The encoding process in linear codes - block and convolutional alike - is a relatively minor task, that can be performed by a generator matrix G or even more simply by means of shift register circuitry corresponding to the generator polynomial(s). The main cause of implementation complexity is therefore due to the decoding procedure.

In comparison, convolutional codes are generally known to provide somewhat better performance than block codes of similar coding rate and decoding complexity. However, many of the advantages of convolutional codes leading to a wide range of commercial applications, are derived from the possibility of economically implementing non-bounded-distance decoding using Viterbi algorithm.

It is therefore fair to suggest that an improvement in the decoding capability of block codes beyond the minimum distance bounds, is likely to improve the performance of block codes to a point where they become more competitive with convolutional codes.

On the other hand, convolutional codes suffer from the following problems which may not apply to block codes.

(a) In the case of convolutional codes, error propagation phenomenon can arise as a result of the feedback mechanism inherent in Viterbi algorithm and sequential decoding. This phenomenon is also inherent in majority-logic decoding when not definitive. However, in the case of block codes, the use of non-definitive majority-logic decoding can also produce error propagation but this will be limited to only one block length.

(b) Whenever a receiver of a convolutionally coded data stream loses synchronism with the transmitter, the problem of synchronization recovery can arise.

Both problems mentioned above can be somehow solved by a periodic termination of the convolutional code. This requires the insertion of a sufficient number of "zero" bits at the end of an information block such that all possible branches of the code tree converge into one node. Once the decoder reaches this node, it can start afresh with a cleared memory. The penalty paid by such a solution is obviously a decrease in coding efficiency due to the additional terminating bits.

Furthermore, one has to be careful about the choice of the generator polynomial for convolutional codes, so that this polynomial does not lead to catastrophic errors [21] .

Block codes are generally more suitable in FEC applications where the information is produced and transmitted or stored in separate data blocks or packets. This is especially so with the case of memory systems or in communication systems in which individual packets are transmitted independantly through different alternative paths. Block coding is particularly more convenient in applications where different portions of a long data block require different degrees of error protection, depending on the nature of carried information. For example raw data may require less error protection than control instructions or addressing information. For such applications, more than one type of FEC block coding will be desirable to utilize within the same data block.

Non-Bounded-Distance Decoding

We indicated earlier that the performance of FEC block codes is subject to further improvement if economically implementable methods of non-bounded-distance decoding can be found. This concept has

been the prime motivation behind the research work presented in this thesis.

In the literature, various decoding algorithms are given for the correction of errors of weight $\tau > t$ with different types of codes such as BCH codes [26] and iterative (product) codes [27]. In general, such algorithms are considerably more complex than distance-bounded algorithms for the same type of codes.

In this thesis, we have concentrated our work on 1-step majority-logic codes as such codes are among the simplest to implement. We shall see that the additional decoding complexity required for correcting errors of weight $\tau > t$ is not significant.

Thesis Outline

First we review in chapter 1 the general topic of random-error correction with block codes. We discuss how a standard array can be used as a "complete" decoding table and how this table can be simplified when using the method of syndrome decoding for the case of linear codes. We then discuss how a decoder implementation can be made even simpler with the class of binary cyclic codes and especially for the case of 1-step majority-logic codes.

We then begin chapter 2 by defining a decoding strategy for minimizing the bit error rate at the decoder output when dealing with errors of weight $t+1$. In view of this strategy we give in Theorem 1 expressions for decoder error performance, and then

discuss the use of two-round decoding for the correction of errors of weight $t+1$ for the case of cyclic codes that are 1-step majority-logic decodable. We see that the two cases of codes with even and odd minimum distance d require somewhat different treatment. Algorithm D1 covers the case of even d, and Algorithm D2 covers the case of odd d, both requiring up to two decoding rounds. Theorems 2 to 4 prove the validity of Algorithm D1 and specify the conditions under which errors of weight $t+1$ are correctable. On the other hand, Theorems 9 to 11 deal with Algorithm D2 in the same way. Lower bounds on the number Q_{t+1} of such correctable errors are given in Theorems 5 and 6 for the case of D1, and in Theorem 12 for the case of D2.

With respect to the two well-known classes of difference-set codes and M-sequence codes, we examine the performance of Algorithm D1 in more details. To do this we define and develop Properties 1 to 5 with respect to the sets of orthogonal check-sums of these codes. We then use such properties to develop further results on the number Q_{t+1} of errors of weight $t+1$ that are correctable by Algorithm D1. In Theorem 7 we indicate the effect of a non-correctable error of weight $t+1$ on the orthogonal check-sums of a difference-set code or an

M-sequence code. Using tnis result we then estimate the exact value of Q_{t+1} for the cases of t=2,3 and 4 , as obtained in Examples 2,3 and 4 . In Theorem 8 we show that Algorithm D1 will in fact correct all uniquely-decodable errors of weight t+1 when used with a difference-set or an M-sequence code.

In Table I we list the actual values of Q_{t+1} and/or its lower bound for some codes of relatively short length n .

In section 2.6 we generalize the concept of non-bounded-distance decoding towards the correction of errors of weight $\tau > t+1$ and thence present Decoding Algorithms D3 and D4 for the cases of even and odd d respectively. Theorem 13 proves the validity of Algorithms D3 and D4 and indicate the conditions for an error to be correctable by these algorithms, while Theorem 14 gives the highest possible weight of a correctable error.

In Decoding Detail D5 , we indicate how many rounds of estimation are required for correcting an error of a given weight. The implementation of Algorithms D1 to D4 is illustrated by way of Flowcharts 1 to 4 respectively where some techniques for shortening the decoding delay are exhibited.

In chapter 3 we study the effect of correcting errors of weight $t+1$ by Algorithms D1 and D2, on the trade-offs among the transmitted power, bandwidth utilization and bit-error rate in random communication channels. On the basis of this and the decoding strategy given in chapter 2, we derive in section 3.3 explicit expressions for the Net Coding Gain (NCG) and Net Error Improvement (NEI) of binary block codes used over binary symmetric channels. We apply these expressions in the case of communication channels having one or none of different carrier modulation schemes, and thus obtain an upper bound for NCG. In section 3.4 we present some illustrative graphical results on the performance of those codes that were studied in chapter 2.

In concluding this thesis we note that for the 1-step majority-logic codes we examined, the decoding error rate of Algorithm D1 tends to become increasingly more dominated by the occurrence of errors of weight $t+2$ as the code length n increases. This highlights an interest in extending the analysis carried in this thesis for Algorithms D1 and D2 to also cover Algorithms D3 and D4.

In passing it may be noted that some of the results given in this thesis have been presented elsewhere [28,29].

CHAPTER 1

REVIEW OF RANDOM-ERROR CORRECTION WITH BLOCK CODES

1.1 General

A linear binary block code V is encoded by taking the product of a k-bit information vector and an $k \times n$ generator matrix G to produce an n-bit code vector. If the minimum distance of V is d, then V will be capable of correcting any error pattern of weight τ or less, and detecting the occurence of -but not necessarily correcting - any error pattern of weight $\lambda > \tau$ or less, where λ and τ are governed by

$$\lambda + \tau = d - 1 ,$$
$$0 \leq \tau \leq t , \quad t = \left\lfloor \frac{d-1}{2} \right\rfloor , \tag{1.1}$$

and $\lfloor \ \rfloor$ indicates the integer portion of the enclosed expression.

In this thesis we shall mainly deal with the combined capability of correcting all errors of weight t or less, as well as detecting and/or correcting certain errors of weight $t+1$, of greater. We note that for the case of even $d=2t+2$, every error pattern of weight $t+1$ is detectable since $\lambda = t+1$ according to (1.1). To this effect we shall

review in the remainder of this chapter the concept of standard array, and then discuss briefly the subject of bounded-distance decoding of majority-logic codes. The subject of non-bounded-distance decoding will be investigated in chapter 2 in connection with 1-step majority-logic codes.

Throughout this thesis we shall adhere to the following notations. All n-bit words will be expressed as polynomials of degree $n-1$ or less. Polynomial addition is taken modulo 2, and polynomial multiplication is taken modulo $(1+x^n)$. We denote V as an (n,k) binary linear block code of minimum distance d. The code V consists of 2^k codewords

$$v(x) = \sum_{i=0}^{n-1} v_i x^i \qquad (1.2)$$

each of which is produced by the encoder at an equal probability of 2^{-k}. A codeword $v(x)$ is then transmitted over a noisy channel which introduces a certain error

$$e(x) = \sum_{i=0}^{n-1} e_i x^i . \qquad (1.3)$$

The received word

$$r(x) = \sum_{i=0}^{n-1} r_i x^i \qquad (1.4)$$

will thus be given by

$$r(x) = e(x) + v(x) \quad . \qquad (1.5)$$

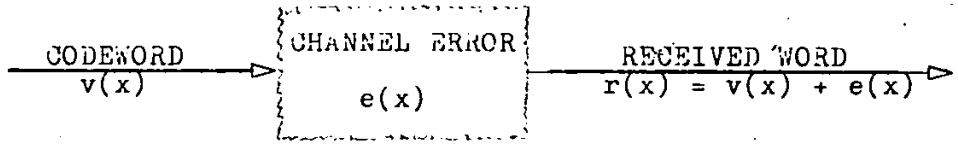
The decoder is then likely to receive any one of 2^n possible words, and will attempt to provide an estimated codeword $\hat{v}(x)$ by deciding which one of the 2^k codewords is the most likely to have been transmitted.

1.2 The Standard Array

The standard array for a given code V, can be defined as a "complete" decoding table which maps every one of the 2^n possibly received words $r(x)$ into one of the 2^k codewords $v(x)$. The concept of the standard array is useful in understanding the "complete" error-correcting capability of a given code V. However, because of the very large size (2^n elements) of the standard array table, its practical usefulness in decoding is limited to very few codes of small length n.

Figure 3 illustrates how a standard array can be constructed. While in principle a standard array can be formed for any group code, the standard array table is usually conceived only for the correction of random errors. In this context a standard array must be capable of correcting all errors of weight t or less when using a t-error-correcting code as well as some errors of weight greater than t for non-perfect codes.

We note that the properties of a linear code dictate that the all-zero word $\theta(x)$ must be a codeword, and that the sum of any two codewords $v_i(x)$ and $v_j(x)$ must also be a codeword, whereas the sum of a codeword in



2^k CODEWORDS	$\theta(x)$	$v_1(x)$	$v_2(x)$	.....	$v_N(x)$
	$e_1(x)$	$e_1(x) + v_1(x)$	$e_1(x) + v_2(x)$	.....	$e_1(x) + v_N(x)$
	$e_2(x)$	$e_2(x) + v_1(x)$	$e_2(x) + v_2(x)$	.....	$e_2(x) + v_N(x)$
CORRUPTED	.	.	.	.....	.
WORDS	.	.	.	.....	.
$2^n - 2^k$	.	.	.	.....	.
	$e_M(x)$	$e_M(x) + v_1(x)$	$e_M(x) + v_2(x)$	.....	$e_M(x) + v_N(x)$
	$M = 2^{n-k}-1$				$N = 2^k - 1$
	2^{n-k} COSET LEADERS				

Figure 3 : The Standard Array of an (n,k) Binary Block Code.

V and a word outside V cannot be a codeword in V. On the basis of such properties, the standard array can be constructed as follows:

1. The first row begins with the codeword $\theta(x)$ followed by all the other codewords $v_1(x), v_2(x), \ldots, v_N(x)$ where $N = 2^k - 1$.

2. The second row begins with a single error pattern $e_1(x)$. The remainder of the second row will then be completed by placing under each codeword $v_i(x)$, for $i = 1, 2, \ldots, N$, the word resulting from taking the sum of $e_1(x) + v_i(x)$. We shall refer to the row thus formed as a coset and to $e_1(x)$ as its leader.

3. The third row begins with another coset leader $e_2(x)$ which will also be a single error pattern and which is not present in the previous rows. This row will then be formed in a similar manner to how the second row was formed.

4. The above process is continued until all possible $\binom{n}{1}$ single error patterns appear as coset leaders.

5. If $t>1$ we choose the next coset leader as a double error pattern and form the coset in the same manner. We can thus form $\binom{n}{2}$ cosets, the leaders of which are all possible double errors.

6. We carry on selecting coset leaders of successively increasing weight until all error patterns of weight t or less appear as coset leaders.
7. For the case of non-perfect codes we continue selecting the next coset leader of smallest possible weight outside the previous rows, and form the cosets in the same fashion until all possible 2^{n-k} cosets are formed. At this point the standard array will be filled with all possible 2^n words.

The standard array may be conceptualized as a decoding table in the following way:
The location of a received word $r(x)$ within a standard array will point to the decoder estimate $\hat{v}(x)$ of the transmitted codeword as being the word at the top of the column containing $r(x)$. At the same time, the standard array will give an estimate $\hat{e}(x)$ of the error as being the leader of the coset containing $r(x)$.

The complete set of errors that are correctable by V is specified by the first column containing all the 2^{n-k} coset leaders. For a t-error-correcting code, the coset leaders will include, starting from the top, all n single-errors followed by all $\binom{n}{2}$ double-errors and so on, down to all $\binom{n}{t}$ errors of weight t.

With the exception of perfect codes, error patterns of weight greater than t will also appear as coset leaders. Among binary linear codes, the only perfect codes known to exist are the single-error-correcting Hamming codes of

$$(n = 2^{m-1} , \quad k = n-m , \quad d = 3)$$

and the (23,12) Golay code of $d=7$.

1.3 Syndrome Decoding

The size of a standard array decoding table can be significantly reduced if we take advantage of the fact that elements of the same coset have the same syndrome. This fact can be seen from the following discussion.

Linear block codes can be completely defined either by their $k \times n$ generator matrix G as already mentioned, or by an $(n-k) \times n$ parity-check matrix H, which can be directly derived from G by noting that

$$GH^T = 0$$

where H^T is the transposed form of H. The product between H^T and any codeword $v(x)$ will give an "all-zero" vector of dimention $n-k$. Since the received word is given by

$$r(x) = v(x) + e(x) \qquad (1.5)$$

we get

$$r(x).H^T = e(x).H^T \quad , \qquad (1.6)$$

where $e(x)$ is the channel error. From (1.6) we see that the product $s(x) \triangleq r(x).H^T$ is independant of

the transmitted codeword $v(x)$. The product $s(x)$ is called the error syndrome.

The above-mentioned fact is one of the key principles underlying syndrome decoding, by which we do not need to operate on all 2^n possible received n-bit words, but rather we store only 2^{n-k} possible (n-k)-bit syndromes. An estimate $\hat{e}(x)$ of the error can be obtained either by using a decoding table or by going through certain combinatorial logic. In either method, an estimate of the transmitted codeword can be obtained from

$$\hat{v}(x) = \hat{e}(x) + r(x) .$$

The implementation of a syndrome decoder can be even further simplified if a linear code possesses the cyclic property. A discussion of the decoding of binary cyclic codes follows in the sequel.

1.4 Decoding of Binary Cyclic Codes

Decoding of binary cyclic codes can be easily implemented by using shift register logic and by taking advantage of their well-understood matnematical structure. A cyclic code has the property that a cyclic shift of any one of its codewords by one bit, is also a codeword. A cyclic code can be completely defined either by its generator polynomial $g(x)$ of degree $n-k$, or by its parity-check polynomial $h(x)$ of degree k where

$$h(x) = \frac{1+x^n}{g(x)} .$$

A given (n,k) cyclic code V can be utilized either in its systematic form or in its nonsystematic form, depending on how an information polynomial $m(x)$ of degree $k-1$, is translated into a codeword $v(x)$. In the systematic form the k information bits will constitute the k highest-order bits of the corresponding codeword which is generated as

$$v(x) = p(x) + x^{n-k} m(x) ,$$

where $p(x)$ is the parity-check portion formed by

$$p(x) = x^{n-k}\, m(x) \bmod g(x) .$$

In the nonsystematic form of V, a codeword is generated as

$$v'(x) = m(x)\, g(x)$$

where no bit in $v'(x)$ can be recognized as a parity-check bit or an information bit. The systematic form of a cyclic code offers more flexibility in decoder implementation, since the decoder can operate separately on the information portion and parity-check portion of a received word $r(x)$.

One method of computing the error syndrome is to take $r(x)$ modulo $g(x)$. The decoder must then produce an estimate $\hat{e}(x)$ of the error. We thus get $\hat{v}(x) = \hat{e}(x) + r(x)$. In order to obtain an estimate $\hat{m}(x)$ of the information polynomial, a decoder for the systematic code can simply take the k highest order bits of $\hat{v}(x)$. For the nonsystematic code, on the other hand, the decoder can obtain $\hat{m}(x)$ by dividing $\hat{v}(x)$ over $g(x)$.

1.5 Majority-Logic Decoding

The parity-check matrix H of a linear block code can be used to derive a set of $n-k$ linearly independant check-sums on subsets of the n bit positions. When substituting the values of the n received bits r_i for the respective terms in these check-sums, an $(n-k)$-bit error syndrome will be created. This in fact is equivalent to taking the product $r(x).H^T$ which has been discussed in section 1.3 .

Suppose that we take all 2^{n-k} possible linear combinations of the $n-k$ linearly independant check-sums. We may then be able to find some J check-sums which are orthogonal on the first received bit r_o , such that r_o appears in every one of the J check-sums whereas every other bit position r_i for $i=1,2,\ldots,n-1$ will appear no more than once. This will enable us to obtain a correct majority-logic estimate of the first error bit e_o if no more than $t = J/2$ received bits are in error. Similarly the values of the following error bits $e_1, e_2, \ldots, e_{n-1}$ can be obtained by making use of the cyclic property of the code and taking majority-logic estimates accordingly. If the code V has a minimum distance $d=J+1$, then V is said to be a 1-step majority-logic code.

On tne basis of the above properties, 1-step majority-logic decoding can be performed by using shift-register circuitry as illustrated in Figure 4 . Here the error correction procedure can be described as follows [22] :

Step 1. The received word is first loaded into the shift register.

Step 2. The J parity check-sums orthogonal on e_o are formed by summing appropriate received bits. This is done by J multiple-input modulo-2 adders.

Step 3. The J orthogonal check-sums are fed into a majority-logic gate. The first received bit r_o is read out of the shift-register and corrected by the output of the majority gate.

Step 4. After Step 3, the shift-register has been shifted one place to the right. Now the second received bit r_1 is in the first stage of the shift register and will be corrected in exactly the same manner as the first received bit was. The decoder repeats Steps 2 and 3 .

Step 5. Similar to Step 4, every one of the remaining received bits $r_2, r_3, \ldots, r_{n-1}$ will be corrected upon successive single shifts to the right of the shift-register contents.

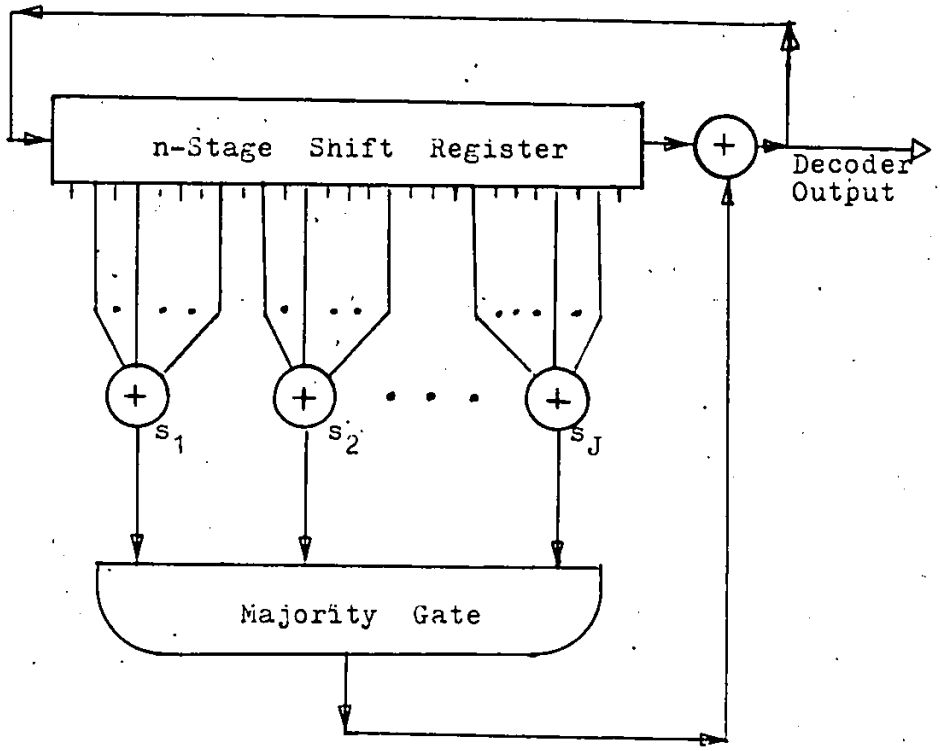


Figure 4: A One-Step Majority-Logic Decoder Using Shift-Register Circuit.

