

1990

# On algorithms for minimal partial realization and BCH decoding

Elias George Rogaris  
*Lehigh University*

Follow this and additional works at: <https://preserve.lehigh.edu/etd>



Part of the [Electrical and Computer Engineering Commons](#)

---

## Recommended Citation

Rogaris, Elias George, "On algorithms for minimal partial realization and BCH decoding" (1990). *Theses and Dissertations*. 5316.  
<https://preserve.lehigh.edu/etd/5316>

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact [preserve@lehigh.edu](mailto:preserve@lehigh.edu).

ON ALGORITHMS FOR MINIMAL PARTIAL REALIZATION AND BCH  
DECODING

by

Elias George Rogaris

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

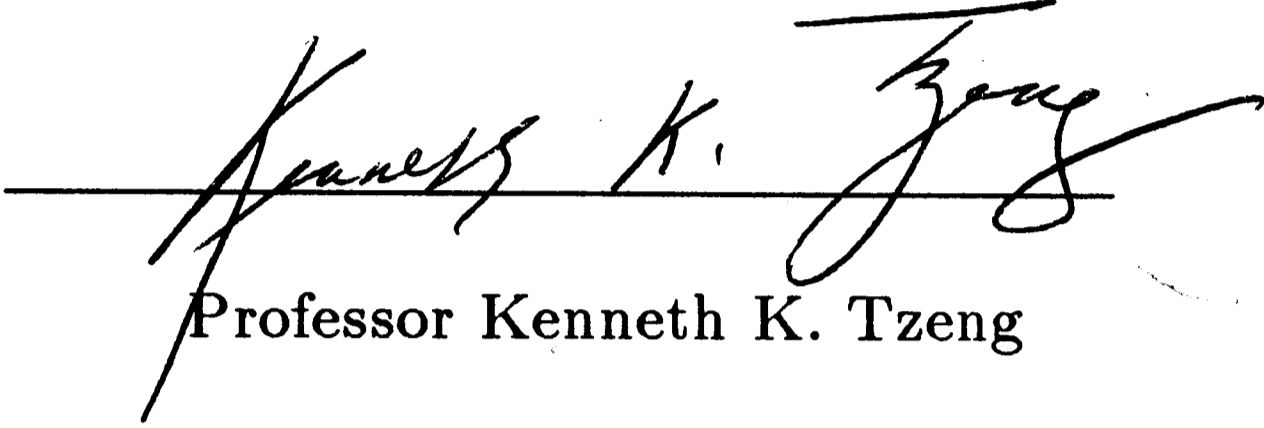
in

Electrical Engineering

Lehigh University

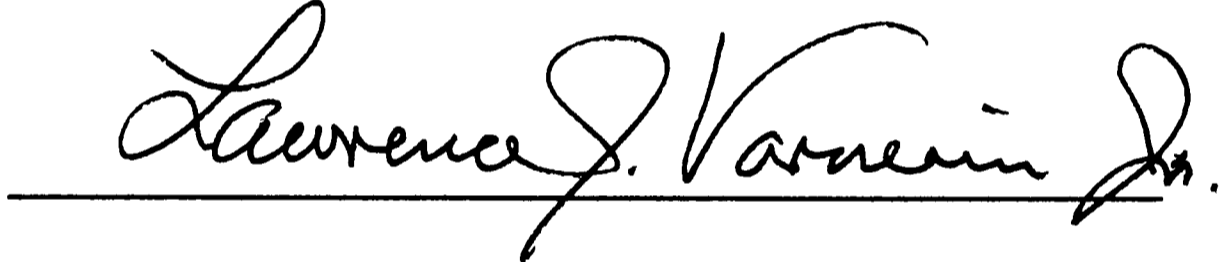
1989

The following thesis is approved for partial fulfillment of the Master's of Science in  
Electrical Engineering.

  
Professor Kenneth K. Tzeng

Advisor

CSEE Department

  
Professor Lawrence J. Varnerin Jr.

Professor Lawrence J. Varnerin Jr.

Department Chair

CSEE Department

## ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Kenneth Tzeng, for his help, advice and guidance during the preparation of this thesis. I would also like to thank G. L. Feng for his assistance in this effort. Finally, I would like to thank Mr. Nicolas Athanassiadis and my parents George and Vassiliki Rogaris for their moral and financial support.

Elias G. Rogaris

## TABLE OF CONTENTS

CERTIFICATE OF APPROVAL	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	v
LIST OF TABLES	vi
ABSTRACT	1
I. INTRODUCTION	2
II. PRELIMINARIES	4
II.1 Fundamentals of Coding	4
1. Block Codes	5
2. Cyclic Codes	6
3. BCH Codes	7
II.2 Algebraic Theory in Linear Systems	9
III. Algorithms	13
III.1 Ho and Kalman Algorithm	14
III.2 Rissanen Algorithm	19
III.3 Conan Algorithm	24
III.4 Berlekamp-Massey Algorithm	27
III.5 Fundamental Iterative Algorithm	30
IV. Relationship between Conan's MPR and FIA	34
V. Conclusion	48
REFERENCES	49
VITA	51

## FIGURES

II.1	Communications System	4
IV.1	Discrepancy Matrix	36
.2	D Matrix for Case 1	40
.3	D Matrix for Case 2	41
.4	MRP Algorithm as derived from FIA	46
.5	Conan's MPR Algorithm	47

## TABLES

I.	Conan's MPR Algorithm Solutions	27
II.	Berlekamp's Algorithm Solutions	30
III.	FIA Solutions	33

## ABSTRACT

The most well known algorithm for decoding BCH codes is the Berlekamp-Massey iterative algorithm. The Berlekamp-Massey iterative algorithm can also be used to solve the realization problem in systems. There are two other algorithms that can be used to decode BCH codes and in solving systems problems. First, the Minimal Partial Realization (MPR) algorithm is designed to solve all types of realization problems and decode BCH codes as well. Second, the Fundamental Iterative Algorithm (FIA) is a more general algorithm than the Berlekamp-Massey iterative algorithm and MPR algorithms and thus the FIA has some additional capabilities. The different versions of the Minimal Partial Realization algorithm are discussed. The relationship between Conan's Minimal Partial Realization algorithm and the Fundamental Iterative Algorithm is shown.



## Chapter I - Introduction

The development of today's communications systems is attributed, among others things, to the use of codes. Codes are being used mainly in order to avoid erroneous transmission, or to achieve secure transmission. Different techniques have been developed for encoding and decoding codes.

One class of codes used extensively in communication systems is the BCH codes. After their discovery in 1960, algorithms were proposed for decoding these codes. The most effective algorithm for this purpose is the Berlekamp-Massey iterative algorithm [1,2] which was introduced in 1968. Because of its effectiveness, the same algorithm was used to solve the realization ~~problem~~ in the control systems area.

An alternative algorithm designed for solving the realization problem in control systems was presented in 1966 by Ho and Kalman [3]. This algorithm is known as the Minimal Partial Realization (MPR) algorithm. In 1971, Rissanen [4] presented an improved version of the MPR algorithm. Generalized versions of the MPR algorithm were presented in 1974 by Dickinson, Morf and Kailath [5] and in 1980 by Conan [6]. Finally, in 1984 Conan [7] presented an effective and easily implemented MPR algorithm, that can be as effective as the Berlekamp-Massey algorithm in control systems applications. This algorithm can be used as an alternative to Berlekamp-Massey algorithm even for BCH codes decoding.

A basic algorithm was introduced in 1984 by Feng and Tzeng [8], named as the Fundamental Iterative Algorithm (FIA). This algorithm is more general than the Berlekamp-Massey algorithm since it can make use of multiple syndrome sequences for decoding cyclic codes.

This thesis includes in Chapter II a review of the background material needed for describing the algorithms used in BCH decoding as well as in systems problem solving. Section one is devoted to error-control coding, including reviews of linear and cyclic codes. It also describes the BCH codes and their decoding problem. Section two introduces the reader to algebraic theory for linear systems. It also includes the necessary definitions and states the problem that needs to be solved.

Chapter III describes the algorithms that can be used effectively for BCH decoding as well as system problem solving. It includes, among others, the Berlekamp-Massey algorithm, the Ho and Kalman algorithm, the Rissanen algorithm, the Conan algorithm, and finally the Fundamental (or Feng-Tzeng) Iterative Algorithm.

Chapter IV presents the relationship between the Minimal-Partial-Realization algorithm developed by Conan and the Fundamental Iterative Algorithm. In particular, Conan's algorithm is derived from the more general Fundamental Iterative Algorithm.

Finally, Chapter V contains the concluding remarks and proposes problems for further research.

## Chapter II - Preliminaries

In this chapter we present some preliminary concepts that are needed to discuss the decoding algorithms or the system solving algorithms. We start with the discussion of error-detecting and error-correcting codes, with some emphasis on BCH codes. The encoding of codes will be presented. In addition, the decoding problem will be stated. The decoding algorithms will be presented in another chapter.

The second half of this chapter is a discussion of the algebraic theory of linear systems. We present the problem in the broad area of linear system theory as well as in the more specific realization theory.

### II.1 Fundamentals of coding

In digital communication systems there is a need to be able to detect and/or correct any errors introduced due to noise in the channel. For this reason error-detection and error-correction codes were developed.

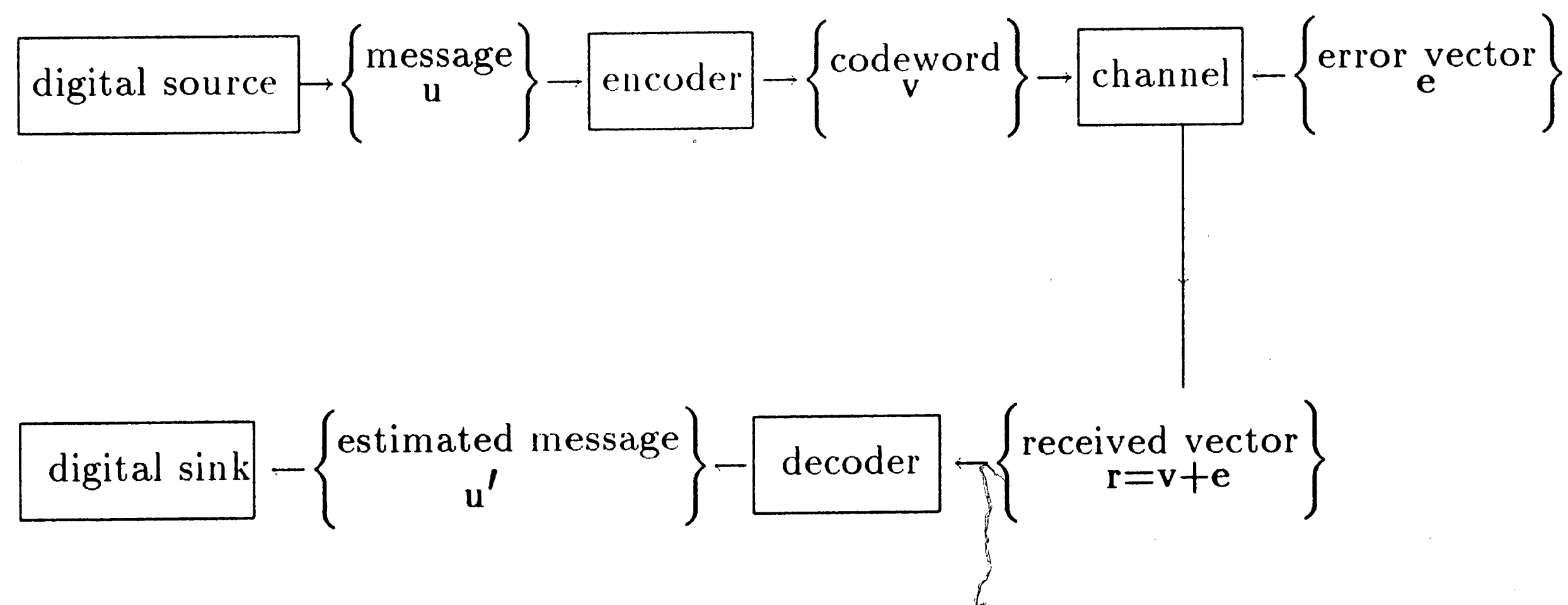


Fig.1.

