On the Decoding of BCH-Codes over Z_{p^m} and Z_p

By

Muhammad Asif

Department of Mathematics
Quaid-i-Azam University
Islamabad, Pakistan
2016

On the Decoding of BCH-Codes over Z_{p^m} and Z_p

By

Muhammad Asif

A DISSERTATION SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF PHILOSOPHY

IN

MATHEMATICS

Supervised By

Prof. Dr. Tariq Shah

Department of Mathematics
Quaid-i-Azam University
Islamabad, Pakistan

2016

On the Decoding of BCH-Codes over Z_{p^m} and Z_p

By

Muhammad Asif

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY IN MATHEMATICS

We accept this thesis as conforming to the required standards

1. _____
Prof. Dr. Tariq Shah
(Supervisor)

2. _____
Prof. Dr. Akbar Azam
(External Examiner)

3. _____
Prof. Dr. Tasawar Hayat
(Chairman)

DEPARTMENT OF MATHEMATICS
QUAID-I-AZAM UNIVERSITY
ISLAMABAD, PAKISTAN
2016

Acknowledgment

I express my sincere thanks towards my guide Professor Dr. Tariq Shah for his continuous help, inspiration and encouragement throughout the thesis work also for providing directions to make progress in this work. Without their invaluable guidance, this work would never have been a successful one.

- I also like to convey my sincere gratitude to Professor Dr. Tasawar Hayat, Head of Department of Mathematics and all faculty members of Department of Mathematics, Quaid-i-Azam University, Islamabad.
- I also would like to thanks my friends specially Muhammad Tanveer, Yasir Naseer, Mubasher Umer, Usman Nazir, Abdullah Naeem, Usman Gillani, Imran Gillani and Wahid for their necessary cooperation in the accomplishment of dissertation.

Last but not least, I would like to dedicate my thesis to my great father, who got embraced "SHAHADAT" for our beloved country during service in Pakistan Army.

Muhammad Asif

Preface

The study of error correcting codes is called Coding theory. This area is discrete applied mathematics which includes the study and discovery of various coding schemes that are used to increase the number of errors that can be corrected during data transmission. Correcting errors is even more important when transmitting data that have been encrypted for security.

The development of data transmission codes starts with the first paper of Claude Shannon "A Mathematical theory of communication" in 1948. He explained in this paper that every communication channel has some capacity. If the rate of data transmission is smaller than the capacity then design of communication system for the channel is possible with the help of data transmission codes. This system has small probability of output errors but Shannon did not give the method for the construction of such type of codes.

The first block codes for this purpose were developed by Hamming in 1950. He represents the class of codes which correct only one error. However in 1954, Muller described the class of codes which correct multiple errors and Reed also in 1954 gave the decoding algorithm for these codes. But both these codes are not good for the Shannon's hypothesis.

The remarkable development in coding theory begins when Bose, Chaudhuri and Hocquenghem in 1960 explain the large class of codes which corrects multiple errors known as BCH codes and Reed-Solomon codes. They explain the BCH codes over finite fields that is Galois fields. In 1972, Blake [3] gave the method for construction of codes over ring $\mathbb{Z}_n$, where n is product of distinct primes But he does not explain the codes over $\mathbb{Z}_{p^m}$, for $m>1$. Further in 1975, Blake [4] discuss Linear codes over the ring $\mathbb{Z}_n$, where $n=p^r$ and p is a prime and $r \in Z^+$. He also define Hamming and Reed-Solomon codes over Galois field by considering their properties.

Spiegel [15] in 1977 shows that codes over $\mathbb{Z}_{p^m}$ can be described in terms of codes over $\mathbb{Z}_p$ and thus we are able to define codes over $\mathbb{Z}_n$ for any positive integer n. In 1979, Shankar [14] constructs the BCH codes over $\mathbb{Z}_{p^m}$. Shankar use $\mathbb{Z}_{p^m}[x]$ which is the polynomial ring in variable x over $\mathbb{Z}_{p^m}$. BCH codes over finite commutative rings with identity are constructed by Andrade and Palazzo [1] in 1998. In the construction techniques of both [1] and [14], the cyclic subgroup of the group of units of an extension ring is specified. Interlando, Palazzo and Elia [6] proposed powerful decoding method which is based on Barlekamp-Massey algorithm for RS and BCH codes over integer residue ring $\mathbb{Z}_{p^m}$. This is very difficult to decoding manually over the higher order integer residue ring and Galois ring.

There are many algebraic concepts which if incorporated in Computer and Information technologies, can have prodigious impacts. For instance, Galois rings and maximal cyclic

subgroups of groups of units of Galois rings. The most widespread application of these concepts can be seen in the coding theory. The aim of using this algebraic structure is to provide the secure data transmission because it makes the communication system more complicated in encoding and decoding process. Galois Ring in coding theory are first used to n length primitive BCH codes. By using the Galois ring of greater size the exhaustive search method becomes more difficult.

So in proposed work, the modified Barlekamp Massey algorithm solved computationally which is used for decoding of BCH codes and Reed-Solomon codes.

Dissertation details are hereunder:

- First chapter serves as the pillar under the foundations of Algebraic Coding Theory. In this chapter, some basic concepts, needed to understand the discussion in the next chapters, are defined. It shall be understood that this work requires some knowledge from both mathematics and computer science. This chapter consists of two sections. In the first section, we define some fundamental Algebraic structures. While in the other section some basics of Coding Theory are discussed.
- In second chapter, the concept encoding and decoding over Galois Field is discussed in detail. This chapter consists of three sections. In the first section, we discuss historical background of BCH codes, Reed Solomon codes and Barlekamp Massey algorithm. In the second section, we explain the encoding and decoding of BCH codes over Galois field with the help of Barlekamp Massey algorithm and extend this algorithm to simplified algorithm for binary BCH codes. While in third section we develop an algorithm in computer language for decoding BCH codes computationally.
- In third chapter, the concept of encoding and decoding of BCH and RS codes over integer residue ring and Galois ring are discussed. This chapter consist of four sections. In the first section, we discuss the historical background of codes over $\mathbb{Z}_{p^m}$. In the second section, we explained the encoding of BCH codes over extension ring of $\mathbb{Z}_{p^m}$ by constructing the maximum cyclic subgroup of group of units of Galois ring. In third section, we explained the decoding procedure of RS and BCH codes over integer residue ring and Galois ring respectively with the help of modified Barlekamp Massey algorithm. While in section four, we develop the program in C# for decoding of RS and BCH codes over integer residue ring and Galois ring respectively.

Contents

Chapter 1

Some Fundamental Algebraic Structures

1.1 Basic concepts

In this part of dissertation we discuss some definitions of algebraic structures. Some part of this section is taken from [7].

Binary Operation: Let $\Phi \neq S$ and "$*$" be a mapping $* : S \times S \to S$ then "$*$" is binary operation if S is closed under the operation '$*$' or $*(s_1, s_2) = s_1 * s_2 \in S \qquad for\ all\ s_1, s_2 \in S.$

Example 1 *: Z , Q, R, C are closed under addition so "$+$" is binary operation of these non empty sets.*

Remark 2 *: Z is not closed under division because if we divide two integers then it is not necessary that result is an integer so division is not binary operation of Z.*

Algebraic Structures

Any set $\Phi \neq S$ which have atleast one binary operation is called algebraic structure.

Semigroup: Let $\Phi \neq S$ with binary operation "$*$" is called semigroup $(S, *)$ if binary operation is associative on S that is if for all $s_1, s_2, s_3 \in S$ then

$$s_1 * (s_2 * s_3) = (s_1 * s_2) * s_3.$$

Example 3 *:Z, Q, R, C are semi groups under the binary operation addition and multiplication.*

Remark 4 *:Z is not semi group under the binary operation subtraction.*

Monoid: A semigroup $(S, *)$ with identity is called monoid that is an element $e \in S$ is called identity of semigroup S if

$s * e = e = e * s \ for \ all \ s \in S$ then S is called monoid.

Example 5 *:Z, Q, R, C are monoid under both binary operations "+"and " $\cdot$ ".*

Remark 6 *:Set of natural numbers N have not monoid under "+".*

1.1.1 Groups

A monoid $(S, *)$ with each element of S is invertible is called group. An element $s_1 \in S$ is called invertible if there exists $s_2 \in S$ such that

$$s_1 * s_2 = e = s_2 * s_1 \quad for \ all \ s_1 \in S,$$

where e is identity of S.

Example 7 *:Q, R, C are groups under "+" and " $\cdot$ " with identities 0 and 1 respectively.*

Remark 8 *:Z is not group under multiplication because inverse of each element of Z is not exist.*

Remark 9 *:$\{0\}$ and $\{1\}$ are trivial groups under "+" and " $\cdot$ " respectively.*

Remark 10 *:The identity and inverse of every element of a group is always unique.*

Abelian Group: A Group $(S, *)$ is abelian if binary operation "$*$" is commutative that is $s_1 * s_2 = s_2 * s_1 \ for \ all \ s_1, s_2 \in S$.

Example 11 *:Q, R, C, V_4 are abelian groups under multiplication.*

Remark 12 *:Group of matrices are not abelian groups under operation multiplication.*

Example 13 *Let $M_n(R)$ denote the collection of all non singular matrices where entries of matrices are come from set of real numbers R forms a non abelian group with respect to multiplication. $M_n(R)$ is also known as $GL(n, R)$ which is general linear group.*

Binary Relation: Let $\Phi \neq S$ then any non empty subset of $S \times S$ is called binary relation.

Equivalence Relation: Any relation which is binary R on a set $\Phi \neq S$ is called equivalence relation if the following conditions are satisfied.

(i) **Reflexive**: Any binary relation is called reflexive if $sRs \; for \; all \; s \in S$ and we also denote it $(s, s) \in R$.

(ii) **Symmetric:** A binary relation is called Symmetric if $s_1 R \; s_2$then $s_2 R s_1 \; for \; all \; s_{1,} s_2 \in S$.

(iii) **Transitive:** A binary relation is called transitive if $s_1 R \; s_2$ and $s_2 R \; s_3$ then $s_1 R s_3 \; for \; all \; s_1, s_2, s_3 \in S$.

Subgroup: Let $\Phi \neq T \subseteq S$ and $(S, *)$ be a group then T is called subgroup of S if T is group itself under the operation "$*$".

We write mathematically as $T \leq S$.

Example 14 *:$\mathbb{Z}$ under the binary operation "$+$" is a subgroup of $(\mathbb{Q}, +), (\mathbb{R}, +), (\mathbb{C}, +)$.*

Remark 15 *:Every group S has atleast two subgroups namely $\{e\}$ and S are called trivial subgroups and anyother subgroup of S is called proper subgroup of S.*

Theorem 16 *:Let S be a group and $T \subseteq S$ then T is called subgroup of S if and only if $t_1 t_2^{-1} \in T \; for \; all \; t_1, t_2 \in T$.*

Cyclic Group

Let S be a group then it is called cyclic group if it is generated by single element of S.

If S is under addition cyclic group then we denote it by,

$$S =< s >= \{ns \ : n \in Z\}.$$

If S is under multiplication cyclic group then we write it as,

$$S =< s >= \{s^n \ : n \in Z\},$$

where s is the generator of the cyclic group.

Example 17 *:Let $n \in \mathbb{Z}^+$ and S be the collection of all nth roots of unity then S is called cyclic group we denote it by $S =< w >= \left\{1, w, w^2, w^3, ..., w^{n-1}\right\}$.*

Remark 18 *:In cyclic group order of group and order of generator is always same.*

Remark 19 *:Cyclic group may have more then one generators.*

Example 20 *:Z is infinite cyclic group under addition it has two generators 1 and -1.*

Remark 21 *:Every abelian group is not cyclic group but converse is true.*

Example 22 *:$V_4$is abelian group but it is not cyclic.*

Theorem 23 *:A group having prime order is always cyclic.*

Normal Subgroups

Let H be a subgroup of a group G then H be a normal subgroup if

$$gH = Hg \ \ \forall g \in G$$

or

$$gHg^{-1} \subseteq H \ \ \forall g \in G$$

or

$$ghg^{-1} \in H \ \ \forall g \in G, h \in H,$$

we denote it as $H \underline{\Delta} G$.

Example 24 *:Centre of a group G is always normal subgroup.*

Example 25 *:Any subgroup is normal subgroup of group G if G is abelian group so $\{e, b\}$ is normal subgroup of V_4.*

Remark 26 *:$\{e\}$ and G are trivial normal subgroups of a group G.*

Remark 27 *:If $H \underline{\Delta} G$ then left and right cosets of H in G are equal for all $g \in G$.*

Theorem 28 *:Any subgroup having index 2 in a group G is normal in G.*

Theorem 29 *:The product of two normal subgroups of group G are normal subgroup of G.*

Quotient or Factor Group

Let $H \underline{\Delta} G$ then the collection of all the left or right cosets of H in G become a group under the operation which is defined as,

$$Hg_1 \cdot Hg_2 = Hg_1g_2 \quad for\ all\ g_1, g_2 \in G,$$

and it is known as factor group and it is represented by G/H that is $G/H = \{Hg \ : \forall g \in G\}$.

Example 30 *:If $G = S_3 = \left\{e, a, a^2, b, ab, a^2b\right\}$ is become a group and $H = \left\{e, a, a^2\right\}$ is normal subgroup of G then factor group is*

$$G/H = \{H, bH\}.$$

Remark 31 *:Q/Z is infinite factor group but its each element have finite order.*

Homomorphism or structure preserving mapping

Let Φ be a mapping from $(G, *)$ to $(J, \circ)$ where G and J are groups under the operations '$*$' and '$\circ$' respectively then Φ is called homomorphism if

$$\Phi(g_1 * g_2) = \Phi(g_1) \circ \Phi(g_2) \quad for\ all\ g_1, g_2 \in G.$$

Kernel of a Homomorphism: let $\Phi : G \longrightarrow J$ be a homomorphism then kernel of the mapping Φ is defined as

$$ker\Phi = \{g \in G : \Phi(g) = e^{\circ}\}.$$

Remark 32 *:If $\ker \Phi$ is zero or identity then Φ is $1-1$.*

Endomorphism: A homomorphism from group $G \to G$ is known as endomorphism.

Monomorphism: A homomorphism $\Phi : G \longrightarrow G^*$ is said to be a monomorphism if it is $1-1$.

Epimorphism: A homomorphism $\Phi : G \longrightarrow G^*$ is called an epimorphism if it is onto.

Isomorphism: A homomorphism $\Phi : G \longrightarrow G^*$ is called an isomorphism if it is one-one and onto.

Automorphism: It is an isomorphism from a group $G \to G$.

Fundamental theorem of Structure preserving mapping

Let G and J be two groups and Φ be a mapping from $G \longrightarrow J$ is an epimorphism then

$$G/\ker \Phi \cong J.$$

2nd Isomorphism Theorem

Let G be a group and $H \leq G$, $K \underline{\Delta} G$ then

$$\frac{H}{H \cap K} \cong \frac{HK}{K}.$$

3rd Isomorphism Theorem

Let G be a group $K \underline{\Delta} G$ and $K \leq H$ and $H \ \underline{\Delta} G$ then $\frac{G/K}{H/K} \cong \frac{G}{H}$.

Permutation: Let $\Phi \neq X$ a bijective mapping $\Phi : X \longrightarrow X$ is said to be a permutation .The set of all permutations on X is represented by S_X. If X contains n elements then the set of all permutation denoted by S_n and if $|X| = n$ then $|S_n| = n!$.

Symmetric Group: The collection of all bijective mappings of a set consisting of n elements together with the usual multiplication of mappings as an algebraic operation is called S_n which is symmetric group of degree n.

Cyclic Permutation: A Permutation of the form $\alpha = \begin{pmatrix} a_1 & a_2 & a_3 & ... & a_k \\ a_2 & a_3 & a_4 & ... & a_1 \end{pmatrix}$ in which $\alpha(a_1) = a_2$, $\alpha(a_2) = a_3...$, $\alpha(a_k) = a_1$ is called cyclic permutation or cycle of length k and written as $\alpha = (a_1 \ a_2 \ a_3 \ ... \ a_k)$.

Transposition: A transposition is cycle having length two.

Example 33 *:A cycle of the form* $(x\ y)$ *is called transposition and* $(x\ y)^2 = I$.

1.1.2 Rings

A $\Phi \neq \widetilde{\mathbb{R}}$ together with binary operations '+' and '$\cdot$' is said to be a ring if it satisfies following conditions.

1) $\widetilde{\mathbb{R}}$ is abelian group under '+'.
2) $\widetilde{\mathbb{R}}$ is semigroup under '$\cdot$'.
3) $(\widetilde{\mathbb{R}}, \cdot)$ is distributive over addition.

Example 34 *:* $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$, $\mathbb{Z}_n$ *are the the examples of rings.*

Commutative Ring: If any Ring $\widetilde{\mathbb{R}}$ satisfies the condition of commutative with respect to "$\cdot$" that is $r_1 \cdot r_2 = r_2 \cdot r_1$ for all $r_1, r_2 \in \widetilde{\mathbb{R}}$ then it is called commutative ring.

Subring: Let $\Phi \neq S \subseteq \widetilde{\mathbb{R}}$ is the subring of $\widetilde{\mathbb{R}}$ if S is a ring itself.

Example 35 $2Z = \{0, \pm 2, \pm 4, \cdots\}$ *is commutative subring of* Z *without multiplicative identity.*

Example 36 *:*$M_n(\widetilde{\mathbb{R}})$ *is not commutative ring. Moreover if* $\widetilde{\mathbb{R}}$ *is commutative then* $M_n(\widetilde{\mathbb{R}})$ *is not a commutative ring with multiplicative identity.*

Remark 37 *:There is also a ring which is not commutative without identity that is* $M_n(2Z)$ *because* $2Z$ *have not multiplicative identity.*

Ideal of Ring

Let $\widetilde{\mathbb{R}}$ be a commutative ring with identity then any $\Phi \neq \widetilde{I} \subseteq \widetilde{\mathbb{R}}$ is called an ideal of ring $\widetilde{\mathbb{R}}$ if it satisfy the following axioms,

$$i)\ \ \widetilde{a}-\widetilde{b}\in\widetilde{I} \qquad for\ all\ \widetilde{a},\widetilde{b}\in\widetilde{I}.$$
$$ii)\ \ \widetilde{r}\,\widetilde{a}\in\widetilde{I} \qquad for\ all\ \widetilde{r}\in\widetilde{\mathbb{R}},\widetilde{a}\in\widetilde{I}.$$

Example 38 *:nZ are ideals of Z.*

Remark 39 *:It is not necessary that every subring is an Ideal but converse is true.*

Example 40 *: A ring of integers is subring of* $\mathbb{Q}$ *but it is not an ideal of* $\mathbb{Q}$.

Remark 41 *:Every field* $\mathcal{F}$ *have only two maximal ideal namely* $\{0\}$ *and* $\mathcal{F}$.

Remark 42 *:Every ring* $\widetilde{\mathbb{R}}$ *is an Ideal of itself.*

Zero Divisors

Let $\widetilde{\mathbb{R}}$ be a ring and $0\neq r_1\in\widetilde{\mathbb{R}}$ is said to be zero divisor from left if there exist $0\neq r_2\in\widetilde{\mathbb{R}}$ such that $r_1r_2=0$ where r_2 is right zero divisor and if $\widetilde{\mathbb{R}}$ is commutative then r_1 and r_2 are zero divisors of each other.

Example 43 *:* 2 *and* 3 *are zero divisors of each other in* Z_6 *because* $2\cdot 3=0$ *under modulo* 6.

Integral Domain

If there does not exist any zero divisor in ring $\widetilde{\mathbb{R}}$ then it is called an Integral domain.

Example 44 *:*$\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R},\mathbb{C}$ *are integral domain because these rings have no zero divisors.*

Remark 45 *:*$M_n(\widetilde{\mathbb{R}})$ *and* Z_4, Z_6, Z_8 *are not integral domains.*

Remark 46 *:Every field is integral domain because field has no zero divisors.*

Unit Element

Let $1\in\widetilde{\mathbb{R}}$ be a ring which is commutative then $U(\widetilde{\mathbb{R}})$ is the set of unit elements or invertible elements in $\widetilde{\mathbb{R}}$ and we defined it as let $0\neq r_1\in\widetilde{\mathbb{R}}$ is called unit element if there exist $0\neq r_2\in\mathbb{R}$ such that $r_1r_2=r_2r_1=1$.