1-step majority-logic decoding is amongst the simplest and fastest known decoding algorithms. Since there are only few codes that are so decodable, there is also an interest in majority-logic decoding with more than one step. A 2-step majority-logic decoder, for instance, will estimate in the first step a set of d-1 orthogonal sums on a particular error bit, e_i. Such a set will then be used in the second decoding step to estimate e_o.

The topic of 1-step majority-logic decoding will be further studied in chapter 2. There we shall see how the inherent capability of non-bounded-distance decoding of majority-logic decoding can be exploited with the use of feedback decoding in order to correct certain errors of weight $\tau \geq t+1$.

CHAPTER 2

ONE-STEP MAJORITY-LOGIC DECODING WITH MULTIPLE ROUNDS

2.1 Introduction

Let V be an (n,k) binary cyclic code of minimum distance d which is majority-logic decodable in one step. Let $t=(d-2)/2$ for d even and $t=(d-1)/2$ for d odd. Then a 1-step majority-logic decoder can estimate each one of the transmitted bits v_i for $i=0,1,..,n-1$, by using a single majority-logic gate. Any error pattern of weight t or less can thus be corrected in one round of n estimations.

Townsend and Weldon [21,30] showed that some errors of weight $T>t$ can also be corrected by using a variable majority-logic threshold and going through as many as 2(d-1) decoding rounds. They gave some computer simulation results indicating how many errors of weights greater than t are so correctable for some quasi-cyclic codes.

In this chapter we consider two-round decoding algorithms and provide analytical results on the number of correctable errors of weight t+1 for the case of cyclic codes. Such errors are more likely to occur than those of greater weights, and may therefore dominate the decoding error performance. It will be therefore interesting to be able to correct as many such errors as possible.

We begin this chapter by discussing a decoding strategy that aims at minimizing the decoded bit error rate when dealing with errors of weight $t+1$. In contrast with the method of Townsend and Weldon, our strategy is to attempt the correction of only such errors of weight $t+1$ that are uniquely decodable in the sense of being coset leaders in a standard array with no other elements in the same coset having similar weight. We shall see that under such a strategy no more than two decoding rounds will be required. In this respect we specify two decoding algorithms D1 and D2 for the two cases of codes with even d and those with odd d, respectively. We then develop some analytical results on the number of errors of weight $t+1$ that are correctable by D1 or D2. We also show that a significant portion of $(t+1)$-errors can in fact be corrected and that such portion tends to approach unity with increasing code length n for the particular codes considered in this chapter.

The above-mentioned results are of practical interest since they indicate that a substantial improvement in coding error performance can be achieved at no additional cost in terms of coding redundancy. The additional implementation complexity is relatively small since the same 1-step majority-logic gate can be used in both decoding rounds.

Finally, in this chapter, we generalize our decoding algorithms to deal with errors of weights greater than t+1 by going through more than two rounds of estimation. We show that up to s+1 rounds will be needed for the correction of errors of weight $\tau \leq t+s$ where $s \leq t-1$.

Although we have addressed in this chapter only the case of cyclic codes, the results developed can also be applied to the quasi-cyclic codes given by Townsend and Weldon [30] and to some codes of Rahman and Blake [31] . The following is a list of some known cyclic codes to which the results developed in this chapter are directly applicable.

1. Difference-set codes [32] :
   $n = 2^{2s} + 2^s + 1$, $n - k = 3^s + 1$, $d = 2^s + 2$.
2. M-sequence (or simplex) codes [33] :
   $n = 2^m - 1$, $k = m$, $d = 2^{m-1}$.
3. BCH code (15,7) d=5 , and its even-weighted subset code (15,6) d=6 [34,35] .
4. Generalized finite-geometry codes [36] : e.g. (85,24) d=22 , (341,45) d=86 , (585,184) d=74 , and (4369,1568) d=274 .

2.2 Decoding Strategy

Let us consider the case of Forward Error Correction (FEC) used over a binary random channel which gives a bit error rate p at the decoder input. Let p_o be the bit error rate at the decoder output after an error correction is attempted. Then for a given p, the value of p_o will depend solely on the error-correction capability of the decoder. In this section we select a decoding strategy for dealing with errors of weight $t+1$.

Corresponding to a transmitted codeword of length n, let τ be the weight of an error pattern $e(x)$ produced by the random channel. An FEC decoder will make a correct estimate of $e(x)$ whenever $\tau \leq t$ where t is as previously defined. In case $\tau \geq t+1$, however, a decoding error may occur. Corresponding to an error pattern of weight τ, let R_τ be the average fraction of erroneously decoded bits over length n. This gives

$$R_\tau = 0 \text{ for } \tau \leq t$$

and

$$0 < R_\tau < 1 \text{ for } t < \tau \leq n .$$

Using the above definitions we get

$$p_o = \sum_{\tau=t+1}^{n} \binom{n}{\tau} p^{\tau}(1-p)^{n-\tau} R_{\tau} \qquad (2.1)$$

In many practical systems we have $p \ll 1/n$. In such cases we can approximate (2.1) to

$$p_o \simeq \binom{n}{t+1} p^{t+1}(1-p)^{n-t-1} R_{t+1} \qquad (2.2)$$

which indicates that the decoder error performance is mainly dominated by the occurrence of non-correctable errors of weight $t+1$.

[NOTE: Approximation (2.2) is good only when $p \ll 1/n$ and R_{t+1} is not very small. As we shall see later in this chapter, R_{t+1} can be made very small by using a non-bounded-distance decoder that corrects a majority of errors of weight $t+1$. Under those circumstances, the effect of errors of weight $t+2$ must also be taken into consideration when determining p_o .]

In the following we shall investigate how to minimize p_o in (2.2) by making R_{t+1} as low as possible.

Let us first refer to the standard array of an (n,k) linear code of minimum distance d, as the basis of our decoding procedure so that the full capability of the code will be available to us. In this context an error pattern $e(x)$ of weight $t+1$ will belong to one of three disjoint sets E_1, E_2 and E_3 which are defined below.

Set E_1 : $e(x) \in E_1$ if and only if $e(x)$ is a coset leader of minimum weight in its coset with no other elements of similar weight $t+1$ in the same coset. Such errors can be said to be uniquely decodable.

Set E_2 : $e(x) \in E_2$ if and only if $e(x)$ has a minimum weight in its coset, with $u > 0$ other elements of similar weignt $t+1$ in the same coset. Such errors can be said to be detectable only.

Set E_3 : $e(x) \in E_3$ if and only if $e(x)$ is part of a coset, the leader of which has a smaller weight than $t+1$. Such an error is neither detectable nor correctable since it leads to another error of a smaller weight as the decoding answer.

Since E_1, E_2 and E_3 are disjoint sets, we have

$$|E_1| + |E_2| + |E_3| = \binom{n}{t+1} \tag{2.3}$$

where $|.|$ indicates the size of the enclosed set.

On the basis of the above definitions, we give Lemmas 1 to 4 below.

LEMMA 1: $e(x) \in E_2$ if and only if $e(x)$ is part of a codeword $v(x)$ of weight $w = 2t+2$.

LEMMA 2: $e(x) \in E_3$ if and only if $e(x)$ is part of a codeword $v'(x)$ of weight $w' = 2t+1$.

Proof to Lemmas 1 and 2 is rather straightforward and will not be given here. Lemma 2 will directly lead to

LEMMA 3: For the case of d even, $E_3 = 0$.

We then combine Lemmas 1 and 2 with (2.3) to get

LEMMA 4: $e(x) \in E_1$ if and only if $e(x)$ forms no part of a codeword of weight $2t+1$ or $2t+2$.

Next we consider the decoder action with respect to an error pattern $e(x)$ belonging to one of the sets E_1, E_2 and E_3. When using the standard array as a decoding table, an error pattern in E_1 will always be corrected as it is considered to be the most likely error pattern to occur among those constituting the same coset. For the same reason, an error pattern in E_3 will always lead to a wrong decoding answer.

On the other hand, an error pattern in E_2 will be detectable but for the sake of error correction, it will leave the decoder without a clear choice as to which element in the respective coset is to be considered as the most likely error pattern. In order to deal with such a situation, two alternative decoding strategies may be considered, as discussed in the following.

DECODING STRATEGY A : When an error pattern $e(x) \in E_2$ is detected, we choose according to a given rule, or arbitrarily, one particular element out of the $u+1$ elements of weight $t+1$ present in the coset that contains $e(x)$.

DECODING STRATEGY B : We use an (n,k) code in its systematic form so that when an error pattern $e(x) \in E_2$ is detected, we take unaltered the k-bit information portion of the received n-bit word, as the best decoder estimate for the k information bits.

Let f be the average value of the Hamming distance between a transmitted codeword and the corresponding n-bit word estimated by the decoder. Then under Strategy A, the decoder will give a wrong answer with respect to an error pattern $e(x) \in E_2$ at a probability of $\frac{u}{u+1}$. This is because the $u+1$ elements of weight $t+1$ contained in the coset of $e(x)$, are all equally likely. From Lemma 1 we note that $2t+2$ bits will be in error at the decoder output if $e(x)$ is erroneously decoded. On the basis of all this we get

$$f_a = \frac{(2t+2)u}{u+1} \tag{2.4}$$

where f_a is the value of f under Strategy A.

On the other hand, Strategy B will leave the number $t+1$ of error bits in $e(x) \in E_2$ unchanged whenever such an error pattern occurs. This means that

$$f_b = t+1 \tag{2.5}$$

where f_b is the value of f under Strategy B.

Since decoding of error patterns from the sets E_1 and E_3 will be the same for both Strategies A and B, and since $f_b < f_a$ for $u > 0$, we conclude that Strategy B

will lead to a lower value of R_{t+1} and hence a smaller bit error rate at the decoder output.

[NOTE: Strategy A would otherwise be applicable if the intent was to minimize the probability of an erroneously decoded codeword, rather than minimizing the decoded bit error rate.]

We must mention here that although any error pattern $e(x) \in E_1$ is theoretically correctable when using the standard array, practical decoders however may not possess the full decoding power of the standard array. In view of this we shall need to modify Strategy B in order to cope with any practical limitations of a given decoder. This will lead us to the following

DECODING STRATEGY C : We use an (n,k) code in its systematic form so that when an error pattern of weight t+1 is detected and found correctable we do the correction, but if such an error cannot be corrected for whatever reason we then take unaltered the k-bit information portion of the received n-bit word.

Theorem 1 below gives expressions for R_{t+1} on the basis of Strategy C . These expressions will be used later in chapter 3 in analyzing the coding gain performance of Decoding Algorithms D1 and D2 specified in sections 2.4 and 2.5 which are based on Decoding Strategy C .

THEOREM 1 : With respect to a binary systematic (n,k) code of minimum distance d , let R_{t+1} denote the average fraction of erroneously decoded bits over length n due to an error pattern e(x) of weight t+1 . Let Q_{t+1} denote the number of error patterns e(x) that are correctable. For the case of odd $d = 2t+1$, let A_d denote the number of code-words of weight d and let $R_{t+1} \triangleq R'$ and $Q_{t+1} \triangleq Q'$. For the case of even $d = 2t+2$, let $R_{t+1} \triangleq R$ and $Q_{t+1} \triangleq Q$. Then Decoding Strategy C will give

$$R' = \left[\frac{tA_d}{n}\binom{2t+1}{t} + \binom{n-1}{t} - \frac{(t+1)Q'}{n}\right] \Big/ \binom{n}{t+1} \qquad \text{(T1a)}$$

and

$$R = \frac{t+1}{n}\left[1 - \frac{Q}{\binom{n}{t+1}}\right] . \qquad \text{(T1b)}$$

Proof:

(a) For odd d , The set E_3 defined earlier, consists of all the error patterns e(x) of weight t+1 that cannot be recognized by the decoder as having weight t+1 .

On the basis of Lemma 2 , the size $|E_3|$ of the set E_3 is equal to the number of error patterns $e(x)$, each of which forms part of a codeword $v(x)$ of weight $d=2t+1$. For every $v(x)$ there are $\binom{d}{t+1}$ error patterns $e(x) \in E_3$. We also know that no $e(x) \in E_3$ can be part of two different codewords $v_1(x)$ and $v_2(x)$ of weight $2t+1$, since otherwise we would have a third codeword of weight $2t < d$ given by $v_3(x) = v_1(x) + v_2(x)$, which is impossible. Therefore we get

$$|E_3| = A_d \binom{d}{t+1} = A_d \binom{2t+1}{t} . \qquad (2.6)$$

As a consequence of Lemma 2 , an error pattern $e(x) \in E_3$ will cause the decoder to erroneously indicate another error pattern of weight t and hence produce a total of $2t+1$ error bits at the decoder output. Thus the total contribution C_1 to the value of R' due to error patterns in E_3 is given by

$$C_1 = A_d \binom{2t+1}{t} \frac{2t+1}{n} \Big/ \binom{n}{t+1} . \qquad (2.7)$$

The remainder of error patterns of weight $t+1$ are those outside E_3 which can be recognized by the decoder as having a weight of $t+1$. Out of these there are $\binom{n}{t+1}-Q'-E_3$ errors each of which will cause, according to Strategy C , $t+1$ error bits at the decoder output. The total contribution C_2 of such error patterns to the value of R' will therefore be given by

$$C_2 = \frac{t+1}{n}\left[\binom{n}{t+1} - Q' - |E_3| \right] / \binom{n}{t+1} . \quad (2.8)$$

By combining (2.6), (2.7) and (2.8) we get (T1a). This proves the first part of Theorem 1 .

(b) For even d , Lemma 3 gives $|E_3| = 0$. Thus there are only $\binom{n}{t+1} - Q$ non-correctable error patterns of weight $t+1$, each producing $t+1$ error bits at the decoder output according to Strategy C . This directly gives (T1b) and thus proves the second part of Theorem 1 .

Q.E.D.

REMARK 1 : R_{t+1} has been upper-bounded in the literature by d/n for the case of odd minimum distance d [23]. This upper bound is obtained by taking the extreme case where all error patterns of weight $t+1$ belong to the set E_3 and hence each will produce d eror bits at the decoder output.

COROLLARY 1: Let V' be an (n,k) cyclic code of odd minimum distance $d' = 2t+1$ generated by $g(x)$. Let V be the expurgated subset $(n,k-1)$ code of even minimum distance $d = 2t+2$, generated by $(1+x)g(x)$. Then Decoding Strategy C gives

$$R' - R \geq A_d \frac{t}{n} \binom{2t+1}{t} \Big/ \binom{n}{t+1} \qquad (C1)$$

where R', R and A_d are as defined in Theorem 1 .

Proof: Because V is a subset code of V', every error pattern correctable by V' is also correctable by V. Therefore

$$Q \geq Q' \qquad (2.9)$$

where Q and Q' are as defined in Theorem 1 . Combining (2.9) with (T1a) and (T1b) leads to (C1), hence proving Corollary 1. Q.E.D.

REMARK 2 : Corollary 1 implies that the performance of an FEC decoder in terms of the bit error rate at its output, can be improved by taking the subset code V at the cost of losing one information bit. An alternative way of taking advantage of the even minimum distance of V, is the well-known hybrid scheme [2, 37] of using both FEC and ARQ such that error patterns of weight t or less are corrected whereas those of weight $t+1$ are only detected for the purpose of retransmission.

2.3 One-Step Majority-logic Definitions

Let $v(x) = \sum_{i=0}^{n-1} v_i x^i$

represent a codeword in a binary cyclic code V which is majority-logic decodable in one step. Suppose that the channel introduces an error of weight τ expressed as

$$e(x) = \sum_{j=1}^{\tau} x^{a_j} ,$$

where $a_j \in \{0,1,\ldots,n-1\}$ and every a_j is distinct. The received word expressed as

$$r(x) = \sum_{i=0}^{n-1} r_i x^i$$

will then be given by

$$r(x) = v(x) + e(x) .$$

Since V is 1-step majority-logic decodable, we can form d orthogonal estimates on the transmitted bit v_o as follows;

$$\hat{v}_o = \text{Majority of} \left\{ \begin{array}{l} s_{o,o} = r_o \\ s_{o,1} = \sum_{j=1}^{u_2} r_{a_{1,j}} \\ \cdot \\ \cdot \\ s_{o,b} = \sum_{j=1}^{u_b} r_{a_{b,j}} \\ \cdot \\ \cdot \\ s_{o,d-1} = \sum_{j=1}^{u_d} r_{a_{d-1,j}} \end{array} \right\} \qquad (2.10)$$

where $a_{b,j} \in \{1,2,\ldots,n-1\}$ for $b = 1,2,\ldots,d-1$ and all terms $r_{a_{b,j}}$ are distinct. Since the code V is cyclic, we can form d orthogonal estimates on any transmitted bit v_i by cyclically shifting all the terms in (2.10) by i positions. In other words we have

$$\hat{v}_i = \text{Majority of} \left\{ \begin{array}{l} s_{i,o} = r_i \\ s_{i,1} = \sum_{j=1}^{u_2} r_{i+a_{1,j}} \\ \cdot \\ \cdot \\ s_{i,b} = \sum_{j=1}^{u_b} r_{i+a_{b,j}} \\ \cdot \\ \cdot \\ s_{i,d-1} = \sum_{j=1}^{u_d} r_{i+a_{d-1,j}} \end{array} \right\} . \qquad (2.11)$$

If among the d estimates at least T estimates agree with one value $\hat{v}_i$, then $\hat{v}_i$ is taken to be the decoding answer for v_i, where $T > d/2$ being the majority threshold.

We note that in the absence of an error, the received word $r(x)$ will equal a codeword $v(x)$, and hence the d estimates in (2.11) on any v_i will unanimously agree with v_i. Therefore in the presence of an error $e(x)$, we can examine the decoder performance by taking the simple case of an all-zero codeword and considering the effect of $e(x)$ alone on the d orthogonal estimates.

The simplest form of a 1-step majority-logic decoder is that using a fixed majority threshold of $T = d - t$; i.e. $T = t+2$ for even d, whereas $T = t+1$ for odd d. Such a decoder will correct any error $e(x)$ of weight $\tau \leq t$ by going through one round of n estimations. However, if certain errors of weight $\tau > t$ are to be corrected then feedback decoding will be necessary in the sense that r_i is complemented when $\hat{v}_i \neq r_i$ before v_{i+1} is to be estimated [21,30]. With feedback decoding it may be necessary to go through more than one round of estimation and to use more than one majority-logic threshold. In this context, we shall discuss in the remainder of this chapter the capability of 1-step majority-logic decoding in correcting errors of weight $\tau > t$.

2.4 General Aspects of Two-Round Decoding

In formulating our decoding algorithms for dealing with errors of weight $t+1$, we shall adopt Decoding Strategy C given in section 2.2 by attempting to correct only an error that is not part of a codeword of weight $2t+1$ or $2t+2$. To be able to correct some of such errors without permitting the possibility of an incorrect majority vote with respect to an error of weight $t+1$ or less, we need to set the initial majority threshold T_1 higher than the error weight; i.e. $T_1 = t+2$. Since such a threshold does not preclude an incorrect majority vote with respect to an error of weight greater than $t+1$, the decoder ought to declare a detectable-only error and stop decoding whenever it sees more than $t+1$ received bits in error; i.e. bits for which $\hat{v}_i \neq r_i$. On the other hand, if the error weight at the end of decoding was $t+1$ or less, we would still need to verify the validity of the decoded answer $\hat{v}(x)$ by testing if $\hat{v}(x)$ modulo $g(x) = 0$, where $g(x)$ is the generator polynomial of the code V. This verification on the decoder answer is required since majority-logic decoding estimates each one of the n transmitted bits v_i separately which when put together, may not necessarily constitute a codeword. Failure to satisfy such a condition will again indicate a detectable-only error. (See Example 1).