### II.1.1. Block Codes

Block codes are widely used in today's communications. Of these codes we restrict our attention to a subclass of block codes, the linear block codes. An  $(n,k)$  linear block code is defined as a  $k$ -dimensional subspace of the vector space of all the  $n$ -tuples over the field  $GF(q)$  if and only if it is formed by  $q^k$  codewords and has a length  $n$ .

Since an  $(n,k)$  linear code is a  $k$ -dimensional subspace of the vector space, it is possible to find  $k$  linearly independent codewords such that they form a  $k \times n$  matrix that operates all codewords and call it the generator matrix  $G$ . If  $G$  is of the form  $(P \ I_k)$  then it is said to be in systematic form. From  $G$  we can obtain the parity check matrix  $H = (I_{n-k}, P^T)$ .  $G$  and  $H$  are related with the property  $G \cdot H^T = 0$ .

In order to determine the random-error-detecting and random-error-correcting capabilities of a code we introduce the minimum distance. First, we define the Hamming weight of  $v$  or  $w(v)$  as the number of nonzero components of  $v$ . The Hamming distance between the codewords  $v$  and  $w$ , denoted  $d(v,w)$ , is defined as the number of places they differ.

Also from the definition of the Hamming distance and the definition of modulo-2 addition we conclude that the Hamming distance between  $v$  and  $w$  is equal to the Hamming weight of the sum  $v$  and  $w$ , that is  $d(v,w) = w(v+w)$ . The minimum distance of a code  $C$ , denoted  $d_{min}$  is defined as

$$d_{min} = \min \{ d(v,w) : v,w \in C, v \neq w \}.$$

Consequently, the minimum distance of a linear code is equal to the minimum weight of the nonzero codewords ( $d_{min} = w_{min}$ ).

The random-error-detecting and random-error-correcting capabilities of a linear block code can be stated as follows.

A linear block code with minimum distance  $d_{min}$  guarantees detecting all error patterns of  $d_{min}-1$  or fewer errors. It is possible that the code can detect more errors for certain patterns. The same code guarantees correcting all error patterns of  $t = \lfloor (d_{min}-1)/2 \rfloor$  ( $t$  is always integer) or fewer errors. Certain error patterns of  $t+1$  or more errors can be corrected under special conditions.

### II.1.2. Cyclic Codes

Cyclic codes are an important subclass of linear codes. These codes are attractive for two reasons: first, encoding and syndrome computation can be implemented by employing shift registers; and second, they have such an algebraic structure that allows us to find many practical methods for decoding theory.

A linear code  $C$  is cyclic if every cyclic shift of a code vector in  $C$  is also a vector in  $C$ . The most important property of the cyclic codes is that, if  $g(x)$  is a polynomial of degree  $n-k$  and is a factor of  $x^n-1$ , then  $g(x)$  generates an  $(n,k)$  cyclic code. The parity check polynomial  $h(x)$  is obtained by  $h(x) = (x^n-1)/g(x)$ . The generator and parity check matrices are of the following form:

$$G = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{n-k-1} & g_{n-k} & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & & g_1 & g_2 & g_3 & & 0 \\ 0 & 0 & \cdots & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix}$$

$$H = \begin{bmatrix} h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & \cdots & h_1 & h_0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & & \vdots \\ 0 & 0 & & h_k & h_{k-1} & h_{k-2} & \cdots & 0 \\ 0 & 0 & \cdots & 0 & h_k & h_{k-1} & \cdots & h_0 \end{bmatrix}$$

### II.1.3. BCH Codes

An important subclass of cyclic codes are the BCH codes. In order to define these codes, let  $\beta$  be an element of  $GF(q^m)$ , and  $l_0$  any nonnegative integer, then a polynomial  $g(X)$  of minimum degree which has as roots the following consecutive powers of  $\beta$ :

$$\beta^{l_0}, \beta^{l_0+1}, \dots, \beta^{l_0+d_0-2}$$

can generate a code of distance  $\geq d_0$ .

Then  $g(x)$  is given by:

$$g(X) = \text{LCM} \{ \Psi_0(X), \Psi_1(X), \dots, \Psi_{d_0-2}(X) \}$$

where  $\Psi_i(x)$  is the minimum polynomial for any  $i$  such that  $0 \leq i \leq d_0-1$ .

Also the length of the code is

$$n = \text{LCM} \{ n_0, n_1, \dots, n_{d_0-2} \}$$

where  $n_i$  is the order of  $\beta^{l_0+i}$  for any  $i: 0 \leq i \leq d_0-1$ . The number of parity checks of such a code is:  $n-k \leq m(d_0-1)$ . The minimum distance  $d_{min}$  of this code is at least  $d_0$ . The code is thus capable of correcting up to  $t$  errors  $\{ t = [(d_0-1)/2] \}$ . If we let  $l_0=1$ ,  $d_0=2t+1$  and  $\beta$  be a primitive element then the BCH code is called primitive.

The decoding procedure of a BCH code like the one mentioned above is

described as follows. When the encoder transmits a BCH codeword

$$v(X) = v_0 + v_1 X + \dots + v_{n-1} X^{n-1}$$

and the received codeword is

$$r(X) = r_0 + r_1 X + r_2 X^2 + \dots + r_{n-1} X^{n-1}$$

If the error pattern is  $e(X)$  then

$$r(X) = v(X) + e(X)$$

The first step of decoding is to find the syndrome  $S$  from the vector  $r(X)$ .

The syndrome is given as

$$S = (S_1, S_2, \dots, S_{2t}) = r \cdot H^T$$

The next step is to find the error-locator polynomial  $\sigma(x)$  from the components of the syndrome  $S$ . In order to find the error location we define the error-locator polynomial as

$$\begin{aligned} \sigma(X) &= (1 + \beta_1 X)(1 + \beta_2 X^2) \dots (1 + \beta_l X^l) \\ &= \sigma_0 + \sigma_1 X + \sigma_2 X^2 + \dots + \sigma_l X^l \end{aligned}$$

when

$$S_1 = \beta_1 + \beta_2 + \dots + \beta_l$$

$$S_2 = \beta_1^2 + \beta_2^2 + \dots + \beta_l^2$$

$$\vdots$$

$$S_{2t} = \beta_1^{2t} + \beta_2^{2t} + \dots + \beta_l^{2t}$$

The problem is reduced to solving the Newton's identities.

$$S_1 + \sigma_1 = 0$$

$$S_2 + \sigma_1 S_1 + 2\sigma_2 = 0$$

$$S_3 + \sigma_1 S_2 + \sigma_2 S_1 + 3\sigma_3 = 0 \quad (\text{binary case only})$$

$$\vdots$$

$$S_l + \sigma_1 S_{l-1} + \dots + \sigma_{l-1} S_1 + l\sigma_l = 0$$

$$S_{l+1} + \sigma_1 S_l + \dots + \sigma_{l-1} S_2 + \sigma_l S_1 = 0$$

or 
$$S_k = -\sum_{j=1}^l \sigma_j S_{k-j} \quad \text{for } k = l+1, l+2, \dots \quad (\text{nonbinary case})$$

Ways to find the error locator polynomial are extensively discussed in the next chapter. The last step to BCH decoding is to determine the numbers  $\beta_1, \beta_2, \dots, \beta_l$  by finding the roots of  $\sigma(X)$  and correct the error in  $r(X)$ . This method is called the Chien search, and it is applicable for the binary case only. For the nonbinary case we find the Forney error magnitudes. We let

$$Z(X) = \sigma_0 + (S_1 + \sigma_1) X + (S_2 + \sigma_1 S_1 + \sigma_2) X^2 + \dots \\ + (S_l + \sigma_1 S_{l-1} + \sigma_2 S_{l-2} + \dots + \sigma_l) X^l$$

Then the error magnitude at location  $\beta_\nu$  is:

$$e_\nu = \frac{Z(\beta_\nu^{-1})}{\prod_{\substack{i=1 \\ i \neq \nu}}^l (1 + \beta_i \beta_\nu^{-1})}$$

This concludes the decoding process.

### II.2.1. Algebraic theory in linear systems

In the earlier days of science, systems work was concerned with problem formulation. In 1945-55 Guillemin gave a new dimension to the systems theory.

According to Guillemin the system we are looking for:

- (A) is a good approximation of the given input/output behavior
- (B) can be built from passive network elements
- (C) has simple internal structure
- (D) can be easily computed

Today, part (A) can be replaced by (A') which is:

- (A') is the simplest exact realization of a given input/output behavior



A system  $\Sigma$  is defined as a physical entity that accepts inputs, emits outputs and has a certain definite internal structure (states). A system  $\Sigma$  is linear if its input/output behavior as well as internal structure is described by linear functions. The systems we are going to consider are only discrete-time systems (systems whose behavior is described at  $t = \dots, -1, 0, 1, \dots \in \mathbb{Z}$  (set of integers)).

An explicit way of defining a system is by using the concept of the state  $x(t)$  of  $\Sigma$  and the rule of the state transition. This allows us to view  $\Sigma$  as a circuit diagram or a computer program. Also, it is the most explicit description of  $\Sigma$ . A system  $\Sigma$  is the triple  $(F, G, H)$  as it is defined by the equations

$$x(t+1) = F x(t) + G u(t) \quad (1)$$

$$y(t) = H x(t) \quad (2)$$

where  $t \in \mathbb{Z}$ ,  $x \in X$  (vector space over  $K$ ),  $u(t) \in K^m$  (set of  $m$ -tuples in  $K$ ),  $y(t) \in K^p$  (set of  $p$ -tuples in  $K$ ).

The next important concept is the input/output map of a system. The input/output map  $f_{\Sigma}$  of  $\Sigma$  is a rule that assigns a uniquely determined output to every finite sequence of inputs. The output occurs one unit of time after the last input. Thus, it is guaranteed that  $\Sigma$  is a causal system:

$$f_{\Sigma} : (u(-r), \dots, u(0)) \mapsto y(1) \quad (3)$$

The terms  $u(-r), \dots, u(0)$  represent the input function at time  $t = -r, \dots, 0$  respectively. We have assumed in (3) that the last input occurred at  $t=0$ . Given that  $\Sigma$  is linear, this implies that  $f_{\Sigma}$  is linear. Thus, we can write

$$f_{\Sigma}(u(-r), \dots, u(0)) = \sum_{k=0}^r A_{k+1}(f_{\Sigma})u(-k) \quad (4)$$

where  $A_{k+1}(f_{\Sigma})$  are  $p \times m$  matrices with elements in  $K$ .

$$\text{Hence } f_{\Sigma} \approx \{ A_1(f_{\Sigma}), A_2(f_{\Sigma}), \dots \} \quad (5)$$

In addition,  $f_{\Sigma}$  is equivalent to the doubly infinite block matrix

$$\mathfrak{H}(f_\Sigma) = \begin{bmatrix} B_1 & B_2 & B_3 & \cdots \\ B_2 & B_3 & B_4 & \cdots \\ B_3 & B_4 & B_5 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (6)$$

which is called the Hankel matrix of  $f_\Sigma$ .

The last concept to be defined is the one of realization. The purpose is to find a system  $\Sigma$  if we are given the input/output map  $f_\Sigma$ . Hence, we say that  $f'$  is a realization of  $\Sigma$  iff

$$A_k(f') = H F^{k-1} G \quad k = 1, 2, 3, \dots \quad (7)$$

The left-hand side of (7) is given by  $f'$  and the right-hand side by  $\Sigma$ . The realizations are not always unique. Therefore, we are interested in minimal realizations ( $\dim X = \text{minimum}$ ).

The problem of minimal realization is stated as follows: Given an infinite sequence  $\{A_1, A_2, \dots\} \approx f$  of  $p \times m$  matrices over  $K$ , find a triple of matrices  $\Sigma_f = (F, G, H)$  over  $K$  such that

$$(i) \quad A_k(f) = H F^{k-1} G \quad \text{is satisfied where } k = 1, 2, \dots$$

$$\text{and } (ii) \quad n = \dim \Sigma_f = \text{size } F = \text{minimum.}$$

This is also called the generalized Fibonacci problem which tries to describe an infinite scalar by means of a simple recursion equation. A subclass of this type of problem is the minimal partial realization of order  $M$  of a complete sequence  $\{A_1, A_2, \dots\}$ .

To avoid any possible confusion we note that there is another problem that is similar to the minimal realization problem; it is called minimal realization of a partial sequence  $\{A_1, A_2, \dots, A_M\}$ .

