SHORTENED CYCLIC BINARY CODES FOR BURST ERROR CORRECTION

by

Tony Zeitoun

Submitted to the Department of Electrical Engineering

in partial fulfilment of the requirements for the degree

of

Master of Science

Department of Electrical Engineering

Faculty of Pure and Applied Science

University of Ottawa

Ottawa, Ontario

June, 1968

ABSTRACT

The thesis is concerned with shortened cyclic codes as applied to burst error correction. They are referred to as SBEC (Shortened Burst Error Correcting) codes. Specifically it will be shown that if the word length n_s of an SBEC code is made equal to $2g + 1 - l_c$, where l_c is the maximum correctable burst length for which the code is considered, then decoding can be accomplished in just two steps, as compared to present decoding techniques of cyclic codes, where the average number of steps is $\frac{n}{2}$, assuming that all correctable errors are equiprobable. The price to be paid in adopting this simple decoding scheme is a degradation in the transmission rate of these SBEC codes, to a value very nearly equal to $\frac{1}{2}$. As regards the burst error correcting capabilities, nothing is of course lost by shortening a cyclic code.

ACKNOWLEDGEMENTS

It was Prof. S. G. S. Shiva who first introduced me to the subject of Coding Theory, and it was he who suggested the problem. His invaluable help through numerous discussions, comments, and constructive criticism, his continuous and sincere encouragement cannot go by unmentioned. For all that I thank him wholeheartedly.

I would like to thank my colleagues R. Roy and P. E. Allard for their helpful suggestions; in particular, G. Seguin who with his perspicacity, provided the atmosphere for numerous heated but scintillating conversations.

My thanks are also directed towards Miss Marion Radford for her patience in meticulously typing such a neat manuscript.

And last but not least, I am grateful for the financial assistance of the National Research Council.

TABLE OF CONTENTS

INTRODUCTION

In an ideal communications system, the symbol that comes out of the decoder should match the symbol that entered the encoder. In a real system (Fig. 1), however, errors occur frequently. It is the purpose of codes to detect and perhaps correct such errors. These codes are not designed to correct every conceivable pattern of errors, but rather the most probable ones.

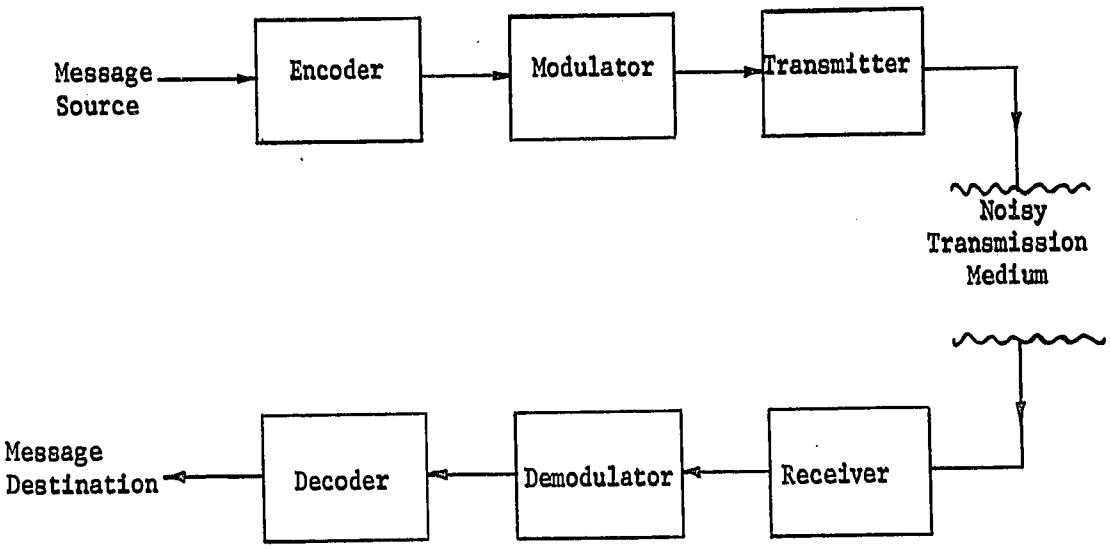


Fig. 1 Block Diagram of a Real Communications or Storage System

The construction of most error correcting codes is based on the assumption that errors occur independently of digit position. In many instances on the other hand, errors do not occur independently. On telephone lines and on magnetic storage systems, for example, errors occur predominantly in bursts due to the prevalence of impulse noise. High error probability then is confined to the duration of the impulse, the error probability being negligible at other times. Similarly, on certain types of fading radio circuits, the signal is buried in noise for short periods of time. Ordinary "static" caused by thunderstorms frequently

displays such burst-like characteristics. Consequently, codes for correcting bursts of errors are required and some remarkably good ones* have been found.

Coding is simply a rearrangement of the information to be transmitted into some other format at the transmitter in accordance with some predetermined conventions. Some of the important reasons [1] for coding are:

- To increase the information capacity of the system.
- To enable error detection, and possibly error correction, to be performed.
- To alleviate transmission problems.

Ideally, the above reasons should lead to one and only one encoding scheme. In practice, though, this one ideal scheme cannot be realized. As a result, there are now a number of coding schemes in use that satisfy one or more of the above mentioned reasons.

If coding is required to combat the errors that occur during transmission, then decoding is required to recover the transmitted information by finding these errors. Consequently the reverse process of coding must be provided at the receiving end, and this is where efficient and simple decoding methods play a major part in communications systems. It is not enough to have a good encoding scheme, we must also try to have a simple and reliable decoding technique whenever possible.

The purpose of this thesis is to describe such a technique for a certain type of errors. Specifically, it will be shown that by modifying the existing BEC (Burst Error Correcting) binary codes, we can arrive at

* Fire Codes, Abramson Codes.....

an efficient and extremely simple decoding technique which can be realized by a simple sequential circuit and which can be easily programmed on a digital computer. The price to be paid for this easy method is a certain degradation in transmission rate.

The thesis is orgainized into three chapters. Chapter 1 is a general survey of codes to which this thesis is related. Chapter 2 discusses the concepts of error control and decoding which lead to chapter 3 where the decoding technique is discussed in full. The main points of chapter 3 have also been covered in a separate technical report [2]. Finally, a digital computer program which realizes the decoding procedure is included in the appendix.

1. SURVEY OF CODES

1.1. Some Algebraic Concepts

A short synopsis will be given, in this section, of some of the important algebraic concepts that are used in developing this thesis. Detailed versions with rigorous proofs can be found in many books on modern algebra [3, 4].

1.1.1. Groups

A group G is a set of objects or elements for which an operation (*) - be it multiplication or addition - is defined and for which the following postulates hold:-

(i) Closure: The operation (*) is used to associate with every pair of elements a,b of G, a 3rd element of the same set a(*)b.

(ii) Associative Law: If a, b, c are elements of G, then

$$[a(*)b]\ (*)c = a(*)\ [b(*)c]$$

(iii) Identity: There is an element e in G called the identity such that if $a \in G$, then

$e(*)\ a = a$ $\quad e = 0$ if the operation is addition

$e = 1$ if the operation is multiplication

(iv) Inverse: If $a \in G$, there exists an element $a^{-1} \in G$ called the inverse of a, such that $a^{-1}\ (*)\ a = e$

In addition to the above, a group is called abelian if $a(*)b = b(*)a$ (Commutative law).

1. SURVEY OF CODES

1.1. Some Algebraic Concepts

A short synopsis will be given, in this section, of some of the important algebraic concepts that are used in developing this thesis. Detailed versions with rigorous proofs can be found in many books on modern algebra [3, 4].

1.1.1. Groups

A group G is a set of objects or elements for which an operation (*) - be it multiplication or addition - is defined and for which the following postulates hold:-

(i) Closure: The operation (*) is used to associate with every pair of elements a,b of G, a 3rd element of the same set a(*)b.

(ii) Associative Law: If a, b, c are elements of G, then

$$[a(*)b]\ (*)c = a(*)\ [b(*)c]$$

(iii) Identity: There is an element e in G called the identity such that if $a \in G$, then

$e(*)\ a = a$ $\quad e = 0$ if the operation is addition

$e = 1$ if the operation is multiplication

(iv) Inverse: If $a \in G$, there exists an element $a^{-1} \in G$ called the inverse of a, such that $a^{-1}\ (*)\ a = e$

In addition to the above, a group is called abelian if $a(*)b = b(*)a$ (Commutative law).

1.1.2. Subgroups

A subset of elements of a group G constitute a subgroup H provided all the postulates for a group are satisfied.

1.1.2. Rings

A ring R is a set of elements for which two operations are defined - addition and multiplication - and for which the following axioms must be satisfied:

(i) The set R is an abelian group under addition.

(ii) Closure: For any two elements a and b$\in$R, the product (a.b) is defined and is an element of R.

(iii) Associative Law: If any 3 elements a,b,c$\in$R then a.(b.c) = (a.b).c

(iv) Distributive Law: If a,b,c$\in$R, then a.(b + c) = ab + ac and (b + c).a = ba + ca

A ring is commutative, if with the multiplicative operation a.b = b.a

1.1.4. Ideals

Just as subgroups are important in the discussion of groups, Ideals play a corresponding role in the theory of rings. An Ideal I is a subset of elements of a ring R with the following two properties:

(i) I is a subgroup of the additive group of R.

(ii) For any element a $\in$ I and r $\in$ R, ar and ra belong to $\mathfrak{I}$.

1.1.5. Residue Classes

Since an ideal is a subgroup, cosets can be formed. However, in the case of ideals these cosets are called residue classes.

1.1.6. Fields

A field is a commutative ring with a unit element (the multiplicative identity) in which every nonzero element has a multiplicative inverse. It is interesting to note that the nonzero elements of a field satisfy all the axioms for a group, and thus form a group, under the operation of multiplication.

1.1.7. Vector Spaces

A set V of elements is called a vector space over a field F (composed of scalar elements), if it satisfies the following axioms:

(i) The set of elements V is an abelian group under addition.

(ii) For any vector v, and any field element c, a product cv which is a vector, is defined.

(iii) For any vectors u and v and any field elements c,d

$c(u + v) = cu + cv$

or $(c + d)v = cv + dv$

(iv) For any vector v and scalar elements c and d,

$(cd)v = c(dv)$

e.g. the set of all n - tuples over a field forms a vector space.

1.1.8. Subspaces

A subset of a vector space V is called a subspace S if all the axioms for a vector space are satisfied.

1.1.9. Null Spaces

A subspace S_1 is called the null space of another subspace S_2 if the set of all n-tuples in S_1 are orthogonal to the set of n-tuples of S_2.

1.2. Binary Block Codes

A block code of length n is defined as a set of binary words, each composed of n binary symbols or n-tuples. The n symbols in every word can be divided into two classes: the information bits (k in number) and the parity check bits [(n - k) in number]. Every word in the code has the information and parity check bits in the same position. There are $N = 2^k$ words in the code in which the information bits' positions can be occupied by any combination of 1's and 0's. The entries in the parity check bits' positions are some linear combination of certain entries in the information bits positions. Hence a code composed of N block words would have the format in Fig. 1.1, where the A_i's are the binary symbols 0 or 1.

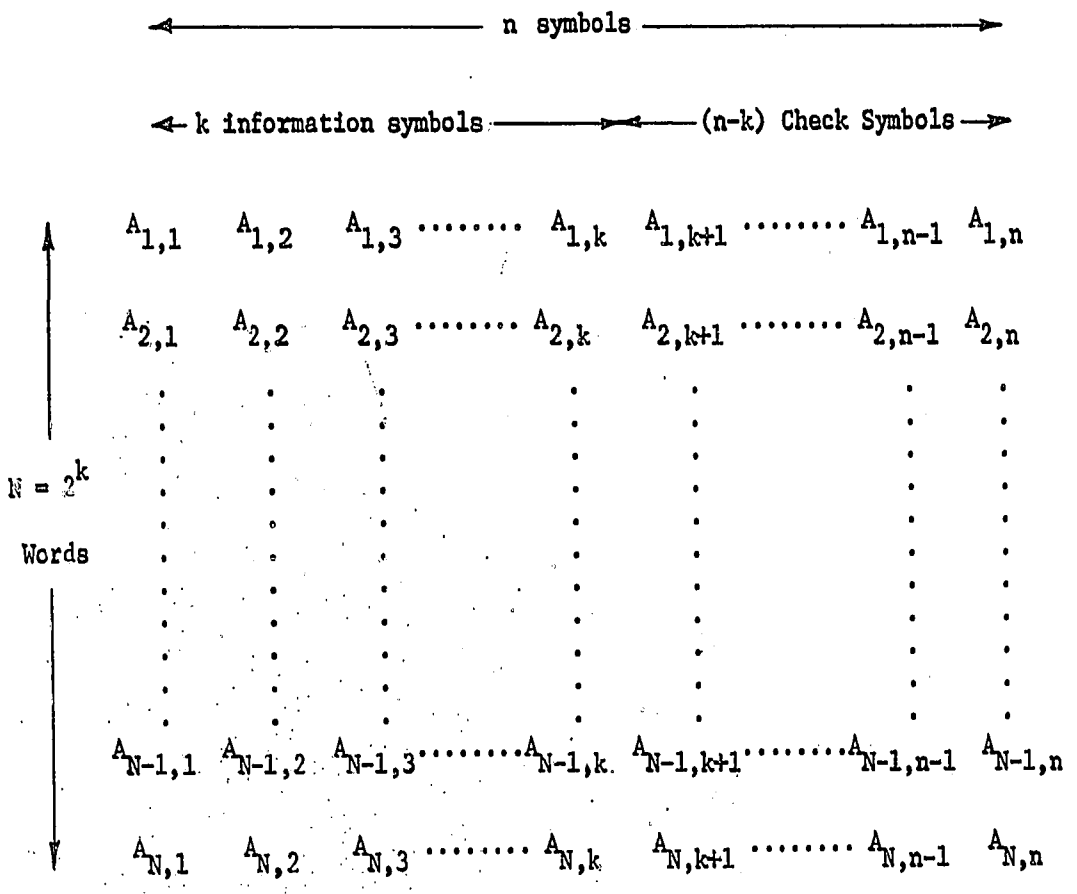


Fig. 1.1 Representation of a Block Code

Slepian [5] has shown that the set of words forming a block code form a group under the operation of modulo 2 addition. Consequently we can form cosets which will give us some data for decoding purposes.