In the following sections 2.5 and 2.6 we shall discuss the mechanism of two-round decoding. We shall see that no more than two decoding rounds will be necessary for the correction of uniquely-decodable errors of weight $t+1$. Whereas the first round is identical for both cases of odd and even minimum distance d , we shall see that the second round will be quite different in these two cases. Therefore for the sake of clarity, we shall discuss the two cases separately in sections 2.5 and 2.6 .

2.5 Two-Round Decoding with Even Minimum Distance

For the case of a code V of even $d = 2t+2$, we specify below Decoding Algorithm D1. This is a two-round algorithm which will be shown to be able to correct all errors of weight $\tau \leq t$ and some errors of weight $t+1$ that are coset leaders of unique weight in their cosets; i. e. uniquely decodable. We then analyze in details the performance of such an algorithm.

2.5.1 Decoding Details

DECODING ALGORITHM D1: We set the majority threshold fixed at $T = t+2$. Using the d orthogonal estimates in (2.11), we take a majority vote on v_i for $i = 0,1,\ldots,n-1$ in that order. In this process we complement r_i by replacing it with r_i+1, if $\hat{v}_i \neq r_i$, and then count the number q of received bits so complemented. Otherwise we leave r_i unchanged; i.e. when either $\hat{v}_i = r_i$ or there is a tie. In this manner, we go through one full round of n estimations. Suppose that what we have at this point is a modified received word $\check{r}(x)$ replacing the original received word $r(x)$. Then beyond this point, we follow one of three possible paths:

(a) We stop decoding and take $\hat{v}(x) = \check{r}(x)$, if $q \leq t+1$, $\check{r}(x)$ modulo $g(x) = 0$, and there was no tie at all during the first round, where $g(x)$ is the generator polynomial of the code V .

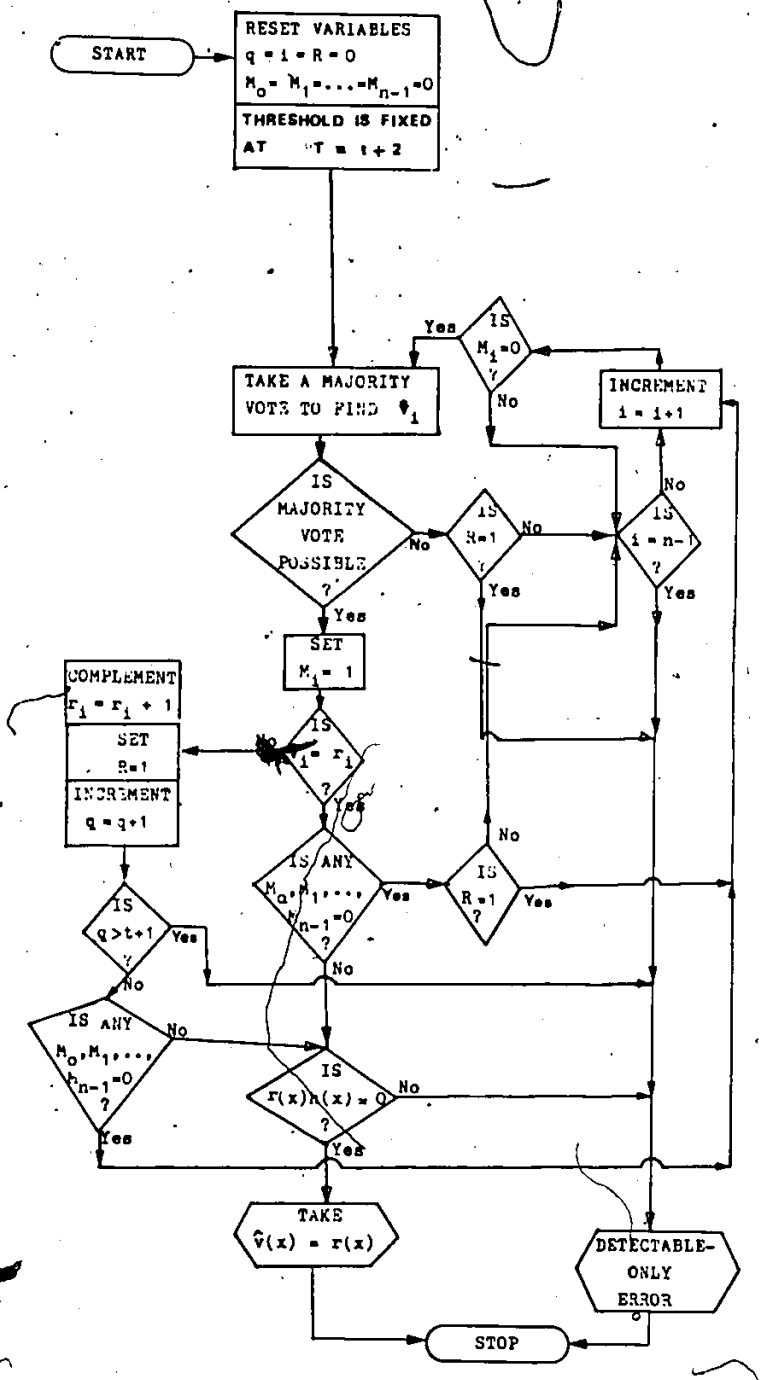
(b) If there was a tie and $0 < q < t+1$, we go through a second round of estimation in order to resolve any remaining ties, by taking a majority vote on the corresponding bit positions. If there has been no tie at all during the second round, $q \leq t+1$, and $r(x)$ modulo $g(x) = 0$, we then take the estimate $\hat{v}(x) = \check{r}(x)$.

(c) In all cases other than (a) and (b) above, we declare a detectable-only error and stop decoding.

The above specifies Decoding Algorithm D1 .

.....

Flowchart 1 shows a specification of Algorithm D1 . In this specification, certain techniques are configured to shorten the decoding time and reduce functional redundancy. We make use of two indices R and M_i for $i = 0,1,\ldots,n-1$. At the beginning of the algorithm we set $R = 0$ to signify the first decoding round. As soon as one of the received bits r_i has been complemented, we set $R = 1$ to signify the second round. Then any tie beyond this point will cause decoding to stop and declare a detectable-only error and thus avoiding further decoding delay. On the other hand, the variable M_i for $i = 0,1,\ldots,n-1$ is used to indicate whether a majority vote has been obtained for $\hat{v}_i$ or not. This is to prevent the decoder from taking a second unnecessary majority vote on an already estimated v_i .



Flowchart 1 : Algorithm D1 for Correcting Errors of Weight $\tau \leq t+1$ for Even d = 2t+2 .

2.5.2 Error-Correcting Capability

In order to examine the error-correcting capability of Decoding Algorithm D1 , we take the simple case of an all-zero codeword and consider the effect of an error $e(x)$ alone on the d orthogonal estimates in (2.11). In the case of an error of weight $\tau \leq t$, Algorithm D1 will obviously stop decoding at the end of the first round with the modified received word being equal to the transmitted codeword $v(x)$. With respect to errors of weight $t+1$, we give Theorems 2,3 and 4 to prescribe the conditions under which error correction by Algorithm D1 is possible. On the other hand, errors of weight $\tau > t+1$ are not permitted by Algorithm D1 which will abort taking a decision and declare a detectable-only error whenever the number q of detected error bits is greater than $t+1$. This is intended to avoid the possibility of an erroneously estimated answer in view of the decoding threshold being high enough only to guarrantee a correct estimate with respect to errors of weight $t+1$ or less.

THEOREM 2: For the case of a code V of even $d = 2t+2$, Algorithm D1 corrects every $(t+1)$-error given by

$$e(x) = \sum_{j=1}^{t+1} x^{a_j} \text{ , where } a_j \in \{0,1,\ldots,n-1\}$$

if and only if there is at least one error position a_j such that, for $i = a_j$, we have a majority vote among the d orthogonal estimates in (2.11); that is for $i = a_j$ the error affects the value of at most t orthogonal estimates.

Proof: Since the threshold T is greater than the error weight, an incorrect majority vote with respect to $e(x)$ is not possible. Suppose that in $e(x)$ there is at least one bit position a_j such that for $i = a_j$ a majority vote on v_i is possible during the first round of estimation. Then for $i > a_j$ what we are left with is an error of weight t so that a majority vote will be obtainable for every $i > a_j$. If $a_j \neq 0$, then for all $i < a_j$ -if any- giving a tie during the first round, a majority vote will be obtainable during the second round. Thus the modified received word after the completion of the second round will indeed be equal to the transmitted codeword $v(x)$.

If there is no $i < a_j$ for which there was a tie during the first round, then decoding stops at the end of the first round and $\check{r}(x)$ is equal to the transmitted code-word $v(x)$. Therefore an error specified by Theorem 2 will always be correctable by Algorithm D1.
On the other hand if no majority vote was possible on any of the error positions during the first round, Algorithm D1 will declare a detectable-only error and stops decoding. Q.E.D.

The proof of Theorem 2 will directly lead to

COROLLARY 2: Algorithm D1 will correct an error $e(x)$ of weight τ during no more than one decoding round of n estimations if either $\tau \leq t$, or $\tau = t+1$ with $e(x)$ satisfying Theorem 2 and including the position $a_1 = 0$.

.....

REMARK 3 : In view of Corollary 2 and the fact that an error of weight $t+1$ is less likely to occur than those of smaller weights, the additional average delay in Decoding Algorithm D1 over a single-round decoding algorithm is insignificant. Furthermore, since Algorithm D1 requires only one majority threshold, its decoding complexity will be in the same order as that of a single-round algorithm for correcting only errors of weight $\tau \leq t$.

REMARK 4 : Further to Remark 3 we note that for the two cases of the (21,11) difference-set code and the (15,6) expurgated BCH code both having $d = 6$, only one round of n estimations is required in Algorithm D1 to correct errors of weight $t+1 = 3$ or less. This can be verified by a close examination of the respective orthogonal relations of these two codes, which reveals that no tie is possible with any triple error satisfying Theorem 2 for correctability by Algorithm D1, hence no necessity for a second decoding round.

In Theorems 3 and 4 given below, we examine in more detail the error types specified in Theorem 2 .

THEOREM 3: Let a codeword $v(x)$ of weight $d = 2t+2$ in an even-weighted code V, be denoted by

$$v(x) = \sum_{j=1}^{d} x^{a_j} \text{ , where } a_j \in \{0,1,\ldots,n-1\} .$$

Let $e'(x)$ be an error of weight $t+1$ which forms part of $v(x)$. Then $e'(x)$ will produce a tie in Algorithm D1 when taking a majority vote on any one of the $t+1$ error positions.

Proof: With respect to (2.11) every sum $r_i+s_{i,b}$ for $i \in \{0,1,\ldots,n-1\}$ and $b \in \{0,1,\ldots,d-1\}$ is a parity-check sum which will be equal to zero when there is no error or when the error is a codeword. Thus in either case a unanimous agreement will be obtained among the d orthogonal estimates in (2.11) for every $\hat{v}_i$, $i=0,1,..,n-1$. In the case of an error of weight d which is a codeword, the only way a unanimous agreement can happen on estimating one of the error positions, is when such an error will affect all orthogonal estimates in (2.11); i.e. each estimate contains one and only one error bit on the right

hand side. As a result of this, any error of weight $\tau \leq d$ which is part of a codeword of weight d will affect τ orthogonal estimates, when estimating one of the error positions. Therefore in the case of an error like $e'(x)$, $t+1$ orthogonal estimates will be affected when taking a majority vote on any error position v_{a_j}, thus producing a tie in view of the majority threshold of Algorithm D1 being $T = t+2$. This completes the proof of Theorem 3. Q.E.D.

THEOREM 4: For the case of a code V of even $d = 2t+2$, Algorithm D1 will only correct an error of weight $t+1$ if such an error is uniquely decodable in the sense that it does not form part of a codeword of weight d.

Proof: The sets E_1, E_2 and E_3 in section 2.2 exhaust all possible error patterns of weight $t+1$. According to Lemma 3, E_3 is an empty set for even d. According to Lemma 1, every error in E_2 forms part of a codeword of weight $2t+2$. Theorem 3 shows that such errors will produce a tie when estimating any one error position, leaving Algorithm D1 unable to estimate the error. Thus the only type of error correctable by D1 must belong to to E_1 and will therefore not form part of a codeword of weight d. Q.E.D.

REMARK 5 : So far we have not shown that Algorithm D1 is optimum in the sense that every error of weight t+1 which does not form part of a codeword of weight d is correctable. However, we shall see later that D1 is optimum in this sense, for the cases of the (15,6) BCH code and the two classes of M-sequence codes and difference-set codes.

2.5.3 Bounds on Decoding Performance

Theorems 5 and 6 below, give lower bounds for the number of errors of weight t+1 that are correctable by Algorithm D1 . In proving these theorems we need to consider errors of the form

$$e_o(x) = 1 + x^{a_2} + x^{a_3} + \ldots + x^{a_{t+1}} , \qquad (2.12)$$

and take any type of (t+1)-error to be expressed as

$$e_c(x) = x^c \, e_o(x) , \qquad (2.13)$$

where $c \in \{0,1,\ldots,n-1\}$.

THEOREM 5: Let N be the size of the set Y_o which consists of all errors $e_o(x)$ having the form of (2.12) corresponding to which there is a majority vote on v_o among the d orthogonal estimates in (2.10). Then the number Q_{t+1} of errors of weight t+1 that are correctable by Algorithm D1 , is lower-bounded by

$$Q_{t+1} \geq \frac{nN}{t+1} . \qquad (T5)$$

Proof: Let Y_c be the set of all errors $e_c(x)$ having the form of (2.13) which are obtained by cyclically shifting every error $e_o(x)$ in Y_o by c positions. Due to the cyclic nature of the code V, an error $e_c(x)$ in Y_c will yield a majority vote among the d estimates in (2.11), for $i = c$, where $c = 0,1,\ldots,n-1$. Thus every error in Y_c is correctable by Algorithm D1 in view of Theorem 2. Therefore the total number of all correctable errors of weight t+1 will be given by

$$Q_{t+1} = |Y| \qquad (2.14)$$

where $|Y|$ denotes the size of the union set Y of all sets $Y_o, Y_1, \ldots, Y_{n-1}$. It is clear that among the sets $Y_o, Y_1, \ldots, Y_{n-1}$ the number of times every error pattern is repeated cannot be more than its weight t+1. In view of this and the fact that

$$|Y_o| = |Y_1| = \ldots = |Y_{n-1}| = N$$

we conclude that

$$|Y| \geq \frac{n.N}{t+1} \qquad (2.15)$$

Combining (2.15) with (2.14) leads to (T5) and hence proves Theorem 5. Q.E.D.

THEOREM 6: For the case of a code V with even minimum distance $d = 2t+2$, the number Q_{t+1} of errors of weight $t+1$ that are correctable by Algorithm D1, is lower-bounded by B_{t+1}, where

$$Q_{t+1} \geq B_{t+1} \triangleq \binom{n}{t+1} - \frac{n}{t+1}\binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t . \qquad (T6)$$

Proof: We prove this theorem by deriving an expression for N as defined in Theorem 5 and substituting the result in (T5). Let Y_o' be the set of all errors having the form of (2.12), for which no majority vote is possible among the d orthogonal estimates in (2.10) during the first round in D1. Let the size of Y_o' be M. Since there are altogether $\binom{n-1}{t}$ errors of the form of (2.12), the number of those giving a majority vote is

$$N = \binom{n-1}{t} - M . \qquad (2.16)$$

It is easy to see that an error of the form of (2.12) will belong to the set Y_o', if and only if the t error bits r_{a_j} for $j = 2,3,\ldots,t+1$, are distributed among

any t out of the $d-1$ orthogonal relations in (2.10) other than $s_{o,o} = r_o$. The value of M will therefore be given by the sum of products of the respective values of u_b corresponding to any combination of t orthogonal relations in (2.10) where $b \in \{1,2,..,d-1\}$. In other words we have

$$M = \sum u_{b_1} u_{b_2} \cdots u_{b_t} \qquad (2.17)$$

with the summation being over all values of b_i where every b_i is distinct; i.e. $1 \leq b_1 < b_2 < \ldots < b_t \leq d-1$. In view of the fact that

$$\sum_{b=1}^{d-1} u_b = n-1 \quad , \qquad (2.18)$$

it can be shown [18] that M is maximum when all the values of u_b are equal and where all the n terms $r_o, r_1, \ldots, r_{n-1}$ are present in (2.10); that is every

$$u_b = \frac{n-1}{d-1} \qquad (2.19)$$

for $b = 1,2,\ldots,d-1$. Using this fact and combining (2.17) with (2.19) will then give

$$M \leq \binom{d-1}{t}\left[\frac{n-1}{d-1}\right]^t = \binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t . \qquad (2.20)$$

Combining (2.20) with (2.16) yields

$$N \geq \binom{n-1}{t} - \binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t . \qquad (2.21)$$

By substituting (2.21) in (T5) we arrive at (T6) above, and thus prove Theorem 6 . Q.E.D.

EXAMPLE 1: As an example for Theorems 5 and 6 we consider the (15,6) expurgated BCH code of d=6 . If we take the reciprocal of the parity-check polynomial as $h^*(x) = 1 + x^3 + x^4 + x^5 + x^6$, we can then get the following orthogonal estimates on v_o :

$$\left.\begin{aligned} s_{o,o} &= r_o , \\ s_{o,1} &= r_1 + r_3 + r_7 , \\ s_{o,2} &= r_2 + r_6 + r_{14} , \\ s_{o,3} &= r_4 + r_{12} + r_{13} , \\ s_{o,4} &= r_8 + r_9 + r_{11} , \\ s_{o,5} &= r_5 + r_{10} . \end{aligned}\right\} \qquad (2.22)$$

We therefore have with respect to (2.10) ,

$$u_b = 3 \text{ , for } b = 1,2,3,4$$

and $$u_b = 2 \text{ , for } b = 5 . \qquad (2.23)$$

According to Theorem 6 , the number of triple errors correctable by D1 is lower-bounded by

$$Q_3 \geq 63 . \qquad (2.24)$$

However, since the result of Theorem 6 is based on the worst case where all the orthogonal estimates have equal number of terms with no missing term, the lower bound of (2.24) can be improved by finding the exact value of N and applying it in Theorem 5 . To do this we consider a triple error of the form

$$e_o(x) = 1 + x^{a_1} + x^{a_2}, \text{ with } 0 < a_1 < a_2 . \qquad (2.25)$$

The number of errors $e_o(x)$ giving a tie; i.e. affecting 3 estimates in (2.22) can be found as

$$M = \binom{4}{2} X3X3 + \binom{4}{1} X2X3 = 78 . \qquad (2.26)$$

Using (2.26) in (2.16) will then give

$$N = \binom{14}{2} - 78 = 13 . \qquad (2.27)$$

By substituting (2.27) in Theorem 5 we get

$$Q_3 \geq 65 . \qquad (2.28)$$

This result coincides with that given in [38] for the total number of triple errors that are uniquely decodable in the sense of not forming part of a codeword of weight 5 or 6 with respect to the BCH (15,7) code of $d = 5$.

A close examination of the expurgated (15,6) code shows that there are only 65 triple errors which do not form part of a codeword of weight 6. We therefore conclude that Algorithm D1 is optimum with respect to this code in the sense of correcting <u>all</u> uniquely-decodable triple errors.

In order to illustrate the need in the decoding algorithm to verify that the decoding answer $\hat{v}(x)$ is in fact a codeword, we consider the case of an all-zero codeword and an error

$$e(x) = 1 + x + x^2 + x^{13} .$$

Such an error would result in a decoding answer of

$$\hat{v}(x) = 1 + x + x^2 + x^9 + x^{13} ,$$

indicating an error of weight 1, i.e.

$$\hat{e}(x) = x^9 .$$

Here $\hat{v}(x)$ is obviously not a codeword since its weight is smaller than the minimum distance (d=6) of the (15;6) code.