Remark 47 *:*$U(\widetilde{\mathbb{R}})$ *is the set of unit elements forms a group under the operation "." and unit element can never be zero divisor.*

Field: Any ring $\widetilde{\mathbb{R}}$ with identity '1' is said to be field if $U(\widetilde{\mathbb{R}}) = \widetilde{\mathbb{R}} \setminus \{0\}$.

Prime Field A field is said to be prime field if it has no proper subfield.

Example 48 *:A field of rational numbers is prime field because it has no proper subfield and* Z_p *is also prime field where* p *is prime.*

Division Ring: A ring $\widetilde{\mathbb{R}}$ is said to be division ring if each non zero element of $\widetilde{\mathbb{R}}$ is unit.

Example 49 *:Every field is division ring.*

Remark 50 *:The unit elements of* Z_n *are those elements of* Z_n *having greatest common divisor is* 1 *with* n.

Nilpotent Element

Let $r \in \widetilde{\mathbb{R}}$ is said to be nilpotent if there exist $n \in \mathbb{Z}^+$ such that $r^n = 0$.

Example 51 *:* 2 *is nilpotent element in* Z_4 *because* $2^2 = 0$ *under modulo 4.*

Remark 52 *:Every zero divisor is not nilpotent element but converse is true.*

Example 53 2 *and* 3 *are zero divisors in* Z_6 *but these are not nilpotent elements so every zero divisor may or may not nilpotent element.*

Principal Ideal: Any Ideal $\widetilde{I}$ is called principal ideal if it is generated by single element.

Polynomial Ring

If $\widetilde{\mathbb{R}}$ is commutative ring with identity and $\mathbb{Z}_{\geq 0}$ is additive monoid then

$$\begin{aligned}
\widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}} &= \left\{ h \mid h : \mathbb{Z}_{\geq 0} \longrightarrow \widetilde{\mathbb{R}} \right\} \\
where\ h &: \mathbb{Z}_{\geq 0} \longrightarrow \widetilde{\mathbb{R}} \text{ is defined by} \\
h(0) &= h_0 \in \mathbb{R} \\
h(1) &= h_1 \\
&\vdots \\
h(n) &= h_n
\end{aligned}$$

so if $h \in \widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}$ then h can be written as $h = (h_0, h_1, h_2, ...h_n, ..)$.

$(\widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}, +, \cdot)$ is ring under the operation addition and multiplication which is defined as follows.

Let for any $f, g \in \widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}$ then ,

i)Addition: $f+g = (f_0, f_1, f_2, ...f_n, ..)+(g_0, g_1, g_2, ...g_n, ..) = (f_0 + g_0, f_1 + g_1, ..., f_n + g_n...)$.

ii)Multiplication: $fg(n) = \sum_{i+j=n} f_i g_j$

This implies that $fg = (f_0g_0, f_1g_0 + f_0g_1, f_2g_0 + f_1g_1 + f_0g_2, ..., f_0g_n + f_1g_{n-1} + ... + f_ng_0, ...)$

If we define a mapping,

$\widetilde{X} : Z_{\geq 0} \longrightarrow \widetilde{\mathbb{R}}$ by

$\widetilde{X}(q) = 0 \qquad if\ q \neq 1$

$\widetilde{X}(q) = 1 \qquad if\ q = 1$

If $\widetilde{X} = \left(\widetilde{X}_0, \widetilde{X}_1, \widetilde{X}_2, ..., \widetilde{X}_n, ...\right)$

then $\widetilde{X} = (0,\ 1,\ 0,\ 0, ..., 0\ , ...)$

and $\widetilde{X}^2 = \widetilde{X}.\widetilde{X} = (0, 1,\ 0, 0, ..., 0, ...)\cdot(0,\ 1,\ 0, 0, ..., 0, ...) = (0.0, 0.1 + 1.0, 0.0 + 1.1 + 0.0, 0, ..., 0, ...)$

$\widetilde{X}^2 = (0, 0, 1, 0, 0, ..., 0, ...)$

$\widetilde{X}^3 = (0, 0, 0, 1, 0, ..., 0, ...)$

$\widetilde{X}^4 = (0, 0, 0, 0, 1, ..., 0, ...)$

.

.

.

$\widetilde{X}^n = (0,\ 0,\ 0, ..., 0,\ 1,\ 0, ...)$

Let $f = (f_0, f_{1,....,}f_n, ...) \in \widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}$

$f = (f_0,\ 0,\ 0, ..., 0,\ ...) + (0,\ f_1,\ 0, ..., 0, ...) + ... + (0,\ 0,\ ..., f_n,\ 0, ...)$ since $\widetilde{\mathbb{R}}$ is imbedded in $\widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}$

$= f_0 + f_1\,(0,\ 1,\ 0, ..., 0,\ ...) + f_2\,(0,\ 0,\ 1, ..., 0, ...) + ... + f_n\,(0,\ 0, ..., 1, 0, ...) + ...$

$= f_0 + f_1\ \widetilde{X} + f_2\ \widetilde{X}^2 + ... + f_n\ \widetilde{X}^n + ...$

then f can be written as

$f = f_0 + f_1\ \widetilde{X} + f_2\ \widetilde{X}^2 + f_3\ ^3 + ... + f_n\ \widetilde{X}^n + ..$

$f\left(\ \widetilde{X}\right) = \sum_{i=0}^{\infty} f_i\ \widetilde{X}^i$

It is called formal power series in $\widetilde{\mathbb{R}}$ and we denote the ring $\widetilde{\mathbb{R}}^{\mathbb{Z}_{\geq 0}}$ by $\widetilde{\mathbb{R}}[[\widetilde{X}]]$ this ring is in one indeterminate $\widetilde{X}$ over the ring $\widetilde{\mathbb{R}}$.if we take finite terms from formal power series and say remaining terms are zeros then it becomes polynomial ring that is,

$\left\{\sum_{i=0}^{n} f_i\widetilde{X}^i \in \widetilde{\mathbb{R}}[[\widetilde{X}]] \mid n \in Z_{\geq 0}\right\}$.

This type of formal power series is called polynomial ring in one indeterminate and we denote it by $\widetilde{\mathbb{R}}[\widetilde{X}]$.

Remark 54 *:*$\widetilde{\mathbb{R}}[\ \widetilde{X}]$ *has finite terms but* $\widetilde{\mathbb{R}}[[\ \widetilde{X}]]$ *has infinite terms so* $\widetilde{\mathbb{R}}[\ \widetilde{X}] \subset \widetilde{\mathbb{R}}[[\widetilde{X}]]$.

Example 55 *:*$\mathbb{Z}[\ \widetilde{X}] \subset\ \mathbb{Q}[\widetilde{X}] \subset \mathbb{R}[\widetilde{X}] \subset \mathbb{C}[\widetilde{X}]$.

Example 56 *:*$\mathbb{Z}_p[\widetilde{X}]$ *is polynomial Ring with finite coefficients and it is also field because coefficient Ring is field.*

Local Ring: If Ring have only one maximal ideal then it is called local Ring.

Example 57 *:Every field is local Ring if* $F[\widetilde{X}]$ *is field then it is local and it has* $\widetilde{X}F[\widetilde{X}]$ *is only maximal ideal.*

Prime Element

A non unit element $0 \neq p$ of an integral domain $\widetilde{\mathbb{R}}$ is called prime element in $\widetilde{\mathbb{R}}$ if $p|r_1r_2$ then either $p|r_1$ or $p|r_2$ where $r_1, r_2 \in \widetilde{\mathbb{R}}$.

Example 58 *:*$2, 3, 5, 7...$ *are prime elements in the ring of integers* $\mathbb{Z}$.

Irreducible Elements

A non unit element $0 \neq q$ of an integral domain $\widetilde{\mathbb{R}}$ is called an irreducible element (atom) in $\widetilde{\mathbb{R}}$ if $q = r_1 r_2$ then either $r_1 \in U(\widetilde{\mathbb{R}})$ or $r_2 \in U(\widetilde{\mathbb{R}})$.

Example 59 *:Prime elements in Z are irreducible elements.*

Example 60 *:*$1 + \widetilde{X}^2 \in Z[\widetilde{X}]$ *is irreducible in* $\mathbb{Q}[\widetilde{X}]$.

Remark 61 *:In PID every irreducible element is maximal ideal.*

Remark 62 *:* $\frac{F[\widetilde{X}]}{(q(x))} \simeq$ *Field if and only if* $q(x)$ *is an irreducible element.*

Theorem 63 *:If* $\widetilde{\mathbb{R}}$ *is an integral domain and* $p, q \in \widetilde{\mathbb{R}}$ *then,*

i) q is irreducible iff (q) is maximal ideal.

ii) Every irreducible is not prime element but converse is true.

iii) If $\widetilde{\mathbb{R}}$ is GCD then every irreducible element is prime.

Example 64 *:The element* 3 *in the quadratic integer ring* $\mathbb{Z}[i\sqrt{5}]$ *is irreducible but it is not prime element because*

$\left(2 + \sqrt{5}i\right)\left(2 - \sqrt{5}i\right) = 3^2$ is divisible by 3 but neither $2 + \sqrt{5}i$ nor $2 - \sqrt{5}i$ is divisible by 3.

Primitive Polynomial

A polynomial with unit content is called primitive polynomial here content of polynomial is GCD of all coefficients of polynomial. If α is primitive root of irreducible polynomial in Galois field then corresponding polynomial is primitive polynomial or called generator of Galois field.

Example 65 *:If* $f = 2\widetilde{X} + 6\widetilde{X}^2 + 12 \in \mathbb{Z}[\widetilde{X}]$ *then* $C(f) = 2$ *so* f *is not primitive polynomial because* 2 *is not unit element of* $\mathbb{Z}$.

Example 66 *: If* $f = 2\widetilde{X} + 1 \in \mathbb{Z}[\widetilde{X}]$ *then it is primitive polynomial because* $C(f) = 1$ *which is unit element of* $\mathbb{Z}$.

Monic Polynomial: A polynomial whose leading coefficient is one is called monic polynomial.

Galois Ring

Let n, m, p be any positive integers, here p is any prime and m is degree of basic irreducible polynomial $f(x)$ then we define Galois ring as, $\frac{\mathbb{Z}_{p^n}[x]}{(f(x))} = \{p_0 + p_1 x + p_2 x^2 + ... + p_{m-1} x^{m-1} : p_0, p_1, ..., p_{m-1} \in \mathbb{Z}_{p^n}\}$.

We denote it as $\mathcal{R} = GR(p^n, m)$ and it is Galois extension ring of $\mathbb{Z}_{p^n}$ having p^{nm} elements.

Galois Field

$GF(p^m) = \frac{\mathbb{Z}_p[x]}{\nu(f(x))} = \{p_0 + p_1 x + p_2 x^2 + ... + p_{m-1} x^{m-1} : p_0, p_1, ..., p_{m-1} \in \mathbb{Z}_p\} = \mathcal{K}$.

It has p^m number of elements, where $\nu(f(x)) = R_p(f(x))$.

$R_p(f(x))$ is a polynomial $f(x)$ which has the coefficient modulo prime p and $\nu(f(x))$ is primitive irreducible polynomial of degree m over $\mathbb{Z}_p$.We denote GR(p^n, m) by $\mathcal{R}$ and $GF(p^m)$ by $\mathcal{K}$ and their multiplicative group of units are denoted by $\mathcal{R}^*$ and $\mathcal{K}^*$ respectively.

It is finite field which means that it has finite number of elements.

If $q = p^n$ where p is prime and $n \in Z^+$ and $f(x)$ is irreducible polynomial of degree m then

$$\frac{F_q[X]}{(f(x))} \simeq F_{q^m} = GF(q^m), \quad \text{where } m \text{ is degree of } f(x).$$

Example 67 *:If* $q = 2^1$ *and* $m = 3$ *then*

$$\frac{Z_2[X]}{< X^3 + X + 1 >} \simeq GF\left(2^3\right) = GF(8),$$

Here $X^3 + X + 1$ is an irreducible element in $Z_2[X]$. Let α is primitive root of $X^3 + X + 1$ then $\alpha^3 + \alpha + 1 = 0$.

$GF(8) = \left\{r + s\alpha + t\alpha^2 : r, s, t \in Z_2 \ , 1 + \alpha + \alpha^3 = 0\right\}$ also $GF(8) \setminus \{0\} = \langle \alpha \rangle$ so elements of $GF(8)$ are as follows,

0

$\alpha^1 = \alpha$

$\alpha^2 = \alpha^2$

$\alpha^3 = \alpha + 1$

$\alpha^4 = \alpha + \alpha^2$

$\alpha^5 = 1 + \alpha + \alpha^2$

$\alpha^6 = 1 + \alpha^2$

$\alpha^7 = 1$

$GF(8) = \left\{0, 1, \alpha, \alpha^2, \alpha + 1, \alpha + \alpha^2, 1 + \alpha + \alpha^2, 1 + \alpha^2\right\}.$

Remark 68 *:Z_n is field iff n is prime and $Z_p = F_p = GF(p)$ where p is prime number.*

Theorem 69 *:$F_{p^r} \subset F_{p^s}$ iff $r|s$.*

Example 70 *: $F_{2^2} \subset F_{2^4}$ because $2|4$.*

1.1.3 Linear Spaces

In this subsection we define some basic definitions of linear algebra which help us to understand this dissertation.

Vector Space

Let $\Phi \neq V$ having two binary operations '+' and scalar multiplication (scalar come from field F) is called vector space over F if the following conditions holds.

i) V is abelian group under '+'.

ii) *for all* $\alpha \in F$ and $v_1 \in V \Longrightarrow \alpha v_1 \in V$.

iii) $\alpha(v_1 + v_2) = \alpha v_1 + \alpha v_2$ *for all* $v_1, v_2 \in V$ and $\alpha \in F$.

iv) $(\alpha + \beta) v_1 = \alpha v_1 + \beta v_1$ *for all* $v_1 \in V$ and $\alpha, \beta \in F$.

v) $\alpha(\beta v_1) = (\alpha\beta) v_1$ *for all* $v_1 \in V$ and $\alpha, \beta \in F$.

vi) There exist $1 \in F$ such that $1 \cdot v_1 = v_1 = v_1 \cdot 1$ *for all* $v_1 \in V$.

Example 71 *:$V = \{(r_1, r_2) \mid r_1, r_2 \in R\}$ is vector space over R.*

Subspace: Let $\Phi \neq S \subseteq V$ then S is called subspace of V if S is vector space itself under the same binary operation of vector space V.

Example 72 *:If vector space $V = R^2$ over R then $S = \{(s, 0) \ : \ s \in R\}$ is subspace of V.*

Remark 73 *: To prove Subspace S we have to check only following conditions*

1) $0 \in S$.
2) $r_1 + r_2 \in S \quad for\ all\ r_1, r_2 \in S$
3) $\alpha r \in S \qquad for\ all\ r \in S$ and $\alpha \in F$.

Remark 74 *:If V_1 and V_2 are two subspaces of V then it is not necessary that $V_1 \cup V_2$ is subspace of V.*

Example 75 *:If $S_1 = \{(r_1, 0) : r_1 \in R\}$ and $S_2 = \{(0, r_2) : r_2 \in R\}$ are two subspaces of R^2 then $S_1 \cup S_2$ is not subspace of R^2.*

Linearly Independent and Linearly Dependent

Suppose that $\sum_{i=1}^{n} \alpha_i v_i = 0$ and not all a_i's are zero then $M = \{v_1, v_2, v_3, ...v_n\}$ is called linearly dependent (where $\alpha_i \in F, v_i \in V$).

If all a_i's are equal to zero then M is called linearly independent.

Example 76 *:Consider vector space $\mathbb{C}$ over C then $\{1 + i, i\}$ is linearly dependent over C.*

Example 77 *:$\{(1, 0), (0, 1)\}$ is linearly independent over R.*

Basis: A subset $M = \{v_1, v_2, v_3, ..., v_n\}$ of a vector space is known as basis of V if $V = \langle M \rangle$ and M is linearly independent set.

Example 78 *:$\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ is a basis of R^3 over R.*

Algebra: A vector space B over the field F is algebra over F if it satisfy the following conditions.

(1) B is a ring.
(2) $\alpha (b_1 b_2) = b_1 (\alpha b_2)$ for all $b_1, b_2 \in B$ and $\alpha \in F$.

1.2 Some Basic Concepts of Coding Theory

Coding theory deals with the problem of errors that occur when a message is transmitted through a channel which is communication channel. The channel might be a television link, telephone line ,recording device or radio .The error may be caused by thermal noise, atmospheric disturbance,faulty equipment,or human negligence. A channel prone to transmission errors is called a noisy channel. An error correcting code is a scheme of encoding the message in such a way that the correct message may be recovered even when errors have taken place during transmission. The general principle of an error correcting code is to add redundancy to the message so that the errors can be calculated and corrected in most cases. The basic definitions of this section is taken from [10].

1.2.1 Codes

Let V be a finite set of q elements and V^n denote the set of all n-tuples of elements of V where n is positive integer greater then 1.So there are q^n elements in V^n which are called words or vectors. Let $\Phi \neq C$ subset of V^n then C is called a q-ary code of length n over V.

In particular if $q = 2$ then the code C is known as binary code , for $q = 3$ the code C is known as ternary code. The elements of C are known as codewords. If C have only one element or $C = V^n$ then C is called trivial code. If every element of C is a vector of the form $vvv...v$ for some $v \in V$ then C is called repetition code. Hence q-ary repetition code contains exactly q codewords.

Example 79 *:*$C = \{0000, 1111\}$ *is binary repetition code of length* 4*. So it contains exactly* 2 *codewords.*

Hamming Distance: Let $\mathbf{w}, \mathbf{v} \in V^n$,$\mathbf{w} = w_1w_2w_3...w_n$,$\mathbf{v} = v_1v_2v_3...v_n$.The hamming distance between the vectors $\mathbf{w}$ and $\mathbf{v}$ denoted by $d(w, v)$,is defined as

$$d(w, v) = |\{i : w_i \neq v_i\}|.$$

Example 80 *: If* $w = 1100$ *and* $v = 1010$ *then hamming distance between* w *and* v *is* 2*. we write it as* $d(1100, 1010) = 2$.

Remark 81 *:The hamming distance satisfies all the conditions of metric space so it is metric on* V^n *and* (V^n, d) *is called metric space.*

Minimum Distance: The least distance between any two different codewords in C are called minimum distance. Mathematically, $d(C) = \min\{d(\overline{w}, \overline{v}) : for \text{ all } \overline{w}, \overline{v} \in C, \overline{w} \neq \overline{v}\}$.

Example 82 *:The minimum distance of* $C = \{000, 101, 100, 111\}$ *is* 1.

Remark 83 *The error-correction and error-detection capabilities of a code can be calculated by the* $d(C)$.

Theorem 84 *:Suppose* C *is a code having minimum distance* d. *let* $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ *then*

(i) There are $d-1$ errors detected in any transmitted codeword in C.

(ii) There are t errors corrected in any transmitted codeword in C.

Example 85 *:Let* $C = \{000, 111\}$ *be a code with* $d = 3$ *then* C *can detect* 2 *errors and corrected only single error.*

Remark 86 *:Every code* C *can be represented as* (n, M, d) *where* n *is the length of code* C *and* M *tells about the number of codewords in* C *and* d *represents minimum distance of* C.

Good Code

The code C is known as good code *if it satisfies,*

1) Length n of the code is smaller.

2) Size of M is very large.

3) $d(C)$ of the code is large.

Remark 87 *Length of the code smaller means transmission of code is very fast and cost of the code is very lower. Size of* M *is very large gives that we can sent more variety of messages and if minimum distance of the code is very large then it can be corrected greater number of errors. The main task of the algebraic theorists is to find such codes whose size* M *and minimum distance is maximum for fixed length* n.

Perfect Code: Suppose $C \subset V^n$ is a code having minimum distance $2t+1$. If for all $w \in V^n$ then there exist $v \in V^n$ such that $d(w,v) \leq t$,then C is known as perfect code.

Example 88 *The binary code* $\{000, 111\}$ *is perfect code with minimum distance* 3.

Remark 89 *If* $C = V^n$ *then it is trivial perfect code with minimum distance* 1.