1.3. Short Introductory Theory of Cyclic Codes

Consider the polynomial $F(x) = x^n - 1$ being represented as the product of two polynomials:

$$x^n - 1 = g(x)\ h(x)$$

where $F(x) = x^n - 1 \in Z_{(0,1)}[x]$

$Z_{(0,1)}[x]$ being the ring of all polynomials whose coefficients are 0 or 1;

h(x) is of degree k, and g(x) of degree (n-k).

Let us form - (Fig. 1.2) - all the residue classes of the ideal generated by x^n-1; (this is the same as forming all the cosets of a subgroup).

	$\{0\}$	=	$0[x^n-1]$	$x^o[x^n-1]$	$x[x^n-1]$ ....
	$\{g_1(x)\}$	=	$g_1(x)+0[x^n-1]$	$g_1(x)+x^o[x^n-1]$	$g_1(x)+x[x^n-1]$ ....
	$\{g_2(x)\}$	=	$g_2(x)+0[x^n-1]$	$g_2(x)+x^o[x^n-1]$	$g_2(x)+x[x^n-1]$ ....
	.		.	.	.
2^n	.		.	.	.
Residue	.		.	.	.
Classes	.		.	.	.
	.		.	.	.
	.		.	.	.
	$\{g_{2^n-1}(x)\}$	=	$g_{2^n-1}(x)+0[x^n-1]$	$g_{2^n-1}(x)+x^o[x^n-1]$	$g_{2^n-1}(x)+x[x^n-1]$....

Fig. 1.2 All Possible Residue Classes of the Ideal Generated by x^n-1

The above array contains all the polynomials in the ring $Z_{(0,1)}[x]$; i.e. each row in the array is a residue class which cannot contain more than one polynomial of degree < n; in other words there are as many residue classes modulo x^n-1 as there are polynomials of degree < n, and that is 2^n.

Referring now to the polynomial

$$x^n-1 = g(x)\ h(x)$$

it would be found that somewhere amongst all these residue classes $\{0\}, \{g_1(x)\}, \{g_2(x)\}, \ldots\ldots \{g_{2^n-1}(x)\}$, there exists a residue class containing the polynomial $g(x)$ which naturally divides $x^n - 1$. As previously mentioned, the array $Z_{(0,1)}[x]$ is a ring under the operations of addition and multiplication. The set of residue classes $\{0\}, \{g_1(x)\}, \{g_2(x)\}, \ldots\ldots \{g_{2^n-1}(x)\}$, modulo $x^n - 1$, also forms a ring; we can therefore speak in terms of ideals.

Let I be an ideal in the algebra of polynomials $Z_{(0,1)}[x]$ modulo $x^n - 1$ and let $g(x) \neq 0$ be the polynomial of least degree such that the residue class $\{g(x)\} \in I$. Then $\{g(x)\}$ generates the ideal I and the polynomial $g(x)$ divides $x^n - 1$. Furthermore, a residue class $\{g_i(x)\} \in I$ iff (if and only if) $g(x)$ divides $g_i(x)$. It can be shown, that every monic polynomial that divides $x^n - 1$ generates an ideal in $Z_{(0,1)}[x]$ modulo $x^n - 1$, and conversely every ideal $I \in Z_{(0,1)}[x]$ has a generator. Furthermore the algebra of polynomials $Z_{(0,1)}[x]$ modulo $x^n - 1$ forms a vector space of dimension n over GF(2), (polynomials having the coefficients 0,1). Also every ideal I in $Z_{(0,1)}[x]$ modulo $x^n - 1$ forms a vector subspace over GF(2).

If $$x^n - 1 = g(x)\, H(x)$$

where $g(x)$ and $h(x)$ are of degree $(n - k)$ and k respectively, then the ideal I generated by $g(x)$ is a vector subspace of dimension k and with basis vectors

$$\{g_1(x)\}, \{xg(x)\}, \{x^2 g(x)\} \ldots\ldots \{x^{k-1} g(x)\}.$$

Definition: A subspace S of $Z_{(0,1)}[x]$, modulo $x^n - 1$, over $Z_{(0,1)}$ is a cyclic subspace or a cyclic code, if for each vector $\{s(x)\} \in S$ the vector $\{xs(x)\}$ is also in S; i.e. the right cyclic shift of every vector in S is also in S.

It has been shown [6] that in the ring of polynomials modulo $x^n - 1$, a subspace is a cyclic code iff it is an ideal. Hence a cyclic code is

completely characterized by its generator polynomial g(x) which divides $x^n - 1$. Furthermore, for every polynominal $g(x) \neq 0$ such that $x^n - 1 = g(x)\, h(x)$, the code can be specified completely by the statement that it is the null space of the ideal generated by $h(x) = \frac{x^n - 1}{g(x)}$.

In matrix representation

$$G(x) = \begin{bmatrix} g(x) \\ xg(x) \\ x^2 g(x) \\ \cdot \\ \cdot \\ \cdot \\ x^{k-2} g(x) \\ x^{k-1} g(x) \end{bmatrix}$$

If $g(x) = A_o + A_1 x + A_2 x^2 + \ldots\ldots\ldots\ldots A_{n-k}\, x^{n-k}$

then

$$G = \begin{bmatrix} A_o & A_1 & \ldots\ldots\ldots\ldots & A_{n-k} & 0 & \ldots\ldots\ldots\ldots & 0 \\ 0 & A_o & \ldots\ldots\ldots\ldots & A_{n-k} & 0 & \ldots\ldots\ldots\ldots & 0 \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ \cdot & & & & & & \\ 0 & \ldots\ldots\ldots\ldots & 0 & A_o & A_1 & \ldots\ldots\ldots\ldots & A_{n-k} \\ 0 & \ldots\ldots\ldots\ldots & 0 & A_o & A_1 & \ldots\ldots\ldots\ldots & A_{n-k} \end{bmatrix}$$

All the rows of G are code vectors. The rank of G is k which is also the dimension of the code. The rows of G being linearly independent, therefore all linear combinations thereof would constitute the code space.

The representation in Fig. 1.3 may be found helpful.

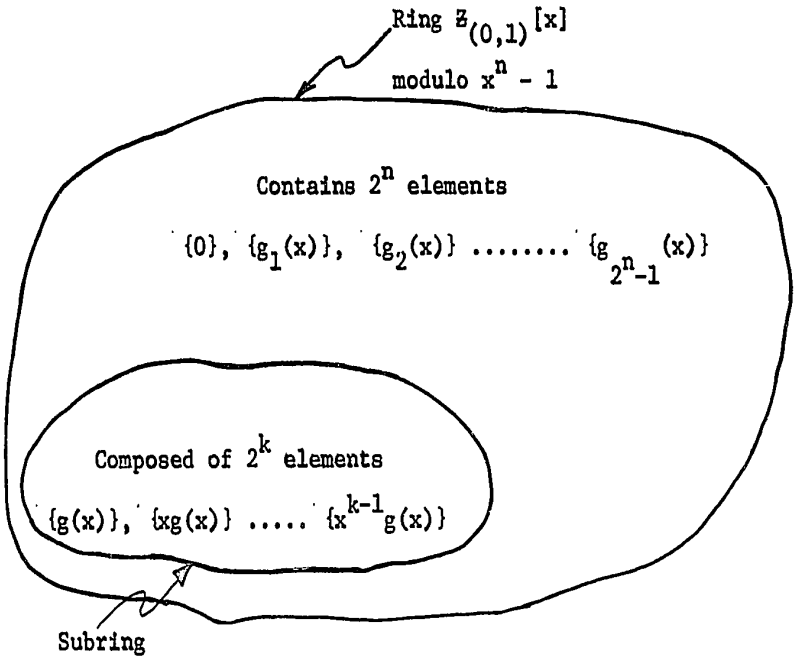


Fig. 1.3 Representation of the ring $Z_{(0,1)}[x]$ containing the subspace of 2^k elements

The ring $Z_{(0,1)}[x]$ modulo $x^n - 1$ forms a vector space over $Z_{(0,1)}$. In the code generated by linear combinations of the vectors in G each code word is an element of the ring $Z_{(0,1)}[x]$ modulo $x^n - 1$, and more specifically of the subspace containing 2^k vector. It is interesting to note here that since a ring is a group under addition, all

manipulations of these code words or code vectors satisfy all the group postulates.

1.3.1. Shortened Cyclic Codes

The existence of an (n,k) linear code implies the existence of a shorter (n - i, k - i) linear code by making the last i information symbols identically zero and deleting them from the code vectors. In the case of cyclic codes, this corresponds to omitting the last i rows and columns of the generator matrix. The resulting code is not cyclic in the true sense of the word, because it is no longer true in every case that the vector formed by shifting a code vector cyclically is also a code vector. We will refer to these codes as Shortened Cyclic Codes.

We already have discussed that cyclic codes are ideals in the algebra of polynomials modulo $x^n - 1$. After deleting the leading i information digits, the resulting codes are still ideals in the algebra of polynomials but modulo some other polynomial f(x). Such codes are called Pseudo-Cyclic Codes; it has been shown [7] that the Shortened Cyclic Codes and the Pseudo-Cyclic Codes are the same.

The [(n - i), (k - i)] generator matrix of the Shortened Cyclic Code, through linear combinations of its (k - i) rows, will generate a code containing $N = 2^{k-i}$ words, each word of length $n_1 = n - i$ digits. It is important to note that the resulting [(n - i), (k - i)], $o < i < k$, code will have the same error correcting capabilities as the parent code; i.e. it will have at least as great a minimum distance as the parent code, and it will correct any error pattern that the original code could correct.

manipulations of these code words or code vectors satisfy all the group postulates.

1.3.1. Shortened Cyclic Codes

The existence of an (n,k) linear code implies the existence of a shorter $(n - i, k - i)$ linear code by making the last i information symbols identically zero and deleting them from the code vectors. In the case of cyclic codes, this corresponds to omitting the last i rows and columns of the generator matrix. The resulting code is not cyclic in the true sense of the word, because it is no longer true in every case that the vector formed by shifting a code vector cyclically is also a code vector. We will refer to these codes as Shortened Cyclic Codes.

We already have discussed that cyclic codes are ideals in the algebra of polynomials modulo $x^n - 1$. After deleting the leading i information digits, the resulting codes are still ideals in the algebra of polynomials but modulo some other polynomial f(x). Such codes are called Pseudo-Cyclic Codes; it has been shown [7] that the Shortened Cyclic Codes and the Pseudo-Cyclic Codes are the same.

The $[(n - i), (k - i)]$ generator matrix of the Shortened Cyclic Code, through linear combinations of its $(k - i)$ rows, will generate a code containing $N = 2^{k-i}$ words, each word of length $n_1 = n - i$ digits. It is important to note that the resulting $[(n - i), (k - i)]$, $o < i < k$, code will have the same error correcting capabilities as the parent code; i.e. it will have at least as great a minimum distance as the parent code, and it will correct any error pattern that the original code could correct.

1.3.2. Bose-Chaudhuri-Hocquenghem Codes

The BCH codes [9, 10, 11] are cyclic codes that are best defined in terms of their generating polynomial. For an arbitrary e and m they are e error correcting and have a word length $n = 2^m - 1$ of which no more than em digits are redundant i.e. parity checks. Thus, the BCH codes cover a wide range in transmission rate ($R = \frac{k}{n} = \frac{\text{No. of information digits}}{\text{No. of digits in word}}$) and error correcting capability. In addition they are a generalization of the Hamming code [8] since for $e = 1$, the Hamming code is obtained in each case.