Next, let us state the minimal partial realization problem. Given a finite sequence  $\{A_1, A_2, \dots, A_M\}$  of  $p \times m$  matrices over  $K$ , find a triple  $\Sigma_M = (F, G, H)$  such that

$$(i) A_k = H F^{k-1} G, \quad k = 1, \dots, M$$

and (ii) size  $F = \text{minimum}$ .

This is called the restricted generalized Fibonacci problem which is known more commonly as the generalized minimal Padé representation.

If we extend the second problem to  $K > M$  then under certain conditions we may be able to solve the realization problem (first problem) given only partial knowledge of the input/output map (second problem).

Most of the important algorithms needed to solve these problems will be discussed in the next chapter.

## Chapter III - Algorithms

In this chapter we describe algorithms to solve the realization problem and to decode BCH codes. Some of these algorithms were derived in algebraic theory for linear systems and some in algebraic coding theory. Since both problems have algebraic foundations we can explore the possibility to find a relationship between them.

The first algorithms derived in algebraic coding theory were proposed first by Peterson in 1960, who found a way to decode the newly discovered BCH codes. Peterson's algorithm was generalized and improved by Gorenstein and Zierler, Chien, Forney, Berlekamp [1], and Massey [2]. From these algorithms, the most efficient are the Berlekamp's iterative algorithm and Chien's search. The Fundamental iterative algorithm [8] presented by Feng and Tzeng in 1983, is able to decode any cyclic code up to Hartmann-Tzeng bound using multiple syndrome sequences. We will describe the Berlekamp's algorithm as well as the Fundamental iterative algorithm later.

The realization problem can be solved by partial-fraction expansions as discussed by Gilbert in 1963. A more general and practical solution was discovered in 1965 by Ho and Kalman [3]. Rissanen in 1971 [4] refined the Ho-Kalman algorithm. It was followed by Dickinson, Morf and Kailath in 1974 [5] who presented an algorithm that could be implemented by hardware.

In 1980, Conan [6] presented a simplified version that could solve the realization problem for a sequence of vectors instead of matrices. This algorithm is the most efficient up to this date. In 1984, Conan [7] presented an algorithm for scalar rational sequences. This algorithm is as simple, effective and easily

implemented as the Berlekamp-Massey iterative algorithm.

Ho-Kalman, Rissanen's and Conan's scalar algorithm will be discussed later since they can be easily explained using scalar sequences. This will help us later to compare the MPR algorithms to BCH decoding algorithms.

### III.1. Ho and Kalman algorithm.

To solve the minimal partial realization problem using the Ho-Kalman algorithm, we first have to present the following lemmas:

*Lemma 1.* Suppose  $f$  has a finite-dimensional realization. Then the induced sequence  $(A_1, A_2, \dots)$  of  $f$  satisfies the relation

$$A_{r+j+1} = -\sum_{i=1}^r \beta_i A_{i+j} \quad j = 0, 1, 2, \dots$$

for some  $\beta_1, \dots, \beta_r \in K$ , where  $r$  may be taken to be  $\geq \deg \Psi_f$

$$\psi_f(z) = z^n + \alpha_1 z^{n-1} + \dots + \alpha_n \quad \text{where } \beta_i = \alpha_{n-i+1}$$

$\psi_f(z)$  is known as a minimal polynomial of  $K$ .

*Lemma 2.* If  $f$  has a finite-dimensional realization then it is realized by

$$H = E_{pr}^p$$

$$G = H_{rr}(f)E_m^{mr}$$

$$F = C = \begin{bmatrix} 0_p & I_p & 0_p & \dots & 0_p \\ 0_p & 0_p & I_p & \dots & 0_p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_p & 0_p & 0_p & \dots & I_p \\ -\beta_1 I_p & -\beta_2 I_p & -\beta_3 I_p & \dots & -\beta_r I_p \end{bmatrix}$$

where  $r$  and  $\beta_1, \dots, \beta_r$  are as given by (Lemma 1) and  $I_p$  and  $0_p$  are the  $p \times p$  identity and zero matrix.

**Lemma 3.** Suppose  $f$  has a finite-dimensional realization  $\Sigma$ . Then

$$\text{rank } H_{N',N}(f) \leq \dim f \leq \dim \Sigma$$

for all positive integers  $N$  and  $N'$ .

The Ho-Kalman algorithm is described next. The following steps lead to a canonical realization of an arbitrary finite-dimensional input/output map  $f$ :

1. Choose an  $r$  such that (lemma 1) holds.
2. Use the invariant factor algorithm  $pr \times pr$  matrix  $P$  and a nonsingular  $mr \times mr$  matrix  $M$  over  $K$  such that:

$$P [H_{rr}(f)] M = \begin{bmatrix} I_n^n & 0_{mr-n}^n \\ 0_n^{pr-n} & 0_{mr-n}^{pr-n} \end{bmatrix} \quad (1)$$

3. Now write down a canonical realization of  $f$  as follows:

$$F = E_{pr}^n P [(\sigma H)_{rr}(f)] M E_n^{mr} \quad (2)$$

$$G = E_{pr}^n P [H_{rr}(f)] E_n^{mr} \quad (3)$$

$$H = E_{pr}^p [H_{rr}(f)] M E_n^{mr} \quad (4)$$

The question that arises is, if you pick an  $r$  (or the pair  $\{N, N'\}$ ) arbitrarily and compute a dynamical system  $\Sigma$ , what are the properties of such a  $\Sigma$ ? Does  $\Sigma$  realize a part of the sequence  $\{A_1, A_2, \dots\}$ ?

As an answer, let us state the *realizability criterion*. Let  $\{A_1, A_2, \dots\}$  be an arbitrary infinite sequence of  $p \times m$  matrices over  $K$  and let  $H$  be the corresponding Hankel matrix. Then  $\Sigma$  given by (2), (3) and (4) realizes the sequence up to and including the term  $A_{N_0}$ , that is (2) holds for  $i = 1, \dots, N_0$ ,

( i ) if and ( ii ) only if there exist positive integers  $N$  and  $N'$  such that

(a)  $N + N' = N_0$  and

(b)  $\text{rank } H_{N',N} = \text{rank } H_{N'+1,N} = \text{rank } H_{N',N+1}$

For the partial realization problem for the scalar case (which is  $m=p=1$ ) we have the following theorem.

*Theorem:* Let  $\{A_1, A_2, \dots\}$  be an arbitrary infinite sequence with fixed  $N_0$  then one of the following three cases will arise:

- (a) The minimal partial realizations are unique
- (b) The problem is overspecified: the minimal partial realization of order  $N_0$  is unique and is at the same time the unique minimal partial realization of order  $M_0 < N_0$ .
- (c) The problem is underspecified: there is an integer  $P_0 > N_0$  such that every minimal partial realization of order  $N_0$  is at the same time the unique minimal partial realization of order  $P_0$  for some arbitrary extension  $\beta_{N_0+1}, \dots, \beta_{P_0}$  of the given sequence. In short, in this case there is a  $(P_0 - N_0)$ -parameter family of minimal realizations.

Case (a) arises if and only if  $N_0=2n$  and

$$\text{rank } H_{nn} = \text{rank } H_{n+1,n'} = n$$

Case (b) arises if and only if

$$\text{rank } H_{nn'} = n-q,$$

where  $q>0$  and  $n'=n$  [ $n'=n+1$ ] when  $N_0=2n$  [ $N_0=2n+1$ ].

Then  $M_0 = 2(n-q)$ .

Case (c) arises if and only if

$$\text{rank } H_{nn'} = n'-q$$

$$\text{rank } H_{n+1,n'} = n'-q+1,$$

where  $q>0$  and  $n'=n$  [ $n'=n+1$ ] when  $N_0=2n$  [ $N_0=2n+1$ ]. Then  $P_0=2(n+q)$ .

In all cases formulas (2), (3) and (4), with  $r=n$ ,  $M_0$ ,  $P_0$ , respectively, provide minimal partial realizations of order  $N_0$ , with the understanding that in case (c)  $P_0 - N_0$  arbitrary parameters  $B_{N_0+1}, \dots, B_{P_0}$  must be added to the sequence  $A_1, \dots, A_{N_0}$ . The dimension of the minimal partial realization is  $n$ ,  $n-q$  and  $n+q$  respectively. (end of theorem).

In order to express  $\Sigma$  in terms of the more conventional transfer function, let us define the Padé approximation problem, which is stated as follows.

Find two polynomials  $\sigma, \chi \in K[z]$ ,  $\deg \sigma < \deg \chi$ , such that the coefficients of the formal power series

$$\frac{\sigma(z)}{\chi(z)} = B_1 z^{-1} + B_2 z^{-2} + \dots$$

agree with a given sequence  $\{A_1, A_2, \dots\}$  up to and including the  $N_0$ th term and such that  $\deg \chi = \text{minimum}$ .

To solve this problem, write  $N_0 = 2n$  or  $2n+1$ . The solution is given by

$$\frac{\sigma(z)}{\chi(z)} = H(zI - F)^{-1}G$$

where  $F, G$  and  $H$  are computed from (2), (3) and (4) with  $\deg \chi = n+q$ , where  $q$  is the "deficiency index"

$$q = \begin{cases} N_0 - n - \text{rank } H_{n, N_0-n} & \text{if } \text{rank } H_{n+1, N_0-n} > \text{rank } H_{n, N_0-n} \\ \text{rank } H_{n, N_0-n} & \text{if } \text{rank } H_{n+1, N_0-n} = \text{rank } H_{n, N_0-n} \end{cases}$$

To illustrate the use of the Ho-Kalman algorithm we present the following numerical example.

*Example 1.* Consider the input/output function generating the scalar partial sequence

$$\{ 1, 1, 1, 2, 1, 3, ?, ??, \dots \}$$

We notice that for  $N=N'=1$  or  $H_{11}$  the realizability criterion is met. Thus



$H_{11}$  has full rank and it can realize exactly the first two terms of the given sequence. The system  $\Sigma$  is  $(1,1,1)$  and its infinite sequence is  $\{1,1,1, \dots\}$ . Since this realization does not realize our sequence we are looking for another one. The remainder sequence is  $\{0,0,0,1,0,2,?,??, \dots\}$ . The smallest full rank matrix is  $H_{44}$  which is a  $4 \times 4$  Hankel matrix:

$$H_{44} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & ? \end{bmatrix}$$

and as we notice its rank is independent of  $?$ . Thus the original sequence has a 4-dimensional realization. However, the 4-dimensional realization requires an  $A_7=?$  and an  $A_8=??$ , which means  $\infty^2$  nonisomorphic minimal realizations. The maximum number of dimensions in order for this sequence to be realized is 5. The minimal realization though requires only 3 dimensions, as we conclude from the realization criterion. Thus, the initial sequence can be written as

$$H_{33} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{which has full rank.}$$

A suitable pair  $(P,M)$  satisfying (1) is given by

$$P = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Then (2), (3) and (4) give

$$F = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad G = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

For this realization  $\tau=2$ ,  $\tau\tau=3$ ,  $\tau\tau\tau=5$ ,  $\tau\tau\tau\tau=2$ ,  $\tau^5=9$ . To get the transfer function we use the results above into the following equation

$$\frac{\sigma(z)}{\chi(z)} = H(zI-F)^{-1}G ;$$

after the calculation

$$\frac{\sigma(z)}{\chi(z)} = \frac{z^2+2z+1}{z^3+z^2-z-2}$$

### III. 2. Rissanen's Algorithm

This algorithm is an improved version of the Ho-Kalman algorithm described above. For a given partial sequence  $\{A_1, \dots, A_N\}$  for each  $N=1,2,3, \dots$  we can find a sequence of minimal partial realizations  $\Sigma_N = (G_N, F_N, H_N)$  such that

$$\dots \Sigma_{N'} \subseteq \Sigma_N \subseteq \dots \quad \text{if } N' < N.$$

The inclusion symbol means that the matrices in  $\Sigma_{N'}$  appear as submatrices of the corresponding ones in  $\Sigma_N$ . Thus, we are looking for an algorithmic solution to the partial realization problem for some  $N$ , but we also want a recursion of  $N$ . Moreover, we look for a solution where each extension of the  $A_i$ 's can be met by calculating just a few new elements. The advantage of this algorithm compared with the B.L. Ho's is that we avoid to calculate each partial realization anew.