Theorem 90 *Suppose* C *is a* q*-ary* (n, M, d) *code having minimum distance* $2t+1$*. Then* C *is a perfect code iff,*

$$M \sum_{j=0}^{t} \binom{n}{j} (q-1)^j = q^n.$$

In particular a binary code is perfect iff $M \sum_{j=0}^{t} \binom{n}{j} = 2^n$.

Example 91 *A binary* $(23, 4096, 7)$ *is perfect code.*

1.2.2 Linear Codes

In coding theory linear codes are very special kind of codes have great importance. They have huge interest for advance objectives in coding theory. Linear codes have many applications in combined coding and modulation. These codes have many algebraic properties. In these codes we take scalars from the finite field $\mathcal{F}$.The set F^n is an n-dimensional vector space over the field $\mathcal{F}$ here n is a positive integer and every element $u \in F^n$ can be written as $u_1u_2u_3...u_n$.

Linear Code: Suppose $\mathcal{F}$ is a finite field and n is a positive integer. Any subspace of F^n is called linear code. If C is a subspace of dimension s,then it is known as $[n, s]$ code. But if the code C with minimum distance then it is denoted as $[n, s, d]$ code.

Example 92 $C = \{00000, 11111\}$ *is a linear* $[5, 1, 5]$ *code over the field* F_2*. But the binary code* $C = \{00, 10, 11\}$ *is not linear because* $10 + 11 = 01 \notin C$.

Remark 93 *There are* q^s *codewords in* q*-ary* $[n, s]$ *code and binary* $[n, s]$ *code consists of* 2^s *codewords. The notion* (n, M, d) *is used for a code of* M *codewords and the notion* $[n, s, d]$ *is used for* s *dimension linear code.*

Generator Matrix

Suppose C is a linear $[n, s]$ code and $\mathcal{G}$ is a $s \times n$ matrix where rows of $\mathcal{G}$ are basis of C. Then $\mathcal{G}$ is known as generator matrix of C. Generator matrix gives us complete linear code because every member of C can be written as linear combination of the rows of $\mathcal{G}$.So row space of $\mathcal{G}$ is code C,that is $C = \{w\mathcal{G} : w \in F^s\}$.

The vectors in the space F^s are of length s and vectors of an $[n, s]$ *code* C are of length n. The size of F^s and C is same, So there is a bijection $\rho : F^s \to C$, defined as, $\rho(w) = w.\mathcal{G}$ for all $w \in F^s$ here elements of domain of ρ are known as messages and elements of range of ρ are called codewords, that is, $w.\mathcal{G}$ is the codeword. The bijection ρ maps a message of length s on to a codeword of length n. The number $n - s$ is redundancy of the code C and its code rate s/n. The mapping ρ is called encoding mapping.

Dual Code: Suppose C is $[n, s]$ code over the field $\mathcal{F}$. Then we can define dual code of C is,

$$C^{\perp} = \{w \in F^n \ : w.v = 0 \text{ for all } v \in C\}.$$

Any two vectors $w, v \in F^s$are orthogonal if and only if $w.v = 0$. If every element of C is orthogonal to itself and every vector of C then a linear code C is called self orthogonal also if $C \subset C^{\perp}$ then it self orthogonal.

Theorem 94 *Suppose C is $[n, s]$ code over $\mathcal{F}$.Then C dual is $[n, n - s]$ code and $\left(C^{\perp}\right)^{\perp} = C$.*

Example 95 *The $C^{\perp}$ of $[4, 1]$ binary code $C = \{0000, 1111\}$ is*

$$C^{\perp} = \{0000, 1100, 1010, 1001, 0110, 0101, 0011, 1111\}$$

and it is $[4, 3]$ code, by finding the those elements of $(F_2)^4$ whose product with the elements of C is zero. It is also self orthogonal because C contained in $C^{\perp}$.

Parity Check Matrix

Suppose C is $[n, s]$ code and $\mathcal{H}$ is a $(n - s) \times n$ matrix which is generator matrix of the $C^{\perp}$.Then $\mathcal{H}$ is known as parity check matrix of the code C.

Theorem 96 *Suppose C is a $[n, s]$ code over $\mathcal{F}$ and $\mathcal{H}$ is a parity check matrix of C,then $C = \{w \in F^n : w\mathcal{H}^T = 0 = \mathcal{H}w^T\}$.*

Theorem 97 *Suppose C is a $[n, s]$ code and $\mathcal{G}$ is a generator matrix and $\mathcal{H}$ is a parity check matrix of the code C then ,*

$$\mathcal{G}\mathcal{H}^T = 0 = \mathcal{H}\mathcal{G}^T,$$

Conversely suppose that $\mathcal{G}$ is a $s \times n$ matrix and $\mathcal{H}$ is $(n-s) \times n$ matrix such that $\mathcal{G}\mathcal{H}^T = 0$. Then $\mathcal{H}$ is a parity check matrix of the code C iff $\mathcal{G}$ is generator matrix of C, where rank of $\mathcal{G}$ is s and rank of $\mathcal{H}$ is $n - s$.

Example 98 *If $C = \{000, 111\}$ then $C^{\perp} = \{000, 110, 011, 101\}$ So unique generator matrix $\mathcal{G} = [1\ 1\ 1]$ and if we take any two non zero vectors from $C^{\perp}$ then they form a parity check matrix that is,*

$$\mathcal{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

The following theorem gives us how to find $\mathcal{G}$ from $\mathcal{H}$ and $\mathcal{H}$ from generator matrix $\mathcal{G}$. This is very useful theorem in algebraic coding theory.

Theorem 99 *Suppose C is $[n, s]$ code. If C has generator matrix $\mathcal{G} = [I_s : B]$ here B is $s \times (n-s)$ matrix, then $\mathcal{H} = \left[-B^T : I_{n-s}\right]$.*

If parity check matrix $\mathcal{H} = [\mathcal{A} : I_{n-s}]$ then generator matrix $\mathcal{G} = \left[I_s : -\mathcal{A}^T\right]$.

Equivalent Code

Let C and $C^{\star}$ be $[n, s]$ codes over the $\mathcal{F}$. If C with generator matrix $\mathcal{G}$ and $C^{\star}$ with generator matrix $\mathcal{G}^{\star}$ then the codes C and $C^{\star}$ are equivalent iff one matrix determined from other matrix by applying the following operations,

(1) By Applying the Elementary operations on the rows of matrix.

(2) By Applying Permutation to the columns of matrix.

(3) By Multiply to any column by a non zero element from the field $\mathcal{F}$.

Remark 100 *These properties of equivalent codes are also holds* $\mathcal{H}$.

Weight of Codeword: The weight of any codeword $u \in C$ is defined as the number of nonzero components in u and it is denoted as $W(u)$.

The $d(C)$ of the code is also defined with the help of weight that is $d(C) = \min\{W(u) : u \in C, u \neq 0\}$.

Example 101 *:Let* $C = \{000, 111\}$ *then* $w(111) = 3$ *and* $d(C) = 3$.

Theorem 102 *Suppose* $\mathcal{H}$ *is a parity check matrix of an* $[n, s]$ *code* C *over the field* $\mathcal{F}$, *The minimum distance of the code* C *and minimal number of linearly dependent columns of* $\mathcal{H}$ *are equal,Consequently,* $d(C) \leq n - s + 1$.

1.2.3 Hamming Codes

In this subsection we define very special kind of linear codes which are hamming codes. We first discuss the particular case of hamming codes and then general q-ary hamming codes.

Binary Hamming Codes

Let $1 \neq m \in Z^+$ and $\mathcal{H}$ be an $m \times (2^m - 1)$ matrix having non zero distinct vectors as its columns in F_2^m. Then the code with $\mathcal{H}$ as its parity check matrix is said to be binary hamming code and it is represented by $\text{Ham}(m, 2)$ .Hence for every given m there are $(2^m - 1)!$ equivalent binary hamming codes.

Since $\mathcal{H}$ is $m \times (2^m - 1)$ matrix, $\text{Ham}(m, 2)$ is a code of length $n = 2^m - 1$ and dimension $s = n - m = 2^m - 1 - m$.Hence $\text{Ham}(m, 2)$ is a $[2^m - 1, 2^m - 1 - m]$ code and $m = n - s$ shows the redundancy of the code.

Example 103 *Suppose* $m = 2$ *then Ham*$(2, 2)$ *is a* $[3, 1]$ *code because here* $n = 2^2 - 1 = 3$ *and* $s = 3 - 2 = 1$. *The parity check matrix of this code* $\mathcal{H} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ *and the generator matrix* $\mathcal{G} = [1 \quad 1 \quad 1]$ *so hence Ham*$(2, 2)$ *is simple binary repetition code that is* $\{000, 111\}$.

q-ary Hamming Codes

Let $\mathcal{F} = F_q$ and $1 \neq m \in Z^+$. Let $n = (q^m - 1)/(q - 1)$ and $\mathcal{H}$ be an $m \times n$ matrix having non zero columns in F^m such that there is not any column which is scalar multiple of other column. Then $[n, n-m]$ code with $\mathcal{H}$ which is its parity check matrix is known as q-ary hamming code and is represented by $Ham(m, q)$.

Theorem 104 *$Ham(m, q)$ is a perfect code with minimum distance* 3.

1.2.4 Cyclic Codes

In this subsection the special class of linear codes will be discussed. In Ring theory and in advanced algebraic structure properties of cyclic codes are very interesting then general linear codes. There are very large class of important codes which are related to cyclic codes.

Cyclic Shift: The mapping $\sigma : F^n \to F^n$ is defined as $\sigma(u_1, u_2, u_3, ..., u_n) = (u_n, u_1, u_2, ..., u_{n-1})$ is called a cyclic shift and this is also linear mapping.

Cyclic Code: A linear code $C \subset F^n$ is said to be cyclic code if $\sigma(u) \in C$ for all $u \in C$.

Example 105 *The code $C = \{000, 100, 010, 001\}$ is binary cyclic code.*

Suppose $\mathcal{F}[X]_n = \{\alpha_0 + \alpha_1 x + \alpha_2 x^2 + ... + \alpha_{n-1} x^{n-1} : \alpha_i \in \mathcal{F}\}$ is the set consisting of all polynomials having degree smaller then n over the field $\mathcal{F}$. The mapping $\rho : \mathcal{F}^n \to \mathcal{F}[X]_n$ is defined as $\rho(u) = u(x)$ for all $u = (u_0, u_1, u_2, ..., u_{n-1}) \in F^n$. This mapping is an isomorphism. Now suppose that $\mathcal{F}[X]$ is the polynomial ring over $\mathcal{F}$. Let $h(x) \in \mathcal{F}[X]$ is an irreducible element over the field $\mathcal{F}$ then the quotient ring,

$$\frac{\mathcal{F}[X]}{< h(x) >} = \{\alpha_0 + \alpha_1 t + \alpha_2 t^2 + ... + \alpha_{n-1} t^{n-1} : \alpha_i \in \mathcal{F}\},$$

is a field, here $t = x + < h(x) >$ so $h(t) = 0$ and $h(x)$ be the degree of n.

Let $h(x) = x^n - 1$ then the quotient ring is ,

$$\frac{\mathcal{F}[X]}{< x^n - 1 >} = \{\alpha_0 + \alpha_1 t + \alpha_2 t^2 + ... + \alpha_{n-1} t^{n-1} : \alpha_0, \alpha_1, \alpha_2, ..., \alpha_{n-1} \in \mathcal{F}\},$$

not field because $h(x) = x^n - 1$ is not irreducible . So if we replace t by x in this ring then it becomes $\mathcal{F}[X]_n$. Hence we prove that $\mathcal{F}[X]_n$ is a ring with $x^n - 1 = 0$ and $\mathcal{F}[X]_n$ is an algebra over the field $\mathcal{F}$..

Theorem 106 *Let* $0 \neq C$ *ideal of* $\mathcal{F}[X]_n$. *Then ,*

1) There exist monic polynomial which is unique $g(x)$ of smaller degree of C.
2) $g(x) | x^n - 1$ in ring $\mathcal{F}[X]$.
3) $g(x) | w(x)$ in ring $\mathcal{F}[X]$ for all $w(x) \in C$.
4) $C =< g(x) >$.

Conversely suppose that C is an ideal and $C =< h(x) >$ where $h(x) \in \mathcal{F}[X]_n$. Then $h(x)$ is polynomial of least degree in C iff $h(x) \mid x^n - 1$ in ring $\mathcal{F}[X]$.

Example 107 *If we have to find the cyclic codes of length 3 and all non trivial ideals of* $\mathcal{F}[Y]_3$,*here* $F = \mathcal{F}_2$ *then the polynomial* $y^3 - 1 = (y-1)\left(y^2 + y + 1\right)$ *here both non trivial factors of* $y^3 - 1$ *are irreducible over the field* $\mathcal{F}_2$ *so we can generate ideals from these factors that is,*

$$< y - 1 >= \{0, 1 + y, 1 + y^2, y^2 + y\},$$
$$< y^2 + y + 1 >= \{0, y^2 + y + 1\},$$

by writing these polynomials as vectors we get the cyclic code of length 3 that is $\{000, 011, 101, 110\}$ and $\{000, 111\}$.

1.2.5 BCH Codes

In this subsection we discuss very important kind of cyclic codes named as BCH codes. BCH stands for Bose Chaudhuri and Hocquenghem.Firstly we discuss some properties of irreducible polynomials and finite fields. Every finite field has order power of some prime p. GF(q) or $\mathcal{F}_q$ denotes the unique field of order $q = p^m$ for some prime p and $m \in Z^+$. If $m = 1$ then $\mathcal{F}_p = Z_p$. The set $\mathcal{F}_q^*$ denotes the set of all non-zero elements of the field $\mathcal{F}_q$ and it becomes a cyclic group of order $p^m - 1$ under multiplication. Hence $b^{p^m - 1} = 1$ or all $b \in \mathcal{F}_q^*$,and $b^{p^m} = b$. If $p^m - 1$ is divisible by any number n, then there is an element $b \in \mathcal{F}_q^*$ whose order is n that

is $o(b) = n$, and then b is called primitive nth root of unity in $\mathcal{F}_q$. Furthermore, $b \in \mathcal{F}_q^*$ is said to be primitive element if $o(b) = p^m - 1$. The characteristic of the field $\mathcal{F}_q$ is p. Hence $pb = 0$ and also $qb = 0$ for all $b \in \mathcal{F}_q$.For the finite field $\mathcal{F}_q$ and $r \in \mathbb{Z}^+$, there exists an irreducible polynomial $h(x) \in \mathcal{F}_q[x]$ of degree r, the quotient ring $\mathcal{F}_q[x]/(h(x))$ is a field of size q^r, denoted by $\mathcal{F}_{q^r}$ or $GF(q,r)$. The field $\mathcal{F}_{q^r}$ or $GF(q,r)$ is called an extension field of $\mathcal{F}_q$ of degree r so hence $b^{p^m} = b$ for all b in $\mathcal{F}_q$ and $q\alpha = 0$ for all $\alpha \in \mathcal{F}_{q^r}$.

Theorem 108 *Let $b_1, b_2, b_3..., b_n \in \mathcal{F}_q$ and $\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n \in \mathcal{F}_{q^r}$ then $(b_1\alpha_1 + b_2\alpha_2 + b_3\alpha_3 + ... + b_n\alpha_n)^q = b_1\alpha_1^q + b_2\alpha_2^q + ... + b_n\alpha_n^q$.*

Let $\alpha \in \mathcal{F}_{q^r}$ or $GF(q,r)$ then there exist unique monic polynomial of least degree $g(x) \in \mathcal{F}_q[x]$ such that $g(\alpha) = 0$.The $g(x)$ is an irreducible and minimal polynomial of α over $\mathcal{F}_q$. If any polynomial $f(x)$ in $\mathcal{F}_q[x]$ such that $f(\alpha) = 0$ then $g(x) \mid f(x)$ and also degree of $g(x)$ divides r. If α is primitive element in $\mathcal{F}_{q^r}$ then degree of $g(x)$ is equal to r.

Theorem 109 *Let α be any element in $\mathcal{F}_{q^m}$. Then $\alpha, \alpha^q, \alpha^{q^2}, \alpha^{q^3}, ...$ have the same minimal polynomial over the field $\mathcal{F}_q$.*

BCH Code

Let c, d, n, $q \in Z^+$ such that $2 \leq d \leq n$ and q is exponent of some prime p and n, q are relatively prime. Suppose r is the least positive integer such that n divides $q^r - 1$ and ξ be the nth root of unity in $\mathcal{F}_{q^r}$. Suppose that $m_j(x) \in \mathcal{F}_q[x]$ are the minimal polynomials corresponding to the ξ^j where $j = c, c+1, ..., c+d-2$. Suppose $\mathcal{C}$ is the cyclic code with generator polynomial which is $\mathbf{g}(x) = \text{lcm}\{m_j(x) : j = c, c+1, ..., c+d-2\}$ in $\mathcal{F}_q[x]_n$. Then C is known as BCH code of length n over $\mathcal{F}_q$ with designed distance d.

Application of BCH Code: BCH codes are used in satellite communication, computer networks, mobile communications and storage systems for example in computer memories like hard disc or the compact disc.

Remark 110 *If $n = q^r - 1$ then the BCH code is known as primitive BCH code and if $c = 1$ then it is called narrow sense BCH code.*

Reed-Solomon Code: A code which is narrow sense primitive BCH is called Reed Solomon code.

Theorem 111 *Suppose that C is a BCH code having length n over $\mathcal{F}_q$ with the designed distance d then,*

$C = \{u(x) \in \mathcal{F}_q[x]_n : u(\xi^j) = 0 \text{ for all } j = c, c+1, c+2, ..., c+d-2\}$.

Equivalently C is the null space of H which is defined as,

$$H = \begin{bmatrix} 1 & \xi^c & \xi^{2c} & . & . & . & \xi^{(n-1)c} \\ 1 & \xi^{c+1} & \xi^{2(c+1)} & . & . & . & \xi^{(n-1)(c+1)} \\ . & . & . & . & & & . \\ . & . & . & & . & & . \\ . & . & . & & & . & . \\ 1 & \xi^{c+d-2} & \xi^{2(c+d-2)} & . & . & . & \xi^{(n-1)(c+d-2)} \end{bmatrix}.$$

Where H is $(d-1) \times n$ quasi parity check matrix over $\mathcal{F}_{q^m}$.

Theorem 112 *Suppose C is a BCH code having designed distance d then the minimum distance is greater then or equal to designed distance that is $d(C) \geq d$.*

Remark 113 *Binary hamming code is also called BCH code.*

Chapter 2

Construction and Decoding of BCH-Codes over Z_p

In this chapter we will explain the construction and decoding procedure of BCH code over the field Z_p by using fast decoding algorithm which is Barlekamp Messey Algorithm. We also explain the historical background of Barlekamp Messy and BCH codes,Reed Solomon Codes.

2.1 Historical Background

There are two men Shannon and Hamming work together at Bell labs and contribute alot in developing adnvancement of algebraic coding theory. Later, in 1959 Hocquenghem and in 1960 Bose and Ray-Chaudhuri independently developed the large class of codes named as BCH codes. After that Reed and Solomon comes with a set of particular types of BCH codes, these codes are better for detecting and correcting burst errors. A decade later Berlekamp come up with a decoding algorithm which was modified in 1969 by James Massey . Later the compact disc was the first storage device which is used for correcting errors in 1982.

2.1.1 History of BCH Codes

There was a French mathematician Alexis Hocquenghem (1908-1990), in an article in 1959 "Codes correcteurs d'erreurs" mentioned the codes that he developed as a "generalization of Hamming's work" . In 1960 Dwijendra Ray-Chaudhuri and his Ph.D. advisor Raj Bose pub-

lished "On a class of error correcting binary group codes" . After Bose, Ray-Chaudhuri and Hocquenghem this class of linear codes are known as BCH codes.

2.1.2 History of Reed Solomon Codes

Irving Reed and Gustave Solomon are American mathematicians. They developed a class of algebraic codes which are called Reed-Solomon codes or RS codes . RS codes are special kind of the larger class of BCH codes but a decade later, by regarding them as cyclic BCH codes, that an effective decoding algorithm which gives the great potential to their wide applications.