Construction:

Given an irreducible polynomial P(x) of degree m over GF(2) (i.e. P(x) has coefficients 0 or 1), a representation of the Galois Field with 2^m elements - $GF(2^m)$ - can be formed. It consists of all polynomials of degree (m - 1) or less. They can be added, modulo 2, term by term in the ordinary way. Multiplication is ordinary, but the answer is reduced modulo 2 and modulo P(x) to a polynomial of degree (m - 1) or less. It can be shown that certain of these polynomials have for their roots primitive elements. A primitive element is one that generates the multiplicative cyclic group (all nonzero elements) of the $GF(2^m)$. The field elements can be thought of as vectors whose components are the coefficients of the polynomials.

One definition and construction procedure for BCH codes has been covered in detail in [9]. An alternative and more general definition [10, 12] tends to show emphatically the cyclic nature of BCH codes.

The most important BCH codes are obtained by letting α be a primitive element of $GF(2^m)$ and (Hamming distance) $d = 2e + 1$. Then $\{f(x)\}$ is a

code vector iff

$$\alpha,\ \alpha^2,\ \alpha^3 \ldots\ \alpha^{2e}$$

are roots of f(x). However, every even power of α is a root of the same minimum function as some previous odd power of α. For example if $m_j(x)$ denotes the minimum function of α^j, then α^2 and α^4 are roots of $m_1(x)$, α^6 is a root of $m_3(x)$, α^8 is a root of $m_1(x)$, α^{10} is a root of $m_5(x)$ and so forth. Therefore an equivalent statement is that {f(x)} is a code vector iff

$$\alpha,\ \alpha^3,\ \ldots\ \alpha^{2e-1}$$

are roots of f(x). Each element α^j of the field is a root of a unique irreducible polynomial $m_j(x)$ of minimum degree. Then f(x) must be divisible by each of the polynomials $m_1(x)$, $m_3(x)$ ..... $m_{2e-1}(x)$ and hence by their least common multiple. Thus the generator of the code

$$G(x) = \text{LCM}[m_1(x),\ m_3(x),\ m_5(x) \ldots\ m_{2e-1}(x)]$$

Since each of the factors $m_j(x)$ is irreducible and has degree no greater than m, the least common multiple of the $m_j(x)$ is simply the product of the polynomials $m_j(x)$ with the duplicates omitted. Duplications are possible and will occur for any α^i and α^j that are roots of the same polynomial $m_i(x)$.

Therefore for any e and m, g(x) will generate a BCH binary code of length $2^m - 1$ which corrects all combinations of e or less errors and has no more than em parity check symbols.

Example

For $m = 4$, $e = 3$, {f(x)} is a code vector iff $\alpha,\ \alpha^2,\ \alpha^3 \ldots\ \alpha^6$ are roots of f(x) where α is a primitive element of $GF(2^4)$. The length of the code is $2^4 - 1 = 15$.

If

$$x^n - 1 = g(x)\, h(x)$$

the generator polynomial of degree $(n - k)$ would be

$$g(x) = m_1(x)\, m_3(x)\, m_5(x)$$

and if α is taken to be a root of $x^4 + x + 1$ then

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)$$

$$g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$$

and $$h(x) = \frac{x^{15} - 1}{g(x)} = 1 + x + x^3 + x^5$$

The generator matrix

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

The resulting code space, obtained by linear combinations of the vectors in G, would have $N = 2^5 = 32$ words each of length $n = 15$, and would be capable of correcting any combination of 3 or fewer errors.

Relatively simple encoding and error correcting methods have been devised [12, 13] for the BCH codes, the study of which gives additional insight into the underlying algebraic structure of these codes. We might mention in passing that in some few cases other codes [14] give a greater number of words than BCH codes.

1.3.3. Abramson Burst Error Correcting Codes for Burst Length ≤ 3

Abramson [15] has described a class of codes which correct all single errors (SE) and double adjacent errors (DAE) and require significantly less checking digits than codes capable of correcting all double errors.

The generator polynomial of these codes has the form

$$G(x) = P(x)\ (1 + x)$$

The code has a maximal length $n = 2^m - 1$, m being the number of parity check digits. The factor P(x) is a primitive polynomial of order m - 1 which allows for single error correction. The factor (1 + x) adds an extra parity check on all digits which makes double adjacent error correction possible. The idea is that the total parity check described by (1 + x), specifies whether there has been a SE or DAE. If there has been a SE, it is taken care of by P(x); if on the other hand there has been a DAE, it will be taken care of by (1 + x).

The first parity check equation is constructed according to

$$n \geq k + m$$

where k is the number of information digits; n and its (m - 1) cyclic shifts then constitute the set of all parity check equations.

Example

For $m = 3$

$$(n = 2^m - 1) \geq k + m$$

and $k = 4 \qquad n = 7$

A sequence having a maximal length $n = 2^m - 1 = 7$ is 1 0 1 1 1 0 0 which when represented as a polynomial, gives as the first parity check equation

$$x_1 + x_3 + x_4 + x_5 = 0$$

The (m - 1) cyclic shifts then give the rest of the parity check equations

$$1\ 0\ 1\ 1\ 1\ 0\ 0 \qquad x_1 + x_3 + x_4 + x_5 = 0$$

$$0\ 1\ 0\ 1\ 1\ 1\ 0 \qquad x_2 + x_4 + x_5 + x_6 = 0$$

$$0\ 0\ 1\ 0\ 1\ 1\ 1 \qquad x_3 + x_5 + x_6 + x_7 = 0$$

Abramson [16] has also attempted to find minimum redundancy binary cyclic codes for burst error correction. He conjectured that the generator polynomial of a BEC (Burst Error Correcting) code capable of correcting a burst error of length $l_c \leq 3$ would be of the form

$$G(x) = P(x)\ (1 + x + x^2)$$

where P(x) is a primitive polynomial of even degree > 2. In as much as this is a conjecture, it appears very likely that there exist primitive polynomials of even degree > 2, which when multiplied by $(1 + x + x^2)$ would generate a BEC code which will correct all bursts of $l_c \leq 3$. The factor $(1 + x + x^2)$ takes care of all possible combinations of error bursts having a length $l_c \leq 3$. The total number of burst patterns of length l_c or less is $2^{l_c - 1}$, and each may start in any of the first $n - l_c + 1$ symbols of a code word. For $l_c = 3$ the total number of burst patterns is

$$2^{l_c - 1} = 4$$

and they are

Single Error SE	Double Adjacent Errors DAE	Split Errors SPE	Triple Adjacent Errors TAE
e	ee	eoe	eee

A lower bound for the necessary number of redundant parity check digits would be

$$2^m \geq 1 + n2^{l_c - 1}$$

m being the number of parity check symbols in a word. Incidentally multiple solid BEC codes have also been dealt with in literature [23, 24].

1.3.4. Fire Codes

The Fire codes [17, 18] are the largest class of BEC codes known. They cover the entire range of burst correction length l_c, burst detection length l_d and word length n. Being cyclic these codes can best be defined in terms of their generator polynomial which has the form

$$G(x) = P(x)(1 + x^{\alpha})$$

where P(x) is an irreducible polynomial of degree m, over GF(2), whose roots have order e which does not divide α. The polynomial G(x) will generate a code having

word length $n = LCM(e, \alpha)$

Information Digits $k = n - (\alpha + m)$

Parity Check Digits $= \alpha + m$

It will correct any error burst of length l_c or less and simultaneously detect any error burst of length l_d or less ($l_d \geq l_c$) provided

$$\alpha \geq l_c + l_d - 1$$

and $l_c \geq m$

For detection purposes only, it can detect two bursts such that

$$\alpha + 1 \geq l_{d_1} + l_{d_2}$$

where the shorter burst among l_{d_1} and $l_{d_2} \leq m$

as well as every single burst l_d provided that $l_d \leq \alpha + m$ (the number of parity checks).

Example:

The code generated by

$$G(x) = (1 + x + x^3)(1 + x^6)$$

where $m = 3 \qquad e = 2^3 - 1 = 7 \qquad \alpha = 6$

has a length $n = \text{LCM}(7,6) = 42$. There are $\alpha + m = 9$ parity check symbols and 33 information sy mbols. The code is capable of correcting any burst of length ≤ 3 and detecting any burst of length ≤ 4, or correcting any burst of length ≤ 2 and detecting any burst of length ≤ 5. For detection purposes only, it is capable of detecting every single burst of length ≤ 9 and every combination of two bursts as long as $(l_{d_1} + l_{d_2}) \leq 7$, and the shorter between l_{d_1} and $l_{d_2} \leq 3$.

Error Correcting Capabilities of Fire Codes

In the previous section we touched briefly upon the error capabilities of Fire codes; we will try here to expand on this and look at the idea behind the design of these codes. The following theorem (a proof of which is found in [19])demonstrates the condition for the effectiveness of Fire codes.

Theorem

A vector that is the sum of a burst of length $\leq l_c$ and a burst of length $\leq l_d$ cannot be a code vector in a Fire code if

$$\alpha \geq l_c + l_d - 1$$

and the shorter burst among l_c and l_d is less or equal to m.

This is equivalent to requiring that a code vector cannot be composed of the sum of a correctable and detectable error burst. In effect, it is a necessary and sufficient condition to assume that the Fire code is able simultaneously to correct every burst of length $\leq l_c$ and to detect every burst of length $\leq l_d$ $(l_d \geq l_c)$. The concept behind the design of Fire codes is the basis to the construction of the generator polynomial which is a generalization of Abramson's polynomial,

$$G(x) = P(x)(1 + x^{\alpha})$$

from which all the properties of the code are derived. Actually the power of the code is centered in the factor $(1 + x^{\alpha})$ The parity checks associated with this factor are α interlaced evenly spaced parity checks. Since the symbols involved in each check are spaced α symbols apart, each of the α parity checks will not be affected by more than one error in any burst of length α or less. Since we can always write

$$\alpha = l_c + l_d - 1$$

then any burst of length $\leq l_c$ will leave at least $l_d - 1$ successive parity checks unaffected, and this alone is sufficient to tell which symbol is at the beginning of the burst. Thus the factor $(1 + x^{\alpha})$ is sufficient to detect a single burst of length $\leq \alpha$ and to determine completely the error pattern for bursts of lengths $\leq l_c$. The additional information required to locate the beginning of an error burst is provided by the factor P(x). This is shown by considering that the sum of two different error bursts cannot be a code vector, i.e. it cannot be divided by G(x). This can only happen when these two error patterns are the same. If their sum is divisible by $(1 + x^{\alpha})$, it is not divisible by P(x), and thus two bursts of the same pattern but different locations are distinquished by the factor P(x).

Fire codes are powerful since they have a high transmission rate $\frac{k}{n}$. Depending on the values of α and m, the transmission rate

$$R = \frac{k}{n} = \frac{n - (\alpha + m)}{n} = 1 - \frac{\alpha + m}{n}$$

is very close to unity, as $n \longrightarrow$ large relative to $(\alpha + m)$. However, for $l_c \leq 3$ the Abramson codes give a higher transmission rate than Fire codes.

It is interesting to note, that Fire codes have even weight since $(1 + x)$ is a factor of $G(x) = P(x)(1 + x^{\alpha})$.

2. ERROR CONTROL

In all codes, the number of information symbols to represent the desired information, is kept to a minimum to preserve channel capacity. Such codes, while adequate for representation of information, may not be suitable for transmission of information due to the introduction of errors from noisy channels.

2.1. Concept of Decoding

Error control then, has to be introduced to find such error. The "ideal observer" decision scheme [20] is one of the earliest and simplest.

2.1.1. "Ideal Observer" Decision Scheme

It consists mainly of having an "ideal observer" (a device that has access to or observes both of the transmitted and received messages) which makes decisions relying on what it observes and what it possesses in its memory; in other words, a received sequence v_j is decoded as a transmitted code word w_i, such that the Hamming distance $d(w_i, v_j)$ is minimum. Thus the decoder must contain a "table" or "code book" whose entries are all binary sequences of the specified code word length, and each possible received sequence is assigned to the closest code word.

The main disadvantage of such a scheme is obvious. The storage requirements for such a decoder are quite severe, requiring a huge memory size for a large number of code words.

2. ERROR CONTROL

In all codes, the number of information symbols to represent the desired information, is kept to a minimum to preserve channel capacity. Such codes, while adequate for representation of information, may not be suitable for transmission of information due to the introduction of errors from noisy channels.

2.1. Concept of Decoding

Error control then, has to be introduced to find such error. The "ideal observer" decision scheme [20] is one of the earliest and simplest.

2.1.1. "Ideal Observer" Decision Scheme

It consists mainly of having an "ideal observer" (a device that has access to or observes both of the transmitted and received messages) which makes decisions relying on what it observes and what it possesses in its memory; in other words, a received sequence v_j is decoded as a transmitted code word w_i, such that the Hamming distance $d(w_i, v_j)$ is minimum. Thus the decoder must contain a "table" or "code book" whose entries are all binary sequences of the specified code word length, and each possible received sequence is assigned to the closest code word.

The main disadvantage of such a scheme is obvious. The storage requirements for such a decoder are quite severe, requiring a huge memory size for a large number of code words.