2.5.4 Difference-Set Codes and M-Sequence Codes

In Theorems 5 and 6 we gave lower bounds on the number of (t+1)-errors that are correctable by Algorithm D1 . We also saw in Example 1 how a knowledge of the orthogonal check-sums of a given code can be used in conjunction with Theorem 5 in order to provide a tighter result than that obtained by Theorem 6 alone. In the same example we saw that Algorithm D1 is optimum in the correction of triple errors when used with the (15,6) double-error-correcting code.

In the sequel we develop some specific results on the performance of Algorithm D1 and show its optimality with respect to the following two well-known classes of 1-step majority-logic codes:

(a) Difference-set codes [32]

$$\begin{aligned} n &= 2^{2s} + 2^{s} + 1 = 4t^2 + 2t + 1 \ , \\ d &= 2^{s} + 2 = 2t + 2 \ , \\ n - k &= 3^{s} - 1 \ . \end{aligned} \tag{2.31}$$

A difference-set code is completely defined by a positive integer s . We shall henceforth refer to such a code as V_D .

(b) M-sequence codes [21,38]

$$n = 2^m - 1 = 4t + 3 ,$$
$$d = 2^{m-1} = 2t + 2 , \qquad (2.32)$$
$$k = m .$$

An M-sequence code is completely defined by a positive integer m. We shall henceforth refer to such a code as V_M.

Before we examine the properties of the orthogonal check-sums of the above-mentioned codes, we shall rewrite expressions (2.10) and (2.11) such that an estimate $\hat{e}_i$ of an error bit at position i can be obtained by taking a majority vote among $d-1$ orthogonal check-sums as follows:

$$\hat{e}_o = \text{Majority of} \left\{ \begin{array}{l} c_{o,1} = r_o + \sum_{j=1}^{u_1} r_{a_{1,j}} \\ c_{o,2} = r_o + \sum_{j=1}^{u_2} r_{a_{2,j}} \\ \cdot \\ \cdot \\ c_{o,b} = r_o + \sum_{j=1}^{u_b} r_{a_{b,j}} \\ \cdot \\ \cdot \\ c_{o,d-1} = r_o + \sum_{j=1}^{u_{d-1}} r_{a_{d-1,j}} \end{array} \right\} \qquad (2.33)$$

$\hat{e}_o = 0$ and

Similarly

$$\hat{e}_i = \text{Majority of } \left\{ \begin{array}{l} c_{i,1} = r_i + \sum_{j=1}^{u_1} r_{i+a_{1,j}} \\ c_{i,2} = r_i + \sum_{j=1}^{u_2} r_{i+a_{2,j}} \\ \cdot \\ \cdot \\ c_{i,b} = r_i + \sum_{j=1}^{u_b} r_{i+a_{b,j}} \\ \cdot \\ \cdot \\ c_{i,d-1} = r_i + \sum_{j=1}^{u_{d-1}} r_{i+a_{d-1,j}} \end{array} \right\} \quad (2.33a)$$

$\hat{e}_i = 0$ and

Next we define C_i as the subset in C, consisting of the $d-1$ orthogonal check-sums in (2.33a), for a given bit position i, where C is the union set of all n subsets C_i for $i = 0,1,\ldots,$.

Using the above definitions, we can now state two well-known properties of the orthogonal check-sums corresponding to difference-set codes and M-sequence codes.

PROPERTY 1: With respect to a code V_D or V_M, every orthogonal check-sum $c_{o,b}$ in (2.33) for $b \in \{1,2,\ldots,d-1\}$, has the same number of terms u other than r_o given by

$$u = \frac{n-1}{d-1} , \qquad \text{(P1)}$$

where $u = 2t$ for the code V_D , (P1a)

and $u = 2$ for the code V_M . (P1b)

PROPERTY 2: With respect to a code V_D or V_M , every check-sum belonging to the subset C_i , also belongs to another subset C_k in C for all $k = j_b + i$, $j_b \neq 0$.

Property 2 , together with the fact that both codes V_D and V_M are completely orthogonizable in one step will then lead to the following.

PROPERTY 3: With respect to a code V_D or V_M , we have

(a) A term r_i appears only in those $d-1$ check-sums that constitute the subset $C_i \in C$.

(b) Every pair of terms (r_i, r_j) for $i \neq j$, appears once and only once throughout the entire set C .

(c) The subset $C_i \in C$ contains all n terms $r_o, r_1, \ldots, r_{n-1}$.

Since there are $\binom{n}{2}$ pairs of terms (r_i, r_j) and since every check-sum in C has $\binom{u+1}{2}$ such pairs, we then get in view of Property 3 the following:

PROPERTY 4: For a code V_D or V_M, the size of C is given by

$$|C| = \binom{n}{2}/\binom{u+1}{2}, \tag{P4}$$

where $|C| = n$ for V_D, (P4a)

and $|C| = \frac{1}{3}\binom{n}{2}$ for V_M. (P4b)

Expression (P4a) is obtained by combining (P4) with (P1a) and (2.31), whereas expression (P4b) is obtained by combining (P4) with (P1b).

Finally we present for the case of difference-set codes the following

PROPERTY 5: With respect to the set C of a code V_D, every check-sum in C has exactly one term in common with any other check-sum in C.

Proof: According to Property 2, a check-sum $c_{i,b}$ as given in (2.33a) is a member of u+1 subsets $C_k \in C$ for $k = i, j_1+i, j_2+i, \ldots, j_u+i$. Then $c_{i,b}$ will have exactly one term in common with any other check-sum belonging to any one of the u+1 subsets C_k.

In other words the term r_i is common between $c_{i,b}$ and all other $d-2$ check-sums in C_i, the term r_{j_1+i} is common between $c_{i,b}$ and all other $d-2$ check-sums in C_{j_1+i}, and so on. On this basis, it is sufficient to prove that the union set

$$C' \equiv C_i \cup C_{j_1+i} \cup C_{j_2+i} \cup \ldots\ldots \cup C_{j_u+i}$$

is identical to the set C of all orthogonal check-sums for the code V_D. We prove this by showing that $|C'| = |C|$, where $|C|$ denotes tne size of C. We note that every two subsets $C_k \in C'$, have only the check-sum $c_{i,b}$ in common, otherwise Property 3 would be contradicted. We therefore have $d-2$ distinct check-sums in every one of the $u+1$ subsets $C_k \in C'$, plus the check-sum $c_{i,b}$ which is common to all the $u+1$ subsets $C_k \in C'$. Thus we get

$$|C'| = 1 + (u+1)(d-2) . \qquad (2.34)$$

Combining (2.34) with (P1a) and (2.31) then gives $|C'| = n$. By referring to (P4a) it is clear that $|C'| = |C|$, implying that $C' \equiv C$. Q.E.D.

.....

In the sequel we shall use Properties 1 to 5 to obtain derivations for the number Q_{t+1} of (t+1)-errors that are correctable by Algorithm D1 with respect to difference-set codes and M-sequence codes. First we use Theorem 6 to obtain numerical values on the lower bound for the correctable fraction of (t+1)-errors. This fraction can be expressed as

$$\frac{B_{t+1}}{\binom{n}{t+1}} = 1 - \frac{\binom{2t}{t-1}}{\binom{n-2}{t-1}} \left[\frac{n-1}{2t+1}\right]^{t-1} . \qquad \text{(T6A)}$$

Table I lists the numerical values of this fraction for some codes of relatively short lengths. In view of the results shown in Table I we can make the following

REMARK 6 : The fraction of (t+1)-errors correctable by Algorithm D1 with respect to a difference-set code or an M-sequence code, approaches unity with increasing code length n . This implies that for longer n , the decoder performance in terms of its decoding error probability, will become increasingly more dominated by the occurrence of errors of weight t+2 , making the code behave more like a (t+1)-error-correcting code rather than a t-error-correcting code.

Next we attempt to obtain some tighter results than that given in Theorem 6. In Example 1 we saw that by using Theorem 5 and applying our knowledge of the orthogonal check-sums of a given code to estimate the exact value of N_t, we can obtain tighter results than Theorem 6. However such a technique will not be applicable to the cases of difference-set codes and M-sequence codes since the lower bound on N given by (2.21) is indeed its actual value in view of Property 1. An alternative approach to that given in Example 1, is to determine the exact value of Q_{t+1} by making use of Properties 1 to 5. This is what we shall do in the sequel, for the cases of $t = 2,3$ and 4.

The results of our forthcoming analysis are also listed in Table I.

Before estimating Q_{t+1} it will be easier to first determine the number P_{t+1} of (t+1)-errors which are not correctable by Algorithm D1. Obviously

$$Q_{t+1} + P_{t+1} = \binom{n}{t+1} . \tag{2.35}$$

We then define a (t+1)-error as

$$e(x) = \sum_{j=1}^{t+1} x^{a_j} . \tag{2.36}$$

TABLE I

FRACTION OF ERRORS OF WEIGHT t+1

CORRECTABLE BY ALGORITHM D1

	(n,k)	d	$Q_{t+1}/\binom{n}{t+1}$ Lower Bound	$Q_{t+1}/\binom{n}{t+1}$ Actual Value
Difference-Set Codes	(21,11)	6	.158	.158
	(73,45)	10	.498	.726
	(273,191)	18	.844	—
	(1057,813)	34	.986	—
M-Sequence Codes	(7,3)	4	0	0
	(15,4)	8	.231	.308
	(31,5)	16	.595	.950
	(63,6)	32	.994	.99990
	(127,7)	64	.993	—
	(255,8)	128	.99997	—
BCH Code	(15,6)	6	.138	.143
Projective-Geometry Codes	(85,24)	22	.866	—
	(585,184)	74	.99992	—

Theorem 7 below indicates how P_{t+1} can be determined.

THEOREM 7: In the case of a difference-set code V_D or an M-sequence code V_M, an error $e(x)$ of the form of (2.36) will be uncorrectable by Algorithm D1, if and only if no check-sum in the set C of all check-sums, contains more than two of the error terms r_{a_j}.

Proof:

(a) Suppose that $e(x)$ is such that no check-sum in C contains more than two of the error terms r_{a_j}. We then consider the subset $C_{a_j} \in C$ of orthogonal check-sums on $\hat{e}_i$ of the form of (2.33a) for $i = a_j$. According to Property 3, this subset consists of all $d-1 = 2t+1$ check-sums containing r_{a_j}, whereas all other t error terms r_{a_k} for $k \neq j$, appear in the subset C_{a_j}. As a result of this, t of the check-sums in C_{a_j} will have r_{a_j} and only one other error term r_{a_k} for $k \neq j$ and thus each sum will be equal to "0". The other $t+1$ check-sums in C_{a_j} will be each equal to "1" since it contains no error term other than r_{a_j}. Therefore,

a tie will occur when using the subset C_{a_j} to take a majority vote on error position a_j . Similarly a tie will also occur when taking a majority vote on every error position a_k for $k \neq j$. Thus according to Theorem 2 , such an error will not be correctable by Algorithm D1.

(b) Suppose that $e(x)$ is such that there is at least one check-sum in C containing one error term r_{a_j} and two or more other error terms r_{a_k} for $k \neq j$. Then by following similar arguments to those used in (a) above, we see that in the subset $C_{a_j} \in C$ there will be $t-1$ or less check-sums equal to "0" and $t+1$ or more check-sums equal to "1". Therefore a majority vote will be possible and according to Theorem 2 , $e(x)$ will be correctable by Algorithm D1 . Hence the only errors that are not correctable by D1 are those described in (a) above. Q.E.D.

.....

Let Z be the set of all errors $e(x)$ of the form of (2.36) that are not correctable by Algorithm D1 .

Then $$P_{t+1} = |Z| \quad . \tag{2.37}$$

For the determination of P_{t+1} and thence Q_{t+1} , we shall individually deal with the cases of $t = 2, 3$ and 4 in the discussion to follow.

I. The case of t=2

When t=2, the error defined by (2.36) reduces to the form

$$e_3(x) = x^{a_1} + x^{a_2} + x^{a_3} \quad . \qquad (2.38)$$

Suppose that $e_3(x)$ belongs to the set of all triple errors that are not correctable by Algorithm D1. Then it follows from Property 3 that there are $\binom{n}{2}$ choices for any given pair of error positions say (a_1, a_2). These $\binom{n}{2}$ possible pairs will appear in $\binom{n}{2}$ check-sums in a one-to-one correspondence. For $e_3(x)$ to satisfy the condition of Theorem 7, the third error term r_{a_3} can be present in any check-sum in C_1, other than the check-sum containing the pair (r_{a_1}, r_{a_2}). There are therefore n-u-1 choices for a_3 excluding all the u+1 terms contained in the check-sum which has the pair (r_{a_1}, r_{a_2}).

As a result of the above we get

$$P_3 = \binom{n}{2} \left(\frac{n-u-1}{3}\right) , \qquad (2.39)$$

where u is given by (P1). Combining (2.39) with (2.35) gives the total number of triple errors correctable by Algorithm D1 as

$$Q_3 = \binom{n}{3} \left(1 - \frac{n-u-1}{n-2}\right) . \qquad (2.40)$$

EXAMPLE 2: Using the result of (2.40) for the (21,11) difference-set code of d=6 , gives Q_3=210 errors. This represents a portion of 0.1579 of triple errors.

II. The case of t=3

When t=3 , an error defined by (2.36) reduces to the form

$$e_4(x) = x^{a_1} + x^{a_2} + x^{a_3} + x^{a_4} \quad . \qquad (2.41)$$

Suppose that $e_4(x)$ belongs to the set of all quadruple errors that are not correctable by Algorithm D1 . On the basis of the discussion carried with respect to the case of t=2 , the number of choices for any three error positions say (a_1,a_2,a_3) is given by P_3 in (2.39). In view of Property 3 , there are $\binom{3}{2}$ check-sums in C , each of which contains a pair from the terms r_{a_1}, r_{a_2} and r_{a_3} . Such check-sums can be denoted by

$$\left.\begin{aligned} c_{1,2} &= r_{a_1} + r_{a_2} + .. + r_{b_1} + \ldots \\ c_{1,3} &= r_{a_1} + r_{a_3} + .. + r_{b_2} + \ldots \\ c_{2,3} &= r_{a_2} + r_{a_3} + .. + r_{b_3} + \ldots \end{aligned}\right\} \qquad (2.42)$$

where all given terms are distinct, and none of r_{b_1}, r_{b_2} or r_{b_3} is part of the error $e_4(x)$. Here the terms r_{b_1}, r_{b_2} and r_{b_3} are used to denote all terms other than the error terms r_{a_j} present in $c_{1,2}$, $c_{1,3}$ and $c_{2,3}$ respectively. The only terms that are repeated in (2.42) are r_{a_1}, r_{a_2} and r_{a_3}, each repeated twice. Thus the total number of distinct terms present in the three check-sums of (2.42) is given by $3(u+1) - 3 = 3u$, where u is given by (P1). According to Theorm 7, the fourth error term r_{a_4} can be present in any check-sum in C outside (2.42). Hence there are $n-3u$ choices for a_4 once a_1, a_2 and a_3 have been chosen for $e_4(x)$. As a result of the above discussion we get

$$P_4 = |Z_4| = P_3 \left(\frac{n-3u}{4}\right) \qquad (2.43)$$

By combining (2.43) with (2.40) and (2.35) we obtain the total number of qudruple errors that are correctable by Algorithm D1 as

$$Q_4 = \binom{n}{4} \left[1 - \left(\frac{n-u-1}{n-2}\right)\left(\frac{n-3u}{n-3}\right)\right] . \qquad (2.44)$$

EXAMPLE 3: Using the result of (2.44) for the (15,4) M-sequence-code of $d=8$, gives $Q_4=420$ errors representing a 0.3077 portion of the total number of quadruple errors. We note that for the case of M-sequence (simplex) codes in general, it is possible to find the number of uniquely-decodable errors of weight $t+1$ [33]. Thus for the (15,4) code, there are 420 uniquely-decodable errors of weight 4 . This is the same as Q_4 estimated above. We therefore conclude that Algorithm D1 is optimum for this code with respect to quadruple errors that are uniquely decodable in the sense of not forming part of a codeword of weight 8 .

III. The case of t=4

To deal with the case of t=4 , we specifically examine the (73,45) difference-set code of d=10 and make use of Property 5 . First we define an error of weight 5 taking the form of

$$e_5(x) = x^{a_1} + x^{a_2} + x^{a_3} + x^{a_4} + x^{a_5} \quad (2.45)$$

and belonging to the set of all such errors that are not correctable by Algorithm D1. Then on the basis of the discussion carried with respect to the case of t=3 , there are P_4 choices for any combination of four error positions say (a_1, a_2, a_3, a_4) where P_4 is as given by (2.43). In view of Property 3 , there are $\binom{4}{2}$ check-sums in C each containing a pair of the terms r_{a_1}, r_{a_2}, r_{a_3} and r_{a_4}. Also according to Property 5 , there is one common term between any two of these $\binom{4}{2}$ check-sums. We can therefore denote such check-sums by

$$\left.\begin{aligned}
c_{1,2} &= r_{a_1} + r_{a_2} + r_{c_1} + .. + r_{b_1} + \cdots \\
c_{1,3} &= r_{a_1} + r_{a_3} + r_{c_2} + .. + r_{b_2} + \cdots \\
c_{2,3} &= r_{a_2} + r_{a_3} + r_{c_3} + .. + r_{b_3} + \cdots \\
c_{1,4} &= r_{a_1} + r_{a_4} + r_{c_3} + .. + r_{b_4} + \cdots \\
c_{2,4} &= r_{a_2} + r_{a_4} + r_{c_2} + .. + r_{b_5} + \cdots \\
c_{3,4} &= r_{a_3} + r_{a_4} + r_{c_1} + .. + r_{b_6} + \cdots
\end{aligned}\right\} \quad (2.46)$$

where all given terms are distinct. Here the terms r_{c_1}, r_{c_2} and r_{c_3} are used to denote all terms which are not part of the error $e_5(x)$ but each one of which is common between two check-sums in (2.46). According to Property 5 , there are only three such terms present in the check-sums of (2.46) . On the other hand, we use the term r_{b_1} to denote any other term present in $c_{1,2}$ but is not part of $e_5(x)$ and is not common with any other check-sum in (2.46). We also use $r_{b_2}, \ldots, r_{b_6}$ for a similar purpose in $c_{1,3}, \ldots ; c_{3,4}$ respectively. In view of this and Theorem 7 , the total number of distinct terms present in (2.46) is given by $4 + 3 + \binom{4}{2}(u-2) = 6u - 5$. According to Theorem 7 the fifth error digit can be present in any check-sum in C outside (2.46). The size of Z_5 is therefore given by

$$P_5 = P_4 \left(\frac{n-6u+5}{5} \right) . \qquad (2.47)$$

Combining (2.47) with (2.43), (2.39) and (2.35) gives the total number of errors of weight 5 that are correctable by Algorithm D1 as

$$Q_5 = \binom{n}{5} \left[1 - \left(\frac{n-u-1}{n-2}\right)\left(\frac{n-3u}{n-3}\right)\left(\frac{n-6u+5}{n-4}\right) \right] . \qquad (2.48)$$

EXAMPLE 4: By using the result of (2.48) for the (73,45) difference-set code of $d=10$, we get $Q_4 = 10,899,629$ errors of weight 5, which represent a fraction of 0.7257 of the total number of errors of weight 5.

The values of $Q_{t+1}/\binom{n}{t+1}$ obtained in Examples 2 to 4 above, are listed in Table I for the sake of comparison with the lower bound of Theorem 6. Extension of the analysis given above for $t>4$, will become increasingly more complicated with higher values of t, and may therefore require some computer simulation. This is a topic which is subject to furtner investigation.

.....

We have so far seen in Examples 1 and 3 that Algorithm D1 is optimum for the two cases of (15,6) BCH code and (15,4) M-sequence code respectively, in the sense of correcting all errors of weight $t+1$ that form no part of a codeword of weight $2t+2$. In the sequel we develop Theorem 8 to show that the optimality of Algorithm D1 in this sense is general to all members of the two classes of difference-set codes and M-sequence codes. To assist in proving Theorem 8 we first give Lemmas 5 to 7.