2.1.3 History of Barlekamp Messey algorithm

Elwyn Berlekamp was a professor of mathematics,computer science and electrical engineering at the University of California, Berkeley. During his studying at MIT in electrical engineering department Claude Shannon was his Ph.D. advisor. An algorithm for decoding of BCH codes was invented by Barlekamp in 1968. There was James Massey which is cryptographer and information theorist who modified this algorithm in 1968 which is named as the Berlekamp-Massey algorithm.

Due to this algorithm it is possible to develop a efficient and fast decoder with a linear feedback shift register (LFSR), however it is not possible until 1982 with the invention of compact disk that the era of digital information as we know it was started. Immink describes that "without error-correcting codes, digital audio would not be technical feasible". Now a days RS codes are used in large scale having many applications which involves transmission of data, like computer networks, wireless communication, GPRS, GSM, telephony, digital video broadcasting and data storage for example hard disk in computers and memory cards in cameras and mobile telephones, and optical storage devices that are CD, Digital Versatile Discs also use Reed-Solomon codes.

2.2 Encoding and Decoding of BCH Codes Over $\mathbf{Z}_p$

In this section we discuss the encoding and decoding schemes over the field. We encode the message by finding the generator polynomial and send it to receiver through noisy channel.

After encoding message we decode that message through Barlekamp Messey Algorithm and correct errors.

2.2.1 Encoding of BCH Codes

If we have to sent a message through a channel which is noisy then we divide the message into parts of k digits and if we have to encode this message then attach $n-k$ check digits or redundant digits to each block to obtain n length codeword such type of code is called (n,k) code. Now the codeword can be send through the noisy channel then there are two possibilities in transmitted codeword either received word is codeword or not, if the received word is not codeword then during transmission there must be an error occurred and then receiver can request to the transmitter that the word would be retransmitted but if the received word is codeword then there is no error occur. It is noted that error correction capabilities of word is less then or equal to the error detection.

How to encode a message: Any message w of k bits can be encoded by following steps,

1) Write a message w into the form of polynomial $w(x)$.
2) Find the generating polynomial $g(x)$.
3) Encoded message $c(x) = w(x)\, g(x)$, here $c(x)$ is n length code polynomial.

Example 114 *Suppose that we encode the message* $w = 11010$ *through the encoder* $[15, 5]$ *and designed distance* $d = 7$ *here* $n = 15$ *and* $k = 5$.

Step1: Message w into the form of polynomial is $w(x) = 1 + x + x^3$.

Step2:-

To find generator polynomial $g(x)$ we construct first Galois field $GF(16)$ which is generated by primitive polynomial $h(x) = x^4 + x + 1$. Assume that ξ is the primitive root of $h(x)$ then $\xi^4 + \xi + 1 = 0$ and $\xi^4 = \xi + 1$

Exponent form	Binary form	elements in the form of polynomials
0	0000	0
ξ	0100	ξ
ξ^2	0010	ξ^2
ξ^3	0001	ξ^3
ξ^4	1100	$1+\xi$
ξ^5	0110	$\xi+\xi^2$
ξ^6	0011	$\xi^2+\xi^3$
ξ^7	1101	$1+\xi+\xi^3$
ξ^8	1010	$1+\xi^2$
ξ^9	0101	$\xi+\xi^3$
ξ^{10}	1110	$1+\xi+\xi^2$
ξ^{11}	0111	$\xi+\xi^2+\xi^3$
ξ^{12}	1111	$1+\xi+\xi^2+\xi^3$
ξ^{13}	1011	$1+\xi^2+\xi^3$
ξ^{14}	1001	$1+\xi^3$
ξ^{15}	1	1

Since ξ is primitive root of unity in $\mathcal{F}_{2^4}$ so $h(x)$ is minimal polynomial corresponding to ξ. Now we find the minimal polynomials of ξ^i corresponding to the value of $i = 1, 2, 3...6$.

Since ξ, ξ^2, ξ^4 have the same minimal polynomial so we can write $m_1(x) = x^4 + x + 1$.

Now we find minimal polynomial for ξ^3, since we know that $\xi^3, (\xi^3)^2, (\xi^3)^{2^2}, \left(\xi^3\right)^{2^3}, ...$ have same minimal polynomial so by using $\xi^{15} = 1$ we get $\xi^3, \xi^6, \xi^9, \xi^{12}$ are roots of $m_2(x)$ so it is expressed as,

$m_2(x) = \left(x - \xi^3\right)\left(x - \xi^6\right)\left(x - \xi^9\right)\left(x - \xi^{12}\right)$

After simplification and using the elements of Galois field we get the minimal polynomial,

$m_2(x) = x^4 + x^3 + x^2 + x + 1$.

Similarly we find minimal polynomial for ξ^5,we know $\xi^5, \left(\xi^5\right)^2, \left(\xi^5\right)^{2^2}, ..,$ have same minimal

polynomial again by using the $\xi^{15}=1$ we get the ξ^5, ξ^{10} are the roots of $m_3(x)$ so it is expressed as,

$m_3(x) = \left(x-\xi^5\right)\left(x-\xi^{10}\right) = x^2+x+1.$

Hence the $g(x)$ of the BCH code is,

$$\begin{aligned} g(x) &= \text{lcm}\{m_i(x) : i = 1,2,3,..,6\}, \\ g(x) &= m_2(x)\, m_3(x)\, m_1(x), \\ g(x) &= \left(1+x+x^2+x^3+x^4\right)\left(1+x+x^2\right)\left(x^4+x+1\right), \\ g(x) &= 1+x+x^2+x^4+x^5+x^8+x^{10} \end{aligned}$$

Step3 : Encoded message is $c(x) = w(x)\, g(x)$,

$c(x) = \left(1+x+x^3\right)\left(1+x+x^2+x^4+x^5+x^8+x^{10}\right),$

$c(x) = 1+x^5+x^6+x^7+x^9+x^{10}+x^{13},$

$c = 100001110110010$ is our desired encoded message of 15 length and this is codeword.

Now if we transmit this message through noisy channel then error may be occur during transmission so we have to find out that errors and then correct these errors this process is called decoding.

2.2.2 Decoding of BCH Codes Over $\mathbf{Z}_p$

Suppose that codeword $\widetilde{c}(x) = \widetilde{c}_0 + \widetilde{c}_1 x + \widetilde{c}_2 x^2 + ... + \widetilde{c}_{n-1}x^{n-1}$ is sent through noisy channel then let error $e(x)$ occurs during transmission so the received code polynomial is $\widetilde{r}(x) = \widetilde{c}(x) + e(x) = \widetilde{r}_0 + \widetilde{r}_1 x + \widetilde{r}_2 x^2 + ... + \widetilde{r}_{n-1}x^{n-1}$.

To decode this received polynomial first we find the syndromes by this formula $\widetilde{S}_i = \widetilde{r}\left(\xi^i\right)$ here $i = 1,2,3,...,2t$.

Since $\widetilde{c}(x)$ is codeword so $\widetilde{c}(\xi^i) = 0$ therefore $\widetilde{S}_i = \widetilde{r}\left(\xi^i\right) = e\left(\xi^i\right)$.

Suppose that $e(x) = x^{l_1} + x^{l_2} + ... + x^{l_\nu}$ here $0 \le l_1 < l_2 < l_3 < ...l_\nu < n$,

$\widetilde{S}_1 = \widetilde{r}\left(\xi^1\right) = e\left(\xi^1\right) = \xi^{l_1} + \xi^{l_2} + ... + \xi^{l_\nu}$ here ν denotes the errors introduced by the channel.

$\widetilde{S}_1 = \xi^{l_1} + \xi^{l_2} + ... + \xi^{l_\nu},$

$\widetilde{S}_2 = \left(\xi^{l_1}\right)^2 + \left(\xi^{l_2}\right)^2 + ... + \left(\xi^{l_\nu}\right)^2,$

.

.

.

$\widetilde{S}_{2t} = \left(\xi^{l_2}\right)^{2t} + \left(\xi^{l_2}\right)^{2t} + ... + \left(\xi^{l_\nu}\right)^2 .$

Here ξ^{l_1}, ξ^{l_2}, $\xi^{l_3}, ..., \xi^{l_\nu}$ are unknowns , any method for solving these equations and find the unknowns is called decoding algorithm for BCH codes.

If we find $\xi^{l_1}, \xi^{l_2}, \xi^{l_3}, ..., \xi^{l_\nu}$ then powers $l_1, l_2, ..., l_\nu$ denotes the error positions in received message.

Let for our simplification $\gamma_k = \xi^{j_k}$ be the error location numbers ,here $1 \leq k \leq \nu$.

$$\begin{aligned}
\widetilde{S}_1 &= (\gamma_1) + (\gamma_2) + ... + (\gamma_\upsilon) \\
\widetilde{S}_2 &= (\gamma_1)^2 + (\gamma_2)^2 + ... + (\gamma_\nu)^2 \\
\widetilde{S}_3 &= (\gamma_1)^3 + (\gamma_2)^3 + ... + (\gamma_\nu)^3 \\
&\vdots \\
\widetilde{S}_{2t} &= (\gamma_1)^{2t} + (\gamma_2)^{2t} + ... + (\gamma_\nu)^{2t} .
\end{aligned}$$

Now we define error locator polynomial as,

$$\begin{aligned}
\sigma(x) &= (1+\gamma_1 x)(1+\gamma_2 x)(1+\gamma_3 x)...(1+\gamma_\nu x), \\
\sigma(x) &= \sigma_0 + \sigma_1 x + \sigma_2 x^2 + ... + \sigma_\nu x^\nu .
\end{aligned}$$

Here the coefficients of x are elementary symmetric functions we defined as,

$\sigma_0 = 1$

$\sigma_1 = \gamma_1 + \gamma_2 + ... + \gamma_\nu$

$\sigma_2 = \gamma_1\gamma_2 + \gamma_2\gamma_3 + ... + \gamma_{\nu-1}\gamma_\nu$

.

.

.

$\sigma_\nu = \gamma_1\gamma_2\gamma_3\gamma_4...\gamma_\nu$.

Remark 115 *The inverses of roots of* $\sigma(x)$ *are error location numbers .*

We can relate elementary symmetric functions with the syndromes by using the newton identities as follows,

$$\begin{aligned} \widetilde{S}_1 + \sigma_1 &= 0 \\ \widetilde{S}_2 + \sigma_1\widetilde{S}_1 + 2\sigma_2 &= 0 \\ \widetilde{S}_3 + \sigma_1\widetilde{S}_2 + \sigma_2\widetilde{S}_1 + 3\sigma_3 &= 0 \end{aligned}$$

.

.

.

$$\widetilde{S}_\nu + \sigma_1\widetilde{S}_{\nu-1} + ... + \sigma_{\nu-1}\widetilde{S}_1 + \nu\sigma_\nu = 0$$

$$\widetilde{S}_{\nu+1} + \sigma_1\widetilde{S}_\nu + ... + \sigma_{\nu-1}\widetilde{S}_2 + \sigma_\nu\widetilde{S}_1 = 0$$

We decode any received message over Galois field is as follows,

Step1: Find syndromes from received polynomial.

Step2:Calculate Error locator polynomial $\sigma(x)$.

Step3:Find error location numbers by finding the inverses of roots of $\sigma(x)$.

Step4:Correct the errors.

Our main step is to calculate $\sigma(x)$ and this can be calculated by Berlekamp Massey algorithm.

2.2.3 Barlekamp Massey Algorithm

This is very fast decoding algorithm and it is used for binary and non binary BCH codes. We start this algorithm by following initial conditions ,

μ	$\sigma^{\mu}(x)$	d_{μ}	l_{μ}	$\mu - l_{\mu}$
-1	1	1	0	-1
0	1	S_1	0	0
1				
2				
.				
.				
.				
$2t$				

Here d_{μ} is nth discrepancy, l_{μ} is degree of $\sigma^{\mu}(x)$, S_1 is first non zero syndrome , t is error correction capabilities.

Important steps to complete this table,

Step1: If $d_{\mu} = 0$ then $\sigma^{\mu+1}(x) = \sigma^{\mu}(x)$ and $l_{\mu+1} = l_{\mu}$.

Step2: If $d_{\mu} \neq 0$ then find $m < \mu$ such that $d_m \neq 0$ and $m - l_m$ has largest value in last column of the table then

$$\sigma^{\mu+1}(x) = \sigma^{\mu}(x) + d_{\mu} d_m^{-1} x^{(\mu-m)} \sigma^{m}(x) \text{ and } l_{\mu+1} = \max(l_{\mu}, l_m + \mu - m)$$

Step3:To find discrepancy use this formula,

$$d_{\mu+1} = S_{\mu+2} + \sigma_1^{(\mu+1)} S_{\mu+1} + ... + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{\mu+2-l_{\mu+1}}.$$

The Polynomial $\sigma^{2t}(x)$ in last row of the table is required error locator polynomial .

After finding the $\sigma^{2t}(x)$ we calculate roots of this polynomial and take inverses of these roots then inverses of roots are known as error location numbers, powers of the error location numbers shows error positions in received code. To correct these errors subtract error vector from received vector then we get corrected codeword. If we have to find the message word then divide corrected codeword by generator polynomial $g(x)$.

Example 116 *Suppose that* $[15,5,7]$ *BCH code generated by* $g(x)=1+x+x^2+x^4+x^5+x^8+x^{10}$. *Suppose that codeword* $\widetilde{c}=100001110110010$ *is sent through channel and the error occur during transmission then received vector is* $\widetilde{r}=110101110110011$. *Find the error in this received vector and then determine corrected codeword also find the message word.*

Solution 117 *Here* $n=15, s=5, d=7$ *so* $t=\left\lfloor\frac{d-1}{2}\right\rfloor=\left\lfloor\frac{7-1}{2}\right\rfloor=3$ *and* $GF(16)=\frac{Z_2[X]}{<x^4+x+1>}$ *here* x^4+x+1 *is primitive polynomial and let* ξ *is primitive root of this polynomial so* $1+\xi+\xi^4=0$.

The received vector in the form of polynomial is ,

$\widetilde{r}(x)=1+x+x^3+x^5+x^6+x^7+x^9+x^{10}+x^{13}+x^{14}$

Step1: Calculate syndromes by using the formula $S_i=\widetilde{r}\left(\xi^i\right)$ here $i=1,2,3,4,5,6$.

$S_1=\widetilde{r}(\xi)=\left(1+\xi+\xi^3+\xi^5+\xi^6+\xi^7+\xi^9+\xi^{10}+\xi^{13}+\xi^{14}\right)\bmod\left(1+\xi+\xi^4\right)=\xi+1=\xi^4$ (from the table of Galois field)

$S_2=\widetilde{r}\left(\xi^2\right)=\left(1+\xi^2+\xi^6+\xi^{10}+\xi^{12}+\xi^{14}+\xi^{18}+\xi^{20}+\xi^{26}+\xi^{28}\right)\bmod\left(1+\xi+\xi^4\right)=\xi^2+1=\xi^8$.

$S_3=\widetilde{r}\left(\xi^3\right)=\left(1+\xi^3+\xi^9+\xi^{15}+\xi^{18}+\xi^{21}+\xi^{27}+\xi^{30}+\xi^{39}+\xi^{42}\right)\bmod\left(1+\xi+\xi^4\right)=\xi^3+\xi^2+1=\xi^{13}$

$S_4=\widetilde{r}\left(\xi^4\right)=\left(1+\xi^4+\xi^{12}+\xi^{20}+\xi^{24}+\xi^{28}+\xi^{36}+\xi^{40}+\xi^{52}+\xi^{56}\right)\bmod\left(1+\xi+\xi^4\right)=\xi$

$S_5=\widetilde{r}\left(\xi^5\right)=\left(1+\xi^5+\xi^{15}+\xi^{25}+\xi^{30}+\xi^{35}+\xi^{45}+\xi^{50}+\xi^{65}+\xi^{70}\right)\bmod\left(1+\xi+\xi^4\right)=0$

$S_6=r\left(\xi^6\right)=\left(1+\xi^6+\xi^{18}+\xi^{30}+\xi^{36}+\xi^{42}+\xi^{54}+\xi^{60}+\xi^{78}+\xi^{84}\right)\bmod\left(1+\xi+\xi^4\right)=\xi^3+\xi^2+\xi=\xi^{11}$ (from table)

$S=\left(\xi^4\ \ \xi^8\ \ \xi^{13}\ \ \xi\ \ 0\ \ \xi^{11}\right)$

Step2: Now We use Berlekamp Massey Algorithm to find error locator polynomial $\sigma(x)$.

This algorithm starts with following initial conditions,

$\mu=-1,\quad \sigma^{-1}(x)=1,\quad d_{-1}=1,\quad l_{-1}=0,\quad \mu-l_\mu=-1-0=-1,$

$\mu=0,\qquad \sigma^0(x)=1,\quad d_0=S_1=\xi^4,\quad l_0=0\quad \mu-l_\mu=0-0=0,$

In this Algorithm we use the following formulas,

(1) $\sigma^{(\mu+1)}(x)=\sigma^\mu(x)+d_\mu d_m^{-1}x^{\mu-m}\sigma^m(x)$.

(2) $d_{\mu+1}=S_{\mu+2}+\sigma_1^{(\mu+1)}S_{\mu+1}+\ldots+\sigma_{l_{\mu+1}}^{(\mu+1)}S_{\mu+2-l_{\mu+1}}$.

(3) $l_{\mu+1}=\max\left(l_\mu,l_m+\mu-m\right)$.

To find $\sigma^1(x)$ put $\mu = 0$ and $m = -1$ in formula(1).

$\sigma^1(x) = \sigma^0(x) + d_0 d_{-1}^{-1} x^1 \sigma^0(x) = 1 + \left(\xi^4\right)(1)^{-1} x(1) = 1 + \xi^4 x,$

also put n=0 and m=-1 in formula(2) then $l_1 = \max\{l_0, l_{-1} + 0 + 1\} = \max\{0, 1\} = 1$

Now find d_1 =?

Put $\mu = 0$ in formula(2) we get , $d_1 = S_2 + \sigma_1^{(1)} S_1 = \xi^8 + \xi^4 \cdot \xi^4 = 0$ here $\sigma_1^{(1)}$ is coefficient of x in $\sigma^1(x)$.

Since $d_1 = 0$ so $\sigma^2(x) = \sigma^1(x)$ and $l_2 = l_1$ this implies that $\sigma^2(x) = 1 + \xi^4 x$ and $l_2 = 1$.

Now we find d_2 =??

Put $\mu = 1$ in formula(2) then $d_2 = S_3 + \sigma_1^{(2)} S_2 + \sigma_2^{(2)} S_1 = (1 + \xi^2 + \xi^3) + \xi^4 \cdot \xi^8 + 0 \cdot \xi^4 = 1 + \xi^2 + \xi^3 + \xi^{12}$,here $\sigma_1^{(2)}$ is coefficient of x in $\sigma^{(2)}(x)$ and $\sigma_2^{(2)}$ is coefficient of x^2 in $\sigma^{(2)}(x)$, from the table of Galois field ξ^{12}=$1 + \xi + \xi^2 + \xi^3$ so we get

$d_2 = 1 + \xi^2 + \xi^3 + 1 + \xi + \xi^2 + \xi^3 = \xi \neq 0$

Since $d_2 \neq 0$ so we find $\sigma^3(x)$ by using the formula(1), Put $\mu = 2$ and $m = 0$ because $d_0 \neq 0$ and $0 - l_0 = 0$ is largest in last column of the table. Therefore, $\sigma^3(x) = \sigma^2(x) + d_2 d_0^{-1} x^{2-0} \sigma^0(x) = \left(1 + \xi^4 x\right) + (\xi)\left(\xi^4\right)^{-1} x^2(1) = 1 + \xi^4 x + (\xi)\left(\xi^{11}\right) = 1 + \xi^4 x + \xi^{12} x.$

Also put $\mu = 2$ in formula(3) we get $l_3 = \max\{l_2, l_0 + 2 - 0\} = \max\{1, 2\} = 2$ and $\mu - l_3 = 3 - l_3 = 3 - 2 = 1$.

Now I find d_3 =???

Put $\mu = 2$ in formula(2) then $d_3 = S_4 + \sigma_1^{(3)} S_3 + \sigma_2^{(3)} S_2 = \xi + \xi^4 . \xi^{13} + \xi^{12} . \xi^8 = 0$ (by using the table of Galois field).