2.1.2. Parity Check Decoding Scheme

The parity check decoding scheme is less severe on memory thus improving the efficiency of the decoding process. It consists of adding symbols to the transmitted message to enable the recipient of the information to detect and, when possible, correct any error that may have perturbed the transmitted message. These additional digits are called parity check digits.

Parity Check Methods for detecting and correcting errors, were first generally developed by R. W. Hamming [8], for single error detection and correction, plus double error detection of independent errors. For single error detection, the basic principle of the parity check method is to transmit an extra digit with each code word, such that all code words contain either an odd or even number of 0's or 1's; the idea being that the receiver will always expect an even or odd number of 0's or 1's in every received word, always assuming that double errors do not occur. For the detection and correction of single binary errors, the basic idea is

a) If we know an error has occurred (by parity check method) and

b) if we know the position of the error, then by simply complementing the digit in that position, we can correct the error. This is accomplished by adding to the information symbols k, m checking symbols such that

$$2^m \geq k + m + 1 \qquad k + m = n$$

Therefore, in order to transmit four information symbols ($k = 4$), we need to have at least 3 parity check symbols ($m = 3$) in each word. If x_i denotes the i^{th} position in a sequence of 7 digits ($n = 7$), we can write the parity check equations (modulo 2 addition)

$$x_1 + x_2 + x_3 + x_5 = 0$$

$$x_1 + x_2 + x_4 + x_6 = 0$$

$$x_1 + x_3 + x_4 + x_7 = 0$$

Thus, if we assume that not more than one error has occurred in all the 7 symbol words, we can find the error position and correct it using the above equations, e.g. if 1 1 0 0 0 1 1 was received, x_6 is in error and the transmitted word is 1 1 0 0 0 0 1.

2.2. Decoding of Cyclic Codes

Efficient decoding of cyclic codes, depends extensively on the way they were generated. We always have to keep in mind, that for a decoder to be able to decipher any received message, it has to know, or have access to, some pertinent information about the code as a whole, e.g. the generator polynomial, capability of error correction of the code, its length, etc. Hence storage requirements are practically nil but the computations are more involved.

2.2.1. Parity Check Decoding

Let us consider the relation

$$x^n - 1 = g(x)\, h(x)$$

where g(x) and h(x) are polynomials of degree (n - k) and k respectively, with coefficients in GF(2), and n maximal, i.e. $n = 2^q - 1$, q being a positive integer.

g(x) will then generate a cyclic code, of length n, containing k information digits, hence $N = 2^k$ code words; $h^*(x)$ (the reciprocal

polynomial of h(x)) will then act as a parity check. Alternately stated, the code generated by g(x) is the null space of the ideal generated by $h(x) = \frac{x^n - 1}{g(x)}$. This last statement is the key to a crude or rather brute force decoding method.

The decoder has to store in its memory all 2^{n-k} parity check syndromes. Then as it receives one message after another, it has to compute whether the received message lies in the null space of the stored parity check syndromes. If it does, then it is a code word; if it does not, then some error must have occurred.

While it looks as a simple and straight forward decoding method, it is at the same time laborious, and most important of all, imposes a severe requirement on the decoder memory size having to store all 2^{n-k} syndromes.

Due to the more complex nature of codes that correct independent errors as related to codes that correct burst errors, (there are as many more different combinations of 1-tuple errors than different error bursts of length 1), methods for independent error decoding of cyclic codes are complex [21]. As a result, somewhat simpler techniques for specific codes only have been developed e.g. Peterson's decoding technique [12] of Bose-Chaudhuri codes etc. On the other hand, due to the simpler nature of BEC codes there exists a few practical decoding techniques for burst error correction of which we will discuss the following two in particular [22].

2.2.2. Decoding Techniques for Burst Error Correction

Method A

Suppose that a certain code vector V(x) = G(x) C(x) was transmitted,

and during transmission a burst of errors $E'(x) = x^j E(x)$ gets added to $V(x)$. The received vector is $R(x) = V(x) + x^j E(x)$. At the decoder, the received vector polynomial is divided by $G(x)$, the generator (of degree g) of the code. If the remainder is zero, then $R(x)$ is a code word and no errors have occurred i.e. $E'(x) = 0$. Otherwise, the remainder contains information about the errors. Since $V(x)$ is a code vector, then the remainder of the division $\frac{V(x)}{G(x)}$ is the same as the remainder obtained by dividing $\frac{E(x)}{G(x)} = A(x) + \frac{B(x)}{G(x)}$ degree of $B(x) < g$. The error correction problem is to find $x^j E(x)$, the actual error, with the receiver having access to some information about the code i.e. $R(x)$, $G(x)$, n an l_c. The procedure is described in Fig. 2.1.

Method B

This is a decoding procedure, similar to the previous one, which in some cases will lead to a simpler circuit realization. It can be adapted to any code for which $G(x)$ can be factored into two polynomials, each of degree at least as great as the length of any correctable burst l_c. Fig. 2.2 outlines the procedure.