With respect to a difference-set code V_D or an M-sequence code V_M we define F to be the set of all errors $f_i(x)$ of weight $t+1$ which are not correctable by Algorithm D1, where

$$f_i(x) = \sum_{j=1}^{t+1} x^{a_{j,i}} , \qquad (2.51)$$

$a_{j,i} \in \{0,1,..,n-1\}$, every $a_{j,i}$ is distinct and $i = 0,1,..,|F|$. Here $|F|$ denotes the size of F. We then define S_{f_i} as the subset of check-sums each containing at least two terms $r_{a_{j,i}}$ for $x^{a_{j,i}} \in f_i(x)$ where S_{f_i} belongs to the set C of all orthogonal check-sums.

LEMMA 5: To every error pattern $f_i(x) \in F$, there corresponds a subset $S_{f_i} \in C$ such that S_{f_i} consists of $\binom{t+1}{2}$ check-sums each containing two and only two error terms $r_{a_{j,i}}$ due to $f_i(x)$.

Proof: Since $f_i(x)$ is an error pattern of weight $t+1$ which is not correctable by Algorithm D1, we conclude from Theorem 7 that there is no check-sum in C which contains more than two error terms $r_{a_1,i}, r_{a_2,i}, \cdots, r_{a_{t+1},i}$ due to $f_i(x)$. According to Property 3(b), every pair of such error terms appears once and only once in C. As a result there will be $\binom{t+1}{2}$ check-sums in C each containing two and only two error terms due to $f_i(x)$. Q.E.D.

LEMMA 6: Let $f_1(x)$ and $f_2(x)$ be two error patterns of weight $t+1$ such that $f_1(x) + f_2(x) = w_1(x)$ where $w_1(x)$ is a codeword of weight $2t+2$ in a code V_D or V_M. Then $f_1(x)$ and $f_2(x)$ belong to F, and the two subsets of check-sums S_{f_1} and S_{f_2} in C corresponding to $f_1(x)$ and $f_2(x)$ respectively are mutually exclusive.

Proof: According to Theorem 4, $f_1(x)$ and $f_2(x)$ must belong to the set F of errors of weight $t+1$ that are not correctable. Then in view of Lemma 5, each one of the two subsets S_{f_1} and S_{f_2} in C consists of $\binom{t+1}{2}$ check-sums with one pair of error terms per check-sum due to $f_1(x)$ and $f_2(x)$ respectively. Now we prove Lemma 6 by way of contradiction. Suppose that there is at least one check-sum c_x which is common between S_{f_1} and S_{f_2}. Since there are no terms common between $f_1(x)$ and $f_2(x)$, then c_x must contain a pair of error terms due to $f_1(x)$ plus another pair of error terms due to $f_2(x)$. In other words c_x contains four terms from the codeword $w_1(x)$. Then in view of Theorem 2 and its proof, Algorithm D1 will be able to correct an error of weight $t+1$ which is part of $w_1(x)$ and contains at least three out of the four terms contained in c_x. This, however, is a contradiction with Theorem 3. Hence Lemma 6 is proven. Q. E. D.

LEMMA 7: Every error pattern $f_i(x) \in F$ is part of a codeword of weight $d = 2t + 2$.

Proof: We consider two error patterns $f_1(x)$ and $f_2(x)$ of weight $t+1$ such that $f_1(x) + f_2(x) = w_1(x)$ where $w_1(x)$ is a codeword of weight $2t+2$. Lemma 6 shows that $f_1(x)$ and $f_2(x)$ belong to F and that their two corresponding subsets S_{f_1} and S_{f_2} in C are mutually exclusive. We then consider another error pattern $f_3(x) \in F$ where $f_3(x) \neq f_1(x)$ or $f_2(x)$. We note from Property 3 that a term r_i appearing in all $d-1$ check-sums of the subset C_i as given by (2.33a), is associated with the other $n-1$ terms r_{i+b} for $b = 1,2,\ldots,n-1$ in a similar way to how another term r_j for $j \neq i$, appearing in all $d-1$ check-sums of C_j, is associated with the $n-1$ terms r_{j+c} for $c = 1,2,\ldots,n-1$. This will lead to the following argument; since we are able to find, for a given $f_1(x) \in F$, another $f_2(x) \in F$ such that the two corresponding subsets S_{f_1} and S_{f_2} in C are mutually exclusive, we must also be able to find at least one $f_4(x) \in F$ in conjunction with any given $f_3(x) \in F$ such that their corresponding subsets S_{f_3} and S_{f_4} in C are also mutually exclusive. It will then follow that

$f_3(x)$ and $f_4(x)$ have no term in common and that an error term $r_{a_j,3}$ due to $f_3(x)$ will appear in t out of the $\binom{t+1}{2}$ check-sums in the subset S_{f_3} but not in any check-sum in the subset S_{f_4}. On the other hand, an error term $r_{a_j,4}$ due to $f_4(x)$ will appear in t check-sums in S_{f_4} but not in S_{f_3}. Thus for a given error term $r_{a_j,3}$, there are $d-1-t = t+1$ check-sums in C outside the subsets S_{f_3} and S_{f_4}, containing the same term $r_{a_j,3}$. The same argument will also be valid for an error term $r_{a_j,4}$ due to $f_4(x)$. On the basis of this, and in view of Property 3 and Theorem 7, we can see that within the set C and outside the subsets S_{f_3} and S_{f_4}, each one of the $t+1$ check-sums containing $r_{a_j,4}$, must contain only one term due to $f_3(x)$ and another term due to $f_4(x)$. Thus if we consider the word $w_2(x) = f_3(x) + f_4(x)$, the terms corresponding to $w_2(x)$ will appear only in pairs throughout the set C of all orthogonal check-sums. Thus all check-sums in C will be equal to zero, implying that $w_2(x)$ is a codeword. Since $f_3(x)$ and $f_4(x)$ have no term in common, $f_3(x)$ must be a part of $w_2(x)$ and the weight of $w_2(x)$ must be $2t+2$. This completes the proof of Lemma 7. Q. E. D.

THEOREM 8: In the case of a difference-set code V_D or an M-sequence code V_M, Algorithm D1 will correct an error $e(x)$ of weight $t+1$ if and only if $e(x)$ is uniquely-decodable in the sense that $e(x)$ is not part of any codeword of weight $d = 2t+2$.

Proof: According to Theorem 4, Algorithm D1 will only correct an error of weight $t+1$ if it is not part of a codeword of weight d. On the other hand, Lemma 7 shows that every error of weight $t+1$ that is not correctable by Algorithm D1 must be part of a codeword of weight d. Therefore both necessary and sufficient conditions of Theorem 8 are satisfied. Q. E. D.

.....

On the basis of Theorem 8, we can now estimate the actual value of Q_{t+1} for any given M-sequence code by using the derivations given in [33] for the number of coset leaders of weight $t+1$. The results thus obtained are listed in Table I for codes of length 63 or less. In comparison with values of the lower bound on Q_{t+1} listed in the same table, we notice that the actual value of $Q_{t+1}/\binom{n}{t+1}$ will approach unity even more rapidly for increasing code length. This will amplify our Remark 6 made with respect to the increasing effect of errors of weight $t+2$ over the decoding performance of Algorithm D1.

2.6 Two-Round Decoding with Odd Minimum Distance

In this section we analyze the performance of Decoding Algorithm D2 which deals with the case of a code V' of odd $d = 2t+1$. This is a two-round algorithm which will be shown to be able to correct all errors of weight $\tau \leq t$ and some errors of weight $t+1$ that are coset leaders of unique weight in their cosets; i.e. uniquely decodable.

2.6.1. Decoding Details

DECODING ALGORITHM D2: First we set the majority threshold at $T_1 = t+2$. For a majority vote, at least T_1 out of the d orthogonal estimates in (2.11) must agree. Otherwise a tie situation arises. On this basis, we take a majority vote on v_i for $i = 0,1,\ldots,n-1$ in that order, leaving r_i unchanged if $\hat{v}_i = r_i$ or if there is a tie, in the sense that no majority vote is possible. If $\hat{v}_i \neq r_i$, we complement r_i by replacing it with r_i+1.

The first decoding round will end either at the first instance we encounter one $\hat{v}_i \neq r_i$ or at $i = n-1$, whichever occurs sooner.

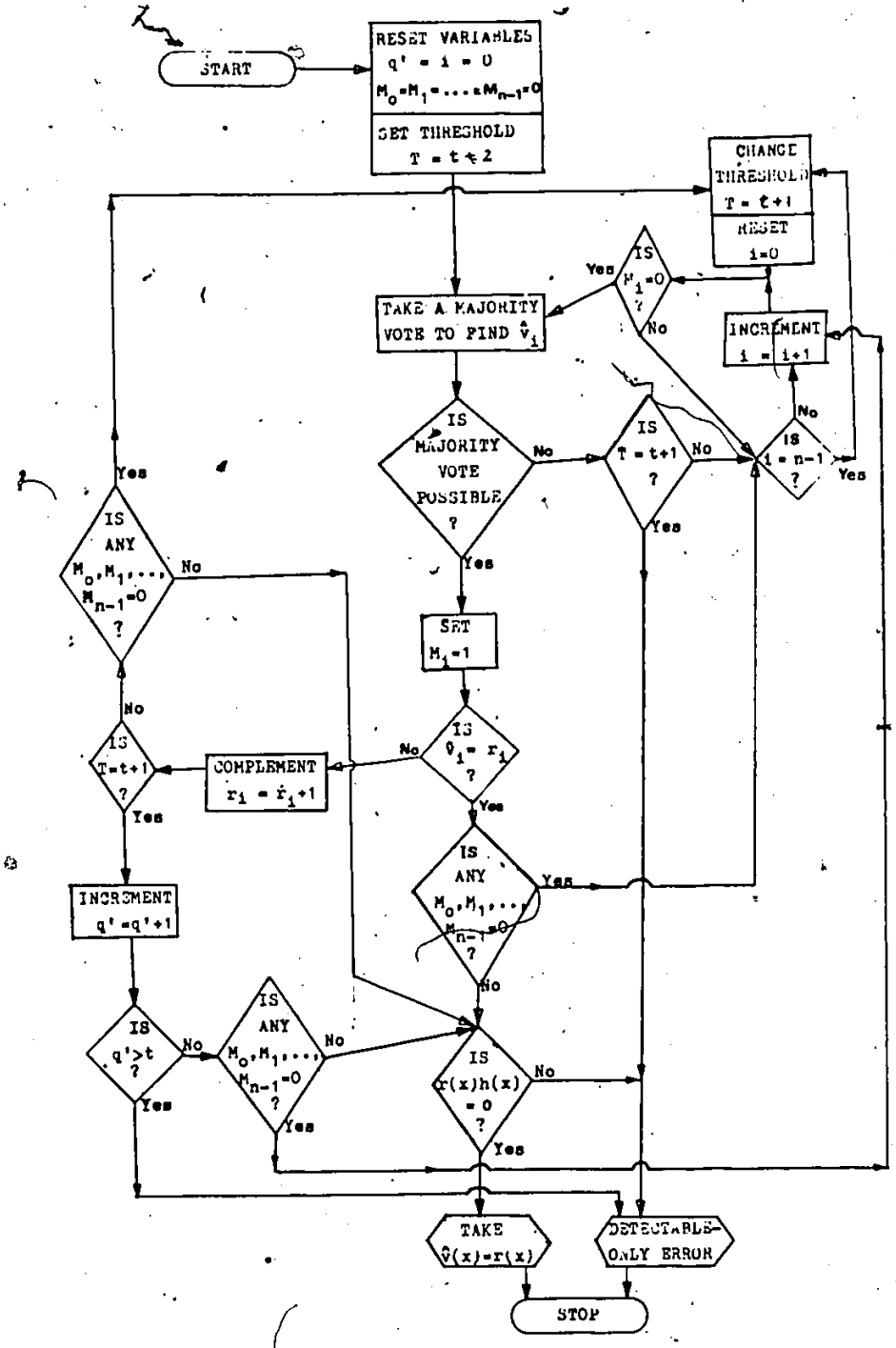
We then go through a second decoding round in case

there are some ties to be resolved. At the beginning of the second round, we lower the majority threshold to $T_2 = t+1$ and proceed to take majority votes on those bits that have not been estimated during the first round. We also count the number q' of the complemented bits in the second round. A tie in the second round will indicate an error which is detectable only. Otherwise we continue taking majority votes until all n bits have been estimated, with the result being a modified received word $\check{r}(x)$. If at this point $\check{r}(x)$ modulo $g(x) = 0$, and $q' = t$, we take $\hat{v}(x) = \check{r}(x)$, where $g(x)$ is the generator polynomial of the code V'. Otherwise we declare a detectable-only error.

The above specifies Decoding Algorithm D2.

.....

Flowchart 2 shows a specification of Decoding Algorithm D2. As in the case of Algorithm D1, we show some simply-implementable techniques to shorten the decoding time. Here the value of the threshold T is used to indicate whether the decoder is in its first or second round. On the other hand, the variable M_i for $i = 0,1,\ldots,n-1$ is used to indicate whether a majority vote has been obtained for $\hat{v}_i$ or not, in order to avoid an unnecessary second majority vote on an already estimated v_i.



Flowchart 2 : Algorithm D2 for Correcting Errors of Weight $\tau \geq t+1$ for Odd $d = 2t+1$.

2.6.2 Error-Correcting Capability

In order to examine the error-correcting capability of Decoding Algorithm D2 , we take the simple case of an all-zero codeword and consider the effect of the error alone on the d orthogonal estimates in (2.11), similar to what we did when we examined Algorithm D1 . In this respect we consider various error situations as follows:

(i) In the case of an error of weight t-1 or less, we have, in the first decoding round, a correct majority vote on every bit v_i for $i = 0,1,\ldots,n-1$. Thus after going tnrough one complete round of estimation, the resulting word $\check{r}(x)$ is equal to the transmitted codeword $v(x)$.

(ii) In the case of an error of weight t , we may or may not have a majority vote on a given bit v_i during the first round. If there is a majority vote it will be always correct. However, we shall be able to get a correct majority vote on every v_i for $i = 0,1,\ldots,n-1$ at the lower majority threshold $T_2 = t+1$. Therefore at the end of decoding, the resulting word $\check{r}(x)$ is equal to the transmitted codeword $v(x)$.

(iii) In the case of an error of weight t+1 , the performance of Algorithm D2 during its first decoding round will be similar to that of Algorithm D1 , since the majority threshold is greater than the error weight making an incorrect majority vote impossible. During the second round, however, Algorithm D2 will be forced to abort taking

a decision on $\hat{v}(x)$ if more than t bits are to be complemented. This is clearly to avoid the possibility of an erroneous decoding during the second round, in view of the decoding threshold $T_2 = t+1$. Thus an error of weight $t+1$ can only be corrected if one of its error positions is removed during the first round. Theorems 9 to 11 given in the sequel will specify the conditions under which an error of weight $t+1$ is correctable by Algorithm D2.

(iv) Errors of weight greater than $t+1$, on the other hand, are excluded from correction by Algorithm D2, since decoding will be aborted if more than a total of $t+1$ error bits are to be complemented. This is done in order to avoid an erroneous estimate, as discussed in case (iii) above.

THEOREM 9: For the case of a code V' of odd $d = 2t+1$, Algorithm D2 corrects every $(t+1)$-error given by

$$e(x) = \sum_{j=1}^{t+1} x^{a_j}, \text{ where } a_j \in \{0,1,\ldots,n-1\}$$

if and only if there is at least one error position a_j such that, for $i = a_j$, we have a majority vote among the d orthogonal estimates in (2.11') during the first round; that is for $i = a_j$ the error affects the value of at most $t-1$ orthogonal estimates.

Proof: Since the threshold of the first decoding round is $T_2 = t+2$ which is greater than the error weight, an incorrect majority vote with respect to $e(x)$ is not possible. Suppose that in $e(x)$ there is at least one bit position a_j such that for $i = a_j$, a majority vote on v_i is possible during the first round of Algorithm D2. Such an error pattern will be reduced to another one of weight t, before the second round begins with a lower threshold $T_2 = t+1$. Then during the second round, all the remaining t error bits can be corrected, leaving $\check{r}(x)$ equal to $v(x)$. On the other hand if no majority vote was possible on any of the error positions during the first round, then no correct answer will be possible during the second round due to the restriction by D2 on the number of complemented bits not to exceed t during this round. Q. E. D.

.....

REMARK 7 : In line with Remark 3 we note that the case of odd d is not different from the case of even d in the sense that the additional error-capability of Algorithm D2 over a single-threshold single-round decoder can be achieved with an insignificant increase in the average decoding delay.

Theorems 10 and 11 are given below to provide more details on the error types specified in Theorem 9 .

THEOREM 10: Let $e'(x)$ be an error of weight $t+1$ which forms part of a codeword $v'(x)$ of weight $2t+2$ in a code V' of odd $d = 2t+1$. Then $e'(x)$ will be declared by Algorithm D2 as a detectable-only error.

Proof: We use similar arguments to those used in proving Theorem 3 for the case of even d , in order to show that when taking a majority vote on any error position a_j in $e'(x)$, using the d orthogonal estimates in (2.11) for $i = a_j$, then either $t+1$ or t such estimates will be affected by the error bits of $e'(x)$. In the latter case the remaining error bit will be missing from (2.11) . Thus in either case, no majority vote is possible on any one of the $t+1$ error digits in $e'(x)$ during the first round of D2 , leaving the error weight unchanged. Such an error will then be declared by Algorithm D2 as detectable-only since the condition of $q' = t$ cannot be met for obtaining a decoder estimate of the transmitted codeword. Q. E. D.

THEOREM 11: For the case of a code V' of odd $d = 2t+1$, Algorithm D2 will only correct an error of weight $t+1$ if such an error is uniquely decodable in the sense that it does not form part of a codeword of weight $2t+1$ or $2t+2$.

Proof: The sets E_1, E_2 and E_3 defined in section 2.2 exhaust all possible error patterns of weight $t+1$. Lemma 2 states that every error belonging to E_3 is part of a codeword of weight $2t+1$. As discussed in section 2.2 an error belonging to E_3 is not correctable since it will lead to an erroneous decoding answer. Lemma 1 states that every error belonging to E_2 forms a part of a codeword of weight $2t+2$, hence such an error will be declared by D2 as detectable-only in accordance with Theorem 10. Thus the only type of error that Algorithm D2 may be able to correct is that belonging to E_1, which does not form a part of a codeword of weight $2t+1$ or $2t+2$.

Q. E. D.

2.6.3 Bounds on Decoding Performance

By using similar analysis to that given for Algorithm D1, we can derive for the case of odd minimum distance, a lower bound B'_{t+1} on the number Q'_{t+1} of errors of weight $t+1$ that can be corrected by Algorithm D2 . Such a lower bound is given in Theorem 12 below.

THEOREM 12: For the case of a code V' with odd minimum distance $d = 2t+1$, the number Q'_{t+1} of errors having weight $t+1$ that are correctable by Algorithm D2 , is lower-bounded by B'_{t+1} where

$$Q'_{t+1} \geq B'_{t+1} \triangleq \binom{n}{t+1} - \frac{n}{t+1}\binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t \quad . \qquad (T12)$$

Proof: A comparison between Theorems 9 and 2 reveals that errors of weight $t+1$ that are correctable by Algorithms D2 and D1 for the cases of d odd and d even respectively, must meet similar conditions in terms of the requirement to have a majority vote possible on one of the error positions during the first decoding round. As a result, Theorem 5 given for the case of d even will be equally valid for the case of d odd. In other words we have

$$Q'_{t+1} \geq \frac{n\,N'}{t+1} \qquad (2.55)$$

where N' is the number of errors $e_o(x)$ having the form of (2.12) corresponding to which there is a majority vote on v_o during the first decoding round. For the sake of convenience we repeat (2.12) below

$$e_o(x) = 1 + x^{a_2} + x^{a_3} + .. + x^{a_{t+1}} \quad . \qquad (2.12)$$

By defining M' as the number of errors of the form of (2.12) which are not included in N' we get

$$N' = \binom{n-1}{t} - M' \quad . \qquad (2.56)$$

It is easy to show that for an error to be included in M', the terms $r_{a_2}, r_{a_3}, .. , r_{a_{t+1}}$ must be present in either t or $t-1$ out of the $d-1$ relations in (2.10) excluding $s_{o,o} = r_o$. Let u_d denote the number of terms missing from (2.10) for a code V' with d odd. Then by using the same arguments used in the proof of Theorem 6 for the case of d even, we can show that M' will be maximum when the number of terms u_b in every one of the $d-1$ relations is equal to u_d. In other words

$$u_b = u_d = \frac{n-1}{d} \quad , \qquad (2.57)$$

for $b = 1,2,\ldots,d-1$. Thus in similarity with the case of d even we obtain, for d odd,

$$M' \leq \binom{d}{t}\left[\frac{n-1}{d}\right]^t = \binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t \quad . \qquad (2.58)$$

Combining (2.55) with (2.56) and (2.58) then leads to (T12) above and will thus prove Theorem 12 . Q.E.D.