Since $d_3 = 0$ so $\sigma^4(x) = \sigma^3(x)$ and $l_4 = l_3 = 2$ so $\mu - l_\mu = 4 - l_4 = 4 - 2 = 2$.

$\sigma^4(x) = 1 + \xi^4 x + \xi^{12} x$

To find d_4 =??

Put $\mu = 3$ in formula(2) then we get $d_4 = S_5 + \sigma_1^{(4)} S_4 + \sigma_2^{(4)} S_3 = 0 + \xi^4 . \xi + \xi^{12} \cdot \xi^{13} = \xi^5 + \xi^{10} = 1.$

Since $d_4 \neq 0$ so we calculate $\sigma^5(x)$ =??

Put $\mu = 4$, and $m = 2$ in formula(1) because $d_2 \neq 0$ and $m - l_m$ is largest value.

$\sigma^5(x) = \sigma^4(x) + d_4 d_2^{-1} x^{(4-2)} \sigma^2(x) = 1 + \xi^4 x + \xi^{12} x + (1)(\xi)^{-1} x^2 \left(1 + \xi^4 x\right)$ =$1 + \xi^4 x + \xi^{12} x + \xi^{14} x^2 \left(1 + \xi^4 x\right)$

$\sigma^5(x) = 1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$ and $l_5 = \max\{l_4, l_2 + 4 - 2\} = \max\{2, 1 + 4 - 2\} = 3.$

To find d_5 put $\mu = 4$ in formula(2) then we get $d_5 = S_6 + \sigma_1^5 S_5 + \sigma_2^5 S_4 + \sigma_3^5 S_3 = \xi^{11} + \xi^4 \cdot 0 + \xi^5 \cdot \xi + \xi^3 \cdot \xi^{13} = 0$.

Since $d_5 = 0$ so $\sigma^6(x) = \sigma^5(x)$ this implies that $\sigma^6(x) = 1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$.

Now putting all the values in the form of table as follows,

μ	$\sigma^{\mu}(x)$	d_{μ}	l_{μ}	$\mu - l_{\mu}$
-1	1	1	0	-1
0	1	ξ^4	0	0
1	$1 + \xi^4 x$	0	1	0
2	$1 + \xi^4 x$	ξ	1	1
3	$1 + \xi^4 x + \xi^{12} x^2$	0	2	1
4	$1 + \xi^4 x + \xi^{12} x^2$	1	2	2
5	$1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$	0	3	2
6	$1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$	$-$	$-$	$-$

Table1

Hence $\sigma^{2t}(x) = \sigma^6(x) = 1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$ is error locating polynomial .

Step3: Find the roots of error locating polynomial by substituting the non zero elements of Galois field , here Galois field is $GF(16)$ so by substituting $1, \xi, \xi^2, \xi^3, \xi^4, \xi^5, ..., \xi^{14}$ we get ξ, ξ^{12}, ξ^{14} are roots of error locator polynomial. Now to find error location number we take inverses of these roots so $\xi^{-1} = \xi^{14}, (\xi^{12})^{-1} = \xi^3, (\xi^{14})^{-1} = \xi$ this implies that ξ, ξ^3, ξ^{14} are error location numbers and powers $1, 3, 14$ shows that error positions in received vector. The error vector is $\mathbf{e}(x) = x + x^3 + x^{14}$.

Step4: The corrected code polynomial is $\widetilde{\mathbf{c}}(x) = \widetilde{\mathbf{r}}(x) - \mathbf{e}(x)$ so,

$\widetilde{\mathbf{c}}(x) = (1 + x + x^3 + x^5 + x^6 + x^7 + x^9 + x^{10} + x^{13} + x^{14}) + \left(x + x^3 + x^{14}\right) = x^{13} + x^{10} + x^9 + x^7 + x^6 + x^5 + 1$

$\widetilde{\mathbf{c}} = 100001110110010$ is corrected codeword.

To find message word we divide corrected code polynomial by generated polynomial ,

$w(x) = \frac{\widetilde{c}(x)}{g(x)} = \frac{x^{13}+x^{10}+x^9+x^7+x^6+x^5+1}{1+x+x^2+x^4+x^5+x^8+x^{10}} = 1 + x + x^3$ this implies that $w = 11010$ is message word of $s = 5$ length which was transmitted after encoding through noisy channel.

2.2.4 Decoding of BCH Codes over Binary Field

Algorithm for binary BCH code to find the error locator polynomial is very efficient because it needs only t steps to find the error locator polynomial. We start Barlekamp Massey algorithm by following conditions ,

μ	$\sigma^{\mu}(x)$	d_{μ}	l_{μ}	$2\mu - l_{\mu}$
$\frac{-1}{2}$	1	1	0	-1
0	1	S_1	0	0
1				
2				
.				
.				
.				
t				

Table2

1) Start with $\mu = -1/2$.

2) If $d_{\mu} = 0$ then $\sigma^{(\mu+1)}(x) = \sigma^{(\mu)}(x)$.

3) If $d_{\mu} \neq 0$ then find $m < \mu$ such that $2m - l_m$ is large as possible in last column of the table and $d_m \neq 0$ then use the formulas,

$$\begin{aligned}
1) \quad \sigma^{(\mu+1)}(x) &= \sigma^{(\mu)}(x) + d_{\mu} d_m^{-1} x^{2(\mu-m)} \sigma^{(m)}(x), \\
2) \quad d_{\mu+1} &= S_{2\mu+3} + \sigma_1^{(\mu+1)} S_{2\mu+2} + \sigma_2^{(\mu+1)} S_{2\mu+1} + ... + \sigma_{l_{\mu+1}}^{(\mu+1)} S_{2\mu+3-l_{\mu+1}}, \\
3) \quad l_{\mu+1} &= \deg\left(\sigma^{(\mu+1)}(x)\right).
\end{aligned}$$

The polynomial $\sigma^{t}(x)$ in last row of the table is our required error locator polynomial.

Example 118 *If we apply this algorithm to above example then we get error locator polynomial after* 3 *steps as follows,*

Step1 is same as above example.

Step 2: Apply Barlekamp Massey algorithm with initial conditions as defined in Table2, So put $\mu = 0$,$m = \frac{-1}{2}$ in formula1 we get,

$$\begin{aligned}
\sigma^{(1)}(x) &= \sigma^{0}(x) + d_0(d_{\frac{-1}{2}})^{-1}x^{2\left(\frac{1}{2}\right)}\sigma^{\frac{-1}{2}}(x), \\
\sigma^{1}(x) &= 1 + d_0(d_{\frac{-1}{2}})^{-1}x, \\
\sigma^{1}(x) &= 1 + \xi^4(1)^{-1}x \\
\sigma^{1}(x) &= 1 + \xi^4 x
\end{aligned}$$

Now put $\mu = 0$ in formula3 we get $l_1 = \deg\left(\sigma^1(x)\right) = 1$.

Now we find d_1 =??

Put $\mu = 0$ in formula2 we get,

$$\begin{aligned}
d_1 &= S_3 + \sigma_1^1 S_1 \\
d_1 &= \xi^{13} + \xi^4 \cdot \xi^8 = \xi^{13} + \xi^{12} = \xi^3 + \xi^2 + 1 + \xi^3 + \xi^2 + \xi + 1 \\
d_1 &= \xi
\end{aligned}$$

Similarly we find $\sigma^2(x)$ by putting $\mu = 1$ and $m = 0$ in formula1 we get ,

$$\sigma^2(x) = 1 + \xi^4 x + \xi^{12}x^2$$

Also from formula2 , $d_2 = 1$ and from formula3, $l_2 = 2$.

To find $\sigma^3(x)$ we substitute $\mu = 2$ and $m = 1$ in formula1 we get, $\sigma^3(x) = 1 + \xi^4 x + \xi^5 x^2 + \xi^3 x^3$ which is required error locator polynomial and this error locator polynomial is same as above example which was after 6 steps. We complete table as follows ,

μ	$\sigma^{\mu}(x)$	d_{μ}	l_{μ}	$2\mu - l_{\mu}$
$\frac{-1}{2}$	1	1	0	-1
0	1	ξ^4	0	0
1	$1+\xi^4 x$	ξ	1	1
2	$1+\xi^4 x+\xi^{12}x^2$	1	2	2
3	$1+\xi^4 x+\xi^5 x^2+\xi^3 x^3$	$-$	$-$	$-$

Table3

If we compare Table3 and Table1 then we get $\sigma^3(x)$ and $\sigma^6(x)$ are same error locator polynomial. So by this algorithm half steps are needed to find error locator polynomial.

Remark 119 *If degree of error locator polynomial is greater then t which is error correction capability of the code then there are more then t errors occur and to locate these errors is not possible in general.*

Remark 120 *The computation required for binary BCH code is one half of the computation required for non binary BCH code.*

Remark 121 *If the number of errors in $\widetilde{r}(x)$ is less then t then it is not necessary to find t steps for error locator polynomial.*

2.3 Computationally Decoding over $\mathbf{Z}_p$

It is very difficult to decode manually over a higher order field so for this we construct algorithm and develop the programs in computer languages which help us for decoding of BCH code in few seconds.

2.3.1 Construction of Galois Field Computationally in C#

Algorithm to find the elements of Galois Field

```
int qVal = Convert.ToInt32(q.Text);
   string polnolq = ppq.Text;
```

```
   Polynomial polynomialppq = new Polynomial(ppq.Text);
   int maxInd = polynomialppq.MaxIndex;
   int FinalQVal = (int)Math.Pow(qVal, maxInd - 1);
   string maxPolinomial = new Polynomial("x^" + maxInd).ToString();
   string[] strIrreducible = new Polynomial(polnolq).ToString().Split('+');
   string[] minPolynomialArr = (new Polynomial(polnolq) - new Polynomial("x^" +
maxInd)).ToString().Split('+');
   for (int j = 0; j < minPolynomialArr.Length; j++)
   {
   int index = minPolynomialArr[j].IndexOf("x");
   string npn = new Polynomial("x^" + j).ToString();
   string piece = string.Empty;
   if (index != -1)
   {
   piece = minPolynomialArr[j].Substring(minPolynomialArr[j].LastIndexOf('x'));
   }
   if (index == 0)
   {
   minPolynomialArr[j] = (qVal - 1) + new Polynomial(minPolynomialArr[j]).ToString();
   }
   else if (index > 0 && npn != "1")
   {
   minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + npn;
   }
   else if (index > 0 && npn == "1")
   {
   minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + piece;
   }
```

```
    else if (index == -1)
    {
    minPolynomialArr[j] = (qVal - Convert.ToInt32(minPolynomialArr[j])).ToString();
    }
    }
    for (int k = 0; k < minPolynomialArr.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = minPolynomialArr[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = minPolynomialArr[k];
    minPolynomialArr[k] = ((Convert.ToInt32(minPolynomialArr[k].Substring(0, index))
% qVal)).ToString() + "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    minPolynomialArr[k] = (Convert.ToInt32(minPolynomialArr[k]) % qVal).ToString();
    }
```

```
}
}
string minPolynomial = string.Join("+", minPolynomialArr);
Polynomial modPolynomial = new Polynomial(minPolynomial);
Polynomial maxpPolynomial = new Polynomial(maxPolinomial);
int MaxPolyValue = 0;
var list = new List<KeyValuePair<string, string>>();
list.Add(new KeyValuePair<string, string>(maxpPolynomial.ToString(), modPolynomial.ToSt
{
maxpPolynomial = maxpPolynomial * new Polynomial("x^1");
Polynomial p1 = modPolynomial * new Polynomial("x^1");
string[] p1Arr = p1.ToString().Split('+');
for (int i = 0; i < p1Arr.Length; i++)
{
if (p1Arr[i].Contains("x^" + maxInd))
{
int index = p1Arr[i].IndexOf("x");
string sstr = p1Arr[i].Replace("x^" + maxInd, "+" + new Polynomial(minPolynomial));
string[] strr = sstr.Split('+');
p1Arr[i] = string.Empty;
if (string.IsNullOrEmpty(strr[0]))
{
strr[0] = "1";
}
for (int j = 1; j < strr.Length; j++)
{
strr[j] = (new Polynomial(strr[0]) * new Polynomial(strr[j])).ToString();
if (string.IsNullOrEmpty(p1Arr[i]))
{ p1Arr[i] += strr[j]; }
else
```

```
    {
    p1Arr[i] += "+" + strr[j];
    }
    }
    }
    }
    Polynomial p2 = new Polynomial(string.Join("+", p1Arr));
    string[] strp2 = p2.ToString().Split('+');
    for (int k = 0; k < strp2.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = strp2[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = strp2[k];
    strp2[k] = ((Convert.ToInt32(strp2[k].Substring(0, index)) % qVal)).ToString()
+ "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
```

```
{
strp2[k] = (Convert.ToInt32(strp2[k]) % qVal).ToString();
}
}
}
modPolynomial = new Polynomial(string.Join("+", strp2));
list.Add(new KeyValuePair<string, string>(maxpPolynomial.ToString(), modPolynomial.ToSt
if (modPolynomial.ToString() == "1")
{
MaxPolyValue = new Polynomial(maxpPolynomial.ToString()).MaxIndex;
break;
}
}
Result.Text = "";
Result.Text += "Galois Field:" + Environment.NewLine;
for (int i = 1; i < maxInd; i++)
{
Result.Text += new Polynomial("x^" + i).ToString() + Environment.NewLine; ;
}
```

2.3.2 Computational check on received word to correct the errors

It is very difficult to check manually the received message of large length have errors or not and to calculate syndromes for decoding the received message. So for this problem we construct an algorithm which tell us received message is code word or not in few seconds and also shows all syndromes for decoding of BCH codes very fast .

To check the received vector have errors or not

If all the syndromes of received vector are zeros then it means that there is no error in received vector or if all the non zero elements of Galois field are roots of received polynomial then it also shows that there is no error in received vector.

Algorithm to calculate all syndromes

```
//To find syndromes
   string[] Syndrome = new string[2*TVal];
   for(int i = 0;i< (2*TVal); i++)
   {
   string[] receiveStrArr = receiveStr.Split('+');
   for (int j = 0; j < receiveStrArr.Length; j++)
   {
   int index = receiveStrArr[j].IndexOf("x");
   if (index > 0)
   {
   int cof = Convert.ToInt32(receiveStrArr[j].Substring(0, index));
   int maxIndRecVec = new Polynomial(receiveStrArr[j]).MaxIndex;
   receiveStrArr[j] = cof.ToString() + "" + new Polynomial("x^" + ((maxIndRecVec
* (i+1)) % MaxPolyValue));
   }else if(index == 0)
   {
   int maxIndRecVec = new Polynomial(receiveStrArr[j]).MaxIndex;
   receiveStrArr[j] = new Polynomial("x^" + ((maxIndRecVec * (i + 1)) % MaxPolyValue)).ToSt
   }
   }
   for(int k = 0;k<receiveStrArr.Length;k++)
   {
   if (list.Where(n => n.Key == receiveStrArr[k]).FirstOrDefault().Key == receiveStrArr[k])
   {
   receiveStrArr[k] = list.Where(n => n.Key == receiveStrArr[k]).FirstOrDefault().Value;
   }
   }
   string receiveStrArrStr = string.Join("+", receiveStrArr);
   Polynomial p1 = new Polynomial(receiveStrArrStr);
```

```
    string[] strp2 = p1.ToString().Split('+');
    for (int k = 0; k < strp2.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = strp2[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = strp2[k];
    strp2[k] = ((Convert.ToInt32(strp2[k].Substring(0, index)) % qVal)).ToString()
+ "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    strp2[k] = (Convert.ToInt32(strp2[k]) % qVal).ToString();
    }
    }
    }
    Polynomial ppp = new Polynomial(string.Join("+",strp2));
    Syndrome[i] = ppp.ToString();
```

```
}
Result.Text += "Syndrome:" + Environment.NewLine;
for(int i=0;i<((2*TVal)); i++)
{
if(list.Where(m=>m.Value == Syndrome[i]).FirstOrDefault().Value == Syndrome[i])
{
Result.Text = "S" + (i + 1).ToString() + " ==> " + list.Where(m => m.Value ==
Syndrome[i]).FirstOrDefault().Key + " = " + Syndrome[i] + Environment.NewLine;
}
else
{
Result.Text += "S" + (i + 1).ToString() + " ==> " + Syndrome[i] + Environment.NewLine;
}
}
}
```

Calculation of inverse in Galois Field Computationally

We can calculate computationally inverse of each non zero element of Galois field in few seconds which helps us to decode the received message. Algorithm to find the inverse in Galois field is follows,

Write the program first for the construction of Galois field as above then we find inverse as,

```
int qVal = Convert.ToInt32(q.Text);
string polnolq = ppq.Text;
Polynomial polynomialppq = new Polynomial(ppq.Text);
int maxInd = polynomialppq.MaxIndex;
int FinalQVal = (int)Math.Pow(qVal, maxInd - 1);
string maxPolinomial = new Polynomial("x^" + maxInd).ToString();
string[] strIrreducible = new Polynomial(polnolq).ToString().Split('+');
string[] minPolynomialArr = (new Polynomial(polnolq) - new Polynomial("x^" +
maxInd)).ToString().Split('+');
```

```
    for (int j = 0; j < minPolynomialArr.Length; j++)
    {
    int index = minPolynomialArr[j].IndexOf("x");
    string npn = new Polynomial("x^" + j).ToString();
    string piece = string.Empty;
    if (index != -1)
    {
    piece = minPolynomialArr[j].Substring(minPolynomialArr[j].LastIndexOf('x'));
    }
    if (index == 0)
    {
    minPolynomialArr[j] = (qVal - 1) + new Polynomial(minPolynomialArr[j]).ToString();
    }
    else if (index > 0 && npn != "1")
    {
    minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + npn;
    }
    string ElementHighPower = string.Empty;
    if (!string.IsNullOrWhiteSpace(list.Where(m => m.Value == ElementValue).FirstOrDefault()
    {
    ElementHighPower = list.Where(m => m.Value == ElementValue).FirstOrDefault().Key;
    }
    else
    {
    ElementHighPower = ElementValue;
    }
    int inversePower = MaxPolyValue - new Polynomial(ElementHighPower).MaxIndex;
    Result.Text = "";
    Result.Text += Environment.NewLine;
```

```
Result.Text += "Inverse of Element is:" + Environment.NewLine;
string pinverse = new Polynomial("x" + inversePower).ToString();
if (!string.IsNullOrWhiteSpace(list.Where(m => m.Key == pinverse).FirstOrDefault().Value
{
Result.Text += pinverse + " = " + list.Where(m => m.Key == pinverse).FirstOrDefault().Va
}
else
{
Result.Text += pinverse;
}
}
else
{
Result.Text += "Invalid element.";
}
}catch(Exception ex)
{
Result.Text = ex.InnerException.ToString();
}
}
}
}
```

Chapter 3

Construction and Decoding of BCH Codes Over Z_{p^m}

In Coding theory, generally design of codes for the reliable transmission of data through noisy channels by using classical and more efficient algebraic techniques. There are many applications of coding theory in different fields for example In magnetic and optical recording, wireless and network communication systems. Error detection and error correction are basic goals in coding theory because during in data transmission there is possible in most cases error must be occurs due to distortion, noise, interference. The main aim of coding theory is to design error control codes. To achieve the aim of coding theory many mathematicians,engineers uses different algebraic techniques to develop the good codes as much as possible.

3.1 Introduction and History of Codes over $\mathbf{Z}_{p^m}$

The development of data transmission codes starts with the first paper of Claude Shannon in 1948. He explained in this paper that every communication channel has some capacity. If the rate of data transmission is smaller then the capacity then design of communication system for the channel is possible with the help of data transmission codes. This system has small probability of output errors but Shannon did not give the method for the construction of such type of codes.

The first block codes for this purpose were developed by Hamming in 1950. He represents

the class of codes which correct only one error. However in 1954, Muller described the class of codes which correct multiple errors and Reed also in 1954 gave the decoding algorithm for these codes. But both these codes are not good for the Shannon's hypothesis.