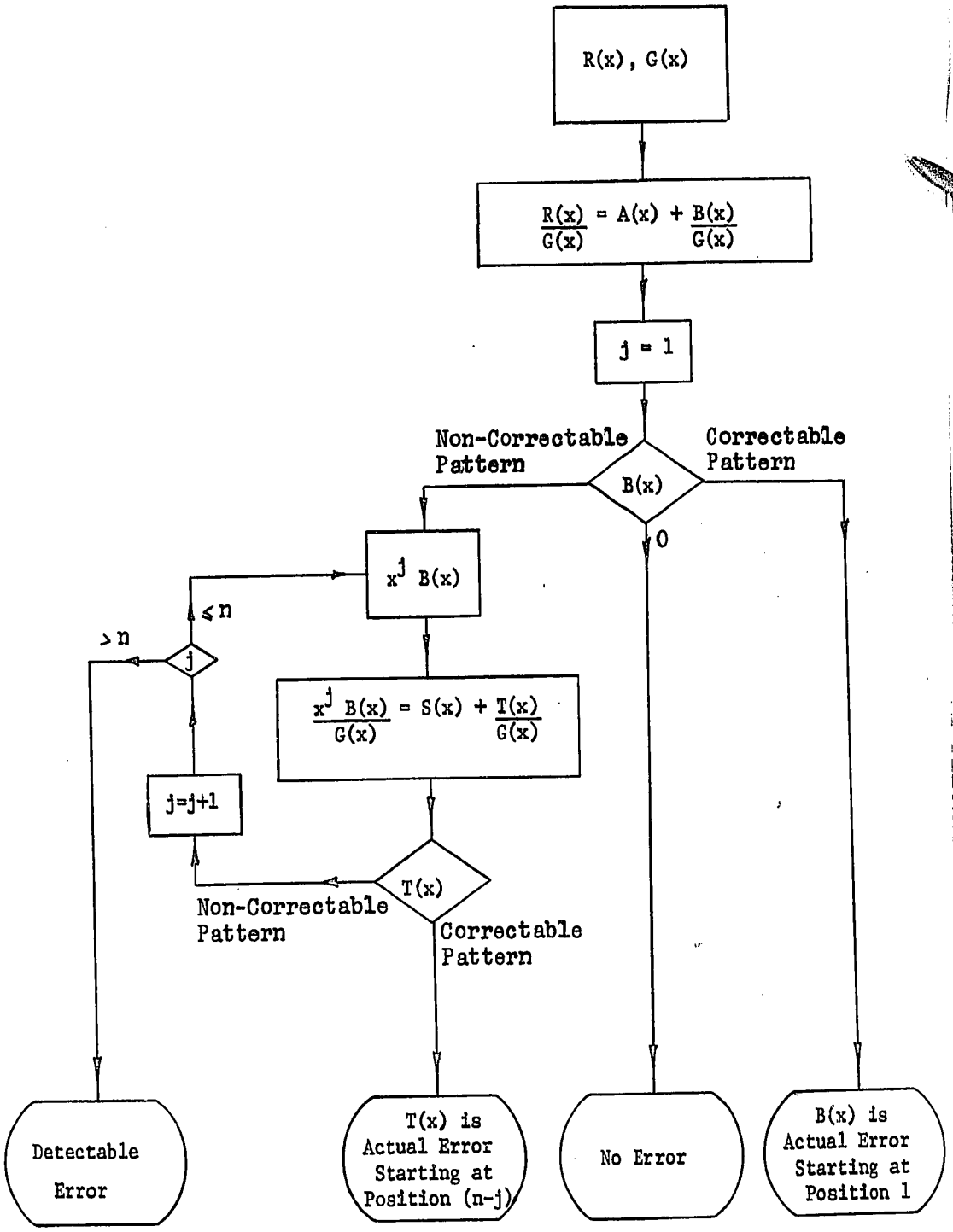


Fig. 2.1 Flow Diagram for Decoding Method A

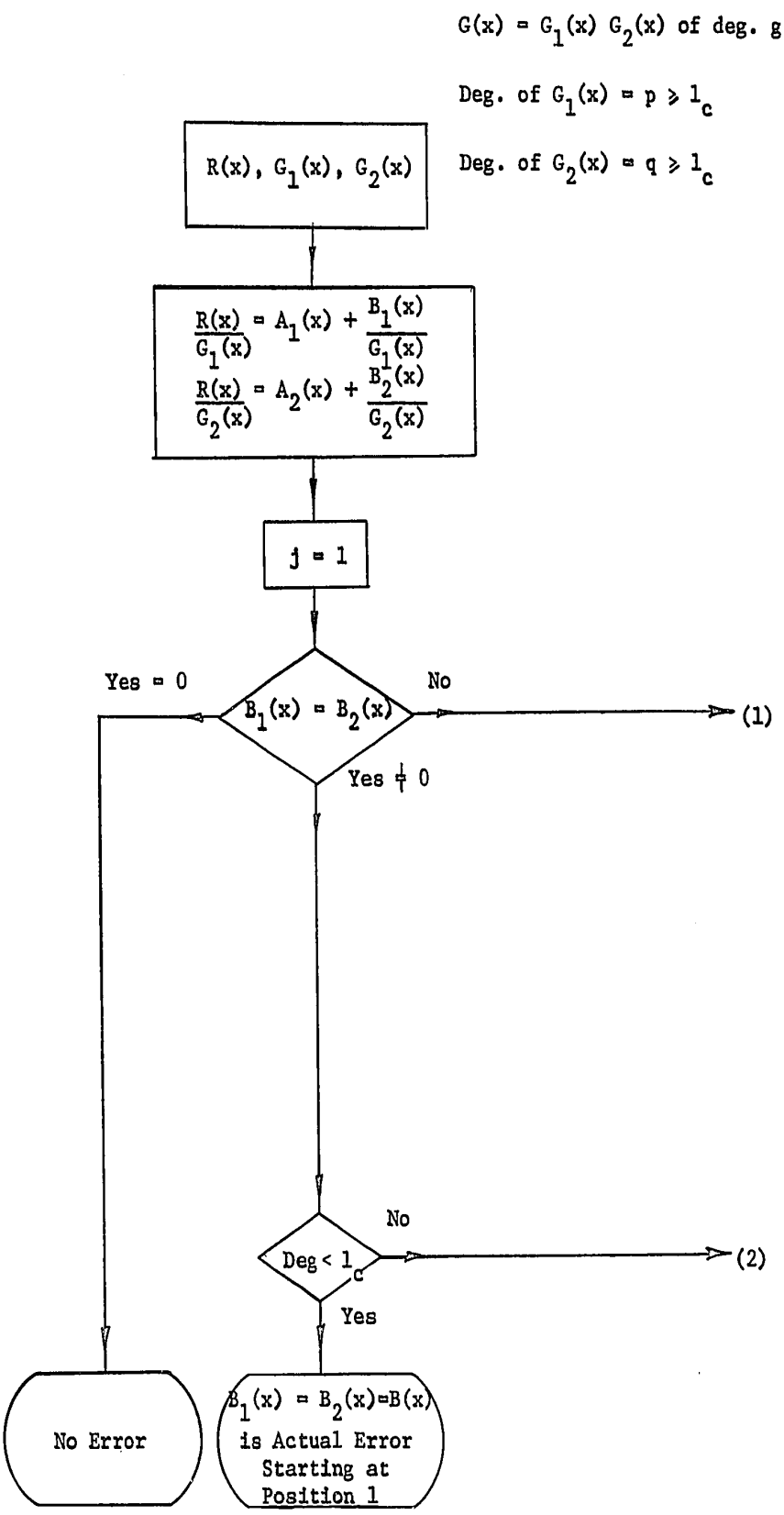


Fig. 2.2 Flow Diagram for Decoding Method B

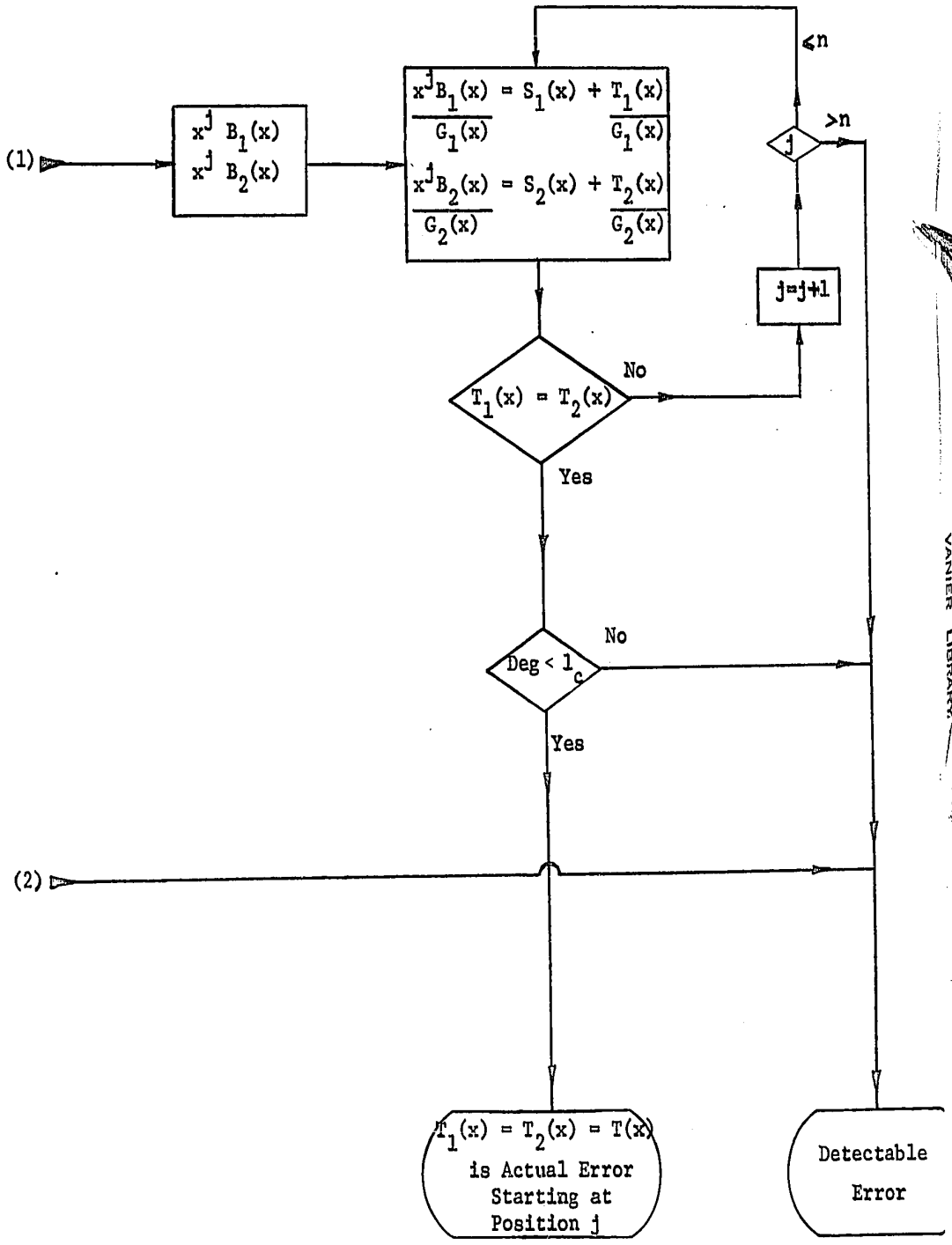


Fig. 2.2(Con't) Flow Diagram for Decoding Method B

Methods A & B give identical results. Practical representation of both decoding methods can be easily realized on a digital computer or by using a shift register circuit; the latter realization is discussed quite fully by Peterson [25].

Both methods are simple and straight forward. They detect and correct burst errors efficiently. Each requires maximally n steps and on the average $\frac{n}{2}$ steps before arriving at some conclusion. However, the time element could be substantial if very long codes are being considered.

In method B two division operations have to be done consecutively for every step, and it also has the disadvantage that it can only be used for codes whose G(x) is a product of two factors, the individual degrees of which have to be $\geq l_c$. This fact alone excludes its application to some versions of important codes e.g. Abramson's code whose $G(x) = P(x)\ (1 + x + x^2)$, where P(x) is a primitive (hence irreducible) polynomial. G(x) will generate a cyclic code that can correct a maximum burst of length $l_c = 3$. Obviously l_c is larger than the degree of one of the factors of G(x), and method B cannot be applied.

Now that we have discussed the construction, and certain properties, of some of the known decoding procedures for burst error correction of cyclic codes, we will introduce a decoding technique which in simplicity far exceeds the above mentioned procedures. It is arrived at by modifying the Cyclic BEC Codes to a certain shorter length. These shortened codes will be referred to as SBEC (Shortened Burst Error Correcting) codes. The penalty we have to pay to obtain this simple decoding technique is a certain degradation in the transmission rate $\frac{k}{n}$. In fact for these SBEC codes $\frac{k}{n}$ can not exceed 1/2.

3. SHORTENED CYCLIC BURST ERROR CORRECTING BINARY CODES

We shall be concerned with binary block codes. The set of code words for such a code is a subset of the set of n-tuples of the form

$$a_1, a_2, \ldots\ldots a_k, \ldots\ldots a_n$$

where each a_i is an element of GF(2).

The symbols $(a_1, a_2, \ldots\ldots a_k)$ are chosen to be the information symbols. The remaining n - k symbols are the parity check symbols. We assume now that, after transmission through a channel, a received n-tuple $(r_1, r_2, \ldots\ldots r_n)$ is obtained which differs from the transmitted n-tuple $(a_1, a_2, \ldots\ldots a_n)$, by a noise sequence $(e_1, e_2, \ldots\ldots e_n)$; i.e.

$$r_i = a_i + e_i \qquad i = 1, 2, \ldots\ldots n$$

where again r_i and e_i are elements of GF(2). It then follows from the last equation that knowledge of the received sequence and the noise sequence suffices to determine the transmitted sequence.

As is well known all (n, k) cyclic codes can be generated by a single appropriate polynomial G(x) of degree g = n - k. If, in the (n x k) generator matrix of such a code, the last i rows and i columns, $i < n - g - 1$, are deleted, the resulting matrix can generate what has been termed a shortened cyclic code. In this chapter we are concerned with shortened cyclic codes as applied to burst error correction. We shall refer to them as SBEC (Shortened Burst Error Correcting) codes. Specifically, we shall show that if the word length n_s of an SBEC code is made equal to $2g + 1 - l_c$, where l_c is the maximum burst length for which the code is considered, then decoding can be done in just two

steps whereas in the case of cyclic codes, as discussed in Ch. 2, it takes on the average, $\frac{n}{2}$ steps, assuming that all correctable errors are equiprobable.

The price paid for this easy decoding is that the transmission rate for SBEC codes, with n_s as defined previously, can at most be 1/2, a value which can be very nearly achieved. As regards burst error correcting capability, nothing is, of course, lost by shortening a cyclic code.

3.1. Theorems and Definitions

Definition

A reciprocal polynomial $f^*(x)$ of any polynomial $f(x)$, is defined as

$$f^*(x) = x^m f\left(\frac{1}{x}\right)$$

where m is an arbitrary integer.

Theorem 1

If $f(x) = f_1(x) + f_2(x)$

$\downarrow$ deg. m $\quad$ $\downarrow$ deg. m_1 $\quad$ $\downarrow$ deg. m_2 $\quad$ $m = \text{Max}\ \{m_1, m_2\}$

then $f^*(x) = f_1^*(x) + f_2^*(x)$

where $f_i^*(x) = x^m f_i\left(\frac{1}{x}\right)$

$i = 1, 2$

$m = m_1, m_2$ being arbitrary integers such that $m = \text{Max}\ \{m_1, m_2\}$

Proof:

$$f^*(x) = x^m f(\tfrac{1}{x})$$

$$= x^m [f_1(\tfrac{1}{x}) + f_2(\tfrac{1}{x})]$$

$$= x^m f_1(\tfrac{1}{x}) + x^m f_2(\tfrac{1}{x})$$

$$\therefore \quad f^*(x) = f_1^*(x) + f_2^*(x)$$

Theorem 2

If $f(x) = f_1(x) \cdot f_2(x)$

$\downarrow$ deg. m $\quad$ $\downarrow$ deg. m_1 $\quad$ $\downarrow$ deg. m_2

then $f^*(x) = f_1^*(x) \cdot f_2^*(x)$

where $f_i^*(x) = x^{m_i} f_i(\tfrac{1}{x}) \qquad i = 1, 2$

and m is an arbitrary integer

such that $m = m_1 + m_2$

Proof:

$$f^*(x) = x^m f(\tfrac{1}{x})$$

$$= x^m f_1(\tfrac{1}{x})\, f_2(\tfrac{1}{x})$$

$$= x^{m_1} x^{m_2} f_1(\tfrac{1}{x})\, f_2(\tfrac{1}{x})$$

$$= x^{m_1} f_1(\tfrac{1}{x})\, x^{m_2} f_2(\tfrac{1}{x})$$

$$\therefore \quad f^*(x) = f_1^*(x)\, f_2^*(x)$$

3.2. Shortened Burst Error Correcting Codes

Let $G(x)$ be the g^{th} degree generator polynomial of a cyclic code which can correct all bursts of length l_c or less and detect all bursts of length l_d or less, $l_d \geq l_c$. Then any code word polynomial can be written as

$$V(x) = G(x)\ C(x) \qquad \ldots\ldots(1)$$

where $C(x)$ is a polynomial of degree $\leq n - 1 - g$, n being the word length. If $E(x)$ is the burst error polynomial, then the received polynomial $R(x)$ is given by

$$R(x) = G(x)\ C(x) + E(x) \qquad \ldots\ldots(2)$$

From (2) we have

$$\frac{R(x)}{G(x)} = C(x) + \frac{E(x)}{G(x)} \qquad \ldots\ldots(3)$$

Now at the receiver, we divide every received $R(x)$ by $G(x)$ to get

$$\frac{R(x)}{G(x)} = A_1(x) + \frac{B_1(x)}{G(x)} \qquad \ldots\ldots(4)$$

where $B_1(x)$ is the remainder. Comparing (3) and (4) we can say that $B_1(x)$ does indeed give the error, if the error has actually occurred within the first g digits of the received word.

On the other hand, if we divide $R^*(x)$ by $G^*(x)$, we get

$$\frac{R^*(x)}{G^*(x)} = A_2(x) + \frac{B_2(x)}{G^*(x)} \qquad \ldots\ldots(5)$$

where $R^*(x) = x^{n-1}\ R\left(\frac{1}{x}\right)$ $\qquad$ n = word length

$$G^*(x) = x^g\ G\left(\frac{1}{x}\right)$$

$B_2(x)$ is the remainder.

We can now say that $B_2^*(x) = x^{n-1} B_2(\frac{1}{x})$ does indeed give the error, if the error has actually occurred within the last g digits of the received word as shown in Fig. 3.1.

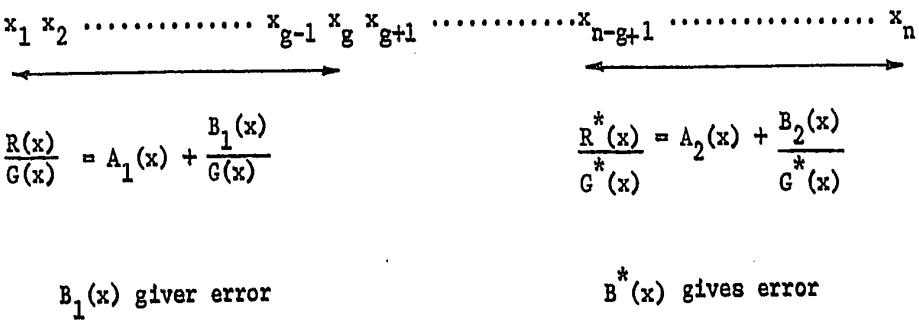


Fig. 3.1.

The question now is as to what happens in the case of errors which are not confined to either the first, or last, g digits. The known [23] decoding procedures make use of $B_1(x)$ repeatedly and require, on the average, $\frac{n}{2}$ steps. As against this, it is clear that the decoding procedure would require just two steps at most if every correctable error is made to appear as $B_1(x)$ or $B_2^*(x)$. This is accomplished by shortening the code to such a length which will be completely spanned by both division steps. In other words the first step will take care of the errors in the first half of the code word while the second step will take care of the errors that occurred in the last half of the code word, with a common region in the middle, the length of which is a fuction of the burst error length l_c. This is shown in Fig. 3.2. where we have a typical code word partitioned in such a way so that the common region in the middle is l_c-1 digits long, keeping in mind that an error burst of length l_c can occur

anywhere from position x_1 to position x_{n-l_c+1}

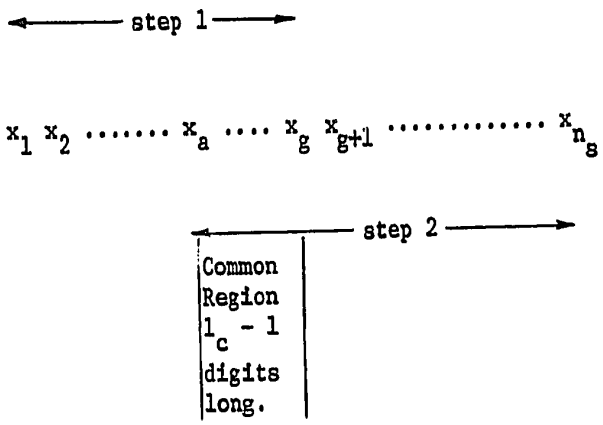


Fig. 3.2.