Before we describe Rissanen's algorithm, let us arrange the entries of the sequence  $\{A_1, A_2, \dots\}$  in the Hankel matrix.

$$A = \begin{bmatrix} f_1 & f_2 & \dots & f_n & \dots \\ f_2 & f_3 & \dots & f_{n+1} & \dots \\ \vdots & \vdots & & \vdots & \\ f_m & f_{m+1} & \dots & f_{m+n-1} & \dots \\ \vdots & \vdots & & \vdots & \end{bmatrix}$$

Let  $A(m,n)$  denote the desired submatrix of  $A$ . The algorithm is based on factorization of  $A(m,n)$  of the following type:

$$A(n,m) = P(n,n) Q(n,m) \quad m \geq n, \text{ rank } A(n,m) \geq n-1, \quad (1)$$

where  $P(n,n)$  is the lower triangular with 1's on the diagonal

$$\rightarrow \begin{bmatrix} f_1 & \dots & f_i & f_{i+1} & \dots & f_m \\ f_2 & \dots & f_{i+1} & f_{i+2} & \dots & f_{m+1} \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & f_{i+n-1} & & \vdots \\ f_n & f_{i+n-1} & f_{i+n} & f_{n+m-1} & & \end{bmatrix} \quad (2)$$

$$= \begin{bmatrix} 1 & & & & & \\ p_{21} & 1 & & & & \\ p_{31} & p_{32} & 1 & & & \\ \vdots & \vdots & & \ddots & & \\ p_{n1} & p_{n2} & \dots & 1 & & \end{bmatrix} \begin{bmatrix} q_{11} & \dots & q_{1i} & q_{1,i+1} & \dots & q_{1m} \\ q_{21} & \dots & q_{2i} & q_{2,i+1} & \dots & q_{2m} \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ q_{n1} & \dots & q_{ni} & q_{n,i+1} & \dots & q_{n,m} \end{bmatrix}$$



- Step 1.* Let  $k$  be the least integer for which  $f_k \neq 0$ . Take  $N=2k+1$  and form  $A(k+1,k+1)$ . It has rank  $\geq k$ .
- Step 2.* Apply the factoring algorithm and find  $P(k+1,k+1)$  and  $Q(k+1,k+1)$  [equation (2)]. If the last row of  $Q$  is nonzero, the rank of  $A(k+1,k+1)$  is  $k+1$ . Increase  $N$  by 2, form  $A(k+2,k+2)$  and continue the factorization. If the last row of  $Q(k+2,k+2)$  is still nonzero, increase  $N$  by 2 and repeat until, say, for  $N=2n-1$  the last row of  $Q(n,n)$  is zero.
- Step 3.* From formulas (3) and (4) calculate the partial realization  $\Sigma_{n-1}$ .
- Step 4.* Increase  $N$  by 1. Continue the factorization for  $A(n,n+1)$ . If the last row of  $Q(n,n+1)$  remains zero for all  $m$ , we have found the realization. (Since the algorithm never stops we place an upper limit for  $m$ .)
- Step 5.* If for some  $m(>n)$  the last element in the last row of  $Q(n,m)$  becomes nonzero, the last picked element  $f_{m+n-1}$  is not realized by partial realization  $\Sigma_{n-1}$ . In this case, pick one point,  $f_{m+n}$ , and form  $A(n+1,m)$ . Continue the factorization, pick one new point and repeat until either  $Q(n',m)$  for some least  $n' \leq m$  has last row zero or  $n' = m$  and the last row is nonzero. In the previous case go to Step 3. In the latter case go to Step 2.

To illustrate the algorithm let us use the same sequence as in the B.L. Ho algorithm:  $\{1,1,1,2,1,3,2,3, \dots\}$

*Example 2.* First, let us demonstrate how the factorization algorithm works. Consider the above sequence in matrix form

$$A(4,5) = \begin{bmatrix} 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 2 & 1 & 3 \\ 1 & 2 & 1 & 3 & 2 \\ 2 & 1 & 3 & 2 & 3 \end{bmatrix}$$

Having  $s(1) = 1$ , we set  $q_{i1} = 0, i > 1$ . Then

$$\begin{aligned} p_{21} \cdot 1 [q_{11}] + 1 [p_{22}] \cdot 0 [q_{21}] &= 1 \rightarrow p_{21} = 1 \\ 1 \cdot 1 [q_{12}] + 1 \cdot 1 [q_{22}] &= 2 \rightarrow q_{22} = 0 \\ 1 \cdot 1 [q_{13}] + 1 \cdot 1 [q_{23}] &= 1 \rightarrow q_{23} = 1 \\ 1 \cdot 2 [q_{14}] + 1 \cdot 1 [q_{24}] &= 3 \rightarrow q_{24} = -1 \\ 1 \cdot 1 [q_{15}] + 1 \cdot 1 [q_{25}] &= 3 \rightarrow q_{25} = 2 \end{aligned}$$

Then  $s(2) = 3, q_{i3} = 0, i > 2$  and

$$\begin{aligned} p_{31} \cdot 1 [q_{11}] &= 1 \rightarrow p_{31} = 1 \\ 1 \cdot 1 [q_{13}] + p_{32} \cdot 1 [q_{23}] &= f_5 = 1 \rightarrow p_{32} = 0 \\ 1 \cdot 2 [q_{14}] + 0 \cdot (-1) [q_{24}] + 1 \cdot q_{34} &= f_6 = 3 \rightarrow q_{34} = 1 \\ 1 \cdot 1 [q_{15}] + 0 \cdot 2 [q_{25}] + 1 \cdot q_{35} &= 2 \rightarrow q_{35} = 1 \end{aligned}$$

Continuing we get the result

$$A(4,5) = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 0 & 1 & & \\ 2 & 1 & -1 & 1 & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & -1 & 2 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In the main algorithm the process develops as follows.

Step 1.  $k=1, f_1=1 \neq 0, N=2 \cdot 1 + 1 = 3$

$$A(2,2) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Step 2.

$$A(2,2) = \begin{bmatrix} 1 & & \\ 1 & 1 & \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$$

The last row of  $Q(2,2)$  is zero, thus we have a minimal realization.

Step 3. From (3) and (4) we find  $\Sigma = (1,1,1)$ .

Step 4. Add  $f_4=2$  and factorize  $A(2,3)$ . Now  $q_{23}=1 \neq 0$ . So the last row of  $Q$  is not all zeros.

Step 5. We pick two points  $f_6$  and later  $f_7$  until the factorization of  $A(4,4)$  gives a  $Q(4,4)$  with last line all zeros.

Step 3. The matrix  $P(4,4)$  has been found. Then,

$$P(3,3) = \begin{bmatrix} 1 & & \\ 1 & 1 & \\ 1 & 0 & 1 \end{bmatrix} \text{ and } P^{-1}(3,3) = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{Also } P_*(3,3) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 2 & 1 & -1 \end{bmatrix}$$

Thus,

$$G_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad F_3 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \end{bmatrix} \quad H_3 = [ 1 \ 0 \ 0 ]$$

### III. 3. Conan's Minimal Partial Realization Algorithm for scalar sequences

Conan's MPR algorithm is the most effective algorithm of this type. In the scalar case, it can be as effective as the Berlekamp-Massey algorithm.

Before we present the algorithm let us first introduce some background information. Let  $S \equiv (s_0, s_1, \dots, s_m, \dots)$  be a semi-infinite sequence over the base field  $F$ , and  $S_m \equiv (s_0, s_1, \dots, s_{m-1})$  the finite subsequence of  $S$  formed by the first  $m$  samples.

An  $m$ th partial realization of  $S$  is a pair of polynomials  $\{d(z), n(z)\}$  with  $d(z)$  monic,  $\deg\{n(z)\} \leq \deg\{d(z)\}$  and  $d(z), n(z)$  relatively prime provided that the system with transfer function  $g(z) = \frac{n(z)}{d(z)}$  has an impulse response whose first  $m$  output symbols match  $S_m$ .

If  $S(z)$  and  $S_m(z)$  are the  $Z$ -transforms of  $S$  and  $S_m$  we can write

$$d(z)S(z) = n(z) + qz^{-(\sigma-d)} + O_{(\sigma-d)}$$

where  $d$  is the degree of the polynomial  $d(z)$  and  $\sigma$  is the order of  $d(z)$  ( $\sigma > d$ );  $q$  is the residual of  $d(z)$  with respect to  $S$  ( $q \neq 0$ ).

The monic polynomial  $d(z)$  yields a minimal partial realization of  $S$ , provided it has order  $\sigma \geq m$  with respect to  $S$  and its degree is minimum. Next,  $d(z)$  is an  $(m+1)$ th partial realization provided the coefficient  $[d(z)S_{m+1}(z)]_{m-d}$  of  $z^{-(m-d)}$  in the product of  $d(z)$  by  $S_{m+1}(z)$  is zero. Also, if  $\{d'_l(z)\}$  is a sequence of  $l$ th minimal partial realization of  $S(z)$  then the following conditions are satisfied.

$$\deg\{d_{m+1}(z)\} = \max[\deg\{d_m(z)\}, m - \deg\{d_m(z)\}]$$

$$\text{if } [d_m(z)S_{m+1}(z)]_{m-\deg\{d_m(z)\}} \neq 0$$

$$\deg\{d_{m+1}(z)\} = \deg\{d_m(z)\} \quad \text{otherwise.}$$



The MPR synthesis algorithm of order  $N$  has as follows:

Step 0 : (Initialization)

$$d(z) \leftarrow 1, n(z) \leftarrow 0, b(z) \leftarrow 0, c(z) \leftarrow -1, d_p \leftarrow 0.$$

Step 1 : if  $m=N$ , stop.

Else compute  $d \equiv [d(z)S_{m+1}(z)]_{m-\deg\{d(z)\}}$ , the coefficient of  $z^{-(m-\deg\{d(z)\})}$  in the product of  $d(z)$  and  $S_{m+1}(z)$ .

Step 2 : (no change)

If  $d = 0$ , go to Step 5.

Else,  $W \leftarrow d_p - \deg\{d(z)\}$ .

If  $W > 0$ , go to Step 4.

Else continue.

Step 3 : (minor change)

$$d(z) \leftarrow d(z) - dzb(z).$$

$$n(z) \leftarrow n(z) - dzc(z).$$

Go to Step 5.

Step 4 : (major change)

$t_1(z) \leftarrow d(z), t_2(z) \leftarrow n(z)$  ( $t_1(z), t_2(z)$  are auxiliary memory cells).

$$d_p \leftarrow \deg\{d(z)\}.$$

$$d(z) \leftarrow z^W d(z) - db(z).$$

$$n(z) \leftarrow z^W n(z) - dc(z).$$

$$b(z) \leftarrow d^{-1}t_1(z).$$

$$c(z) \leftarrow d^{-1}t_2(z).$$

Step 5 : (initialization for the next step)

$$d_p \leftarrow d_p + 1, m = m + 1, \text{ go to Step 1.}$$

For better illustration let us present an example using the MPR algorithm.

*Example 3.* For comparison purposes we use the same sequence as in Ho-Kalman and Rissanen's algorithm. The results of the algorithm are shown below.

Table I. Conan's MPR algorithm solutions

m	s <sub>m</sub>	d(z)	n(z)	d	b(z)	c(z)	d <sub>p</sub>
0	1	1	0	1	0	-1	0
1	1	1	1	1	0	-1	1
2	1	z	z+1	1	1	1	1
3	2	z-1	z	1	1	1	2
4	1	z <sup>2</sup> -z-1	z <sup>2</sup> -1	-2	z-1	z	2
5	3	z <sup>2</sup> +z-3	z <sup>2</sup> +2z-1	-2	z-1	z	3
6	2	z <sup>3</sup> +z <sup>2</sup> -z-2	z <sup>3</sup> +2z <sup>2</sup> +z	0	$\frac{-z^2}{2} - \frac{z}{2} + \frac{3}{2}$	$\frac{-z^2}{2} - z + \frac{1}{2}$	3
7	3	"	"	0	"	"	4
8	-	"	"	-	"	"	5

The solution is: 
$$S_m(z) = \frac{z^3 + 2z^2 + z}{z^3 + z^2 - z - 2}$$

The advantage of this algorithm is that it requires less memory and fewer computations in order to be implemented. The implementation is very simple since it can be achieved using a shift register (similar to the Berlekamp-Massey algorithm).

### III. 4. Berlekamp-Massey algorithm

As we mentioned in the previous chapter the Berlekamp-Massey algorithm is the most effective algorithm up to date for decoding BCH codes. An introduction to the operation of the algorithm follows.