.....

The result of Tneorem 12 is clearly identical to that given in Theorem 6 . In view of this we give the following corollary.

COROLLARY 3: The set of errors of weight $t+1$ that can be corrected by Algorithm D2 when using an (n,k) code V' of odd d, is identical to the set correctable by Algorithm D1 when using the even-weighted $(n,k-1)$ subset code V.

Proof: With reference to the d orthogonal relations of the form of (2.11), we note that the set S_1 of the $2t+2$ orthogonal relations for the even-weighted subset code, include the set S_2 of all $2t+1$ orthogonal relations for the parent code of odd d, with the addition of one relation consisting of all the terms missing from S_2 [19]. This means that when Theorem 12 applies to the (n,k) code V' of minimum distance $2t+1$, we shall also have Theorem 6 applicable to the subset $(n,k-1)$ code V of minimum distance $2t+2$. Q. E. D.

.....

EXAMPLE 5: As an example for Theorem 12 we consider the (15,7) BCH code of $d=5$ which is the parent code of that given in Example 1. If we take the reciprocal of the parity-check polynomial as $h^*(x) = 1 + x + x^3 + x^7$, we can get the same set of orthogonal estimates on v_o as given in Example 1 excluding the estimate $s_{o,5} = r_5 + r_{10}$. In other words we have, for the (15,7) code, the following estimates on v_o ;

$$\left.\begin{aligned} s_{o,o} &= r_o \ , \\ s_{o,1} &= r_1 + r_3 + r_7 \ , \\ s_{o,2} &= r_2 + r_6 + r_{14} \ , \\ s_{o,3} &= r_4 + r_{12} + r_{13} \ , \\ s_{o,4} &= r_8 + r_9 + r_{11} \ . \end{aligned}\right\} \qquad (2.59)$$

We therefore have, with respect to (2.10) ,

$$u_b = 3 \quad \text{for} \quad b = 1,2,3,4 \ . \qquad (2.60)$$

If we directly apply (2.60) in (T12) to obtain a lower bound on the number Q'_{t+1} of triple errors corrected by D2 , we get $Q'_3 \geq 63$. However, since the derivation of (T12) is based on condition (2.57) which is not met for this code, the lower bound on Q'_3 can be improved by finding an exact value of M' , as we already did in Example 1 . Thus in view of Corollary 3 we can use (2.28) from Example 1 directly to obtain

$$Q'_3 = Q_3 \leq 65 \ . \qquad (2.61)$$

We can then make a similar comment to that made in Example 1 regarding the optimality of Algorithm D2 in the sense of being able to correct all uniquely-decodable errors of weight 3 .

2.7 Decoding with Multiple Rounds

In the previous sections we analyzed Decoding Algorithms D1 and D2 with respect to their ability to correct errors of weight $t+1$, for the case of even and odd minimum distance respectively. Either algorithm will undergo up to two decoding rounds of majority-logic estimation. In this section we generalize the decoding method to allow for the correction of errors of weights greater than $t+1$ by going through more than two decoding rounds. For this purpose we give two decoding algorithms D3 and D4 for the cases of even and odd minimum distance respectively. We then examine in some details the types of errors of weights greater than $t+1$ that are correctable by Algorithms D3 and D4 .

2.7.1 Decoding Details

DECODING ALGORITHMS D3 AND D4 : Both algorithms begin the first decoding round with a majority-logic threshold of $T_1 \geq t+2$, and then proceed in taking majority votes on each $\hat{v}_i$ for $i = 0,1,\ldots,n-1$ in a similar manner to that described for Algorithm D2 . For each bit position i, an index M_i is assigned which is initially set at $M_i = 0$ indicating that an estimate of v_i is yet to be obtained. Every time a majority vote occurs for $\hat{v}_i$, the

value of M_i is changed to "one" in order to prevent taking another majority vote for the same i during the forthcoming rounds of estimation. In case there is a tie in the sense that no majority vote is possible for i, we leave r_i unchanged and $M_i = 0$. In case $\hat{v}_i = r_i$, we also leave r_i unchanged but set $M_i = 1$. Otherwise we complement r_i and set $M_i = 1$, if $\hat{v}_i \neq r_i$. The first decoding round will end either at the first instance we encounter one $\hat{v}_i \neq r_i$ or at $i = n-1$, whichever occurs sooner. We then lower the majority-logic threshold to $T_2 = T_1 - 1$ and go through a second round of estimation in case one or more bits v_i remain to be estimated. As long as there is some $M_i = 0$ for one or more positions i, the above decoding process will be recursively applied with the majority-logic threshold lowered by "one" at the beginning of every new round. In other words, at the beginning of the m'th round, the threshold is lowered to

$$T_m = T_1 - m + 1 \quad . \tag{2.65}$$

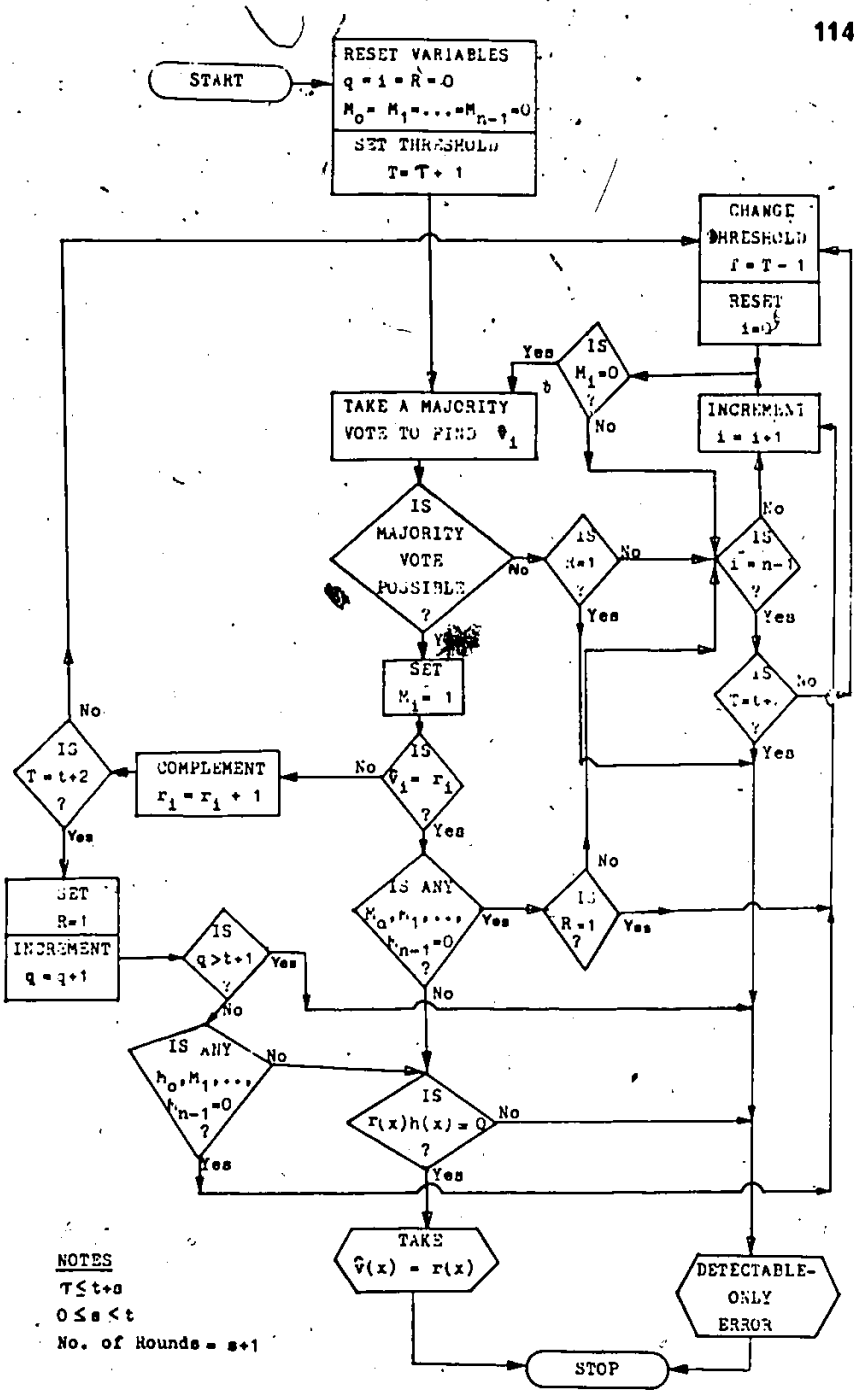
Suppose that after going through $m = T_1 - t - 2$ rounds, and hence reaching a threshold of $T_m = t+2$, there is still some $M_i = 0$ for one or more i. Then the

remaining steps in decoding will depend on whether d is even or odd. For the case of even d , Algorithm D3 will apply which will henceforth follow exactly the same steps as those described for Algorithm D1 .

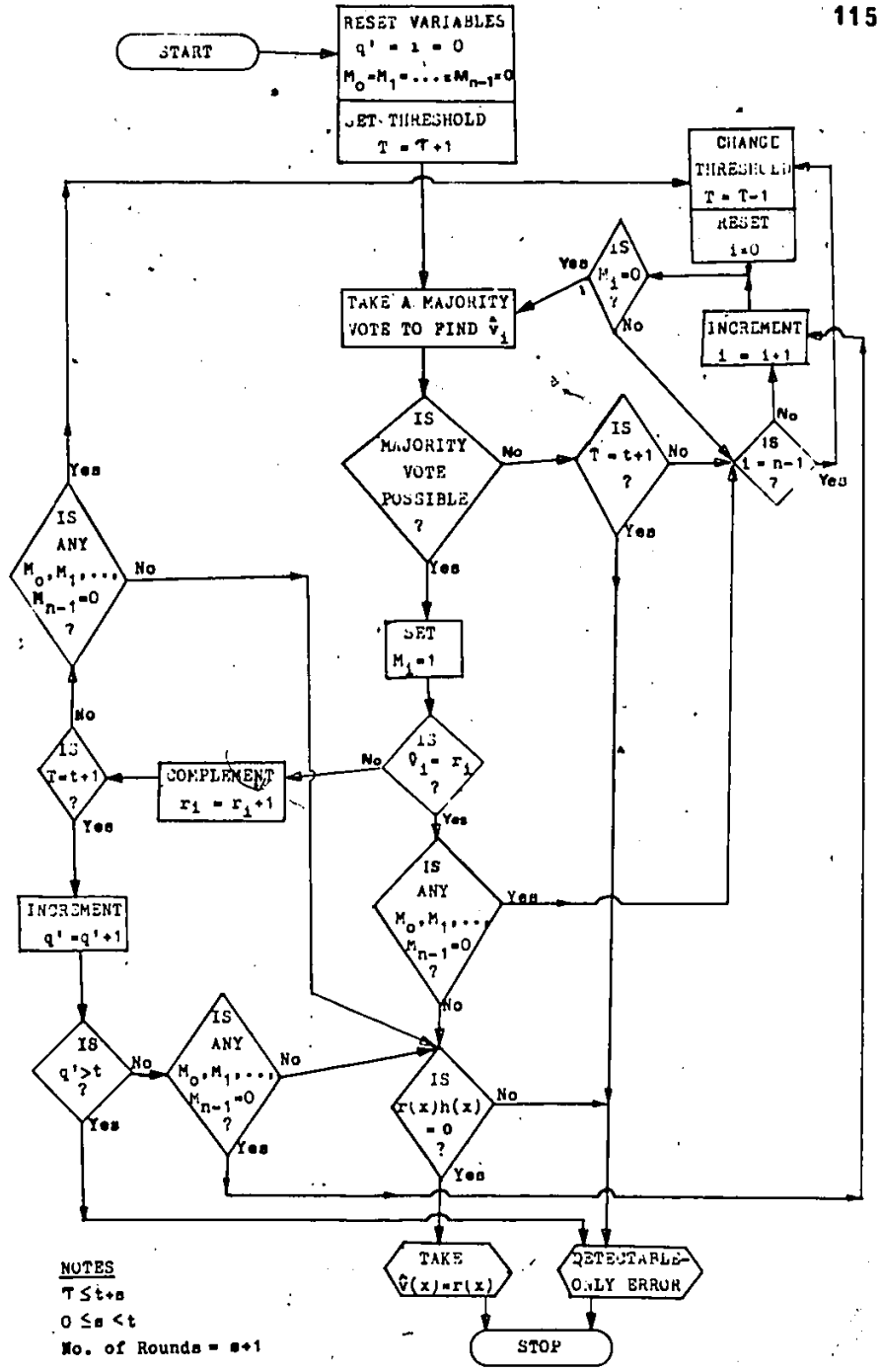
Otherwise for odd d Algorithm D4 will apply which will henceforth follow the same steps as those given for Algorithm D2 . This concludes the description of Algorithms D3 and D4 .

.....

Flowcharts 3 and 4 specify Algorithms D3 and D4 respectively . The variables q , q', R , M_i and i are as previously defined with respect to Algorithms D1 and D2 . If we take the special case where $T_1 = t+2$, then Algorithms D3 and D4 will reduce to Algorithms D1 and D2 respectively, each undergoing up to two rounds of estimation.



Flowchart 3 : Algorithm D3 for Correcting Errors of Weight τ or Less, for Even d , $\tau > t$.



Flowchart 4 : Algorithm D4 for Correcting Errors of Weignt τ or Less, for Odd d , $\tau > t$.

2.7.2 Error-Correcting Capability

In the sequel we examine the capability of Algorithms D3 and D4 to correct an error of weight $\tau \geq t+1$. To this effect we first introduce Lemmas 8 and 9 which will then lead to Theorems 13 to 15. These theorems give the conditions under which an error is correctable, and also indicate an upper bound on the weight of a correctable error. We finally give Decoding Detail D5 which specifies, for a given τ, the required initial majority-logic threshold and the corresponding number of decoding rounds.

LEMMA 8: For the case of codes with d even and d odd respectively, Algorithms D3 and D4 will go through $T_1 - t$ decoding rounds or fewer, where T_1 is the initial majority-logic threshold.

Proof: In the case where $T_1 = t+2$, Algorithms D3 and D4 will reduce to D1 and D2 respectively, requiring two decoding rounds for the correction of errors of weight $t+1$. On the other hand D1 and D2 will not attempt to correct errors of weight $\tau > t+1$.

In the case where $T_1 = t+2+z$, $z \geq 0$, Algorithms D3 and D4 will require $2+z$ decoding rounds for the correction of errors of weight $t+1+z$, whereas D3 and D4 will not attempt to correct errors of higher weight. Thus D3 and D4 will go through no more than $2+z = T_1-t$ decoding rounds. Q. E. D.

LEMMA 9: For the cases of d even and d odd respectively, Algorithms D3 and D4 will correct an error of weight τ only if

$$\tau \leq T_1 - 1 - b \text{ , for } 0 \leq b < T_1 - t \text{ ,}$$

where b is the number of decoding rounds during which no received bit is complemented, and T_1 is the initial majority-logic threshold.

Proof: In order for Algorithm D3 or D4 to correct all the error bits, τ received bits must be complemented. According to Lemma 8 , there are at most $T_1 - t$ decoding rounds with no more than one received bit to be complemented in each round - down to the second last round. During the very last round, however, no more than t received bits can be complemented without causing the decoder to declare a detectable-only error. Thus the greatest number of complemented bits will be $T_1 - 1 - b$.

Q. E. D.

THEOREM 13: Let an error of weight T be such that with respect to Algorithm D3 and D4 , there are b decoding rounds at each of which we have a tie among the d orthogonal estimates of the form of (2.11) for every $i = a_j$, where r_{a_j} is a received error bit which has not been complemented during the preceeding rounds of estimation. Then for the two cases of codes with d even and d odd, Algorithms D3 and D4 will respectively correct every error of this sort, if and only if

$$T \leq T_1 - 1 - b \ , \quad \text{for} \quad 0 \leq b < T_1 - t \ ,$$

where T_1 and b are as defined in Lemma 9 .

Proof: Lemma 9 gives a necessary condition for the correctability of an error of a given weight T . In order to prove Theorem 13 we then need to show that the condition of Lemma 9 is also sufficient. Suppose that the first instance when a received bit r_i is complemented, takes place at the c'th decoding round. Here we have $1 \leq c \leq T - t$, according to Lemma 8 . Since there are only b rounds during which no received bit is complemented, we have

$$1 \leq c \leq b + 1 \ . \qquad (2.66)$$

Combining (2.65) with (2.66) gives a majority-logic threshold at round c or

$$T_c = T_1 - c + 1 \geq T_1 - b \quad . \tag{2.67}$$

This together with Lemma 9 gives $T_c \geq \tau+1$, which ensures correct majority votes during the first c rounds. Thus as soon as one received bit is complemented during round c , the residual error contained in the modified received word $\check{r}(\bar{x})$ will nave a weight of

$$\check{\tau}_{c+1} = \tau - 1 \leq T_1 - 2 - b \quad . \tag{2.68}$$

Concurrently the threshold will be lowered to $T_{c+1} = T_c - 1$ in order to begin round c+1 . According to (2.67) and. (2.66) we have

$$T_{c+1} \geq \tau - b - 1 \geq \check{\tau}_{c+1} + 1 \quad , \tag{2.69}$$

which will again ensure correct majority votes during round c+1 . During this round one of two possible situations will occur;

(a) A received bit is complemented and thus the weight of the residual error is reduced to $\check{\tau}_{c+2} = \check{\tau}_{c+1} - 1$. This means that the new threshold for round c+2 will still be higher than the weight of the residual error;

i.e. $$T_{c+2} \geq \check{\tau}_{c+2} + 1 \quad . \tag{2.70a}$$

(b) No received bit is complemented ; i.e. $\check{\tau}_{c+2} = \check{\tau}_{c+1}$. This situation can only happen if $c \leq b$, since what we will have so far is c rounds during which no received bit has been complemented. Under such condition the majority threshold during round $c+2$ will be given by

$$T_{c+2} = T_1 - c - 1 \geq T_1 - b - 1 \geq \check{\tau}_{c+2} + 1 \qquad (2.70b)$$

which will ensure correct majority votes similar to situation (a) above.

We can thus pursue similar arguments to show that all majority votes taken will be correct throughout the forthcoming rounds with the weight of the residual error being consistently smaller than the majority threshold. When we arrive at the beginning of the last round we have, in the case of odd d , a threshold of $t+1$, implying no more than t residual error bits according to (2.70a) and (2.70b) . These error bits can therefore be complemented during the last round. In the case of even d , on the other hand, the threshold of $t+2$ will not change while going from the second last round to the last round. This does not, however, alter the arguments carried for odd d with respect to the weight of the residual error, implying a similar situation during the last round to that for d odd. On the basis of these arguments we conclude that the condition of Lemma 9 is sufficient for the correct decoding of an error of weight τ.

Q. E. D.

THEOREM 14 : For the case of codes with d even or d odd respectively, the weight τ of an error correctable by Algorithm D3 or D4 is upper-bounded by

$$\tau \leq \tau_{max} \leq 2t - 1$$

provided that the initial threshold T_1 is set at

$$T_1 \geq \tau + 1 \quad .$$

Proof: From the proof of Lemma 9 it is clear that for a given T_1 , the highest value τ_{max} of the error weight is given by $\tau_{max} = T_1 - 1$ where $b = 0$. In this situation one error term r_{a_j} needs to be complemented in each decoding round. However r_{a_j} can only be corrected when its value is in disagreement with at least T_1 out of the $d-1$ estimates $s_{i,1}, s_{i,2}, \ldots, s_{i,d-1}$ in the form of (2.11) for $i = a_j$. This requires that $T_1 \leq d-1$, meaning that $\tau_{max} \leq d-2$. Then the only way to obtain a majority vote on one $\hat{v}_{a_j}$ corresponding to an error term r_{a_j} is for the other error terms to appear in pairs among the $d-1$ orthogonal estimates $s_{i,1}, s_{i,2}, \ldots, s_{i,d-1}$ for $i = a_j$. In other words τ_{max} must be odd which means that $\tau_{max} = d-3$ for d even and $\tau_{max} = d-2$ for d odd, giving $\tau_{max} = 2t-1$ for both cases of odd and even d .