Therefore if we partition at x_{g+1}, then

$$(g + 1 - a) + 1 = l_c$$

and $$a = g + 2 - l_c$$

and the length n_s would have to be such that the length of the second half, between x_{n_s} and x_a, is equal to g

i.e. $$g = n_s - a + 1$$

and $$n_s = 2_g + 1 - l_c \quad \text{.....(6a)}$$

And as a result the corresponding number of information digits k_s is

$$k_s = g + 1 - l_c \quad \text{.....(6b)}$$

Fig. 3.3. shows a typical code word in the new shortened code.

$x_1\ x_2 \ldots\ldots\ldots\ldots\ldots x_{n_s-g+1} \ldots\ldots x_g\ \ x_{g+1} \ldots\ldots\ldots\ldots x_{n_s}$

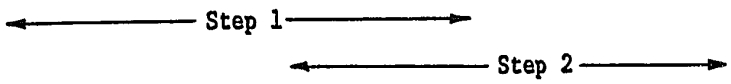


Fig. 3.3.

Combining (6a) and (6b), we get the expression, for the transmission rate, that

$$\frac{k_s}{n_s} = \frac{g - l_c + 1}{2_g - l_c + 1} \longrightarrow \frac{1}{2}$$

as g increases for a given l_c.

3.3. Decoding Procedure

At this point, let us go back to (4) and (5). These can be rewritten as

$$R(x) = A_1(x)\ G(x) + B_1(x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)$$

$$R^*(x) = A_2(x)\ G^*(x) + B_2(x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(8)$$

From (8) we have

$$[R^*(x)]^* = [A_2(x)\ G^*(x)]^* + B_2{}^*(x)$$

or

$$R(x) = [A_2(x)\ G^*(x)]^* + B_2{}^*(x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

Since $A_2(x)$ can be treated as the reciprocal of some $A_3(x)$, (9) can be rewritten as

$$R(x) = [A_3^*(x)\ G^*(x)]^* + B_2^*(x)$$

or

$$R(x) = A_3(x)\ G(x) + B_2^*(x) \quad \ldots\ldots(10)$$

Combining (7) and (10)

$$B_1(x) + B_2^*(x) = [A_1(x) + A_3(x)]\ G(x) \quad \ldots\ldots(11)$$

In (11) the right hand side is some code word polynomial, say, $V_t(x)$. Therefore (11) can be written as

$$B_1(x) + B_2^*(x) = V_t(x) \quad \ldots\ldots(12)$$

Recalling that no code word can be the sum of two correctable error patterns, or the sum of a correctable pattern and a detectable pattern, we can now say, on the basis of (12), that if $B_1(x)$ is correctable, then $B_2^*(x)$ is not correctable, and vice versa.

Now let us consider an SBEC code with length as prescribed by (6a). Assuming that only correctable and detectable errors can occur, and remembering that in such a code, every correctable error has to occur as $B_1(x)$ or $B_2^*(x)$ only, the decoding procedure would be as follows:

Step 1

Get $B_1(x)$, by reducing $R(x)$ modulo $G(x)$; that is

$$\frac{R(x)}{G(x)} = A_1(x) + \frac{B_1(x)}{G(x)}\ .$$

If $B_1(x) = 0$, then no error has occurred.

If $B_1(x)$ = Correctable pattern, then $B_1(x)$ is the actual error.

If $B_1(x)$ = Non correctable pattern, proceed to step 2.

Step 2

Get $B_2(x)$, by reducing $R^*(x)$ modulo $G^*(x)$; that is,

$$\frac{R^*(x)}{G^*(x)} = A_2(x) + \frac{B_2(x)}{G^*(x)}$$

If $B_2(x)$ = Correctable pattern, then $B_2{}^*(x) = x^{n_s-1} B_2(\frac{1}{x})$

is the actual error.

If $B_2(x)$ = Non Correctable pattern, then a detectable, but not correctable error has occurred.

A flow diagram of the procedure, is shown in Fig. 3.4.

3.4. Some Comments About the Length of the Shortened Code

With reference to (6a) and (6b) we have, as already stated, the transmission rate given by

$$\frac{k_s}{n_s} = \frac{g + 1 - l_c}{2g + 1 - l_c} \quad \text{.....(13)}$$

It is clear that the rate tends to the maximum value of $\frac{1}{2}$ as g gets larger for a given l_c. In other words, it is desirable to have as large a g as possible.

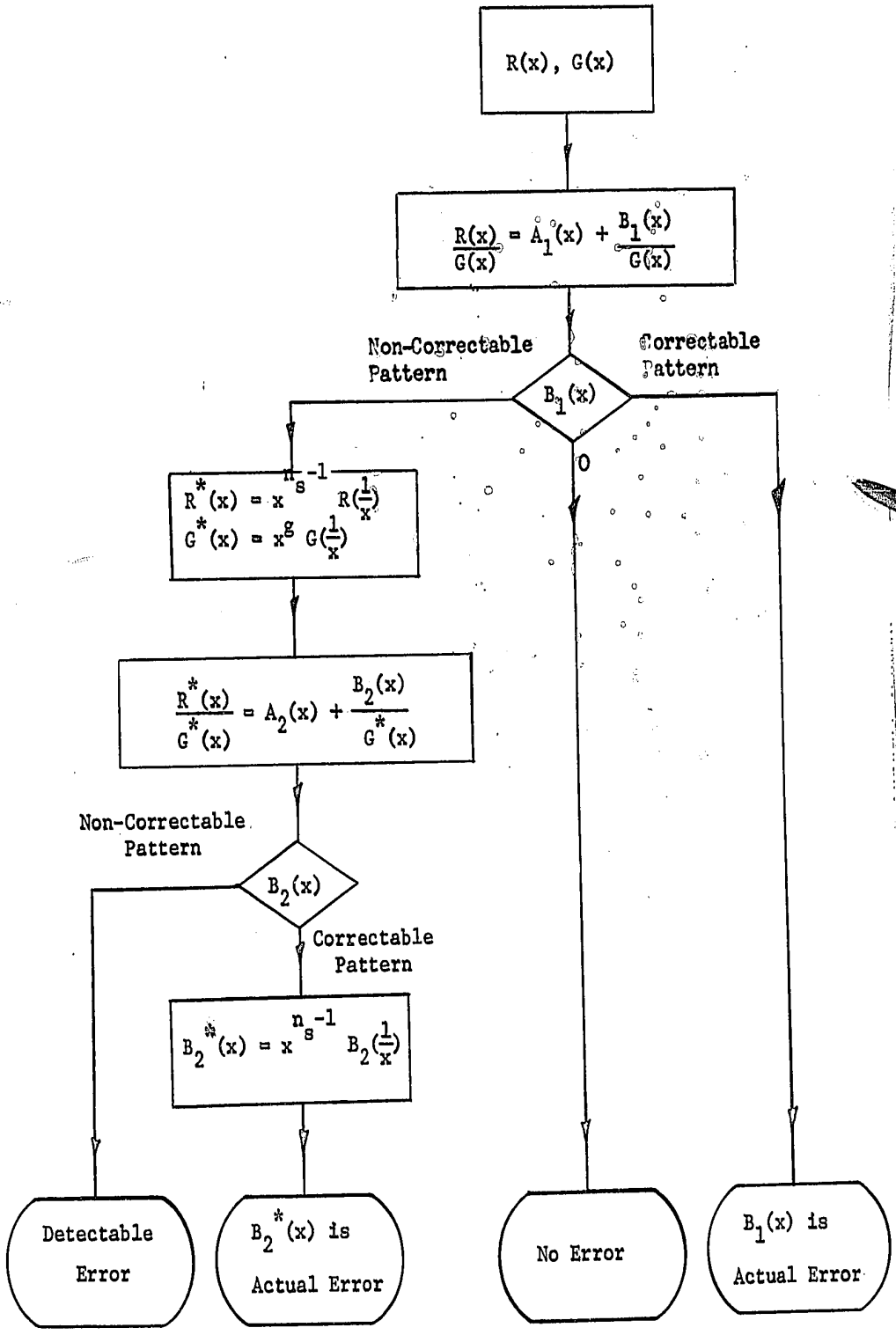


Fig. 3.4 Flow Diagram of Decoding Procedure

Abramson codes, $l_c = 3$, are generated by

$$G_A(x) = P_A(x)(1 + x + x^2) \quad \ldots\ldots(14)$$

Where $P_A(x)$ is a primitive polynomial of degree even q.

For these codes,

$$n_s = 2g + 1 - l_c$$

$$= 2(q + 2) + 1 - l_c$$

or for $l_c = 3$

$$n_s = 2(q + 1)$$

and

$$k_s = g + 1 - l_c$$

$$= q + 2 + 1 - 3$$

or

$$k_s = q$$

so that, the transmission rate for Abramson code is

$$\frac{k_s}{n_s} = \frac{1}{2} \quad \frac{q}{q + 1} \quad \ldots\ldots(15)$$

From (15), it is clear that the maximum possible rate of $\frac{1}{2}$, can be nearly achieved without making q unduly large, assuming that the code exists for the chosen q.

For Fire codes, the generating polynomial

$G_F(x)$ has the form

$$G_F(x) = P_F(x)(1 + x^{\alpha}) \quad \ldots\ldots(16)$$

where $P_F(x)$ is a primitive polynomial of degree $\geq l_c$, whose roots have order $\underline{e}$ which does not divide α.

Also $\alpha \geq l_c + l_d - 1$

$$l_d \geq l_c.$$

Taking $l_d = l_c + \beta \qquad \beta > 0$

and $\alpha = l_c + l_d - 1$

or $\alpha = 2l_c + \beta - 1$

we can write

$$g = l_c + \alpha = 3l_c + \beta - 1 \quad \ldots\ldots (17)$$

Then

$$n_s = 2g + 1 - l_c$$

or

$$n_s = 5l_c + 2\beta - 1 \quad \ldots\ldots (18)$$

and

$$k_s = g + 1 - l_c$$

or $k_s = 2l_c + \beta \quad \ldots\ldots (19)$

so that the transmission rate for a Fire code is

$$\frac{k_s}{n_s}_F = \frac{2\,l_c + \beta}{5\,l_c + 2\beta - 1} \quad \ldots\ldots (20)$$

To get as high a rate as possible in (20) 2β has to be $\gg 5\,l_c - 1$ for a given l_c. When $l_c = 3$, the rate for a Fire code is given, from (20), by

$$\left(\frac{k_s}{n_s}\right)_F = \frac{1}{2} \cdot \frac{\beta + 6}{\beta + 7} \quad \text{...(21)}$$

Using (15) and (21) we can get a comparison between the lengths of Fire and Abramson Codes if both transmission rates were equal for the same $1_c = 3$. Equating (15) to (21), we can write

$$\beta = q - 6$$

We found the expression for the shortened length of an Abramson code for $1_c = 3$ to be

$$(n_s)_A = 2(q + 1)$$

From (18) for $1_c = 3$ $\quad (n_s)_F = 2(\beta + 7)$

Using $\beta = q - 6$, we notice that

$$(n_s)_A = (n_s)_F = 2(q + 1)$$

i.e. for $1_c = 3$ a Fire code is as long as an Abramson code for the same transmission rate.

3.5. An example

As an example of an SBEC code, let us consider the Fire code generated by

$$G_F(x) = (1 + x + x^3)(1 + x^6) \quad \text{..............................(22)}$$

for which we have

$$g = 9, \quad 1_c = 3, \quad 1_d = 4, \quad \beta = 1$$

With reference to (18), (19), and (20)

$$n_s = 16, \qquad k_s = 7$$

and $$\frac{k_s}{n_s} = \frac{7}{16} = 0.4375.$$

The rate could have been, of course, increased by taking a higher β, as against the present value of 1.

To see how the decoding procedure works, let us consider a code word polynomial, say

$$V(x) = G_F(x)(1 + x^3)$$

which after using (22) becomes

$$V(x) = 1 + x + x^4 + x^7 + x^{10} + x^{12} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(23)$$

During transmission, an error sequence gets added to V(x). Suppose the error is

$$E(x) = x^6 + x^9 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(24)$$

which is detectable but not correctable.

The received polynomial is

$$R(x) = V(x) + E(x),$$

or

$$R(x) = 1 + x + x^4 + x^6 + x^7 + x^9 + x^{10} + x^{12} \quad \ldots\ldots\ldots\ldots\ldots(25)$$

Using the information which it has access to,

i.e. R(x), G(x), l_c and n, the decoder has to detect and possibly correct the error, using the above procedure.