For a given sequence  $S = (s_1, s_2, s_3, \dots)$  there is a polynomial  $\sigma(z) = 1 + \sum_{i=1}^l \sigma_i z^i$  such that it relates the syndrome  $S$  with the error locator polynomial  $\sigma(z)$  in the following way

$$s_k + \sigma_1 s_{k-1} + \sigma_2 s_{k-2} + \dots + \sigma_l s_{k-l} = 0$$

which are the Newton's equations.

In a similar way with the MPR algorithm the sequence  $S$  can be expressed as a ratio of two polynomials,  $\sigma(z)$  and  $\omega(z)$  such that

$$[1 + S(z)] \sigma(z) = \omega(z) \quad (1)$$

Since our sequence  $S$  is limited up to  $2t$  terms ( $S = \{s_1, s_2, s_3, \dots, s_{2t}\}$ ) then the key equation (equation (1)) becomes

$$[1 + S(z)] \sigma(z) = \omega(z) \pmod{z^{2t+1}} \quad (2)$$

Berlekamp's algorithm solves equation (2) for polynomials  $\sigma(z)$  and  $\omega(z)$  by breaking the problem into smaller pieces. Then, we have a sequence of equations

$$(1 + S) \sigma^{(k)} \equiv \omega^{(k)} \pmod{z^{k+2}}$$

where for each  $k = 0, 1, 2, \dots, 2t$  we have

$$\sigma^{(k)} = \sum_i \sigma_i^{(k)} z^i \quad \text{and} \quad \omega^{(k)} = \sum_i \omega_i^{(k)} z^i.$$

The solution of the problem is accomplished by the Berlekamp's algorithm which is stated below.

Initially define  $\sigma^{(0)} = 1$ ,  $\tau^{(0)} = 1$ ,  $\omega^{(0)} = 1$ ,  $\gamma^{(0)} = 0$ ,  $D(0) = 0$ ,  $B(0) = 0$ .

Proceed recursively as follows. If  $S_{k+1}$  is unknown, stop; otherwise define  $\Delta_1^{(k)}$

as the coefficient of  $z^{k+1}$  in the product  $(1 + S) \sigma^{(k)}$  and let

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

$$\omega^{(k+1)} = \omega^{(k)} - \Delta_1^{(k)} z \tau^{(k)}$$

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

and  $B(k) = 0$ , set

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

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

$$\tau^{(k+1)} = z \tau^{(k)}$$

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

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

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

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

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

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

*Example 4.* To illustrate the operation of this algorithm let us use the same sequence as in previous algorithms. The sequence is  $\{1, 1, 1, 2, 1, 3, 2, 3, \dots\}$ .

Then

$$\sigma^{(k)} = 1 + \sigma_1 z + \sigma_2 z^2 + \dots,$$

$$\omega^{(k)} = \omega_0 + \omega_1 z^1 + \omega_2 z^2 + \dots,$$

$$(1 + S) = 1 + s_1 z + s_2 z^2 + \dots;$$

the results of the algorithm appear in the following table.

Table II. Berlekamp algorithm solutions

S	k	$\Delta_1^{(k)}$	$\sigma^{(k)}$	$\tau^{(k)}$	$\omega^{(k)}$	$\gamma^{(k)}$	D(k)	B(k)
-	0	1	1	1	1	0	0	0
1	1	0	1-z	1	1	1	1	1
1	2	0	1-z	z	1	z	1	1
1	3	1	1-z	z <sup>2</sup>	1	z <sup>2</sup>	1	1
2	4	-2	1-z-z <sup>3</sup>	1-z	1-z <sup>3</sup>	1	3	0
1	5	-1	1+z-z <sup>2</sup> -2z <sup>3</sup>	z-z <sup>2</sup>	1+2z-z <sup>3</sup>	z	3	0
3	6	0	1+z-z <sup>2</sup> -2z <sup>3</sup>	z <sup>2</sup> -z <sup>3</sup>	1+2z+z <sup>2</sup> -z <sup>3</sup>	z <sup>2</sup>	3	0
2	7	0	"	z <sup>3</sup> -z <sup>4</sup>	"	z <sup>3</sup>	3	0
3	8	-	"	-	"	-	-	-

The solution is:  $1+z-z^2-2z^3$ . (The polynomial  $\omega(z)$  is discarded in binary case.)

### III. 5. Fundamental iterative (Feng - Tzeng) algorithm

Unlike the previous algorithms the Fundamental Iterative Algorithm can be used to decode cyclic codes up to the Hartmann-Tzeng bound. It can also find a solution for a multiple set of sequences.

The algebraic problem for which the Fundamental iterative algorithm has been designed is as follows. Let  $\{a_1, a_2, \dots, a_l, a_{l+1}, \dots\}$  be a sequence over an arbitrary field  $F$ . When  $l \geq 0$ , the  $(l+1)$ th symbol  $a_{l+1}$  is said to be linearly dependent on the first  $l$  symbols  $a_1, a_2, \dots, a_l$  if there exist  $c_1, c_2, \dots, c_l$  in  $F$ , which are not all zero, such that

$$a_{l+1} = \sum_{j=1}^l c_j a_{l+1-j} \quad (1)$$

Next, let us define the polynomial  $f^{(l+1)}(x)$  with degree at most  $l$  and length  $l+1$  such that

$$f^{(l+1)}(x) = 1 + c_1x + c_2x^2 + \dots + c_lx^l \quad (2)$$

$$\text{and } f^{(l+1)}(a_{k+1}) = a_{k+1} + c_1a_k + \dots + c_la_{k+1-l} \text{ for } k \geq l$$

Then equation (1) can be written as  $f^{(l+1)}(a_{l+1}) = 0$  indicating the linear dependence of  $a_{l+1}$  to the previous symbols.

Let us now introduce multiple sequences  $a_{i1}, a_{i2}, a_{i3}, \dots$  for  $i = 1, 2, 3, \dots, M$  over a field  $F$ . Just as in the single sequence case we need to determine the smallest  $l$  and  $c_1, c_2, \dots, c_l$  such that

$$a_{i,l+1} = \sum_{j=1}^l c_j a_{i,l+1-j} \text{ for } i = 1, 2, 3, \dots, M. \quad (3)$$

An alternative way to state the problem is the following. We need to find the smallest  $l$  such that, for all sequences, the  $(l+1)$ th symbol in each sequence is dependent on the first  $l$  symbols. In addition, we need to find the  $c_j$ 's for  $j = 1, 2, \dots, l$ .

Also, the problem can be stated in polynomial form. In this case we are trying to find the polynomial  $f^{(l+1)}(x)$  of shortest length  $l+1$  such that

$$f^{(l+1)}(a_{i,l+1}) = 0 \text{ for } i = 1, 2, 3, \dots, M. \quad (4)$$

In matrix form the problem is stated as follows.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1,l+1} & \dots \\ a_{21} & a_{22} & \dots & a_{2,l+1} & \dots \\ \vdots & \vdots & & \vdots & \\ \vdots & \vdots & & \vdots & \\ a_{M1} & a_{M2} & & a_{M,l+1} & \dots \end{bmatrix} \begin{bmatrix} c_l \\ c_{l-1} \\ \vdots \\ c_1 \\ 1 \\ 0 \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ 0 \\ \vdots \end{bmatrix} \quad (5)$$

Let  $A = [a_{ij}]$  denote the matrix with  $M$  semi-infinite sequences as rows. Then

by (5) the problem in matrix form is to find the smallest  $l$  such that the first  $l+1$  columns of  $A$  are linearly dependent and, also, find the  $c_j$ 's pertaining to this linear dependence.

The fundamental algorithm is presented as follows.

Step (1).  $0 \rightarrow l, l \rightarrow f^{(l+1)}(x), l \rightarrow i$

Step (2). Compute  $d_{i,l+1} = f^{(l+1)}(a_{i,l+1})$

Step (3). If  $d_{i,l+1} = 0$ , then

(a) if  $i + (l+1) = M$ , stop.

(b) otherwise  $i + 1 \rightarrow i$  and back to (2)

Step (4). If  $d_{i,l+1} \neq 0$ , then

(a) if there exists  $d_{i,l'+1} \in D$  for  $1 \leq l' \leq l$  then

$$f^{(l+1)}(x) - \frac{d_{i,l+1}}{d_{i,l'+1}} x^{l-l'} f^{(l'+1)} \rightarrow f^{(l+1)}(x)$$

and return to (3a)

(b) otherwise,  $d_{i,l+1}$  is stored in  $D$

$f^{(l+1)}(x)$  is stored in  $C$

$l + 1 \rightarrow l, 1 \rightarrow i$  and return to (2)

*Example 5.* To illustrate the operation of the Fundamental algorithm let us present the following example. In order to be able to compare the Fundamental algorithm with the previous algorithms, let us use the same sequence with all previous examples;  $\{1, 1, 1, 2, 1, 3, 2, 3, \dots\}$

In matrix form the sequence becomes the following matrix.

$$A = \begin{bmatrix} 1 & 1 & 1 & 2 & 1 & 3 & 2 & 3 & x & x \\ 1 & 1 & 2 & 1 & 3 & 2 & 3 & x & x & x \\ 1 & 2 & 1 & 3 & 2 & 3 & x & x & x & x \\ 2 & 1 & 3 & 2 & 3 & x & x & x & x & x \\ 1 & 3 & 2 & 3 & x & x & x & x & x & x \\ 3 & 2 & 3 & x & x & x & x & x & x & x \\ 2 & 3 & x & x & x & x & x & x & x & x \\ 3 & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \end{bmatrix}$$

The results of the Fundamental iterative algorithm are listed in the table below.

Table.III. FIA solutions

$i$	$l$	$f^{(l+1)}$	$d_{i,l+1}$
1	0	$1 \in C^1$	$1 \in D$
1	1	$1-x$	1
2	1		0
3	1	$1-x \in C^2$	$1 \in D$
1	2		0
2	2	$1-x \in C^3$	$1 \in D$
1	3	$1-x-x^3$	1
2	3	$1+x-2x^2-x^3$	-2
3	3	$1+x-x^2-2x^3$	-1
4	3	-	0

The algorithm stops giving the final result as the polynomial  $f^{(l+1)}(x)$ .

Thus, the final solution is:  $f^{(l+1)}(x) = 1+x-x^2-2x^3$ .



## Chapter IV - From FIA to Conan's MPR algorithm

In this chapter we describe the relationship between Conan's MPR algorithm and the Fundamental Iterative Algorithm (FIA). Since the FIA is a more general algorithm, the MPR can be derived from the FIA by eliminating the unnecessary steps. The process is parallel to the derivation of the Berlekamp-Massey iterative algorithm from the FIA.

We start from the generalized Newton's identities

$$s_k + d_{L-1}s_{k-1} + \dots + d_1s_{k-L-1} + d_0s_{k-L} = 0$$

$$\text{or } s_k = - \sum_{j=1}^L d_{L-j}s_{k-j} \quad \text{for } k = L+1, L+2, \dots \quad (1)$$

which relate the syndrome terms  $s_j$ ,  $j = 1, 2, \dots$  to the coefficients  $d_i$ ,  $i = 0, 1, 2, \dots, L-1$  of the error-locator polynomial  $d(z) = z^L + d_{L-1}z^{L-1} + d_{L-2}z^{L-2} + \dots + d_0$  where  $L$  stands for the number of errors.

The primary objective in the BCH decoding procedure is to determine the smallest  $L$  and the  $d_{L-i}$  coefficients ( $i = 1, 2, \dots, L$ ), such that (1) is satisfied. This is also the objective of the minimal partial realization. The order of the minimal partial realization corresponds to the number of known syndrome terms which is also related to the BCH bound. If the known syndrome numbers are from  $s_1$  to  $s_{\delta_0-1}$  where  $\delta_0$  is the BCH bound, then equation (1) in matrix form for  $L+1 \leq k \leq \delta_0-1$  becomes

$$\begin{bmatrix}
 s_1 & s_2 & \cdots & s_{L+1} & \cdots \\
 \vdots & \vdots & & \vdots & \\
 s_{\delta_0-L-1} & & \cdots & s_{\delta_0-1} & \cdots \\
 \vdots & & & & \\
 s_{\delta_0-1} & & & & 
 \end{bmatrix}
 \begin{bmatrix}
 d_0 \\
 \vdots \\
 d_{L-1} \\
 1 \\
 0
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 \vdots \\
 0 \\
 \vdots
 \end{bmatrix}
 \quad (2)$$

Let  $S$  denote the matrix of the syndrome terms. Then, we have to find the smallest  $L$  such that  $L+1$  columns of  $S$  are linearly dependent and also to find the  $d_{L-j}$  coefficients,  $j = 1, 2, 3, \dots, L$ .