The remarkable development in coding theory begins when Bose, Chaudhuri and Hocquenghem in 1960 explain the large class of codes which corrects multiple errors known as BCH codes and Reed-Solomon codes. They explain the BCH codes over finite fields that is Galois fields. In 1972, Blake [3] gave the method for construction of codes over ring Z_n, where n is product of distinct primes p_j, from cyclic codes over $GF(p_j)$. But he does not explain the codes over $\mathbb{Z}_{p^m}$, for $m > 1$. Further in 1975, Blake [4] discuss Linear codes over the ring Z_n, where $n = p^r$ and p is a prime and $r \in Z^+$. He also define Hamming and Reed-Solomon codes over Galois field by considering their properties.

Spiegal [14]in 1977 shows that codes over $\mathbb{Z}_{p^m}$ can be described in terms of codes over $\mathbb{Z}_p$ and thus we are able to define codes over $\mathbb{Z}_n$ for any positive integer n. In 1979, Shankar [13]constructs the BCH codes over $\mathbb{Z}_{p^m}$. Shanker use $\mathbb{Z}_{p^m}[x]$ which is the polynomial ring in variable x over $\mathbb{Z}_{p^m}$, he also construct BCH code for arbitrary integer M, where $M = \prod_{i=1}^{l} p_i^{k_i}$. BCH codes over finite commutative rings with identity are constructed by Andrade and Palazzo [1]in 1998. In the construction techniques of both [1] and [13], the cyclic subgroup of the group of units of an extension ring is specified. Interlando, Palazzo and Elia [6] proposed powerful decoding method which is based on Barlekamp-Massey algorithm for RS and BCH codes over integer residue ring $\mathbb{Z}_{p^m}$.Andrade and Palazzo [1] explain decoding procedure of $C(n, \eta)$ with same algorithm that can correct all errors up to Hamming weight t.This is very difficult to decoding manually over the higher order integer residue ring and Galois ring so we have solve the decoding of BCH code and Reed-Solomon code by using Barlekamp Massey algorithm computationally.

3.2 BCH Codes over Galois Ring: Construction

The first algorithm for construction of BCH codes over Galois Ring was given by Priti Shankar. He construct the codes over $\mathbb{Z}_{p^m}$ by the method which is same as the construction of codes over $GF(p)$. For this purpose, extension of Galois rings is used, where few conditions of extension

of Galois field are lost. To explain important properties of Galois extension rings, let $\mathbb{Z}_{p^m}[x]$ be the ring of polynomial and $h(x)$ be the irreducible polynomial of degree r over $\mathbb{Z}_{p^m}$ and also over $GF(p)$. Then $\mathcal{R} = GR(p^m, r) = \mathbb{Z}_{p^m}[x] / < h(x) >$ is called Galois extension of $\mathbb{Z}_{p^m}$ of dimension r. For certain value of n, $x^n - 1$ can be written as into linear factors over $GR(p^m, r)$, where n is such that $\gcd(n, p^m) = \gcd(n, p) = 1$. With the help of these factors we determine the cyclic and BCH codes over $\mathbb{Z}_{p^m}$. Zero divisors of $GR(p^m, r)$ form an abelian group under "+" having elements of degree $r-1$ or less. The coefficients of these polynomials are zero divisors in $\mathbb{Z}_{p^m}$. It means $GR(p^m, r)$ is a local ring. The Units of $GR(p^m, r)$ are those polynomials having atleast one coefficient is unit in $\mathbb{Z}_{p^m}$. The units of $GR(p^m, r)$ form a multiplicative group and it is represented by $\mathcal{R}^*$. It is an abelian group and can be written as direct product of cyclic groups. We are looking for the maximal cyclic subgroup G_n, whose elements are the zeros of $x^n - 1$.

The following results are important for construction of G_n and BCH codes:

Theorem 122 *[6, Theorem 2] There is unique G_n of $\mathcal{R}^*$ of order relatively prime to p. This cyclic subgroup has order $p^r - 1$.*

Theorem 123 *[6, Theorem 3] Let α generates the cyclic subgroup having order n in $\mathcal{R}^*$, such that $\gcd(n, p) = 1$. Then the polynomial $x^n - 1$ can be factorized as $x^n - 1 = (x - \alpha)\left(x - \alpha^2\right) \cdots (x - \alpha^n)$ iff $R_p(\alpha)$ has order n in $\mathcal{K}^*$, which is the multiplicative subgroup of $\mathcal{K} = GF(p^r)$.*

Corollary 124 *[6, Corollary 1] Assume that α generates the cyclic subgroup having order n in R^*, then the polynomial $h(x)$, which divides $x^n - 1$ having coefficient in $\mathbb{Z}_{p^m}$ can be factorized over G_n as $h(x) = (x - \alpha^{e_1})(x - \alpha^{e_2}) \cdots (x - \alpha^{e_l})$ iff $R_p(h(x))$ can be factorized over $GF(p^r)$ as $R_p(h(x)) = (x - (R_p(\alpha))^{e_1})(x - (R_p(\alpha))^{e_2}) \cdots (x - (R_p(\alpha))^{e_l})$.*

Theorem 125 *[6, Theorem 4] Let $\overline{\alpha} = R_p(\alpha)$ generate a subgroup which is cyclic of order n in $\mathcal{K}^*$.Then α generate cyclic subgroup of order $n.d$ in $\mathcal{R}^*$, where $d \geq 1$, and $G_n =< \alpha^d >$ of $\mathcal{R}^*$.*

Lemma 126 *[1, Lemma 3.1] Let α be a primitive element of G_n. Then the differences $\alpha^{l_1} - \alpha^{l_2}$ are units in $\mathcal{R}$ if $0 \leq l_1 \neq l_2 \leq n - 1$.*

On the basis of above results, the generator polynomial $g(x)$ of cyclic BCH code of length n in $GR(p^m, r)$ can be calculated as

$$g(x) = \text{lcm}\left(\widetilde{M}_1(x), \widetilde{M}_2(x), ..., \widetilde{M}_{2t}(x)\right)$$

where $\widetilde{M}_i(x)$, $1 \leq i \leq 2t$ are the minimal polynomials of α^{b+i} over $\mathbb{Z}_{p^m}$, for some $b \geq 0$ and $t \geq 1$. The polynomials $M_i(x)$ over $GR(p^m, r)$ are calculated by the method similar to the calculation of $m_i(x)$ (minimal polynomial over $GF(p^r)$). And the parity check matrix of BCH code having generator polynomial $g(x)$ is of the form

$$H = \begin{bmatrix} 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ 1 & \alpha^{b+2} & \alpha^{2(b+2)} & \cdots & \alpha^{(n-1)(b+2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+2t} & \alpha^{2(b+2t)} & \cdots & \alpha^{(n-1)(b+2t)} \end{bmatrix}. \tag{3.1}$$

The minimum distance is guaranteed by following theorem:

Theorem 127 *The* min*imum distance of the BCH code is defined in 3.1 is greater then or equal to* $2t + 1$.

Generator polynomial of BCH Code and Maximum Cyclic Subgroup over Galois ring

The BCH codes over Galois rings are constructed corresponding to each Galois field. For Galois fields $GF(p^m)$ of order p^m, there are many Galois rings $GR(p^n, m)$ of order p^{mn}, where $n \in Z^+$ and m is the degree of monic irreducible polynomial $f(x)$. The polynomial $f(x)$ is irreducible in Galois rings and $\overline{f(x)} = R_p(f(x))$ is primitive irreducible in the corresponding Galois field. By increasing the value of n, we have codes having more codewords. Some other observations are made by considering the following example.

Example 128 *For the rings* $\mathbb{Z}_{2^n}$, $\mathbb{Z}_2$ *is the residue field. Since* $f(x) = x^3 + x + 1$ *be a monic and irreducible polynomial over* $\mathbb{Z}_{2^n}$ *and* $\overline{f(x)} = x^3 + x + 1$ *is irreducible over* $\mathbb{Z}_2$, *so it is basic irreducible polynomial. Galois ring* $\mathcal{R} = GR(2^n, 3) = \frac{\mathbb{Z}_{2^n}[x]}{(f(x))}$ *of order* 2^{3n} *and corresponding*

Galois field of order 8 *becomes* $\mathcal{K} = \frac{\mathbb{Z}_2[x]}{(\overline{f(x)})}$. *Let* α *be the root of* $\overline{f(x)}$, *then* $\alpha^3 + \alpha + 1 = 0$. *By calculating the powers of* α *modulo* 2 *and modulo* $\overline{f(\alpha)}$, *we have* $\alpha^7 = 1$. *The powers of* α *represents number of non zero elements in* $GF\left(2^3\right)$. *Now we consider* $\alpha^3 = -\alpha - 1$ *modulo* $2^2, 2^4, 2^5$ *and modulo* $f(u)$,*and again modulo* $2^2, 2^4, 2^5$ *then* $14, 56, 112$ *powers of* α *are equal to* 1 *respectively. It implies that the Maximal cyclic subgroup* G_7 *is generated by* α^2, α^8 *and* α^{16} *in Galois rings* $GR\left(2^2, 3\right), GR\left(2^4, 3\right)$ *and* $GR\left(2^5, 3\right)$ *respectively. The possible generator polynomials* $g^n(x)$ *of primitive BCH codes* C_m^n *of length* 7 *in rings* $GR\left(2^n, 3\right)$ *are:-*

$g^2(x)$ *in* $GR\left(2^2, 3\right)$	$g^4(x)$ *in* $GR\left(2^4, 3\right)$	$g^5(x)$ *in* $GR\left(2^5, 3\right)$
$x^3 + 2x^2 + x + 3$	$x^3 + 6x^2 + 5x + 15$	$x^3 + 6x^2 + 5x + 31$
$x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$	$x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$	$x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

Table : Generator Polynomials of BCH codes of length 7

It means that possible codes of length 7 *are* $(7, 4)$ *and* $(7, 1)$ *having minimum distances* 3 *and* 7, *error correction capabilities* 1 *and* 3, *and code rates* 0.571 4 *and* 0.142 8 *respectively. Further, it is observed that if we take* $f(x) = x^3 + 3x + 1$ *and* $f(x) = x^3 + 5x + 3$ *as the irreducible polynomial over* $\mathbb{Z}_{2^2}$ *and* $\mathbb{Z}_{2^3}$ *respectively, the generator polynomials and all above results remains the same. By considering the other example of primitive BCH codes,*

Example 129 *Suppose that* $f(x) = x^3 + x + 1$ *is irreducible polynomial over* $\mathbb{Z}_8$. *We have to construct BCH codes over extension ring of* $\mathbb{Z}_8$ *which is* $\frac{\mathbb{Z}_8[x]}{<x^3+x+1>} \simeq GR(8, 3)$. *Firstly we construct maximum cyclic subgroup of* $GR(8, 3)$ *which is isomorphic to* $\mathcal{K}^* = GF(8) \setminus \{0\}$. *Let* α *be the primitive root of the polynomial* $f(x)$ *so* $\alpha^3 + \alpha + 1 = 0$.

This implies that $\alpha^3 = -\alpha - 1 = 7\alpha + 7$ *now by taking powers of* α *until we get* 1 *it means*

that we have to find the order of α *in* $GR(8,3)$ *.*

$\alpha^4 = 7\alpha + 7\alpha^2$	$\alpha^{13} = 7\alpha^2 + 3$	$\alpha^{22} = 6 + 5\alpha + 4\alpha^2$
$\alpha^5 = 1 + \alpha + 7\alpha^2$	$\alpha^{14} = 1 + 4\alpha$	$\alpha^{23} = 4 + 2\alpha + 5\alpha^2$
$\alpha^6 = 1 + 2\alpha + \alpha^2$	$\alpha^{15} = \alpha + 4\alpha^2$	$\alpha^{24} = 3 + 7\alpha + 2\alpha^2$
$\alpha^7 = 7 + 2\alpha^2$	$\alpha^{16} = 4 + 4\alpha + \alpha^2$	$\alpha^{25} = 6 + \alpha + 7\alpha^2$
$\alpha^8 = 6 + 5\alpha$	$\alpha^{17} = 7 + 3\alpha + 3\alpha^2$	$\alpha^{26} = 1 + 7\alpha + \alpha^2$
$\alpha^9 = 6\alpha + 5\alpha^2$	$\alpha^{18} = 4 + 3\alpha + 3\alpha^2$	$\alpha^{27} = 7 + 7\alpha^2$
$\alpha^{10} = 3 + 3\alpha + 6\alpha^2$	$\alpha^{19} = 5 + \alpha + 3\alpha^2$	$\alpha^{28} = 1$
$\alpha^{11} = 2 + 5\alpha + 3\alpha^2$	$\alpha^{20} = 5 + 2\alpha + \alpha^2$	
$\alpha^{12} = 5 + 7\alpha + 5\alpha^2$	$\alpha^{21} = 7 + 4\alpha + 2\alpha^2$	

Hence order of α *in* $GR(8,3)$ *is* 28 *so* G_s *maximum cylic subgroup is generated by* α^4 *and order of* G_s *is* 7 *by theorem 122. Therefore* $G_7 =< \beta = \alpha^4 >$ *and elements of* G_7 *are given as ,*

$\beta = 7\alpha + 7\alpha^2$
$\beta^2 = 6 + 5\alpha$
$\beta^3 = 5 + 7\alpha + 5\alpha^2$
$\beta^4 = 4 + 4\alpha + \alpha^2$
$\beta^5 = 5 + 2\alpha + \alpha^2$
$\beta^6 = 3 + 7\alpha + 2\alpha^2$
$\beta^7 = 1$

This implies that $G_7 = \{1, \beta, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6\}$ *and these all elements are roots of* $x^7 - 1$ *modulo* 8*. Now we construct BCH code for designed distance* $d = 4$ *.So find the minimal polynomial of* β^i *where* $i = 1, 2, 3$*.Since* β, β^2, β^4 *have same minimal polynomial so* $m_1(x) = (x - \beta)\left(x - \beta^2\right)\left(x - \beta^4\right) = x^3 + 6x^2 + 5x + 7$ *. Now* $\beta^3, \beta^6, \beta^5$ *have same minimal polynomial so* $m_2(x) = \left(x - \beta^3\right)\left(x - \beta^5\right)\left(x - \beta^6\right) = x^3 + 3x^2 + 2x + 7$ *therefore* $g(x) = m_1(x) \cdot m_2(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ *is generator polynomial of BCH code corrosponding to Galois*

Ring. If we have to find the generator polynomial for corrosponding Galois field then there is no need to construct all minimal polynomial just take modulo p of coefficients of $g(x)$ which is already constructed for Galois ring so we get $\overline{g(x)} = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ for the Galois field $Gf\left(2^3\right)$.

Remark 130 *If α is generator of $GF(p, m)$, then $\alpha^{p^{n-1}}$ generator of cyclic subgroup in $GR(p^n, m)$.*

Remark 131 *The generator polynomials $g^n(x)$ of C_m^n are converted to generator polynomials $g^{n-1}(x)$ of C_m^{n-1} by reducing the coefficients of $g^n(x)$ modulo p^{m-1}.*

Remark 132 *Error correction capability and code rate remains same for codes in Galois rings $GR(2^n, m)$ and in corresponding Galois fields $GF(2^m)$. But the number of codewords in Galois ring are greater then the Galois field.*

Reed-Solomon Codes over Z_{p^m}

Let $\beta \in Z_{p^m}$ be a primitive element of the integers modulo p. Suppose $n = p-1$ and a positive integer m relatively prime to n then according to Blake [4] we define the matrix $\widetilde{H}$ as

$$\widetilde{H} = \begin{bmatrix} 1 & \beta^m & \beta^{2m} & \dots & \beta^{(n-1)m} \\ 1 & \beta^{m+1} & \beta^{2(m+1)} & \dots & \beta^{(n-1)(m+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \beta^{m+d-2} & \beta^{2(m+d-2)} & \dots & \beta^{(n-1)(m+d-2)} \end{bmatrix}$$

Theorem 133 *The null space of $\widetilde{H}$ over Z_{p^m} is a code C having length $n = p-1$ and minimum Hamming distance d and dimension $(p-d)$.*

Hence $\widetilde{H}$ is a parity check matrix of a $(p-1, p-d, d)$ Reed-Solomon code over Z_{p^m}. We suppose $m = 1$ and $d = 2t+1$ here $t \leq \left\lfloor \frac{n-1}{2} \right\rfloor$, So we have t-error correcting Reed Solomon codes.

3.3 Decoding procedure of Reed-Solomon and BCH-Codes

In this section we solve the major issues related to decoding on Reed-Solomon codes and BCH codes of n length which has t errors correction capability whose minimum hamming distance

$\geq 2t+1$. We take Ring $\mathcal{R} = Z_{p^m}$ in the case of Reed-Solomon codes and take $\mathcal{R} = GR(p^m, r)$ in the case of BCH codes. The symbol β will be used to represent primitive element of Z_p or G_n.

Interlando, Palazzo and Elia [6] presents the decoding procedure which is based on Berlekamp-Massey algorithm for Reed-Solomon and BCH codes over $\mathbb{Z}_{p^m}$, where m is positive integer. This algorithm is further extended by Andrade and Palazzo [1] for $C(n^{'}, \eta)$. This code corrected all errors up to Hamming weight and decoding procedure have following steps.

(1) *Syndromes calculation with the help of parity check matrix.*

(2) *Calculation of elementary symmetric functions from syndromes.*

(3) *Calculate the error location numbers.*

(4) *For RS or BCH codes over Z_{p^m}, the error values or error magnitudes to be calculated..*

To explain these steps, let $\widetilde{c} = (\widetilde{c}_1, \widetilde{c}_2, ..., \widetilde{c}_n)$ be a n length transmitted vector and $\widetilde{\mathbf{r}} = (\widetilde{r}_1, \widetilde{r}_2, ..., \widetilde{r}_n)$ be a received vectors, then error vector is given by $\mathbf{e} = (e_1, e_2, ..., e_n) = \widetilde{r} - \widetilde{c}$. By considering α as primitive element of G_n, the above steps are analyzed as:

Step 1: Syndromes calculation Calculation of syndromes is very simple just take product of received vector and transpose of parity check matrix as,

$$\widetilde{S} = \widetilde{\mathbf{r}} \cdot \widetilde{\mathbf{H}}^t$$

There are total $2t$ syndromes are calculated.