Going through step 1, we get

$$B_1(x) = 1 + x + x^3 + x^7 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(26)$$

which is not correctable. Therefore we go to step 2 and get

$$B_2(x) = 1 + x^2 + x^3 + x^8 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(27)$$

which is not correctable also. Therefore we conclude that a detectable, but not correctable, error has occurred.

On the other hand, suppose the error pattern is

$$E(x) = x^7 + x^9 \quad \ldots\ldots(28)$$

which is correctable. Then the received polynomial

$$R(x) = V(x) + E(x)$$
$$= 1 + x + x^4 + x^9 + x^{10} + x^{12} \quad \ldots\ldots(29)$$

Going through step 1, in the decoding procedure

$$B_1(x) = 1 + x + x^3 + x^6 \quad \ldots\ldots(30)$$

which is not correctable.

Step 2 gives

$$B_2(x) = x^6 + x^8 \quad \ldots\ldots(31)$$

which is correctable.

Therefore the error pattern is given by

$$B_2^*(x) = x^7 + x^9 \quad \ldots\ldots(32)$$

where $B_2^*(x) = x^{n_s-1} B_2\left(\frac{1}{x}\right)$ $\qquad n_s = 2_g + 1 - 1_c = 16$

and the original transmitted vector is obtained by adding

$B_2^*(x) = E(x)$ to the received vector $R(x)$

$R(x) + E(x) = V(x) + E(x) + E(x) = V(x)$.

4. CONCLUDING REMARKS

From the material presented in the thesis, we can make the following concluding remarks:

(i) An SBEC code of length n_s given by (6a) is advantageous from the point of view of decoding; the procedure consists of at most two steps, whereas in the case of parent cyclic codes, it requires on the average $\frac{n}{2}$ steps.

(ii) Because of the shortened length, the amount of computation in each of the two steps, is considerably less than in the case of computing the remainder $B_1(x)$ in a cyclic code.

(iii) The price to be paid for the above advantages, is a degradation in the transmission rate; at most it can be one half. It may, however, be noted that this maximum value can very nearly be achieved.

(iv) It was observed that in the case of $l_c = 3$, a Fire code is as long as an Abramson code, for the same transmission rate.

(v) This simple decoding procedure may have various uses in a number of codes. The possible use of SBEC codes in synchronization problems is being investigated presently by S.G.S. Shiva and G. Seguin.

REFERENCES

1. Reza, F. M., "An Introduction to Information Theory", McGraw-Hill Co., Inc., New York 1961.
2. Shiva, S. G. S., and T. Zeitoun, "Shortened Cyclic Burst Error Correcting Binary Codes of Certain Lengths". Technical Report No. 68-5, Dept. of Electrical Engineering, Univeraity of Ottawa, Ottawa, Canada, May 1968.
3. Birkhoff, G., and S. MacLane, "A Survey of Modern Algebra", MacMillan, New York, 1941.
4. van der Waerden, B. L., "Modern Algebra", Volume 1, translated by F. Blum, Frederick Ungar Publishing Co., New York, 1949.
5. Slepian, D., "A class of Binary Signalling Alphabets", Bell System Technical Journal, 35, 203-234, 1956.
6. Peterson, W. W., "Error Correcting Codes:, The M.I.T. Press, 138, 1961.
7. op. cit. 159-160.
8. Hamming, R. W., "Error Detecting and Error Correcting Codes", Bell Systems Technical Journal, 29, 147-160, 1950.
9. Bose, R. C., and D. K. Chaudhuri, "On a Class of Error Correcting Binary Group Codes", Information and Control, 3, 68-79, 1960.
10. Bose, R. C., and D. K. Chaudhuri, "Further Results on Error Correcting Binary Group Codes", Information and Control, 3, 279-290, 1960.
11. Hocqueughem, A., "Codes Correcteurs d'Erreurs", Chiffres, 2, 147-156, 1959.

12. Peterson, W. W., "Encoding and Error Correction Procedures for the Bose-Chaudhuri Codes", IRE Transactions on Information Theory, Sept. 1960.

13. Peterson, W. W. "Error Correcting Codes", The M.I.T. Press, 169-180, 1961.

14. Shiva, S. G. S., "Certain Group Codes", Proceedings of the IEEE, Vol. 55, No. 12, 2162-2163, Dec. 1967.

15. Abramson, N.M., "A Class of Systematic Codes for Non-Independent Errors", IRE Transactions, IT-5, 150-157, 1959.

16. Abramson, N. M., "Error Correcting Codes from Linear Sequential Networks", presented at the Fourth London Symposium on Intromation Theory, August 1960.

17. Fire, P., "A Class of Multiple-Error-Correcting Binary Codes for Non-Independent Errors", Sylvania Report RSL-E-2, Sylvania Reconnaissance Systems Laboratory, Mountain View, California, 1959.

18. Peterson, W. W., "Error Correcting Codes", The M.I.T. Press, 183-186, 1961.

19. op. cit., 184-185.

20. Ash, R. B., "Information Theory" Interscience Publishers, Division of John Wiley and Sons, 92, 1965.

21. Peterson, W. W., "Error Correcting Codes", The M.I.T. Press, 201-204, 1961.

22. op. cit, 189-196

23. Shiva, S. G. S. and C. L. Cheng, "Multiple Solid Burst Error Correcting Binary Codes", (Correspondence) to appear in IEEE Transaction on Information Theory, Nov. 1968 or Jan. 1969.

24. Schillinger, A. G., "A Class of Solid Burst Error Correcting Codes", Research Report PIBMRI-1223-64, Polytechnic Institute of Brooklyn, New York, April 1964.

25. Peterson, W. W., "Error Correcting Codes", The M.I.T. Press, 191-196, 1961.

APPENDIX

The appendix consists of a typical application of the decoding procedure performed on the IBM 360/65 digital computer. The program is written in FORTRAN IV language and is easily described by flow chart in Fig. A.1.

The computer is being considered to form part of the receiving section of a real communications system. It may be visualized as being connected to the receiver, thus accepting as input a block of code words which may or may not have been perturbed by noise after a simulated noisy transmission channel. The purpose of the program then is to perform the two steps of the decoding procedure on each received word and give as output the corrected code word. In the case of those received words which may have been corrupted by a non-correctable error pattern, a note to that effect will be printed. Therefore, a glance at the computer print-out sheet will tell the operator which words were received wrongly and their corrected version if possible. In the case of occurrence of non-correctable errors, the operator's attention will be drawn to the printed note to that effect, which would enable him to put in a retransmission request.

The code being considered is an Abramson cyclic burst error correcting binary code capable of correcting any error burst of length $l_c \leq 3$. It is generated by the polynomial

$$G_A(x) = (1 + x + x^4)(1 + x + x^2)$$

or

$$G_A(x) = 1 + x^3 + x^4 + x^5 + x^6$$

This polynomial will generate an (n_s, k_s) cyclic BEC code of length

$$n_s = 2g + 1 - l_c = 10$$

having k information digits where

$$k = g + 1 - l_c = 4$$

The corresponding (10x4) generator matrix will consist of the coefficients of the generating polynomial with its 3 cyclic shifts i.e.

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

One of the advantages of this decoding procedure is that at the receiving side, we only need to store the degree of G(x), g = 6 in this case, and the maximum length of error bursts which this code is capable of correcting, i.e. $l_c = 3$. The following is the computer program, with the printout results.