Matrix  $S$  can be seen as a special form of matrix  $A$  with  $a_{ij} = S_{i+j-1}$  and  $M = \delta_0 - 1$ . Comparing {III.5.(3)} and {III.5.(5)} with (1) and (2) we have the following correspondence.

$$\begin{aligned}
 f^{(l+1)}(x) &\leftrightarrow d(z) \\
 a_{ij} = S_{i+j-1} &\text{ or } S_k = a_{i,k-i+1} \\
 c_j &\leftrightarrow d_{L-j} \\
 f^{(l+1)}(a_{i,L+1}) = 0 &\leftrightarrow d(S_{L+i}) = 0 \\
 i = 1, 2, \dots, M &\quad i = 1, 2, \dots
 \end{aligned}
 \quad (3)$$

Thus, the FIA will provide an iterative procedure for BCH decoding or for finding the minimal partial realization. Since  $S_j$  is not known for  $j > \delta_0 - 1$ , the algorithm should stop after  $s_{\delta_0-1}$  is considered. Therefore, the first adjustment of the FIA is that the test " $i < M$ ?" should be replaced by " $i+L < \delta_0 - 1$ ?".

As we noticed from the example in section (III. 5),  $d_{13}$  is the same as  $d_{22}$  i.e. the discrepancy  $d_{13}$  is shifted to  $d_{22}$ . It is also observed that when  $s_1=s_2=\dots=s_{v-1}=0$  but  $s_v \neq 0$ , then not only we have  $d_{11}=d_{21}=\dots=d_{v-1,1}=0$  but  $d_{v1} = d_{v-1,2} \leq \dots = d_{1v} = S_v \neq 0$  as shown in Fig.1.

$$\begin{bmatrix} 0 & 0 & \dots & \dots & x \\ 0 & & & x & \\ \vdots & & & & \\ 0 & x & & & \\ x & & & & \end{bmatrix}$$

Fig.1. Discrepancy Matrix

Recall that from (3) we have  $a_{ij} = S_{i+j-1}$ , and thus  $a_{ij} = a_{uv}$  if and only if  $i+j=u+v$ . In general,  $a_{ij} = a_{i-1,j+1} = \dots = a_{1,j+i-1} = S_{i+j-1}$ . For the case in Fig.1., we have  $a_{v1} = a_{v-1,2} = \dots = a_{1v} = S_v$ . Because  $f^{(1)}(x) = 1$ , then  $d_{v1} = d_{v-1,2} = \dots = d_{1v} = S_v \neq 0$ .

As a result of the property of discrepancy shifting, we can skip many steps of the FIA. In addition, only one item needs to be stored in Tables D and C respectively at any given time. When these modifications are incorporated in this application, the FIA becomes the Conan's MPR algorithm.

The modifications are discussed in the following. For cross reference, the notations in (3) will help to observe the links between FIA and MPR algorithm.

Let us consider the situation after the  $m$ th step of iteration when  $s_1, s_2, \dots, s_m$  have been considered and (1) is satisfied for  $L+1 \leq k \leq m$ . Let us call

the smallest  $L$  and the corresponding column polynomial  $f^{(L+1)}(x)$ , after this step of iteration,  $L_m$  and  $d^{(m)}(z)$  respectively. Then (1) is satisfied for  $L_m + 1 \leq k \leq m$  and we have

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_{L_m+1} \\ s_2 & s_3 & \cdots & s_{L_m+2} \\ \vdots & \vdots & & \vdots \\ s_{m-L_m} & \cdots & & s_m \end{bmatrix} \begin{bmatrix} d_0^{(m)} \\ \vdots \\ d_{L_m-1}^{(m)} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (4)$$

or equivalently

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,L_m+1} \\ a_{21} & & & a_{2,L_m+1} \\ \vdots & & & \vdots \\ a_{m-L_m,1} & \cdots & & a_{m-L_m,L_m+1} \end{bmatrix} \begin{bmatrix} c_m^{(L_m+1)} \\ \vdots \\ c_1^{(L_m+1)} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (5)$$

Therefore, we have  $d^{(m)}(z) = z^{L_m} + d_{L_m-1}^{(m)} z^{L_m-1} + \cdots + d_0^{(m)}$

and  $f^{(L_m+1)}(x) = 1 + c_1^{(L_m+1)} x + \cdots + c_{L_m}^{(L_m+1)} x^{L_m}$

with  $d^{(m)}(S_k) = 0$  for  $L_m+1 \leq k \leq m$  (6)

and  $f^{(L_m+1)}(a_{i,L_m+1}) = 0$  for  $1 \leq i \leq m-L_m$  (7)

Observe that the polynomial  $d^{(m)}(x)$  is indexed by the subscript of the syndrome

terms  $S_m$  subtracted from  $L_m$  ( $d_{L_m-j}$ ) while the polynomial  $f^{(L_m+1)}(x)$  is indexed by the column number  $L_m+1$ . We notice as we apply the FIA that  $f^{(L_m+1)}(x)$  may be modified as the elements in column  $L_m+1$  are processed. If we desire an exact correspondence between the two polynomials, we can add a row index to the column polynomial  $f^{(L_m+1)}(x)$ , so that after the  $m$ th step of iteration (when the element  $S_m$  at column  $L_m+1$  and row  $m-L_m$  is processed),  $f^{(L_m+1)}(x)$  is designated as  $f^{(m-L_m, L_m+1)}(x)$ . Then  $f^{(m-L_m, L_m+1)}(x) = d^{(m)}(z)$ .

In general we have

$$f^{(ij)}(x) = d^{(i+j-1)}(z) \quad (8)$$

Also notice that in (4) and (5) there are  $m-L_m$  equations satisfied by the syndrome terms  $s_1$  to  $s_m$ . Hence, the sequence  $\{s_1, s_2, \dots, s_m\}$  is divided into  $m-L_m$  recurrent subsequences of length  $L_m+1$ . If only one equation is satisfied as in case {III.5.(5)} for nonrecurrent sequences, we have  $m-L_m=1$  and  $L_m+1=m$ . Then  $f^{(L_m+1)}(x) = f^{(1, L_m+1)}(x) = d^{(L_m+1)}(z) = d^{(m)}(z)$ . The indices we adopted are thus agreeable.

At the  $(m+1)$ th step, the algorithm is to find the smallest  $L_{m+1}$  and  $d^{(m+1)}(x)$  such that

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_{L_{m+1}+1} \\ s_2 & s_3 & \cdots & \vdots \\ \vdots & & & \\ s_{m+1-L_{m+1}} & & & s_{m+1} \end{bmatrix} \begin{bmatrix} d_{L_{m+1}}^{(m+1)} \\ \vdots \\ d_1^{(m+1)} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (9)$$

or equivalently, the  $f^{(m+1-L_{m+1}, L_{m+1}+1)}(x)$  of shortest length  $L_{m+1}+1$  such that

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,L_{m+1}+1} \\ a_{21} & & & \vdots \\ \vdots & & & \\ a_{m+1-L_{m+1},1} & \cdots & & a_{m+1-L_{m+1}+1} \end{bmatrix} \begin{bmatrix} c_{L_{m+1}}^{(L_{m+1}+1)} \\ \vdots \\ c_1^{(L_{m+1}+1)} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (10)$$

$$\text{Let } d_{ij} = d_{i+j-2} \quad (11)$$

$$\text{Then } d_{i,L_{m+1}} = d_{i+L_{m-1}}$$

$$= f^{(L_{m+1})}(a_{i,L_{m+1}})$$

$$= d^{(m)}(S_{L_{m+1}+i})$$

From (4) and (5), we have for  $i = 1, 2, \dots, m-L_m$ , respectively

$$d_{i,L_{m+1}} = d_{i+L_{m-1}} = 0$$

When  $i = m-L_m+1$ , we have

$$\begin{aligned} d_{m-L_m+1,L_{m+1}} &= f^{(L_{m+1})}(a_{m-L_m+1,L_{m+1}}) \\ &= d_m \\ &= d^{(m)}(S_{m+1}) \\ &= s_{m+1} + d_{L_{m-1}}^{(m)} s_m + \cdots + d_0^{(m)} s_{m-L_m} \end{aligned} \quad (12)$$

which is referred to as the  $m$ th discrepancy.

If  $d_{m-L_m+1,L_{m+1}} = d_m = 0$ , then there is no change of  $f^{(L_{m+1})}(x)$  and  $d^{(m+1)}(z) = d^{(m)}(z)$  with  $L_{m+1} = L_m$ .

When  $d_{m-L_m+1,L_{m+1}} \neq 0$ , suppose there exists  $d^{(m)}(z)$  for  $1 \leq l \leq m$  such that (1) is satisfied for  $L_l+1 \leq k \leq l$  but not for  $k = l+1$ . Then there exist  $i'$  and  $L' = L_m$  such that  $d_{i',L'+1}^{i'} = d_{i',L_{m+1}}^{i'} = d_{i'+L_l-1}^{i'} = 0$  for  $1 \leq i' \leq l-L_l$ , but

when  $i' = l - L_l + 1$ ,  $d_{i'} = d_{l-L_l+1, L_l+1} = d_m \neq 0$ .

Next we examine the following two cases.

Case 1:  $i = m - L_m + 1 > i' = l - L_l + 1$

In this case, the D matrix appears as in Fig.2.

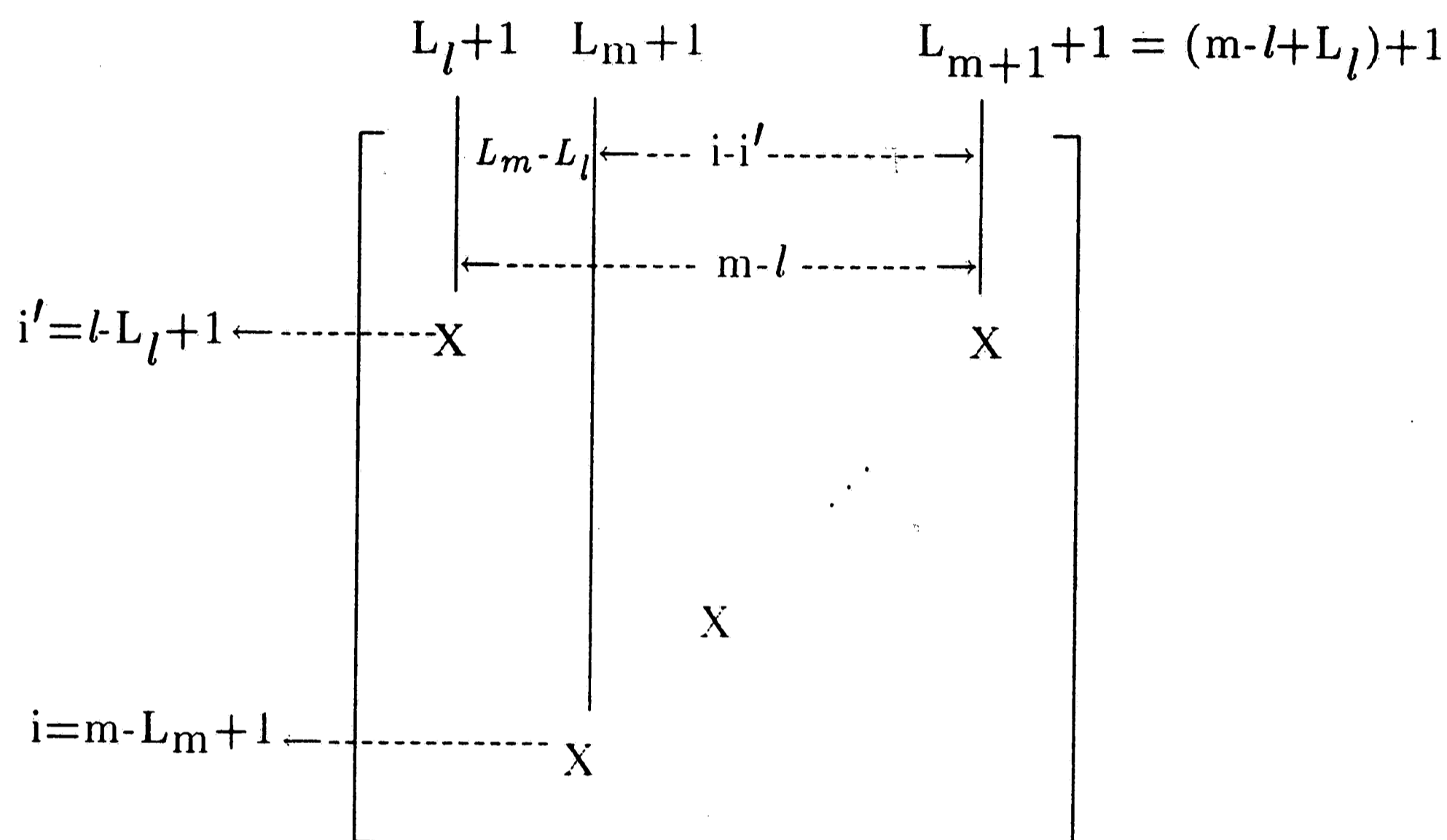


Fig.2. D matrix for Case 1

$$\begin{aligned} \text{Hence, } d_m &= d_{m-L_m+1, L_m+1} \\ &= d_{m-L_m, L_m+2} \\ &= d_{l-L_l+1, m-l+L_l+1} \end{aligned}$$