Step 2: Calculation of elementary symmetric functions The elementary symmetric functions $\overline{\sigma_1}, \overline{\sigma_2}, ..., \overline{\sigma_v}$ are defined as coefficients of the polynomial

$$(X - Z_1)(X - Z_2)...(X - Z_v) = X^v + \overline{\sigma_1} X^{v-1} + ... + \overline{\sigma_{v-1}} X + \overline{\sigma_v}$$

where v represents number of errors introduced by the channel. These functions are obtained by finding a solution $\overline{\sigma_1}, \overline{\sigma_2}, ..., \overline{\sigma_v}$ of the following equations over R with minimum possible v.

$$\widetilde{S}_{j+v} + \widetilde{S}_{j+v-1}\overline{\sigma_1} + ... + \widetilde{S}_{j+1}\overline{\sigma_{v-1}} + \widetilde{S}_j \overline{\sigma_v} = 0 \quad \text{for } j = 1, 2, ..., 2t - v, \tag{3.2}$$

Where $\widetilde{S}_1, \widetilde{S}_2, ..., \widetilde{S}_{2t}$ are the sequence of syndromes. This solution is obtained by modified Berlekamp-Massey algorithm which holds for commutative rings with identity. The solution is unique iff error magnitudes are unit in ring $\mathcal{R}$. It is an iterative algorithm because at μth step, we have to determine $l_{\boldsymbol{\mu}}$ values $\overline{\sigma_i^{(\boldsymbol{\mu})}}$ such that the following $\mu - l_{\boldsymbol{\mu}}$ equations holds with $l_{\boldsymbol{\mu}}$ possibly small and $\overline{\sigma_0^{(\boldsymbol{\mu})}} = 1$.

$$\begin{aligned} \widetilde{S}_{\mu}\overline{\sigma_0^{(\boldsymbol{\mu})}} + \widetilde{S}_{\boldsymbol{\mu}-1}^{(\boldsymbol{\mu})}\overline{\sigma_1^{(\boldsymbol{\mu})}} + ... + \widetilde{S}_{\boldsymbol{\mu}-l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}\overline{\sigma_{l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}} = 0 \\ \widetilde{S}_{\boldsymbol{\mu}-1}\overline{\sigma_0^{(\boldsymbol{\mu})}} + \widetilde{S}_{\boldsymbol{\mu}-2}\overline{\sigma_1^{(\boldsymbol{\mu})}} + ... + \widetilde{S}_{\boldsymbol{\mu}-l_{\boldsymbol{\mu}}-1}\overline{\sigma_{l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}} = 0 \\ \vdots \\ \widetilde{S}_{l_{\boldsymbol{\mu}}+1}\overline{\sigma_0^{(\boldsymbol{\mu})}} + \widetilde{S}_{l_{\boldsymbol{\mu}}}\overline{\sigma_1^{(\boldsymbol{\mu})}} + ... + \widetilde{S}_1\overline{\sigma_{l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}} = 0 \end{aligned} \tag{3.3}$$

At nth stage the solution is represented by the polynomial $\overline{\sigma^{(\boldsymbol{\mu})}(x)} = \overline{\sigma_0^{(\boldsymbol{\mu})}} + \overline{\sigma_1^{(\boldsymbol{\mu})}}x + \overline{\sigma_{l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}}x^{l_{\boldsymbol{\mu}}}$ and nth discrepancy $d_{\boldsymbol{\mu}}$ is defined by $d_{\boldsymbol{\mu}} = \widetilde{S}_{\boldsymbol{\mu}+1}\overline{\sigma_0^{(\boldsymbol{\mu})}} + \widetilde{S}_{\boldsymbol{\mu}}\overline{\sigma_1^{(\boldsymbol{\mu})}} + ... + \widetilde{S}_{\boldsymbol{\mu}+1-l_{\boldsymbol{\mu}}}\overline{\sigma_{l_{\boldsymbol{\mu}}}^{(\boldsymbol{\mu})}}$. The modified Berlekamp-Massey algorithm is explained by following results:

Lemma 134 *[6, Lemma 1]Let $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$ be a solution to the first μ power sums and has next discrepancy $d_{\boldsymbol{\mu}} \neq 0$ Let*

$$\overline{\sigma^{(m)}(x)} = 1 + \sigma_1^{(m)} \cdot x + \cdots + \sigma_{l_m}^{(m)} \cdot (x)^{l_m}$$

be any polynomial solution to the first m power sums, where $1 \leq m < \mu$, such that the linear equations $d_{\boldsymbol{\mu}} - y.d_m = 0$ in $\mathcal{R}$ have solutions in y. Then the polynomial,

$$\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} = \overline{\sigma^{(\boldsymbol{\mu})}(x)} - y.(x)^{\boldsymbol{\mu}-m}.\overline{\sigma^{(m)}(x)}$$

is a solution to the first $\mu + 1$ power sums. Further, $l_{\boldsymbol{\mu}+1} = \max\{l_{\boldsymbol{\mu}}, l_m + \mu - m\}$.

Lemma 135 *[6, Lemma 2]Suppose that $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$, $l_{\boldsymbol{\mu}}$, and $d_{\boldsymbol{\mu}} \neq 0$ are defined as in above lemma and $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ be the any polynomial solution satisfying $\mu + 1 - l_{\boldsymbol{\mu}+1}$ power sums. Then let*

$$\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} = \overline{\sigma^{(\boldsymbol{\mu})}(x)} - a.(x)^{\boldsymbol{\mu}-m}.\overline{\sigma^{(m)}(x)},$$

where a is unit in $\mathcal{R}$ and $\overline{\sigma_0^{(m)}} = 1$. Then the polynomial $\overline{\sigma^{(m)}(x)}$ is a polynomial solution to the

first $m - l_m$ equations of (3.3) and has next discrepancy satisfying $d_{\boldsymbol{\mu}} + a \cdot d_m = 0$ and $l_m = l_{\boldsymbol{\mu}+1} - (\mu - m)$.

Theorem 136 *[6, Theorem 6]Let $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$ be a solution at the μth stage and $\overline{\sigma^{(m)}(x)}$ be one of the prior minimal solutions, where $1 \leq m < \mu$, such that $d_{\boldsymbol{\mu}} - y.d_m = 0$ have solutions in y and $m - l_m$ have largest value in last column of the table. Further, suppose that $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is modified by the following method. If $d_{\boldsymbol{\mu}} = 0$ then*

$$\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} = \overline{\sigma^{(\boldsymbol{\mu})}(x)} \text{ and } l_{\boldsymbol{\mu}+1} = l_{\boldsymbol{\mu}}. \tag{3.4}$$

If $d_{\boldsymbol{\mu}} \neq 0$ then

$$\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} = \overline{\sigma^{(\boldsymbol{\mu})}(x)} - y.(x)^{\boldsymbol{\mu}-m}.\overline{\sigma^{(m)}(x)} \text{ and } l_{\boldsymbol{\mu}+1} = \max\{l_{\boldsymbol{\mu}}, l_m + \boldsymbol{\mu} - m\}. \tag{3.5}$$

If there is not any solution $\overline{D^{(\boldsymbol{\mu}+1)}(x)}$ with degree smaller than $\max\{l_{\boldsymbol{\mu}}, l_m + \mu - m\}$, and the coefficient of the smallest exponent of x in $\overline{D^{(\boldsymbol{\mu}+1)}(x)} - \overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is a zero divisor in $\mathcal{R}$, then at $(\mu+1)$th stage $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ is a minimal polynomial solution.

Proof. :If $d_\mu = 0$ then $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} = \overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is minimal solution because $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is minimal solution. Now we suppose that if $d_\mu \neq 0$ then by 3.5 $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ is known. By 134 $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ is a polynomial solution of degree $l_{\boldsymbol{\mu}+1} = \max\{l_{\boldsymbol{\mu}}, l_m + \mu - m\}$.

Now we show that this solution is minimal,

From above lemma if $m - l_m \geq \mu - l_\mu$ then $l_{\mu+1} = l_\mu$ and at $(\mu+1)$th stage $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ is a minimal solution.

If $m - l_m < \mu - l_\mu$ then $l_{\mu+1} = \max\{l_\mu, l_m + \mu - m\} = l_m + \mu - m > l_\mu$.

Now if we take $\overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ is minimal solution then assume that there exist a polynomial $\overline{D^{(\boldsymbol{\mu}+1)}(x)}$ of degree d such that $l_\mu \leq d < l_m + \mu - m$ and the coefficient of the smallest exponent of x in $\overline{D^{(\mu+1)}(x)} - \overline{\sigma^{(\mu)}(x)}$ is unit in $\mathcal{R}$.

We consider two cases,

1) If $d = l_\mu$ then from 135 there is a solution $\overline{\sigma^{\left(\bar{m}\right)}(x)}$ with $l_{\bar{m}} = d - \left(\mu - \bar{m}\right)$ that is $\bar{m} - l_{\bar{m}} = \mu - l_\mu$. From our supposition $m - l_m < n - l_n$ so $m - l_m < \bar{m} - l_{\bar{m}}$. So $m - l_m$ was taken to be the largest of the value $k - l_k$ for the previous solutions which shows a contradiction.

2) If $d > l_\mu$ then from above lemma $d = l_{\underline{m}} + \mu - \bar{m}$. However because $m - l_m \geq \bar{m} - l_{\underline{m}}$ so $d = \mu - \left(\bar{m} - l_{\underline{m}}\right) \geq n - (m - l_m) = l_{\mu+1} > d$

that is $d > d$ which shows a contradiction.

Hence if the coefficient of the smallest power of x in $D^{(\mu+1)}(x) - \sigma^{(\mu)}(x)$ is a unit in$\mathcal{R}$ then $\sigma^{(\boldsymbol{\mu}+1)}(x)$ is a minimal solution. ■

Modified Berlekamp Massey Algorithm

We take in this algorithm syndromes as a input and output of this algorithm will be elementary symmetric functions which satisfies equations 3.3 for minimum υ.We start this algorithm by following initial conditions as in [19],

$\overline{\sigma^{(-1)}(x)} = 1, \quad l_{-1} = 0, \quad d_{-1} = 1, \ \overline{\sigma^{(0)}(x)} = 1, \ l_0 = 0$, and $d_0 = \widetilde{S}_1$, here $\widetilde{S}_1$ is first non zero syndrome. Further steps of this algorithm are follow as

Step1) $0 \rightarrow \mu$.

Step2) If $d_{\boldsymbol{\mu}} = 0$, then $\overline{\sigma^{(\boldsymbol{\mu})}(x)} \rightarrow \overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$, $l_{\boldsymbol{\mu}} \rightarrow l_{\boldsymbol{\mu}+1}$ and then go to step 5.

Step3) If $d_{\boldsymbol{\mu}} \neq 0$, then find a $m < \mu$, such that $d_{\boldsymbol{\mu}} - y d_m = 0$ has solution in y and $m - l_m$ has the greatest value. Then

$$\begin{aligned} \overline{\sigma^{(\boldsymbol{\mu})}(x)} - y \cdot x^{\mu-m} \cdot \sigma^{(m)}(x) &\rightarrow \overline{\sigma^{(\boldsymbol{\mu}+1)}(x)} \\ \max\{l_\mu, l_m + \mu - m\} &\rightarrow l_{\mu+1}. \end{aligned}$$

Step4) If $l_{\boldsymbol{\mu}+1} = \max\{l_{\boldsymbol{\mu}}, \mu + 1 - l_{\boldsymbol{\mu}}\}$, then go to step 5, else find the solution $\overline{D^{(\boldsymbol{\mu}+1)}(x)}$ of minimum degree in the range $\max\{l_{\boldsymbol{\mu}}, \boldsymbol{\mu} + 1 - l_{\boldsymbol{\mu}}\} \leq l \leq l_{\boldsymbol{\mu}+1}$, such that $\overline{\sigma^{(m)}(x)}$ is defined by $x^{\boldsymbol{\mu}-m} \cdot \overline{\sigma^{(m)}(x)} = \overline{D^{(\boldsymbol{\mu}+1)}(x)} - \overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is a solution for the first m power sums and $d_m = -d_{\boldsymbol{\mu}}$, with $\overline{\sigma_0^{(m)}}$ is a zero divisor in $\mathcal{R}$. If such type of solution is found then $\overline{D^{(\boldsymbol{\mu}+1)}(x)} \rightarrow \overline{\sigma^{(\boldsymbol{\mu}+1)}(x)}$ and $l \rightarrow l_{\boldsymbol{\mu}+1}$.

Step5). If $\mu \leq 2t - 2$, then $\widetilde{S}_{\boldsymbol{\mu}+2} + \widetilde{S}_{\boldsymbol{\mu}+1}\overline{\sigma_1^{(\boldsymbol{\mu}+1)}} + ... + \widetilde{S}_{\boldsymbol{\mu}+2-l_{\boldsymbol{\mu}+1}}\overline{\sigma_{l_{\boldsymbol{\mu}+1}}^{(\boldsymbol{\mu}+1)}} \rightarrow d_{\boldsymbol{\mu}+1}$.

Step6). $\mu+1 \rightarrow \mu$, if $\mu \leq 2t - 1$ then go to step 2, else stop the algorithm.

The coefficients $\overline{\sigma_1^{2t}}, \overline{\sigma_2^{2t}}, ..., \overline{\sigma_{l_\mu}^{2t}}$ of $\overline{\sigma^{(2t)}(x)}$ satisfy equation 3.2. This algorithm allows $\overline{\sigma^{(\boldsymbol{\mu})}(x)}$ is updated from $\overline{\sigma^{(m)}(x)}$ whose discrepancy may be a non invertible element in commutative ring $\mathcal{R}$.

Step 3: Determining error location numbers There is one extra step over ring to find error location number rather then over field because in ring $\mathcal{R}$ the solution of equation 3.2 is not unique in general. The reciprocal of polynomial $\overline{\sigma^{2t}(x)}$ (output of Barlekamp Massey algorithm) is $\rho(x)$ which may not be the correct error locator polynomial. So to find error location numbers first computing the roots $x_1, x_2, ..., x_v$ of $\rho(x)$ and then select from $z_0 = \alpha^0, z_1 = \alpha^1, z_2 = \alpha^2, ...z_{n-1} = \alpha^{n-1}$ such that $z_i - x_i$ are zero divisors in ring $\mathcal{R}$ where i is a positive integer less than or equal to $n-1$. These $z_i's$ are correct error location numbers.

Step 4: Calculation of error magnitudes To calculate error magnitudes $y_1, y_2, ..., y_v$ we use Forney's method from [5] where error magnitudes defined as:

$$y_j = \frac{\sum_{l=0}^{v-1} \overline{\sigma_{j,l}} \cdot \widetilde{S}_{v-l}}{\sum_{l=0}^{v-1} \overline{\sigma_{j,l}} \cdot z_j^{v-l}} \; where \; j = 1, 2, ..., v.$$

The coefficients $\overline{\sigma_{j,l}}$ are calculated by $\overline{\sigma_{j,i}} = \overline{\sigma_i} + z_j \cdot \overline{\sigma_{j,i-1}}$, where $i = 0, 1, ..., v-1$ and $\overline{\sigma_0} = \overline{\sigma_{j,0}} = 1$.

Example 137 *Consider a two error correcting Reed-Solomon code over Z_{121}. Here $n = p-1 = 11-1 = 10$ and $t = 2$ then $d = 2t+1 = 5$. and $k = p-2 = 9$. Let $\beta = 2$ be a primitive element of Z_{11}. Hence we have $(10, 9, 5)$ code with parity check matrix is,*

$$\widetilde{H} = \begin{bmatrix} 1 & 2 & 4 & 8 & 16 & 32 & 64 & 7 & 14 & 28 \\ 1 & 4 & 16 & 64 & 14 & 56 & 103 & 49 & 75 & 58 \\ 1 & 8 & 64 & 28 & 103 & 98 & 58 & 101 & 82 & 51 \\ 1 & 16 & 14 & 103 & 75 & 111 & 82 & 102 & 59 & 97 \end{bmatrix}$$

Now suppose that codeword $\widetilde{c} = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$ is transmitted and the vector $\widetilde{r} = (0\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 9)$ is received. We have to decode this vector by using modified Berlekamp Massey algorithm .

Step 1: To calculate syndrome we use this method

$$\widetilde{S} = \widetilde{r} \cdot \widetilde{H}^t = (17 \quad 20 \quad 4 \quad 107)$$

Step 2: Calculation of polynomial which satisfy equation 3.2 by using Barlekamp Massey algorithm so we find $\overline{\sigma^{2t}(x)}$ as follows,

μ	$\overline{\sigma^{\mu}(x)}$	d_{μ}	l_{μ}	$\mu - l_{\mu}$
-1	1	1	0	-1
0	1	17	0	0
1	$1+104x$	94	1	0
2	$1+70x$	73	1	1
3	$1+x+84x^2$	97	2	1
4	$1+61x+49x^2$	–	–	–

Hence $\overline{\sigma^4(x)} = 49x^2 + 61x + 1$ is required polynomial.

Step 3: Calculation of error positions and error location numbers.

First we calculate reciprocal of $\overline{\sigma^4(x)}$ which is $\overline{\rho(x)} = x^2 + 61x + 49$ then we find roots of this polynomial by substituting $x = \{0, 1, 2, 3, 4, ..., 120\}$ so we get $x_1 = 32$ and $x_2 = 28$ are roots of $\overline{\rho(x)}$. Now we select those z_i from the elements $\{\beta^0, \beta^1, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \beta^7, \beta^8, \beta^9\}$ that is $\beta^0 = 1, \beta^1 = 2, \beta^2 = 4, \beta^3 = 8, \beta^4 = 16, \beta^5 = 32, \beta^6 = 64, \beta^7 = 7, \beta^8 = 14, \beta^9 = 28$ such that $z_i - x_i$ are zero divisors in $\mathcal{Z}_{121}$ therefore $z_1 = \beta^5 = 32$ and $z_2 = \beta^9$ such that $z_1 - x_1$ and $z_2 - x_2$ are zero divisors in $\mathcal{Z}_{121}$. Hence $z_1 = \beta^5 = 32$ and $z_2 = \beta^9 = 28$ correct error location numbers and powers of β represents error positions in received vector so here at $5th$ and $9th$ position error occurs.

Step 4: Calculation of error magnitudes,

To find calculation of error magnitudes first we find correct elementary symmetric function as ,

$(x - z_1)(x - z_2) = (x - 32)(x - 28) = x^2 + 61x + 49$ this implies that $\overline{\sigma_1} = 61$ and $\overline{\sigma_2} = 49$. Now we apply Forney's procedure to find error magnitudes ,

$$
\begin{aligned}
y_1 &= \frac{\overline{\sigma_{1,0}} \cdot \widetilde{S}_2 + \overline{\sigma_{1,1}}\widetilde{S}_1}{\overline{\sigma_{1,0}} \cdot z_1^2 + \overline{\sigma_{1,1}} \cdot z_1} \\
\overline{\sigma_{1,0}} &= 1, \quad \overline{\sigma_{1,1}} = \overline{\sigma_1} + z_1\overline{\sigma_{1,0}} \\
y_1 &= \frac{1 \cdot (20) + (93)\,(17)}{1 \cdot (32)^2 + (93)\,(32)} \\
y_1 &= \frac{28}{7} = 28\,(7)^{-1} = 28\,(52) \bmod 121 \\
y_1 &= 4
\end{aligned}
$$

Similarly we find y_2 as

$$y_2 = \frac{\widetilde{S}_2 + \overline{\sigma_{2,1}} \cdot \widetilde{S}_1}{(z_2)^2 + \overline{\sigma_{2,1}} \cdot z_2},$$

By putting the values we get $y_2 = 9$.

Hence the error vector e =(0 0 0 0 0 0 4 0 0 0 9) this implies that encoded message is $v = \widetilde{r} - e$ = (0 0 0 0 0 0 0 0 0 0 0)

Example 138 *In this example we discuss the decoding of BCH codes over Galois Ring.*

Suppose C is a $(8, 3)$ BCH code which have 2 error correction capability over $\mathbb{Z}_9$ and extension ring of $\mathbb{Z}_9$ is $\mathcal{R} = GR\,(9, 2) = \frac{\mathbb{Z}_9[x]}{<x^2+x+2>}$.

We have to decode received message over $\mathcal{R} = GR\,(9, 2)$ for this first we find maximum cyclic subgroup of order $3^2 - 1$ which is G_8. Let $\beta = 2 + 8x$ is generator of maximum cyclic subgroup. So hence $G_8 = \{1, \beta, \beta^2, \beta^3, \beta^4, \beta^5, \beta^6, \beta^7\}$. Suppose that codeword v = (0 0 0 0 0 0 0 0) is transmitted through the channel may be noisy then the received message is $\widetilde{r}$ = (0 3 0 0 0 0 6 0) . We decode this received message by using Barlekamp Messey algorithm .First we check that r is codeword or not. We compute the parity check matrix $\widetilde{H}$ of order $2t \times n$,here $t = 2$ and $n = 8$ so order of H is 4×8.

$$\widetilde{H} = \begin{bmatrix} 1 & \beta & \beta^2 & \beta^3 & \beta^4 & \beta^5 & \beta^6 & \beta^7 \\ 1 & \beta^2 & \beta^4 & \beta^6 & 1 & \beta^2 & \beta^4 & \beta^6 \\ 1 & \beta^3 & \beta^6 & \beta & \beta^4 & \beta^7 & \beta^2 & \beta^5 \\ 1 & \beta^4 & 1 & \beta^4 & 1 & \beta^4 & 1 & \beta^4 \end{bmatrix}$$

If $\widetilde{r} \cdot \widetilde{H}^t = 0$ then there is no error occurs in other words if syndrome is zero then received message is codeword.

Decoding Procedure:-

Step1: Calculation of Syndrome as,

$\widetilde{S} = \widetilde{r} \cdot \widetilde{H}^t = (3 \;\; 3x \;\; 3 \; 3)$

Step2: Calculation of polynomial which satisfy the equation 3.2 by using the modified Barlekamp Massey algorithm apply to syndrome we get the following table.