Meaning of Symbols Used in Computer Program

IDEG = Degree of Generating Polynomial

LC = Maximum Length of Burst Pattern

NW = Number of Received Words

ILHV = Length of every Received Word

IDIV = Received Polynomial coefficients

IGX = Generating Polynomial coefficients

IRDIV = Reciprocal of Received Polynomial

IRGX = Reciprocal of Generating Polynomial

IRX = Remainder of Division Operation in Step 1

IRRX = Remainder of Division Operation in Step 2

IEROR = Actual Error

ICV = Corrected Vector

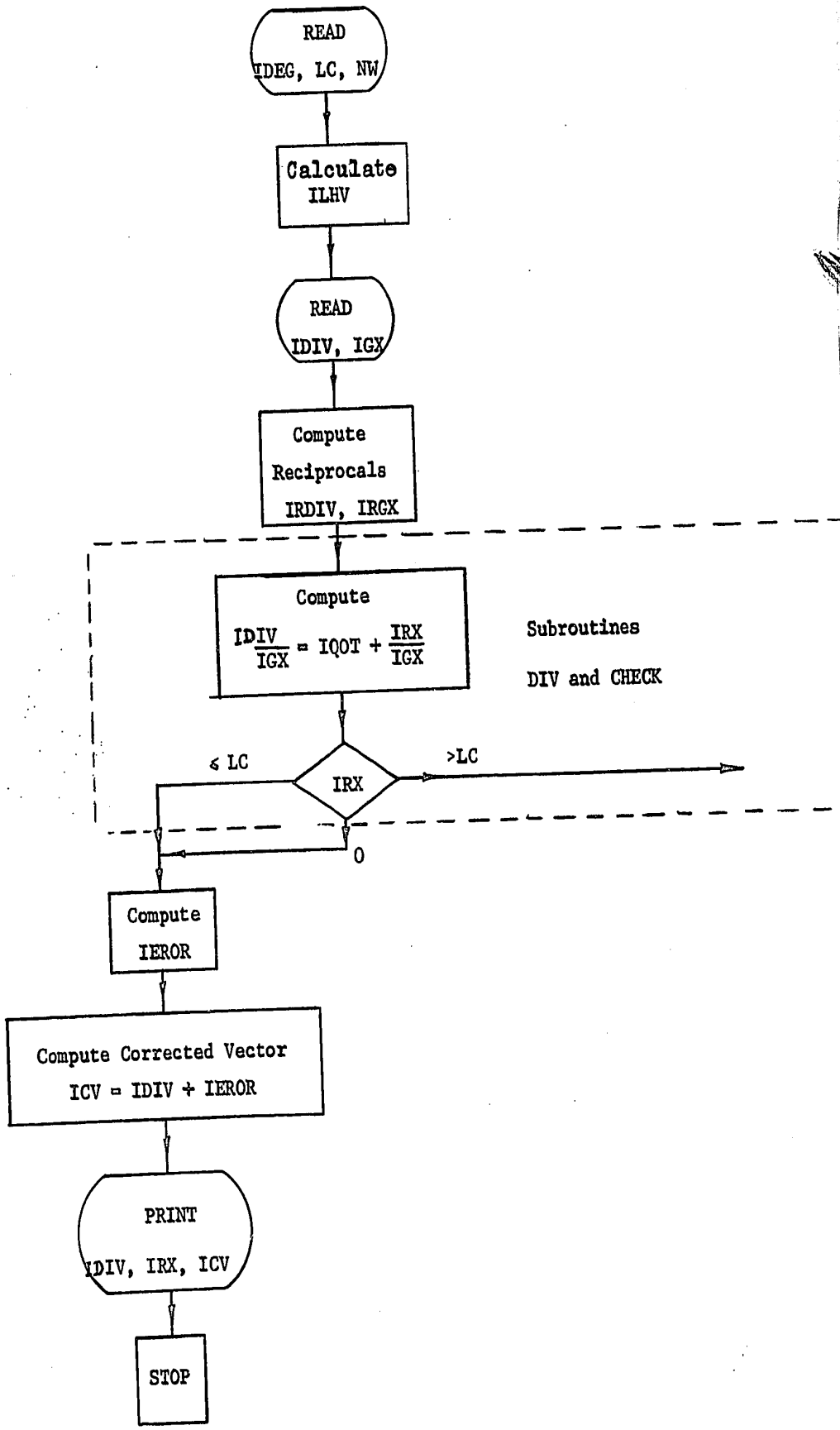


Fig. A.1 Computer Program Flow Chart of Decoding Procedure

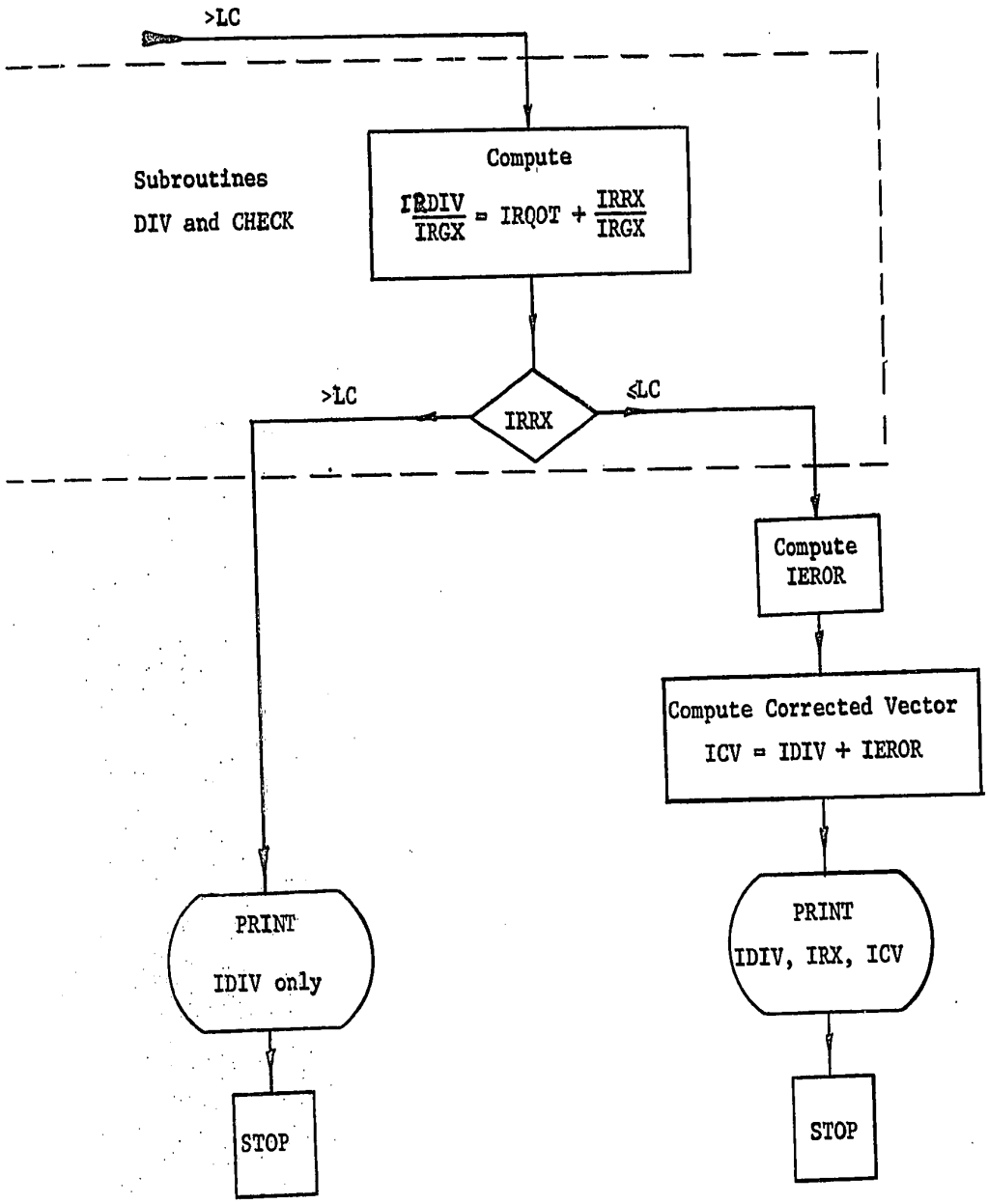


Fig. A.1 Con't Computer Program Flow Chart of Decoding Procedure

```
      DIMENSION IGX(20),IRGX(20),IRX(10),IRRX(10),IV(50,20),IDIV(20),IRD
     1IV(20),IQOT(20),IRQOT(20),IEROR(20),ICV(20)
      DO 11 L=1,4
C     READ DEG. OF GEN. POLYNOMIAL, MAX. LENGTH OF ERROR BURST,
C     AND NUMBER OF RECEIVED CODE WORDS.
      READ(1,1) IDEG,LC,NW
    1 FORMAT(3I10)
      ILHV=2*IDEG-LC+1
      K=IDEG+1
C     READ DIGITS OF ALL RECEIVED WORDS
      READ(1,2)((IV(I,J),J=1,ILHV),I=1,NW)
    2 FORMAT(10I1)
C     READ DIGITS OF GEN. POLYNOMIAL
      READ(1,2)(IGX(I),I=1,K)
      WRITE(3,60)
   60 FORMAT(1H1,25X,'DECODING  PROCEDURE'//)
      WRITE(3,61)
   61 FORMAT(1H ,34X,'FOR'//)
      WRITE(3,62)
   62 FORMAT(1H ,12X,'SHORTENED CYCLIC BURST ERROR CORRECTING BINARY COD
     1ES'//)
      WRITE(3,63)
   63 FORMAT(1H ,12X,'SIMULATED TRANSMISSION THROUGH A NOISY CHANNEL OF
     1A S'//)
      WRITE(3,64)
   64 FORMAT(1H ,12X,'SHORTENED VERSION OF AN ABRAMSON CODE WHOSE'//)
      WRITE(3,65)
   65 FORMAT(1H ,12X,'GENERATOR POLYNOMIAL IS:'//)
      WRITE(3,66)
   66 FORMAT(1H ,22X,'G(X)=(1+X+X4)(1+X+X2)'//)
      WRITE(3,67)
   67 FORMAT(1H ,17X,'OR   G(X)=(1+X3+X4+X5+X6)'//)
      WRITE(3,68)
   68 FORMAT(1H ,12X,'THE CODE IS CAPABLE OF CORRECTING ALL ERROR'//)
      WRITE(3,69)
   69 FORMAT(1H ,12X,'BURSTS OF LENGTH LC=3 OR LESS.'//)
      WRITE(3,70)
   70 FORMAT(1H ,12X,'THE LENGTH OF EACH WORD IN THE CODE IS     NS=15'//
     1)
      WRITE(3,71)
   71 FORMAT(1H ,12X,'THE NUMBER OF INFORMATION DIGITS IS        KS= 4'//
     1)
      WRITE(3,72)
   72 FORMAT(1H ,12X,'THE NUMBER OF WORDS IN THE CODE SPACE IS  N =16'//
     1)
      WRITE(3,4)
    4 FORMAT(1H1,12X,'RECEIVED  VECTOR',10X,'ERROR  VECTOR',11X,'CORRECT
     1ED  VECTOR'///)
      NUM=ILHV-IDEG
C     GET RECIPROCAL OF GENERATING POLYNOMIAL
      DO 5 I=1,K
    5 IRGX(I)=IGX(K-I+1)
      DO 10 I=1,NW
      DO 15 J=1,ILHV
      IEROR(J)=0
      IDIV(J)=IV(I,ILHV-J+1)
C     GET RECIPROCAL OF RECEIVED VECTORS
   15 IRDIV(J)=IV(I,J)
```

```
C     CALLING DIVISION SUBROUTINE, STEP 1 OF DECODING PROCEDURE
      CALL DIV(IDIV,IGX,ILHV,IDEG,NUM,IRX,IQOT)
C     CALLING SUBROUTINE FOR CHECKING IF REMAINDER IS A CORRECTABLE
C     ERROR PATTERN
      CALL CHECK(IRX,IDEG,LC,$41,$40)
C     STEP 2 OF DECODING PROCEDURE
   40 CALL DIV(IRDIV,IRGX,ILHV,IDEG,NUM,IRRX,IRQOT)
      CALL CHECK(IRRX,IDEG,LC,$45,$50)
   41 DO 42 J=1,IDEG
C     OBTAIN ERROR PATTERN
   42 IEROR(ILHV-IDEG+J)=MOD(IEROR(ILHV-IDEG+J)+IRX(J),2)
      DO 20 J=1,ILHV
C     OBTAIN CORRECTED VECTOR
   20 ICV(J)=MOD((IDIV(J)+IEROR(J)),2)
C     PRINT : RECEIVED VECTOR,ERROR VECTOR AND CORRECTED VECTOR
      WRITE(3,100)(IRDIV(J),J=1,ILHV),(IEROR(ILHV-J+1),J=1,ILHV),(ICV(
     1ILHV-J+1),J=1,ILHV)
  100 FORMAT(1H0,10X,10I2,5X,10I2,5X,10I2///)
      GO TO 10
   45 DO 43 J=1,IDEG
   43 IEROR(J)=MOD(IEROR(J)+IRRX(IDEG-J+1),2)
      DO 25 J=1,ILHV
   25 ICV(J)=MOD((IDIV(J)+IEROR(J)),2)
      WRITE(3,100)(IRDIV(J),J=1,ILHV),(IEROR(ILHV-J+1),J=1,ILHV),(ICV(
     1ILHV-J+1),J=1,ILHV)
      GO TO 10
C     IF ERROR PATTERN IS NON CORRECTABLE , PRINT SO.
   50 WRITE(3,110)(IRDIV(J),J=1,ILHV)
  110 FORMAT(1H0,10X,10I2,5X,'A NON CORRECTABLE ERROR HAS OCCURRED'///)
   10 CONTINUE
   11 CONTINUE
      RETURN
      END
```

****

```
      SUBROUTINE DIV(IDID,IDIR,M,L,K,IRER,IQOT)
      DIMENSION IDID(20),IDIR(20),IRER(10),IQOT(20),IDI(20)
      DO 131 I=1,M
  131 IDI(I)=IDID(I)
      DO 132 I=1,K
      IQOT(I)=IDI(I)
      DO 133 J=1,L
  133 IDI(I+J)=MOD((IDI(I+J)+IQOT(I)*IDIR(J+1)),2)
  132 CONTINUE
      DO 134 I=1,L
  134 IRER(I)=IDI(K+I)
      RETURN
      END
```

****

```
      SUBROUTINE CHECK(IRER,M,N,*,*)
      DIMENSION IRER(10)
      K=0
      I=0
  128 I=I+1
      IF(K-1)135,120,135
  135 IF(IRER(I))125,125,110
  110 IF(I+N-M)115,115,125
  115 I=I+N
      K=1
  120 IF(IRER(I))125,125,130
  125 IF(M-I)126,126,128
  126 RETURN 1
  130 RETURN 2
      END
```

****

6 3 16

0010101111
0010011110
0000010001
0100101000
0010110101
0110100011
0100101101
1001010000
0100110111
1011100110
1101001001
1101001101
1101111011
1111011100
1100010101
0000001110
1111001

DECODING PROCEDURE

FOR

SHORTENED CYCLIC BURST ERROR CORRECTING BINARY CODES

SIMULATED TRANSMISSION THROUGH A NOISY CHANNEL OF A

SHORTENED VERSION OF AN ABRAMSON CODE WHOSE

GENERATOR POLYNOMIAL IS:

$$G(X)=(1+X+X4)(1+X+X2)$$

OR $G(X)=(1+X3+X4+X5+X6)$

THE CODE IS CAPABLE OF CORRECTING ALL ERROR

BURSTS OF LENGTH LC=3 OR LESS.

THE LENGTH OF EACH WORD IN THE CODE IS NS=10

THE NUMBER OF INFORMATION DIGITS IS KS= 4

THE NUMBER OF WORDS IN THE CODE SPACE IS N =16

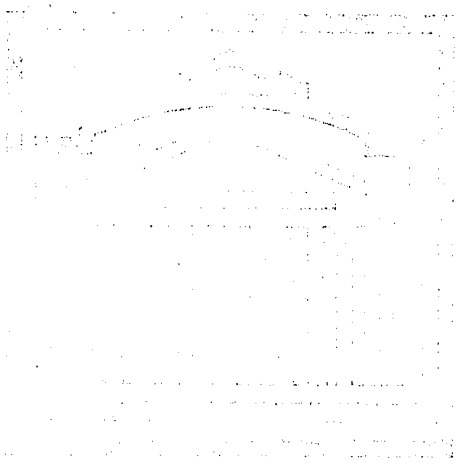
RECEIVED VECTOR	ERROR VECTOR	CORRECTED VECTOR
0 0 1 0 1 0 1 1 1 1	0 0 1 1 1 0 0 0 0 0	0 0 0 1 0 0 1 1 1 1
0 0 1 0 0 1 1 1 1 0	0 0 0 0 0 0 0 0 0 0	0 0 1 0 0 1 1 1 1 0
0 0 0 0 0 1 0 0 0 1	0 0 1 1 0 0 0 0 0 0	0 0 1 1 0 1 0 0 0 1
0 1 0 0 1 0 1 0 0 0	0 0 0 0 0 1 0 1 0 0	0 1 0 0 1 1 1 1 0 0
0 0 1 0 1 1 0 1 0 1	A NON CORRECTABLE ERROR HAS OCCURRED	
0 1 1 0 1 0 0 0 1 1	0 0 0 0 0 0 0 0 0 1	0 1 1 0 1 0 0 0 1 0
0 1 0 0 1 0 1 1 0 1	0 0 1 1 0 0 0 0 0 0	0 1 1 1 1 0 1 1 0 1
1 0 0 1 0 1 0 0 0 0	0 0 0 0 1 0 1 0 0 0	1 0 0 1 1 1 1 0 0 0
0 1 0 0 1 1 0 1 1 1	1 1 0 0 0 0 0 0 0 0	1 0 0 0 1 1 0 1 1 1
1 0 1 1 1 0 0 1 1 0	0 0 0 0 0 0 0 0 0 0	1 0 1 1 1 0 0 1 1 0
1 1 0 1 0 0 1 0 0 1	A NON CORRECTABLE ERROR HAS OCCURRED	
1 1 0 1 0 0 1 1 0 1	A NON CORRECTABLE ERROR HAS OCCURRED	

1101111011 0001110000 1100001011

1111011100 0000000110 1111011010

1100010101 0010000000 1110010101

0000001110 0000001110 0000000000



VITA

Name:	**ZEITOUN, Tony**
Date of birth:	**February 4, 1941.**
Place of birth:	**Nazareth Israël.**
Education:	
Secondary:	**Terra Santa, 1958.**
High School:	**Nazareth Israël**
B.A.Sc. (E.E.)	**University of Ottawa, 1963.**