Thus,  $d_m$  can be lined up with  $d_l$  at row  $l - L_l + 1$  and column  $m - l + L_l + 1$  and be eliminated by letting  $L_{m+1} = L_m + i - i'$  and

$$f^{(L_{m+1}+1)}(x) = f^{(L_m+1)}(x) - \frac{d_{i, L_m+1}}{d_{i', L_l+1}} x^{(L_m - L_l) + (i - i')} f^{(L_l+1)}(x)$$

or equivalently by letting

$$d^{(m+1)}(z) = z^{(m-l) - (\deg\{d_m(z)\} - \deg\{d_l(z)\})} d^{(m)}(z) - q_m q_l^{-1} d^{(l)}(z)$$

when  $\deg\{d_{m+1}(z)\} = m - (l - \deg\{d_l(z)\})$

This implies that we can skip all steps between the position at row  $n-l_n+1$  and column  $l_n+1$  and the position at row  $m-l_m+1$  and column  $n-m+l_m+1$ .

$$\text{Case 2 : } i = m-L_m+1 \leq i' = l-L_l+1$$

In this case the situation in the  $D$  matrix appears as in Fig.3.

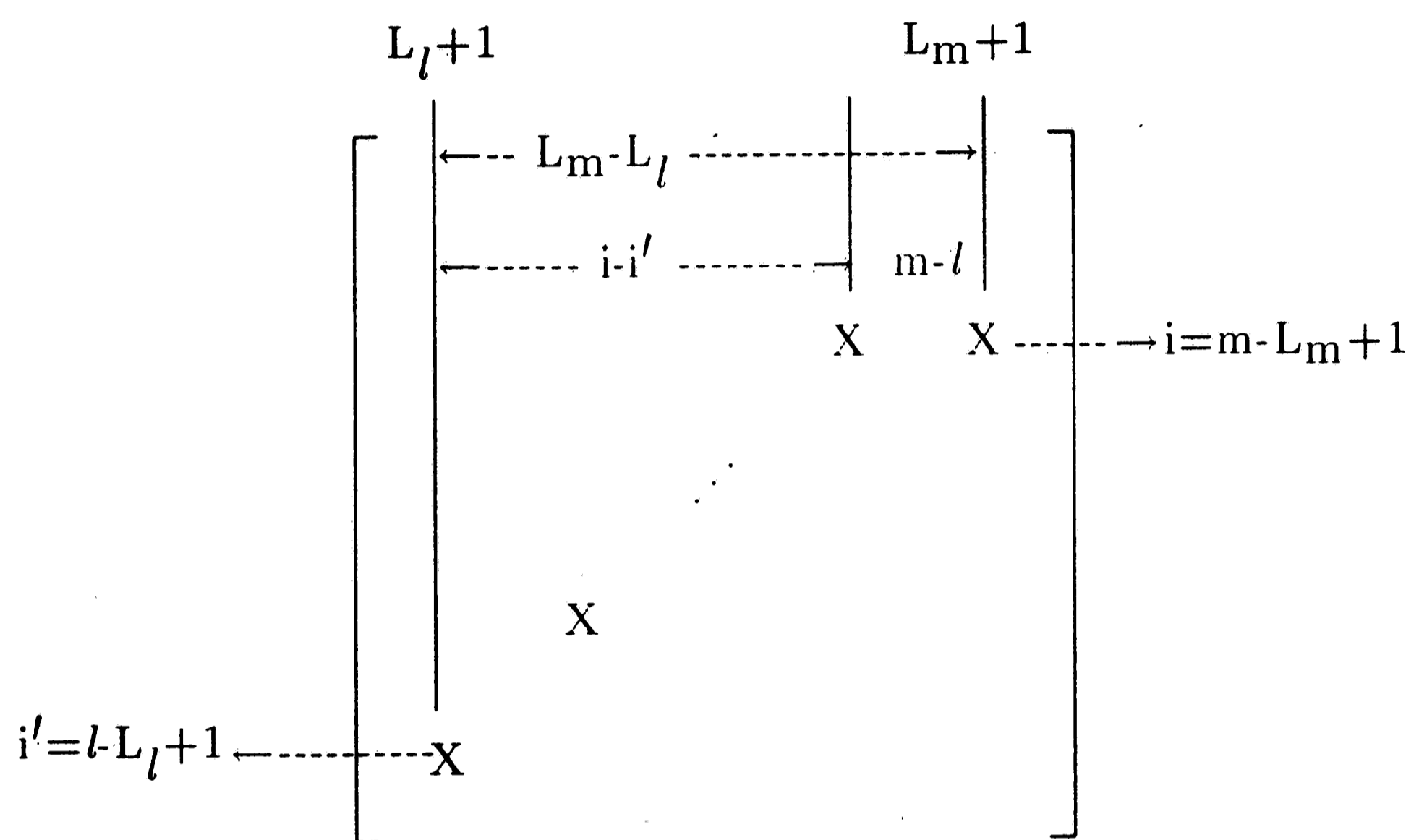


Fig.3.  $D$  matrix for Case 2

$$\begin{aligned} \text{Then we have } d_l &= d_{l-L_l+1, L_l+1} \\ &= d_{l-L_l, L_l+2} \\ &= d_{l-L_m+1, l-m+L_m+1} \end{aligned}$$

Therefore, there exists a  $d_l$  at row  $m-L_m+1$  and  $d_m = d_{m-L_m+1, L_m+1}$  can be eliminated when  $f^{(m-L_m, L_m+1)}(x)$  is replaced by

$$f^{(m-L_m+1, L_m+1)}(x) - \frac{d_{i, L_m+1}}{d_{i', L_l+1}} x^{(L_m-L_l)+(i-i')} f^{(L_l+1)}(x)$$

or equivalently by letting

$$d^{(m+1)}(z) = z^{(m-l)-(\deg\{d_m(z)\}-\deg\{d_l(z)\})} d^{(m)}(z) - q_m q_l^{-1} d^{(l)}(z)$$



when  $\deg\{d_{m+1}(z)\} = \deg\{d_m(z)\}$

To complete the derivation, we need to find an initial solution  $d^{(l)}(z)$  with  $d_l \neq 0$  (or  $f^{(L'+1)}(x)$  with  $d_{i',L'+1} \neq 0$ ). Suppose  $s_1 = s_2 = \dots = s_{v-1} = 0$  and  $s_v \neq 0$  as exemplified in Fig.1., then we have  $d_{v,1} = d_{v-1,2} = \dots = d_{1,v} = s_v \neq 0$ . Hence the next step is to consider the element at row  $L$  and column  $v+1$ . Thus we have as initial conditions

$$\begin{aligned} L' = 0, \quad i' = v, \quad f^{(L'+1)}(x) = f^{(1)}(x) = 1 \\ d_{i',L'+1} = d_{v,1} = s_v \neq 0; \quad L = v, \quad i = 1 \end{aligned} \quad (13)$$

$$\text{Then } d_{i,L+1} = d_{1,v+1} = s_{v+1}$$

which is equivalent to

$$\begin{aligned} l = v-1, \quad d^{(l)}(z) = 1, \quad L_l = 0, \quad d_l = s_{l+1} = s_v \neq 0; \\ m = v, \quad d^{(m)}(z) = 1, \quad L_m = v, \quad d_m = s_{m+1} = s_{v+1} \end{aligned} \quad (14)$$

In addition, we may add an  $s_0=1$  to the syndrome sequence resulting in a

$$S' = \begin{bmatrix} s_0 & s_1 & s_2 & \dots \\ s_1 & s_2 & s_3 & \dots \\ \vdots & & & \\ s_{d_0-1} & \dots & & \end{bmatrix}$$

which can be considered either as

$$(1) \left[ \begin{array}{c|c} \begin{matrix} s_0 \\ s_1 \\ s_2 \\ \vdots \end{matrix} & S \end{array} \right] \quad \text{or (2)} \left[ \begin{array}{c} \begin{matrix} s_0 & s_1 & s_2 & \dots \end{matrix} \\ \hline S \end{array} \right]$$

In the first configuration,  $s_0$  is considered as  $a_{10}$  while in the second configuration as  $a_{01}$ . In either case, the next element to be considered should be  $s_1 = a_{11}$  in matrix  $S$ . Following the basic algorithm the initial conditions for the first case are

$$\begin{aligned} L' &= -1 & i' &= 1, & f^{(L'+1)}(x) &= f^{(0)}(x) = 1 \\ d_{(i', L'+1)} &= d_{1,0} = s_0 = 1 \\ L &= 0 & i &= 1 & f^{(L+1)}(x) &= f^{(1)}(x) = 1 \\ d_{i, L+1} &= d_{11} = s_1 \end{aligned} \tag{15}$$

while for the second case, we have

$$\begin{aligned} L' &= 0, & i' &= 0 & f^{(L'+1)}(x) &= f^{(1)}(x) = 1 \\ d_{i', L'+1} &= d_{0,1} = s_0 = 1 \\ L &= 0 & i &= 1 & f^{(L+1)}(x) &= f^{(1)}(x) = 1 \\ d_{i, L+1} &= d_{11} = s_1 \end{aligned} \tag{16}$$

which is equivalent to

$$\begin{aligned} l &= -1, & d^{(l)} &= d^{(-1)}(z) = 1, & L_l &= L_{-1} = 0 \\ d_l &= d_{-1} = s_0 = 1 \\ m &= 0, & d^{(m)}(x) &= d^{(0)}(z) = 1 & L_m &= L_0 = 0 \\ d_m &= d_0 = d^{(0)}(s_1) = s_1 \end{aligned} \tag{17}$$

which also corresponds in terms of the notation of [ 7 ] to

$$\begin{aligned} l &= -1 & d_l(z) &= d_{-1}(z) = 1 & \deg\{d_l(z)\} &= \deg\{d_{-1}(z)\} = 0 \\ d_l &= d_{-1} = s_0 = 1 \\ m &= 0 & d_m(z) &= d_0(z) = 1 & \deg\{d_m(z)\} &= \deg\{d_0(z)\} = 0 \\ d_m &= d_0(z) = d_0(s_1) = s_1 \end{aligned} \tag{18}$$

Notice that these initial conditions do not agree with Conan's MPR algorithm initial conditions. If we apply these conditions in Conan's MPR algorithm  $d_1(z)$  becomes zero and stays zero for the following steps. That is why we adopt a set of

initial conditions that lead to a feasible solution. This is achieved when Step 4 of Conan's MPR algorithm produces a nonzero value.

$$d(z) = z^i d(z) - s_i b(z) = z^i$$

This means  $b(z) \equiv d_{-1}(z) = 0$ , not 1. In addition,  $n(z) = 0$  in order for the 0th realization to be minimal. The realization is minimal in the MPR algorithm only when the  $z^0$  coefficient of  $n(z)$  is zero. Then to avoid  $n(z)$  to stay zero the following equation should be able to produce nonzero values.

$$n(z) = z^i n(z) - s_i c(z) = s_i$$

This implies  $c(z) = -1$ . Hence, we establish a set of initial conditions for the MPR algorithm.