Q. E. D.

THEOREM 15 : Every error $e(x)$ of weight τ that is correctable by Algorithm D3 or D4 , is uniquely decodable in the sense that there is no polynomial $e'(x) \neq e(x)$ of weight $\tau' \leq \tau$ that makes $e'(x) + e(x)$ a codeword.

Proof: We prove Theorem 15 by way of a contradiction. Suppose that there is such a polynomial $e'(x)$ of weight $\tau' \leq \tau$ which makes

$$e(x) + e'(x) = v(x)$$

where $v(x)$ is a non-zero codeword. This means that $e'(x)$ will produce the same majority vote as produced by $e(x)$ on every bit position $i = 0,1, \ldots ,n-1$. Therefore some of the majority votes will be incorrect with respect to $e'(x)$. This is impossible because $e'(x)$ has a weight $\tau' \leq \tau$ which satisfies Theorem 13 for being correctable by Algorithm D3 or D4 . Theorem 15 must therefore be valid. Q. E. D.

In light of Theorems 13 to 15 we can now discuss the behaviour of Algorithm D3 or D4 with respect to an error $e(x)$ of weight τ under each of the following circumstances;

(i) $\tau \leq d - T_1$

(ii) $d - T_1 < \tau \leq t$

(iii) $t < \tau \leq T_1 - 1$

(iv) $T_1 - 1 < \tau \leq n$

It is clear from Lemma 8 and Theorem 14 that

$$t + 1 < T_1 \leq 2t \quad . \tag{2.71}$$

(i) For the case of $\tau \leq d - T_1$, we have in the first round of estimation, a correct majority vote for every bit position $0, 1, \ldots, n-1$. As a result, decoding will be completed after going through one full round with the decoder answer being that $\hat{v}(x) = \check{r}(x)$.

(ii) For the case of $d - T_1 < \tau \leq t$, we may or may not have a majority vote for a given position during a given m'th round, for which we have a majority threshold $T_m > d - t$. However, if for any given recieved bit r_i a tie has occurred during each one of the first m rounds, a majority vote will be finally possible for every r_i once the majority threshold goes down to $d-t$. We may note that the majority threshold will be equal to $d-t$ at the last round of Algorithm D4 and the second last round of Algorithm D3. Regardless of which round will provide a majority vote on any given r_i, such a vote

will always be correct, since the threshold is always greater than the error weight τ . Thus at the end of decoding, the resulting word $\check{r}(x)$ is equal to the transmitted codeword $v(x)$.

(iii) In the case where $t < \tau \leq T_1 - 1$, the decoder can only correct those errors that are uniquely decodable as indicated in Theorem 15 . The mechanism by which such errors are corrected is already covered in the proof of Theorem 13 . Those errors which are not correctable by Algorithm D3 or D4 , will either lead to a detectable-only situation or will cause the decoder to estimate another error $\hat{e}(x) \neq e(x)$ of smaller weight such that $\hat{e}(x) + e(x)$ is a codeword. Obviously the decoder cannot do any better than choosing $\hat{e}(x)$ which is a coset leader of the same coset in which $e(x)$ is an element.

(iv) Finally in the case where $\tau > T_1 - 1$, no error can be correctly estimated by Algorithm D3 or D4 , resulting in a similar situation to the case of noncorrectable errors discussed in (iii) above.

Within the context of the discussion carried so far, we conclude this section by mentioning the following;

DECODING DETAIL D5 : When Algorithms D3 and D4 are to be used with codes of d even and d odd respectively, for the correction of uniquely-decodable errors of weight

$$\tau \leq t + s ,$$

we first set the initial majority threshold of Algorithms D3 and D4 at

$$T_1 = t + s + 1$$

and then go through $s+1$ or fewer rounds of majority-logic estimation accordingly, where

$$0 \leq s < t .$$

2.8 Summary of Main Results in Chapter 2

In this chapter we examined the capability of 1-step majority-logic decoding in correcting errors of weight $T > t$, for the case of t-error-correcting binary cyclic codes.

We first developed Decoding Strategy C which is aimed at minimizing the decoded bit error rate when dealing with errors of weight $t+1$. To facilitate the estimation of decoded bit error rate we defined R_{t+1} as the average fraction of erroneously decoded bits over length n due to an error of weight $t+1$. Then on the basis of the above strategy, we gave in Theorem 1 two expressions for R_{t+1} for the two cases of odd and even minimum distance.

In an attempt to improve the decoding performance by reducing the value of R_{t+1}, we considered two-round Decoding Algorithms D1 and D2 for the two cases of codes with d even and d odd respectively. In both algorithms the decoder takes a majority vote among d orthogonal estimates on any transmitted bit v_i such that at least T estimates must agree to produce a decoding answer where $T > d/2$ being the majority threshold, and $i = 0,1,..,n$ with n being

the code length. Algorithm D1 requires a fixed majority threshold of $T = t+2$ for the case of d even. For the case of d odd, Algorithm D2 requires two thresholds $T_1 = t+2$ and $T_2 = t+1$ during the first and second rounds respectively. We noted that no more than two decoding rounds were needed for Algorithms D1 and D2, and that only one round of estimation would be required for correcting an error of weight $\tau \leq t$ in the case of D1 or $\tau \leq t-1$ in the case of D2.

The main results of analyzing the performance of these two decoding algorithms are given in Theorems 2, 4 and 6 for Algorithm D1, and Theorems 9, 11 and 12 for Algorithm D2. We found the results to be similar for both D1 and D2. A combined summary of such results is given below.

Theorems 2 and 9 state that Algorithms D1 and D2 can correct every error of weight $t+1$ that contains at least one error position a_j such that when taking a majority vote on v_i for $i=a_j$, the error affects the value of at most $d-t-2$ orthogonal estimates.

Theorems 4 and 11 state that Algorithms D1 and D2 can only correct an error of weight $t+1$ if such

an error is uniquely decodable in the sense that it does not form part of a codeword having either a weight $2t+1$ for the case of d odd, or a weight $2t+2$ for both cases of odd and even d.

Theorems 6 and 12 give a lower bound B_{t+1} for the number Q_{t+1} of errors of weight $t+1$ that are correctable by Algorithms D1 and D2 as

$$Q_{t+1} \geq B_{t+1} \triangleq \binom{n}{t+1} - \frac{n}{t+1}\binom{2t+1}{t}\left[\frac{n-1}{2t+1}\right]^t .$$

Numerical values of this upper bound are listed in Table I for some 1-step majority-logic codes of even minimum distance and relatively short length n. In order to gain an insight of how near, the lower bound B_{t+1} is to the actual value of Q_{t+1}, we estimated Q_{t+1} for four example codes after a detailed examination of the properties of their orthogonal check-sums.

For the two classes of difference-set codes and M-sequence codes, we used certain properties of the orthogonal check-sums in proving Theorems 7 and 8, and hence showing that Algorithm D1 is optimum in its

capability of correcting every uniquely-decodable error of weight $t+1$. On the basis of this , we estimated the exact values of Q_{t+1} for two more M-sequence codes by using previously reported results on the coset enumeration of such codes.

Finally we examined the capability of 1-step majority-logic decoding in correcting errors of weight greater than $t+1$ by going through more than two rounds. We considered the multiple-round decoding Algorithms D3 and D4 for the case of d even and d odd respectively. These two algorithms are in fact generalizations of the two-round Algorithms D1 and D2 .

Theorem 13 prescribes the conditions under which an error is correctable by Algorithm D3 and D4 . Theorem 14 gives an upper bound for the weight τ of such an error to be

$$\tau \leq \tau_{max} \leq 2t - 1$$

provided that the initial threshold T_1 is set at

$$T_1 \geq \tau + 1 \quad .$$

Theorem 15 shows that every error $e(x)$ of weight τ that is correctable by Algorithm D3 or D4 , is uniquely decodable in the sense that there is no polynomial $e'(x) \neq e(x)$ of weight $\tau' \leq \tau$ that makes $e(x)+e'(x)$ a codeword.

We then see from Decoding Detail D5 that we need $s+1$ or fewer rounds of majority-logic estimation for the correction of uniquely-decodable errors of weight $\tau' \leq t + s$ where $0 \leq s < t$ after setting the initial majority threshold at

$$T_1 = t + s + 1 \quad .$$

CHAPTER 3

IMPROVEMENT IN ERROR PERFORMANCE WITH CORRECTION OF (t+1)-ERRORS

3.1 General

In this chapter we derive certain expressions to evaluate the error performance of coded systems and to determine the improvement achieved with the correction of some errors of weight $t+1$. We then use the numerical results given in Table I, in order to graphically illustrate such error performance.

First we consider the case of memory systems and then analyze in more details the case of random communication systems where we take into account the system trade-offs among transmitted power, bandwidtn utilization and bit error rates. For the latter case we shall derive explicit expressions for the measures on coding error performance already defined in chapter 1; i.e. the Net Coding Gain (NCG) and Net Error Improvement (NEI). The graphical results on these measures are given in this chapter only to serve as an illustration of how to estimate the improvement in decoding performance achieved by Algorithm D1 over a bounded-distance decoder.

For the sake of convenience we repeat below some of the definitions and expressions given earlier in chapter 2 .

p = bit error rate at the decoder input.

p_o = bit error rate at the decoder output after an error correction is attempted.

R_τ = average fraction of erroneously decoded bits over length n , corresponding to an error pattern of weight τ .

$$p_o = \sum_{\tau=t+1}^{n} \binom{n}{\tau} p^\tau (1-p)^{n-\tau} R_\tau \quad . \qquad (3.1)$$

3.2 Performance in Memory Systems

In the case of memory systems [39] the improvement in error performance obtained with Algorithm D1 or D2 over that with a bounded-distance decoder can be directly measured in terms of the reduction ratio ρ in the value of p_o. In other words,

$$\rho = \frac{p_o \text{ (with bounded-distance decoding)}}{p_o \text{ (with Algorithm D1 or D2)}} \qquad (3.2)$$

It is possible to simplify the analysis and derive an expression for ρ independant of p if we make the assumptions that $p \ll R_{t+1}/n$, to get an approximation of p_o as

$$p_o \simeq \binom{n}{t+1} p^{t+1} (1-p)^{n-t-1} R_{t+1} . \qquad (3.3)$$

The value of R_{t+1} depends on the error-correction capability of the decoder. According to Theorem 1, we have for the case of even d,

$$R_{t+1} = \frac{t+1}{n} \left[1 - \frac{Q_{t+1}}{\binom{n}{t+1}} \right], \qquad (3.4)$$

where Q_{t+1} is the number of errors of weignt t+1 that are correctable by the decoder. Thus a bounded-distance decoder will have $Q_{t+1} = 0$ giving

$$R_{t+1} = \frac{t+1}{n} \quad . \tag{3.5}$$

From (3.2) to (3.5) it is clear that

$$\frac{1}{\rho} \simeq 1 - \frac{Q_{t+1}}{\binom{n}{t+1}} \tag{3.6}$$

for $p << R_{t+1}/n$.

We list in Table II the approximate values of ρ for the same codes given before in Table I .

TABLE II

ERROR-REDUCTION RATIO (ρ) IN MEMORY SYSTEM DUE TO (t+1)-ERROR CORRECTION

$$\rho \doteq 1 \Big/ \left[1 - \frac{2_{t+1}}{\binom{n}{t+1}} \right]$$

	(n,k)	d	Lower Bound	Actual Value
Difference-Set Codes	(21,11)	6	1.2	1.2
	(73,45)	10	2.0	3.7
	(273,191)	18	6.4	——
	(1057,813)	34	71.4	——
M-Sequence Codes	(7,3)	4	1.0	1.0
	(15,4)	8	1.3	1.4
	(31,5)	16	2.5	20
	(63,6)	32	9.4	10^4
	(127,7)	64	143	——
	(255,8)	128	3.3×10^4	——
BCH Code	(15,6)	6	1.2	1.2
Projective-Geometry Codes	(85,24)	22	7.5	——
	(585,184)	74	1.3×10^4	——

3.3 Performance in Communication Systems

For the case of communication systems we take the idealized model of binary symmetric channel as discussed earlier in the introduction of this tnesis. In Figure 1 two coding performance measures were illustrated; i.e. Net Coding Gain (NCG) and Net Error Improvement (NEI) . A close examination of this figure reveals that

$$NCG = \frac{E(p_o)}{\varepsilon_u} \tag{3.7}$$

and

$$NEI = \frac{P(\varepsilon_u)}{p_o} \tag{3.8}$$

where $p = P(\varepsilon)$ and $\varepsilon = E(p)$ are the two reciprocal functions, either of which will specify the relationship between ε and p . We can now obtain explicit expressions for NCG and NEI by combining (2.1) with (3.7) and (3.8) respectively as follows.

$$NCG = \frac{\sum_{\tau=t+1}^{n} \binom{n}{\tau} p^{\tau}(1-p)^{n-\tau} R_{\tau}}{\varepsilon} \tag{3.9}$$

and

$$NEI = \frac{P(\varepsilon)}{\sum_{\tau=t+1}^{n} \binom{n}{\tau} p^{\tau}(1-p)^{n-\tau} R_{\tau}} \qquad (3.10)$$

where $p = P(\frac{k}{n} \varepsilon)$ in accordance with the notation of Figure 1.

In order to obtain some illustrative graphical results on NCG and NEI , we shall consider the performance of Algorithm D1 with respect to some of the even-weighted codes listed in Table I . In doing this we shall make the assumption that $p << R_{t+1}/n$. As we indicated in Remark 6 , the fraction of correctable errors of weight $t+1$ for some of the codes of Table I , is very close to "one" . This makes the performance of Algorithm D1 more dependant on the occurrence of errors of weight $t+2$, especially in the case where p is in the same order of magnitude as R_{t+1} . Since no errors of weight $t+2$ will be corrected by D1 , and since p is assumed very small, we can neglect the effect of errors of weight $t+3$ and approximate (3.9) and (3.10) to the following;

$$NCG \simeq \frac{E\left[\binom{n}{t+1} p^{t+1}(1-p)^{n-t-1} R_{t+1} + \binom{n}{t+2} p^{t+2}(1-p)^{n-t-2} R_{t+2}\right]}{\varepsilon} \qquad (3.9a)$$

and

$$NEI \simeq \frac{P(\varepsilon)}{\binom{n}{t+1} p^{t+1}(1-p)^{n-t-1} R_{t+1} + \binom{n}{t+2} p^{t+2}(1-p)^{n-t-2} R_{t+2}} \qquad (3.10a)$$

The value of R_{t+1} or its upper bound can be readily obtained by substituting in Theorem 1 (expression (3.4) above) the value of $Q_{t+1}/\binom{n}{t+1}$ or its lower bound respectively as taken from Table I. On the other hand, the upper bound for R_{t+2} is given in Theorem 16 below. When we apply the upper bounds of R_{t+1} and R_{t+2} in (3.9a) and (3.10a) we can respectively get lower bounds for NCG and NEI.

THEOREM 16 : When Strategy C is used in conjunction with Algorithm D1 for the case of a systematic 1-step majority-logic code V of even minimum distance $d = 2t+2$, the average fraction R_{t+2} of erroneously decoded bits over length n due to an error pattern of weight $t+2$, is upper-bounded by

$$R_{t+2} \leq \frac{2t+2}{n} \quad . \qquad \text{(T16)}$$

Proof: Since Algorithm D1 does not correct any error of weight $\tau > t+1$, an error of weight $t+2$ will always produce an erroneous answer in one of two possible ways.

(a) When such an error is part of a codeword of weight $w \leq 2t+3$, it will cause the decoder to estimate an error of weight $w - (t+2)$ or less. However, since the code is even-weighted we have $w = 2t+2$. As a result, there will be $2t+2$ bits in error within the n-bit received word.

(b) When the error is not part of a codeword of weight $2t+2$, it will cause the decoder to leave the received word unchanged in accordance with Strategy C. As a result, there will be $t+2$ bits in error within the n-bit received word.

The upper bound of (T16) will then be obtained by taking the worst case of (a) above. Q. E. D.

3.4 An Upper Bound for Binary AWGN Channels

In the following we examine the performance of coding in binary channels with Additive White Gaussian Noise (AWGN) using either one or none of PSK, FSK or ASK modulation and either coherent or non-coherent detection [40,41]. The relationship between p and ε for each of these cases is given below;

I. $$p = \frac{1}{2}\,\mathrm{erfc}\sqrt{\varepsilon}\ , \tag{3.11}$$

for polar baseband (non-return-to-zero), and for phase-shift keying (PSK) with coherent detection.

II. $$p = \frac{1}{2}\,\mathrm{erfc}\sqrt{\frac{\varepsilon}{2}}\ , \tag{3.12}$$

for amplitude-shift keying (ASK) with coherent detection, and for frequency-shift keying (FSK) with coherent detection.

III. $$p = \frac{1}{2}\exp(-\frac{\varepsilon}{2})\ , \tag{3.13}$$

for FSK with non-coherent detection.

IV. $$p \simeq \frac{1}{2}\left[1+\frac{1}{\sqrt{2\pi\varepsilon}}\right]\exp(-\frac{\varepsilon}{2})\ , \tag{3.14}$$

for ASK with non-coherent detection.

Here $\mathrm{erfc}(y)$ is the complementary error function of y.

Next we examine the behaviour of NCG with decreasing values of p. To do this we use the approximation

$$\operatorname{erfc}(x) = \frac{\exp(-y^2)}{y\sqrt{\pi}} \text{ , for large } y \text{ .} \qquad (3.15)$$

If we combine (3.15) with (3.11) to (3.14) respectively and then apply the results in (3.9) to take the limit of NCG as p approaches zero we find that

$$\lim_{p \to 0} \text{NCG} = B_g = \frac{k}{n}(t+1) \text{ .} \qquad (3.16)$$

where NCG increases with decreasing p. The simplicity of the upper bound B_g on the Net Coding Gain is remarkable. This bound can serve as a useful and simple measure in determining the ultimate capability of a given (n,k,t) code in improving the performance of an AWGN binary channel. It is obvious that codes giving a value of B_g smaller than "one" can be readily discarded from further consideration for an implementation with such a channel.

3. Graphical Results

In this section we provide some illustrative results on the performance of the 1-step majority-logic codes listed in Table I of chapter 2, by using the derivations already given in this chapter. Of particular interest to the main theme of this thesis is the improvement in coding performance due to the correction of errors of weight $t+1$, over that of a bounded-distance decoder.

First we estimate the upper bound B_g on the Net Coding Gain given by (3.16). We note that for the case of M-sequence codes we have $n = 2^m-1$, $k = m$ and $t = 2^{m-2} - 1$, which gives

$$B_g = \frac{m}{4}\left(1+\frac{1}{n}\right) .$$

This means that the useful value of $B_g > 1$ can only be achieved with $m > 4$; i.e. $n > 15$. This is just one example to illustrate the usefulness of the performance measure B_g in discarding undesirable codes from further consideration for the AWGN channel. Figure 5 plots B_g versus bandwidth expansion factor n/k for some of the codes listed in Table I. This graph shows a distinct advantage of longer codes within a given class, where we both obtain a higher value for B_g and a lower bandwidth expansion factor. Since the value of B_g for a given (n,k,t) code is independant of what decoding technique is used, we cannot use B_g for comparing between

the performance of Algorithm D1 and that of a bounded-distance decoding algorithm.

In order to present some illustrative graphical results on NCG and NEI , we choose the case of non-coherent FSK because of its analytical simplicity. The results plotted in Figure 6 are obtained by combining (3.13) with (3.9a) and (3.10a) respectively, where a value of $p = 10^{-5}$ has been chosen as an example. The results are not representative, however, since they largely depend on the type of modulation scheme and the value of p . Nevertheless, our exercise in obtaining such results does serve the purpose of giving an example on how to use the data provided in Table I to obtain a comparative evaluation of coding performance in communication channels, with and without the use of non-bounded-distance decoding techniques.