μ	$\overline{\sigma^{\mu}(Z)}$	d_{μ}	l_{μ}	$\mu - l_{\mu}$
-1	1	1	0	-1
0	1	3	0	0
1	$1+6Z$	$3x$	1	0
2	$1+(6+8x)Z$	$3x$	1	1
3	$1+(5+8x)Z+3Z^2$	$6x$	2	1
4	$1+(3+8x)Z+(2x)Z^2$	$-$	$-$	$-$

Hence $\overline{\sigma^4(Z)} = 1+(3+8x)Z+(2x)Z^2$

Step3: Calculation of correct error location number,

Take reciprocal of $\overline{\sigma^4(Z)}$ then polynomial is $\overline{\rho(Z)} = Z^2+(3+8x)Z+2x$ is correct error locator polynomial. Now we find roots of this polynomial by substituting all the elements of Galois ring so we get $Z_1 = 5+8x$ and $Z_2 = 1+2x$ are roots of $\overline{\rho(Z)}$. Now from the elements of G_8, $\beta^0 = 1, \beta^1 = 2+8x, \beta^2 = 2+4x, \beta^3 = 3+x, \beta^4 = 8, \beta^5 = 7+x, \beta^6 = 7+5x, \beta^7 = 6+8x$ Choose those elements X_i such that $X_i - Z_i$ are zero divisors in $\mathcal{R}$ $=GR(9,2)$, So select $X_1 = \beta^1 = 2+8x$ such that $X_1 - Z_1 = 2+8x-5-8x = 6$ is zero divisor in $GR(9,2)$ similarly we take $X_2 = \beta^6 = 7+5x$ such that $X_2 - Z_2 = 6+3x$ is zero divisor in $GR(9,2)$. Therefore X_1 and X_2 are correct error location numbers and powers of β shows error positions, Hence error occur at position 1 and at position 6. To find correct elementary symmetric function we take $(Z-X_1)(Z-X_2) = (Z-(2+8x))(Z-(7+5x)) = Z^2+(5x)Z+(6+8x)$ this implies that $\overline{\sigma_1} = 5x$ and $\overline{\sigma_2} = 6+8x$.

Step4: Computing the error magnitudes,

To calculate error magnitudes we apply Forney's procedure to Syndrome and elementary symmetric function as like in above example so we get error magnitude Y_1=3 and $Y_2 = 6$. Therefore the error is $e = (0\ 3\ 0\ 0\ 0\ 0\ 6\ 0)$ Hence the corrected codeword $v = \widetilde{r} - e = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$.

3.4 Computationally Decoding of RS and BCH Codes over $\mathbf{Z}_{p^m}$

Decoding of BCH and RS codes manually is very difficult and time consuming. If we have higher order integer residue ring then it is very complicated to calculate manually the elements of Galois ring and decoding of RS and BCH codes. So to resolve this problem we construct an algorithm to decode the received message over the Z_{p^m}.

3.4.1 Calculation the Elements of Galois Ring in C#

By using this algorithm we can find the all elements of Galois ring in few seconds So this help us to fast decoding of RS and BCH codes. The algorithm is as,

```
// INPUT THE VALUE OF q AND IRREDUCIBLE POLYNOMIAL
int qVal = Convert.ToInt32(q.Text);
string polnolq = ppq.Text;
Polynomial polynomialppq = new Polynomial(ppq.Text);
int maxInd = polynomialppq.MaxIndex;
int FinalQVal = (int)Math.Pow(qVal, maxInd - 1);
string[,] qaloisRing = new string[FinalQVal, qVal];
//Find the elements of Galois Ring
for (int i = 0; i < FinalQVal; i++)
{
for (int j = 0; j < qVal; j++)
{
if (i == 0)
{
qaloisRing[i, j] = j.ToString();
```

```
    }
    else
    {
    Polynomial p1 = new Polynomial("x" + qaloisRing[(i - 1), j]);
    string[] strArr = p1.ToString().Split('+');
    for (int strCounter = 0; strCounter < strArr.Length; strCounter++)
    {
    if (strArr[strCounter].Contains(qVal.ToString()))
    {
    string a = strArr[strCounter].Substring(0, strArr[strCounter].IndexOf("x"));
    int ii = strArr[strCounter].IndexOf("x");
    string b = strArr[strCounter].Substring(ii);
    if (a == qVal.ToString())
    {
    Polynomial Coefficient = new Polynomial("x") * new Polynomial(b);
    strArr[strCounter] = Coefficient.ToString();
    }
    }
    }
    for (int strCounter = 0; strCounter < strArr.Length; strCounter++)
    {
    if (strArr[strCounter].Contains("x^" + maxInd.ToString()))
    {
    strArr[strCounter] = strArr[strCounter].Replace("x" + maxInd, "x^" + (maxInd
- 1) + "x").ToString();
    }
    }
    qaloisRing[i, j] = new Polynomial(string.Join("+", strArr)).ToString();
    }
    }
```

```
}
int intCounter = 0;
Result.Text = "";
for (int i = 0; i < FinalQVal; i++)
{
for (int j = 0; j < qVal; j++)
{
Result.Text += "Pol " + (intCounter++) + "=" + qaloisRing[i, j] + "\t\t";
```

3.4.2 Computationally Find the inverse of element in Galois Ring

It is not possible that to find the inverse of each element in Galois ring because in ring there may be some zero divisors so each element of Galois ring is not invertible. So it is very difficult to find the inverse of element manually in Galois ring that's why we construct an algorithm to find inverse of the elements of Galois ring in few seconds so this algorithm save our time and help us to decode the received vector. Algorithm is as,

```
int qVal = Convert.ToInt32(q.Text);
string inverseVal = inverse.Text;
string polnolq = ppq.Text;
Polynomial polynomialppq = new Polynomial(ppq.Text);
int maxInd = polynomialppq.MaxIndex;
int FinalQVal = (int)Math.Pow(qVal, maxInd - 1);
string[,] qaloisRing = new string[FinalQVal, qVal];
string maxPolinomial = new Polynomial("x" + maxInd).ToString();
string[] strIrreducible = new Polynomial(polnolq).ToString().Split('+');
string[] minPolynomialArr = (new Polynomial(polnolq) - new Polynomial("x^" +
maxInd)).ToString().Split('+');
for (int j = 0; j < minPolynomialArr.Length; j++)
{
int index = minPolynomialArr[j].IndexOf("x");
string npn = new Polynomial("x^" + j).ToString();
```

```
    string piece = string.Empty;
    if (index != -1)
    {
    piece = minPolynomialArr[j].Substring(minPolynomialArr[j].LastIndexOf('x'));
    }
    if (index == 0)
    {
    minPolynomialArr[j] = (qVal - 1) + new Polynomial(minPolynomialArr[j]).ToString();
    }
    else if (index > 0 && npn != "1")
    {
    minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + npn;
    }
    else if (index > 0 && npn == "1")
    {
    minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + piece;
    }
    else if (index == -1)
    {
    minPolynomialArr[j] = (qVal - Convert.ToInt32(minPolynomialArr[j])).ToString();
    }
    }
    for (int k = 0; k < minPolynomialArr.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = minPolynomialArr[k];
```

```
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = minPolynomialArr[k];
    minPolynomialArr[k] = ((Convert.ToInt32(minPolynomialArr[k].Substring(0, index))
% qVal)).ToString() + "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    minPolynomialArr[k] = (Convert.ToInt32(minPolynomialArr[k]) % qVal).ToString();
    }
    }
    }
    string minPolynomial = string.Join("+", minPolynomialArr);
    Polynomial modPolynomial = new Polynomial(minPolynomial);
    Polynomial maxpPolynomial = new Polynomial(maxPolinomial);
    for (int i = 0; i < FinalQVal; i++)
    {
    for (int j = 0; j < qVal; j++)
    {
    if (i == 0)
    {
```

```
qaloisRing[i, j] = j.ToString();
}
else
{
Polynomial p1 = new Polynomial("x" + qaloisRing[(i - 1), j]);
string[] strArr = p1.ToString().Split('+');
for (int strCounter = 0; strCounter < strArr.Length; strCounter++)
{
if (strArr[strCounter].Contains(qVal.ToString()))
{
string a = strArr[strCounter].Substring(0, strArr[strCounter].IndexOf("x"));
int ii = strArr[strCounter].IndexOf("x");
string b = strArr[strCounter].Substring(ii);
if (a == qVal.ToString())
{
Polynomial Coefficient = new Polynomial("x") * new Polynomial(b);
strArr[strCounter] = Coefficient.ToString();
}
}
}
for (int strCounter = 0; strCounter < strArr.Length; strCounter++)
{
if (strArr[strCounter].Contains("x^" + maxInd.ToString()))
{
strArr[strCounter] = strArr[strCounter].Replace("x^" + maxInd, "x^" + (maxInd
- 1) + "x").ToString();
}
}
qaloisRing[i, j] = new Polynomial(string.Join("+", strArr)).ToString();
}
```

```
    }
    }
    Result.Text = "";
    bool bResult = true;
    for (int i = 0; i < FinalQVal; i++)
    {
    for (int j = 0; j < qVal; j++)
    {
    Polynomial ppppp = new Polynomial(inverseVal) * new Polynomial(qaloisRing[i,
j]);
    string[] inverserArr = ppppp.ToString().Split('+');
    for (int l = 0; l < inverserArr.Length; l++)
    {
    if (maxInd == new Polynomial(inverserArr[l]).MaxIndex)
    {
    string myString = inverserArr[l];
    int index = myString.IndexOf("x");
    if (index > 0)
    {
    inverserArr[l] = (new Polynomial(inverserArr[l].Substring(0, index)) * modPolynomial).To
    }
    else
    {
    inverserArr[l] = (modPolynomial).ToString();
    }
    }
    }
    Polynomial p4 = new Polynomial(string.Join("+", inverserArr));
    inverserArr = p4.ToString().Split('+');
    for (int k = 0; k < inverserArr.Length; k++)
```

```
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = inverserArr[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = inverserArr[k];
    inverserArr[k] = ((Convert.ToInt32(inverserArr[k].Substring(0, index)) % qVal)).ToString
+ "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    inverserArr[k] = (Convert.ToInt32(inverserArr[k]) % qVal).ToString();
    }
    }
    }
    string pinverse = new Polynomial(string.Join("+", inverserArr)).ToString();
    if (pinverse == "1")
    {
    Result.Text += Environment.NewLine + "Inverse = " + qaloisRing[i, j];
```

```
bResult = false;
break;
}
}
}
if (bResult)
{
Result.Text += Environment.NewLine + "It is not invertible.";
}
}
}
}
```

3.4.3 Computationally Calculation of Maximum Cyclic Subgroup of Group of units of Galois Ring

For decoding of BCH codes over Galois ring we need a maximum cyclic subgroup of group of units of Galois ring because every element of Galois ring is not invertible and we decode the BCH code with the help of maximum cyclic subgroup. It is very time consuming process of construction of G_s manually and transmission of data is very slow. So we develop the algorithm in computer language which give us maximum cyclic subgroup in few seconds and with this program we transmit and receive the data very fast. Program in C# is as follows,

```
int qVal = Convert.ToInt32(q.Text);
 int nVal = Convert.ToInt32(nValText.Text);
 string polnolq = ppq.Text;
 Polynomial polynomialppq = new Polynomial(ppq.Text);
 int maxInd = polynomialppq.MaxIndex;
 int FinalQVal = (int)Math.Pow(qVal, maxInd - 1);
 string[] recmatrix = new string[nVal];
 string maxPolinomial = new Polynomial("x^" + maxInd).ToString();
 string[] strIrreducible = new Polynomial(polnolq).ToString().Split('+');
```

```
    string[] minPolynomialArr = (new Polynomial(polnolq) - new Polynomial("x^" +
maxInd)).ToString().Split('+');
    for (int j = 0; j < minPolynomialArr.Length; j++)
    {
    int index = minPolynomialArr[j].IndexOf("x");
    string npn = new Polynomial("x^" + j).ToString();
    string piece = string.Empty;
    if (index != -1)
    {
    piece = minPolynomialArr[j].Substring(minPolynomialArr[j].LastIndexOf('x'));
    }
    if (index == 0)
    {
    minPolynomialArr[j] = (qVal - 1) + new Polynomial(minPolynomialArr[j]).ToString();
    }
    else if (index > 0 && npn != "1")
    {
    minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + npn;
    }
    else if (index > 0 && npn == "1")
    {
    minPolynomialArr[j] = (-Convert.ToInt32(minPolynomialArr[j].Substring(0, index))
+ qVal).ToString() + "" + piece;
    }
    else if (index == -1)
    {
    minPolynomialArr[j] = (qVal - Convert.ToInt32(minPolynomialArr[j])).ToString();
    }
    }
```

```
    for (int k = 0; k < minPolynomialArr.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x" + pc).ToString();
    string myString = minPolynomialArr[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = minPolynomialArr[k];
    minPolynomialArr[k] = ((Convert.ToInt32(minPolynomialArr[k].Substring(0, index))
% qVal)).ToString() + "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    minPolynomialArr[k] = (Convert.ToInt32(minPolynomialArr[k]) % qVal).ToString();
    }
    }
    }
    string minPolynomial = string.Join("+", minPolynomialArr);
    Polynomial modPolynomial = new Polynomial(minPolynomial);
    Polynomial maxpPolynomial = new Polynomial(maxPolinomial);
```

```
int MaxPolyValue = 0;
var list = new List<KeyValuePair<string, string>>();
list.Add(new KeyValuePair<string, string>(maxpPolynomial.ToString(), modPolynomial.ToS
{
maxpPolynomial = maxpPolynomial * new Polynomial("x^1");
Polynomial p1 = modPolynomial * new Polynomial("x^1");
string[] p1Arr = p1.ToString().Split('+');
for (int i = 0; i < p1Arr.Length; i++)
{
if (p1Arr[i].Contains("x^" + maxInd))
{
int index = p1Arr[i].IndexOf("x");
string sstr = p1Arr[i].Replace("x^" + maxInd, "+" + new Polynomial(minPolynomial));
string[] strr = sstr.Split('+');
p1Arr[i] = string.Empty;
if (string.IsNullOrEmpty(strr[0]))
{
strr[0] = "1";
}
for (int j = 1; j < strr.Length; j++)
{
strr[j] = (new Polynomial(strr[0]) * new Polynomial(strr[j])).ToString();
if (string.IsNullOrEmpty(p1Arr[i]))
{ p1Arr[i] += strr[j]; }
else
{
p1Arr[i] += "+" + strr[j];
}
}
}
```

```
    }
    Polynomial p2 = new Polynomial(string.Join("+", p1Arr));
    string[] strp2 = p2.ToString().Split('+');
    for (int k = 0; k < strp2.Length; k++)
    {
    for (int pc = 0; pc < maxInd; pc++)
    {
    string npn = new Polynomial("x^" + pc).ToString();
    string myString = strp2[k];
    int index = myString.IndexOf("x");
    string piece = "";
    if (index > 0 && index <= (myString.Length - 1))
    {
    piece = myString.Substring(myString.LastIndexOf('x'));
    if (piece == npn)
    {
    string nsigmastr = strp2[k];
    strp2[k] = ((Convert.ToInt32(strp2[k].Substring(0, index)) % qVal)).ToString()
+ "" + npn;
    }
    }
    else if (index == 0)
    {
    }
    else if (index == -1)
    {
    strp2[k] = (Convert.ToInt32(strp2[k]) % qVal).ToString();
    }
    }
    }
```

```
    modPolynomial = new Polynomial(string.Join("+", strp2));
    list.Add(new KeyValuePair<string, string>(maxpPolynomial.ToString(), modPolynomial.ToS
    if (modPolynomial.ToString() == "1")
    {
    MaxPolyValue = new Polynomial(maxpPolynomial.ToString()).MaxIndex;
    break;
    }
    }
    int dVal = MaxPolyValue / nVal;
    string[] Gn = new string[nVal];
    Gn[0] = "x^" + MaxPolyValue;
    for (int i = 1; i < nVal; i++)
    {
    Gn[i] = new Polynomial("x^" + (dVal * i)).ToString();
    }
    Result.Text = "";
    Result.Text += "G" + nVal + " Maximum Cyclic Subgroup:" + Environment.NewLine;
    Result.Text += Environment.NewLine;
    for (int i = 0; i < nVal; i++)
    {
    if (list.Where(n => n.Key == Gn[i]).FirstOrDefault().Value != null)
    {
    Gn[i] = list.Where(n => n.Key == Gn[i]).FirstOrDefault().Value;
    }
    Result.Text += "Polynomial" + i + "=> " + Gn[i] + Environment.NewLine + " "
+ Environment.NewLine;
    }
    Result.Text += Environment.NewLine;
    }
   }
```

Conclusion

In second chapter of my dissertation we have done decoding of BCH codes over Galois field by using Barlekamp Massey algorithm and extend this algorithm to simplified algorithm for binary BCH codes. We conclude that decoding of BCH codes over binary field is required half steps for computing as compare to decoding over non binary BCH codes. We also done decoding of BCH codes over Z_p computationally. So we develop an algorithm to check the received word have errors or not and to calculate the syndromes for determining errors. These algorithms help us for data transmission very fastly and we can correct errors during transmission of codes very fast. Correcting errors is even more important when transmitting data that have been encrypted for security. In chapter three we have done decoding of RS and BCH codes over integer residue ring computationally. In this correspondence we described the decoding procedure of BCH and RS codes over integer residue ring with the help of modified Barlekamp Massey algorithm. The difference between modified Barlekamp Massey algorithm and original Barlekamp Massey it always allows to update the minimal solution at step n by making use of previous solution whose discrepancies can even be non units in $\mathcal{R}$ and for minimality of new solution one extra step had to be introduced. The solution provided by the modified Barlekamp Massey algorithm is not unique in general, then in Galois field reciprocal of the polynomial may not be the correct error locator polynomial. The complexity of decoding procedure of BCH and RS codes over Galois field and Galois ring essencially the same. We solve the modified Barlekamp Massey algorithm computationally which remove the complexity of decoding procedure. By using the computational approach transmission of data and decoding of RS and BCH codes is very fast and communication channel for data transmission code work fastly and correct errors during transmission.

Bibliography

[1] A.A. Andrade and R. Palazzo Jr., Construction and decoding of BCH codes over finite rings, *Linear Algebra Appl.*, 286, (1999), 69-85.

[2] A.A. Andrade and R. Palazzo Jr., Linear codes over finite rings, *TEMA Tend. Mat. Apl. Comput.*, 6(2) (2005), 207-217.

[3] I.F. Blake, Codes over certain rings, *Inform. Control.*, 20, (1972), 396-404.

[4] I.F. Blake, Codes over integer residue rings, *Inform. Control.*, 29, (1975), 295-300.

[5] G.D. Forney Jr., On decoding BCH codes, *IEEE Trans. Inform. Theory*, IT-11(4), (1965), 549-557.

[6] J.C. Interlando, R. Palazzo Jr., M. Elia, On the decoding of Reed-Solomon and BCH codes over integer residue rings, *IEEE Trans. Inform. Theory,* IT-43 (1997) 1013-1021.

[7] A.V. Kelarev., Ring Constructions and Applications, *World Scientific, River Edge*, New York (2002).

[8] A.V. Kelarev, Algorithms for computing parameters of graph-based extensions of BCH codes, *Journal of Discrete Algorithms, 5* (2007), 553-563.

[9] B.R. McDonald, Finite rings with identity, *Marcel Dekker*, New York (1974).

[10] S.R. Nagpaul and S.K. Jain, Topics in applied abstract algebra, *The Brooks/ Cole Series in Advanced Mathematics*, (2004).

[11] T. Shah, Amanullah and A.A Andrade, A decoding procedure which improves code rate and error corrections, *Journal of Advanced Research in Applied Mathematics*, 4(4) (2012), 37-50

[12] T. Shah, M. Khan and A. A. Andrade, A decoding method of an n length binary BCH code through $(n+1)n$ length binary cyclic code, *An. Acad. Bras. Cienc* 85(3) (2013), 345-354.

[13] P. Shankar, On BCH codes over arbitrary integer rings, *IEEE Trans. Inform. Theory*, IT-25(4), (1979), 480-483.

[14] E. Spiegel, Codes over $\mathbb{Z}_m$, *Inform. Control*, 35 (1977), 48-51.

[15] E. Spiegel, Codes over $\mathbb{Z}_m$, Reviseted, *Inform. Control*, 37 (1978), 100-104.

[16] C. E. Shannon, *A mathematical theory of communication*, Bell syst. Tech. J., 27 (1948), 379−423 (part I); 623−656 (part II).

[17] R. W. Hamming, *Error detecting and error correcting codes*, Bell syst. Tech. J., 29 (1950), 147−160.

[18] David F. Anderson, Multiplicative Ideal Theory in Commutative Algebra 2006, pp 21-37.

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