The solutions of Fundamental and Conan's MPR algorithms are not always the same as we can see from the examples (III.3 and III.5). The 3rd partial realization of the MPR algorithm agrees with the FIA because the realization is minimal and therefore unique. However, the 4th partial realization is not minimal. When this happens the degree of the solution polynomial of the FIA is greater by {1} compared with the solution polynomial of the MPR algorithm. The solutions of the FIA and MPR algorithms are equivalent since the problems they are solving are equivalent. The FIA problem satisfies the equation

$$A(x) \sigma(x) \equiv \omega(x) \pmod{x^N}. \quad (A(x), \sigma(x), \omega(x) \text{ are polynomials}).$$

The MPR algorithm satisfies

$$S(z) d(z) \equiv n(z) \pmod{z^{-N+d}}$$

$S(z)$ ,  $d(z)$ ,  $n(z)$  are polynomials and  $d = \deg\{d(z)\}$ . The same problem in FIA notation is

$$x^{-N} A(x) \sigma(x) \equiv \omega(x) x^{-N} \pmod{1}.$$

For a further comparison, we show the flowcharts of the MPR as derived

from the FIA and the MPR algorithm in Fig.4. and Fig.5., and thus, we have concluded our derivation of Conan's MPR algorithm from the FIA.

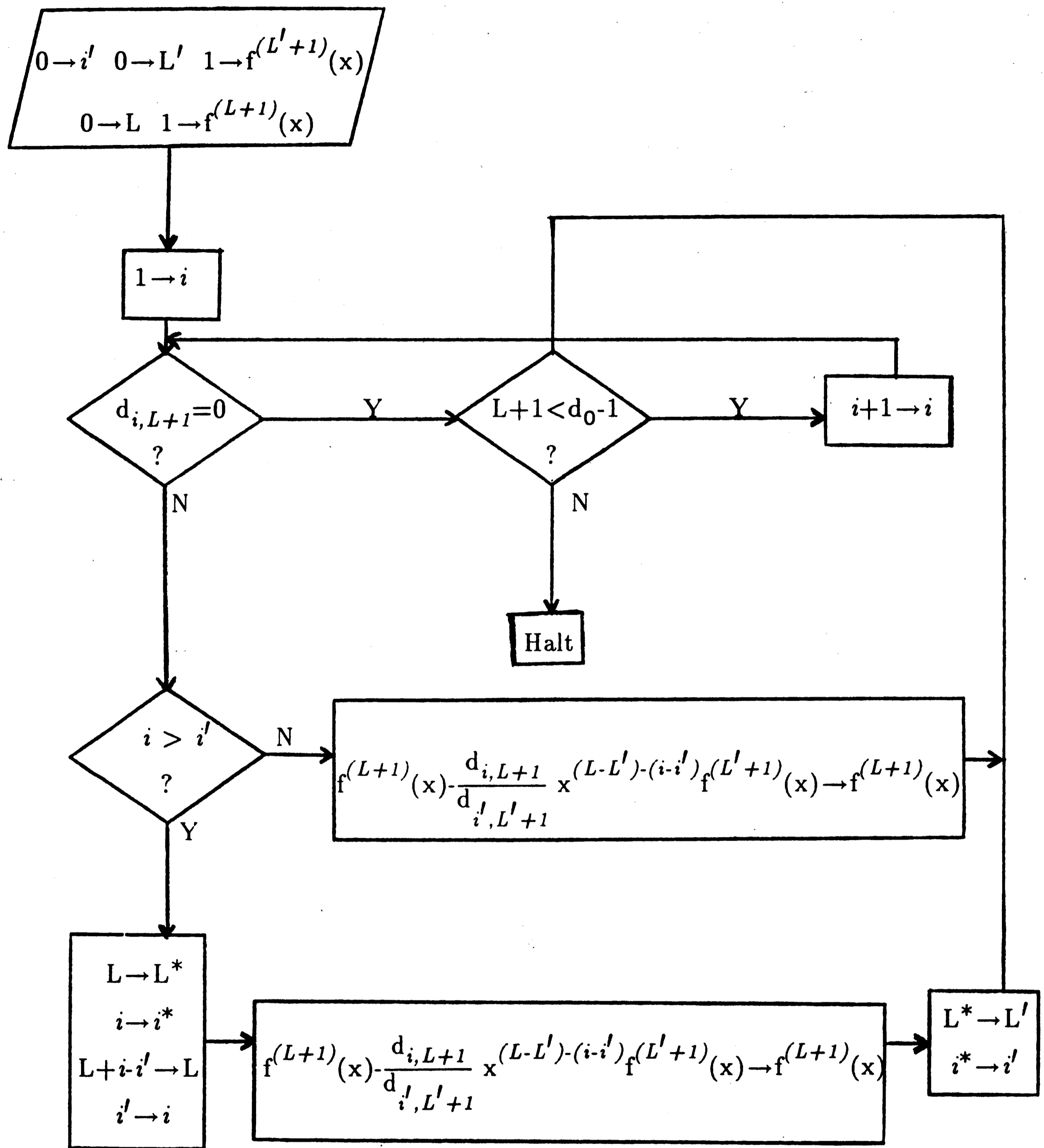


Fig. 4. Conan's MPR Algorithm as derived from FIA

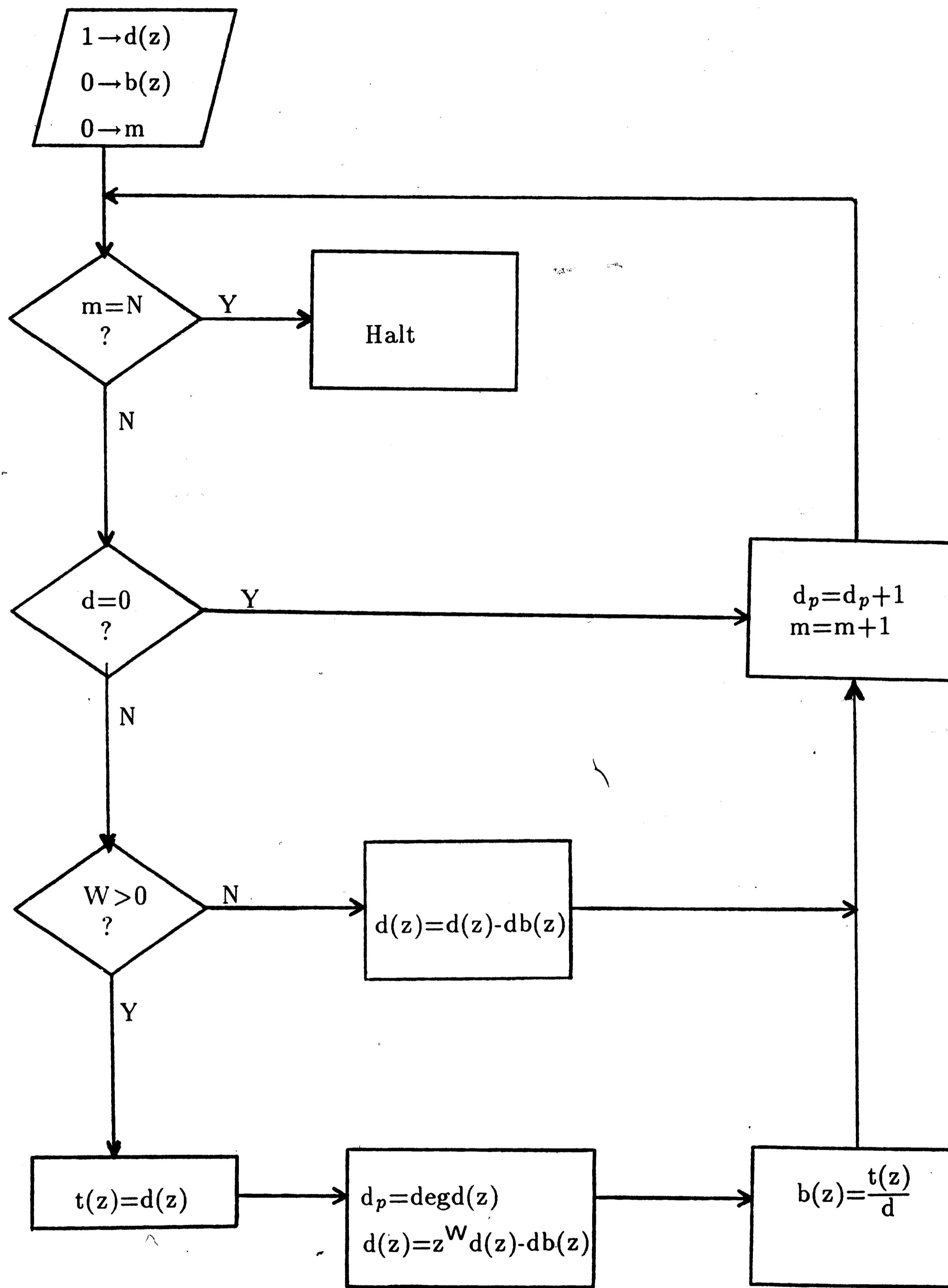


Fig 5. Conan's MPR Algorithm

## Chapter V - Conclusion

We discussed so far three algorithms that solve the minimal partial realization problem, and two algorithms which are used to decode BCH codes. Since the problems are equivalent, any of these five algorithms can be used to solve the realization problem or to decode BCH codes.

Some of these algorithms have more capabilities than the rest. The MPR algorithms can realize a sequence of  $p \times m$  matrices. Conan's MPR algorithm and Berlekamp-Massey algorithm can be implemented by a shift register. The FIA can decode cyclic codes beyond the BCH bound up to the Hartmann-Tzeng bound. The FIA can also decode a set of multiple scalar sequences.

A problem for further research is to develop a generalized version of Conan's MPR algorithm to decode codes beyond the BCH bound. Also it can be expanded to solve a set of semi-infinite sequences. An even further expansion is to develop an algorithm that can solve a set of semi-infinite sequences of matrices.

Another problem is to generalize the FIA to solve a set of semi-infinite sequences of matrices.

## REFERENCES

1. Berlekamp, E.R. Algebraic Coding Theory. New York: McGraw-Hill, 1968.
2. Massey, J.L. "Shift-Register Synthesis and BCH Decoding" IEEE Trans. on Inf. Theory, vol. IT-15, January (1969), pp. 122-127.
3. Ho, B.L. and Kalman, R.E. "Effective construction of linear state variables from input/output functions" Regelungstechnik, vol. IT-14, (1966), pp. 545-548.
4. Rissanen, J. "Recursive identification of linear systems" SIAM J. Control, vol. IT-9, August (1971), pp. 420-430.
5. Dickinson, B.W. and Morf, M. and Kailath, T., "A minimal realization algorithm for matrix sequence" IEEE Trans. Automat. Control, vol. IT-19, (1974), pp. 31-38.
6. Conan, J. "Structural properties of convolutional codes: An algorithmic approach with applications to linear multivariable system theory" Ph.D. dissertation, Dept. Elec. Eng., McGill University, Montreal, 1980.
7. Conan, J. "A recursive procedure for the solution of the minimal partial realization problem for scalar rational sequences," Rev. Roumaine Math. Pures Appl., vol. IT-30, (1985), pp. 625-645.
8. Feng, G.L. and Tzeng, K.K. "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis" IEEE Inter. Sympos. on Inform. Theory, St. Jovite, Quebec, September (1983).
9. Kalman, R.E. and Falb, P.L. and Arbib, M.A. Topics in Mathematical System Theory. New York: McGraw-Hill, 1969.
10. Lin, S. and Costello, D.J.Jr.. Error Control Coding: Fundamentals and Applications. Englewood Cliffs, N.J.: Prentice-Hall, 1983.



11. Kalman, R.E. and Declaris, N.. Aspects of Network and System Theory. New York: Holt, Rienhart and Wiston, New York, 1971.

## VITA

Elias G. Rogaris was born on December 27, 1962 in Athens, Greece. From 1981 to 1984 he attended Ulster County Community College in Stone Ridge, New York where he received his Associate of Science in Engineering in June 1984. From 1984 until 1987 he attended the Ohio State University in Columbus, Ohio where he received his Bachelor of Science in Electrical Engineering in June 1987. Since 1987, he has been a graduate student at Lehigh University in Bethlehem, Pennsylvania, in the Department of Computer Science and Electrical Engineering where he is working on his Master of Science degree in Electrical Engineering under Professor K.K. Tzeng. His interests are communications systems and algebraic coding theory.