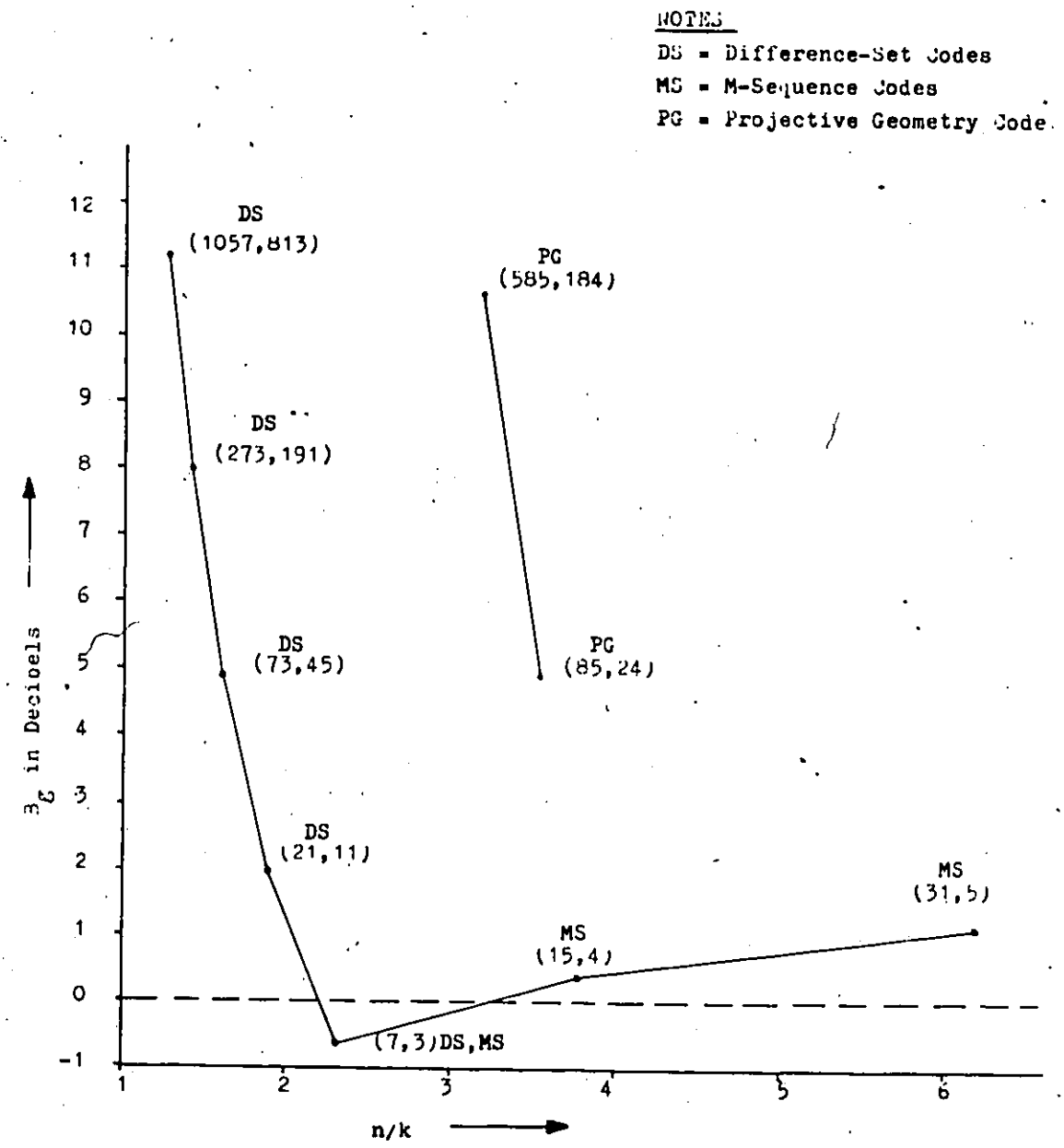


Figure 5 : Upper Bound (B_g) on Net Coding Gain in Binary AWGN Channels.

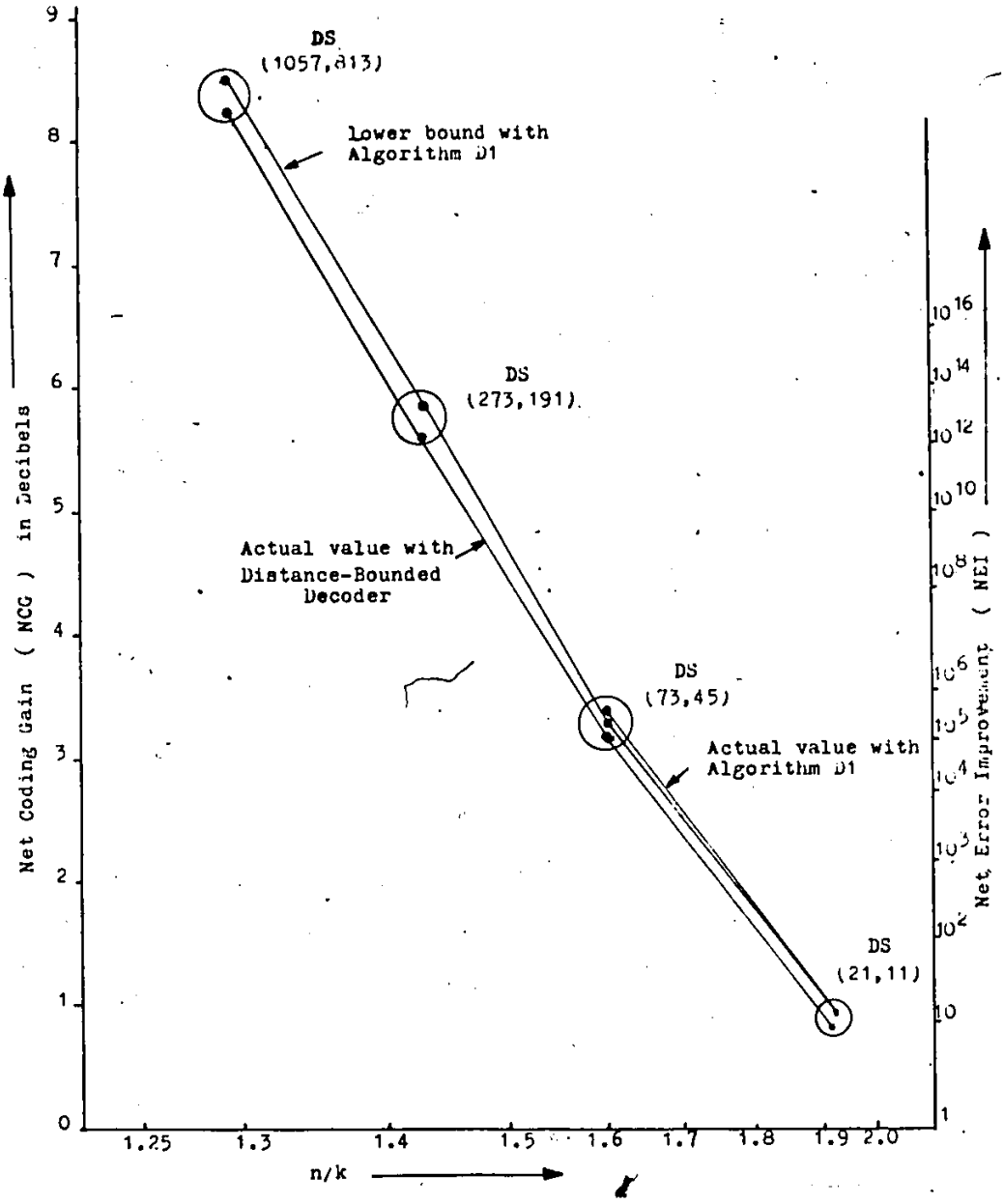


Figure 6 : Coding Performance in Binary Non-Coherent FSK Channels at $p = 10^{-5}$.

CONCLUDING REMARKS

The main empnasis in this thesis has been on analyzing the capability of 1-step majority-logic decoding in correcting errors of weight $\tau > t$ in general and $\tau = t+1$ in particular. We saw in chapter 2 that by using a code in its systematic form, we can minimize the decoding error probability with respect to errors of weight $t+1$ by only correcting those errors that are uniquely decodable in the sense of not forming part of a codeword of weight $2t+1$ or $2t+2$. On this basis we gave Decoding Algorithms D1 and D2 for the two cases of even and odd minimum distance d respectively. These algoritnms are capable of correcting uniquely-decodable errors of weight $t+1$, and undergo up two round of majority-logic estimation.

We discussed the performance of Algorithms D1 and D2 and developed lower bounds on the number Q_{t+1} of correctable errors of weight $t+1$. We noted that D2 corrected the same set of errors with respect to an (n,k) code of $d=2t+1$, as that correctable by D1 with respect to the even-weighted subset $(n,k-1)$ code of $d=2t+2$. We also noted that no more than two decoding roundswere needed for Algorithms D1 and D2 and that

only one round of estimation would be required for correcting an error of weight $\tau \leq t$ in the case of D1 or $\tau \leq t-1$ in the case of D2 . This implies that the average decoding delay in D1 or D2 , is only slightly higher than that in a bounded-distance decoder.

For the two specific cases of the (15,6) BCH code and the (21,11) difference-set code, both of $d=6$, we saw that no more than one decoding round would be needed for the correction of errors of weight $t+1=3$. There, the capability of correcting errors of weight $t+1$ is found inherent in the single-round version of a 1-step majority-logic decoder using feedback.

We observed that Algorithms D1 and D2 were optimum in the case of the (15,7) BCH code and its subset (15,6) code, both of $t=2$, in being able to correct every triple error that is uniquely decodable.

After a detailed examination of the properties of orthogonal check-sums for the two classes of difference-set codes and M-sequence codes, we estimated the exact value of Q_{t+1} for $t = 2, 3$ and 4 . The analysis for the cases of $t>4$ is subject to further investigation where computer simulation may be required. Nevertheless, we used the properties of orthogonal check-sums to show

that Algorithm D1 is optimum in its capability of correcting every uniquely-decodable error of weight $t+1$ for the two classes of difference-set codes and M-sequence codes. On the basis of this, we estimated the exact values of Q_{t+1} for two more M-sequence codes by using previously reported results on the coset enumeration of such codes .

The analysis carried in this thesis is based on Decoding Strategy C which requires to use a code in its systematic form and will correct only those errors that are uniquely decodable, whereas leaving the received word unchanged for the case of a detectable-only error. However, certain applications might require the use of a code in its non-systematic form with the need to correct as many (t+1)-errors as possible regardless whether they are uniquely decodable or not. Under such circumstances we could amend Algorithm D1 by lowering the decoding threshold to $T_2 = t+1$ during the second round, and also amend both algorithms D1 and D2 by removing the requirement of having to correct no more than t error bits during the second round. Such amendments would obviously increase the number of correctable (t+1)-errors beyond those estimated in our analysis, to cover some errors which are not uniquely decodable. As an example, in the case of the (15,7)

BCH code of $d=5$, [18] has shown that 37 triple errors containing the zeroth position can be corrected by a single-round decoder using feedback. This is much higher than the number of uniquely-decodable triple errors containing the zeroth position which is only 13 as we saw in Example 1.

A tabulation of the values of Q_{t+1} and its lower bound indicates that for increasing code length n, the correctable fraction of errors of weight $t+1$ approaches "one" which causes the decoding error probability to be increasingly more dominated by the occurrence of errors of weight $t+2$ for larger n. It is therefore interesting to investigate the capability to correct errors of weight $t+2$ or greater. To do this we generalized Algorithms D1 and D2 in the form of Algorithms D3 and D4 respectively, where D3 and D4 can correct an error of weight up to $\tau_{max} = 2t-1$ by going through t or fewer rounds of estimation. As an example, if errors of weight $t+2$ are to be corrected we must start the decoding algorithm with a majority threshold of $T_1 = t+3$ and then follow the same decoding principles as in Algorithms D1 and D2. We then lower the threshold to $T_2 = t+2$ as soon as one received bit is complemented and go through another round as if it is the first round of Algorithm D1 or D2, and so on.

Analyzing the performance of Algorithms D3 and D4 with respect to the number of correctable errors of weight $T > t+1$ would clearly be more complicated than the analysis we gave for $T \leq t+1$. This topic warrants further investigation, especially for larger n where the large majority of errors of weight $t+1$ are correctable, causing the decoder performance to be sensitive to those errors of greater weights.

In chapter 3 we derived certain expressions to evaluate the error performance of coded systems and to determine the improvement achieved with the correction of errors of weight $t+1$. We then used the numerical results given in chapter 2 in order to graphically illustrate the error performance.

First we considered the case of memory systems and gave an approximation of the ratio of error rate reduction achieved by Algorithm D1 over a bounded-distance decoder. We then considered the case of random communication systems and gave a more detailed analysis taking into account the trade-offs among transmitted power, utilized bandwidth and bit error

rate. We derived explicit expressions for the Net Coding Gain (NCG) and Net Error Improvement (NEI). We also found that the value of NCG is theoretically limited by the simple upper bound $B_g = k(t+1)/n$ for the case of binary AWGN channels regardless of what modulation scheme is utilized.

We used the above-mentioned derivations on coding performance to provide some graphical results on 1-step majority-logic decoding with and without the correction of (t+1)-errors. Since both NEI and NCG are functions of the channel characteristics, we chose an example of non-conerent FSK system to illustrate the behaviour of NEI and NCG . On the other hand, the analytical results of chapter 3 provide some useful tools for evaluating the coding performance of a given t-error-correcting code when being considered for a particular memory system or communication system. These results are also useful for determining the amount of improvement in error performance that can be achieved with the correction of (t+1)-errors.

REFERENCES

[1] I.M. Jacobs, "Practical applications of coding", IEEE Trans. Inform. Theory, Vol. IT-20, pp. 305-310, May 1974.

[2] D. Wiggert, Error-Control Coding and Applications, Dedham, MA: Artech House, 1978.

[3] R.T. Chien, "Memory error control: beyond parity", IEEE Spectrum, pp. 18-23, July 1973.

[4] L. Levine and W. Meyers, "Semiconductor memory reliability with error detecting and correcting codes", Computer, pp. 43-50, October 1976.

[5] M.P. Ristenbatt, "Alternatives in digital communications", Proc. IEEE, Vol. 61, No. 6, pp. 703-721, June 1973.

[6] R.T. Chien, "Block-coding techniques for reliable data transmission", IEEE Trans. Comm. Tech., Vol. COM-19, pp. 743-751, October 1971.

[7] M.W. Williard, "Introduction to redundancy coding", IEEE Trans. Vehicular Tech. Vol VT-27, No. 3, pp. 86-98, August 1978.

[8] R.C. Montgomery, "Simple hardware approach to error detection and correction", Computer Design, pp. 109-118, November 1978.

[9] H.O. Burton and D.D. Sullivan, "Errors and error control", Proc. IEEE, Vol. 60, No. 11, pp. 1293-1300, November 1972.

[10] J.D. Ralphs, "Limitations of error-detection coding at high error rates", Proc. IEEE, Vol. 118, No. 3/4, March/April, 1971, pp. 409-416.

[11] J.A. Heller and I.M. Jacobs, "Viterbi decoding for satellites and space communications", IEEE Trans. Comm. Tech., Vol. COM-19, No. 5, pp. 835-849, October 1971.

[12] L.N. Lee, "Error-Correction coding for commercial satellite channels", ICC-81 Conference Records, paper 02.5, June 1981.

[13] P.J. Mabey, "Mobile radio data transmission - coding for error control", IEEE Trans. Vehicular Tech., Vol. VT-27, No. 3, pp. 99-109, August 1978.

[14] P.E. Allard, V.K. Bhargava and G.E. Séguin, "Realization, economic and performance analysis of error-correcting codes and ARQ systems for broadcast Telidon and other Videotex transmission", DOC Contract Report, June 15, 1981.

[15] "North American Broadcast Teletext Specification, CBS, June 1981.

[16] F. Kuo and N. Abramson, editors, Computer Communications, Prentice Hall, 1973.

[17] R. Swanson, "Matrix technique leads to direct error code implementation", Computer Design, pp. 101-108, August 1980.

[18] E.L. Wall, "Applying the Hamming code to microprocessor-based systems", Electronics, pp. 103-110, Nov. 22, 1979.

[19] A. Heimlich and J. Korelitz, "Memory finds and fixes errors to raise reliability of microcomputers", Electronics, pp. 168-172, January 3, 1980.

[20] F.J. MacWilliams and N.J.A. Sloane, "The Theory of Error-Correcting Codes", Amsterdam, Netherlands: North-Holland, 1977.

[21] W.W. Peterson and E.J. Weldon, "Error-Correcting Codes", 2nd ed., Cambridge, MA: MIT Press, 1972.

[22] S. Lin, "An Introduction to Error-Correcting Codes", Englewood Cliffs, N.J.: Prentice-Hall, 1970.

[23] J.H. Blythe and K. Edgecombe, "Net coding gain of error-correcting codes", Proc. IEE, Vol. 122, No. 6, pp. 609-614, June 1975.

[24] V.K. Bhargava, "Coding gain of a class of error-correcting codes", IEEC & E '77 Proceedings, Paper 77024, Session 2, pp. 26-27, 1977.

[25] A.J. Viterbi, "Convolutional codes and their performance in communication systems", IEEE Trans. Comm. Tech., Vol. COM-19, pp. 751-772, October 1971.

[26] Van der Horst, J.A. and T. Berger, "Complete decoding of triple-error-correcting binary BCH codes", IEEE Trans. Inform. Theory, Vol. IT-22, pp. 138-147, March 1976.

[27] M. Goldberg, "Easily decoded error-correcting codes and techniques for their generation", Ph.D. dissertation, Dept. of Elec. Eng., Imperial College, London, England, January 1972.

[28] Z.A. Muscati and S.G.S. Shiva, "On the performance of certain block error-correcting codes in binary channels", Proceedings of 17th Annual Allerton Conference on Communications, Control, and Computing, pp. 810-818, October 1979.

[29] Z.A. Muscati and S.G.S. Shiva, "1-step majority-logic decoding with two rounds of estimation", ICC-81 Conference Records, paper 65.7, June 1981.

[30] R.L. Townsend and E.J. Weldon, "Self-orthogonal quasi-cyclic codes", IEEE Trans. Inform. Theory, Vol. IT-13, pp. 183-195, April 1967.

[31] V.K. Bhargava, "Some codes of Rahman and Blake for computer applications", IEEE Trans. Computers, Vol. C-27, pp. 765-767, August 1978.

[32] E.J. Weldon, "Difference-set cyclic codes", B.S.T.J., pp. 1045-1055, September 1966.

[33] N.J.A. Sloane and R.J. Dick, "On the enumeration of cosets of first order Reed Muller codes, ICC-71 Proceedings, paper 36, June 1971.

[34] C. Leferriere and S.G.S. Shiva, "On 1-step majority-logic decoding", Proc. IEE, Vol. 124, No. 6, pp. 527-528, June 1977.

[35] S.G.S. Shiva and S.E. Tavares, "On binary majority-logic decodable codes", IEEE Trans. Inform. Theory, Vol. IT-20, pp. 131-133, January 1974.

[36] C.R.P. Hartmann, J.B. Ducey and L.D. Rudolph, "On the structure of generalized finite-geometry codes", IEEE Trans. Inform Theory, Vol.IT-20, pp. 737-745, November 1976.

[37] E. Guevara, et al, "Even-weighted codes for error detection", Electronics Letters, Vol. 12, No. 10, May 1976.

[38] E.R. Berlekamp, Algebraic Coding Theory, New York, N.Y.: McGraw-Hill, 1968.

[39] L. Levine and W. Meyers, "Semiconductor memory reliability with error detecting and correcting codes", Computer, pp. 43-50, October 1976.

[40] J.J. Spilker Jr., Digital Communications by Satellites, Englewood Cliffs, N.J.: Prentice-Hall, 1977.

[41] Z.P. Peyton Jr., Communication System Principles, Reading, MA: Addison-Wesley, 1976.