MATHEMATICAL STRUCTURES FOR DECODING PROJECTIVE GEOMETRY CODES

by

LOIS ELLEN WRIGHT

A thesis submitted for the Degree of Doctor
of Philosophy in the Faculty of Engineering
University of London

DEPARTMENT OF ELECTRICAL ENGINEERING,
IMPERIAL COLLEGE OF SCIENCE AND TECHNOLOGY,
UNIVERSITY OF LONDON

May, 1977.

SYNOPSIS

In recent years, a significant amount of research in the field of Coding Theory has led to the development of error-correcting codes for communication systems. Such codes make it possible to correct and/or detect a large percentage of the errors which may occur during the transmission of information through a communication channel. However, in general, these codes are not used in existing communication systems due to the inherent complexity of their associated decoders. The objective of this thesis is to overcome this difficulty for a particular set of codes by developing a simplified decoding method which provides error-protection in a communication system without the associated cost of a complex decoder. In particular, a simplified decoding method for certain Projective Geometry codes is presented. The decoding algorithm, based on the results of an extensive analysis of the mathematical structure of Projective Geometries, significantly decreases the complexity of the standard decoder of the Projective Geometry Codes. Moreover, the decoder can be implemented simply and with little additional cost to the system.

The thesis begins with an introduction to basic Coding Theory and Algebra concepts. This is followed by a detailed study of Majority Logic Decoding and Projective Geometry (PG) Codes. The mathematical structures, or orbits, used for decoding are developed and analysed. The simplified decoder is applied to the (63,41) order-3 PG code and the (255,218) order-5 PG code. A comparison between the standard Majority Logic Decoder and the proposed orbit decoder is made. Modifications to the orbit decoder for increasing the distance of the codes are described. The thesis is concluded with a discussion of further research questions and conclusions drawn from this work.

ACKNOWLEDGEMENTS

I wish to express my appreciation to my supervisor, Dr. Laurie Turner, for the discussions of, and the excellent suggestions to, the content of my research work.

Financial assistance was received from both the National Research Council of Canada and Imperial Oil Limited.

TABLE OF CONTENTS

Part I

Part II

LIST OF TABLES

Table Page

LIST OF FIGURES

PART I
CHAPTER 1: INTRODUCTION

1.1 General Introduction

Data communication systems are rapidly becoming more and more prevalent as technology provides the increasingly sophisticated equipment necessary for their existence. The majority of communication networks rely heavily upon digital computers and their peripheral devices, which have very low tolerance to errors in received information. Consequently, the application of Coding Theory, as a means of guaranteeing the reliability of transmitted information, independent of the parameters of the machinery, has become a practical way of overcoming errors within these systems.

Previously, error-correcting codes were restricted to highly specialized areas such as space communication or remote control of machinery, where the occurrence of an error could be disasterous. Coding was not introduced into less specialized systems because of two major factors:

1) the complexity of existing decoding schemes, and

2) the cost of the associated circuitry. In the past few years both of these problems have been reduced significantly. Much research has gone into both the simplification of existing codes and the development of new, less complicated codes. And, of even more consequence, major technological advances have decreased dramatically the cost and size of solid state electronic devices. Together, these have made coding a feasible solution to errors within a communication system.

1.2 Communication Channel and Error Correction

A communication system consists of five major components, the source, the encoder, the channel, the decoder and the destination, as shown in the flow diagram in Figure 1.1.1.

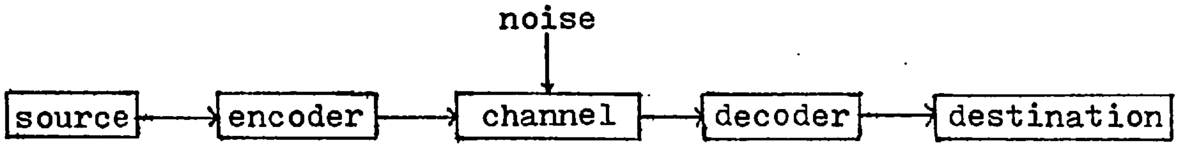


Figure 1.1.1 Communication System

If no noise were injected into the system, codes would be unnecessary. However, in all realistic communication channels, noise, to varying degrees, is a factor.

In most instances, the source consists of binary or decimal digits grouped in such a way that a source alphabet can be defined. A message, consisting of letters from the source alphabet, is forwarded to the encoder, which transforms it into a signal acceptable to the channel. This is typically in the form of electrical pulses, restricted by such channel parameters as power, bandwidth and duration. The message is then input to the channel where it is subject to errors from channel noise. The message, possibly perturbed by errors, is received by the decoder. Based on this input, the decoder must decide what message has been sent. The decoded message is then passed on to the destination.

Shannon has shown that a communication channel, as depicted above, has a capacity for the transmission of information.[45] In fact, if the rate of the source is less than the capacity of the channel, then a set of signals can be chosen for the encoder such that the probability of error by the decoder can be made arbitrarily small. The aforementioned rate of the code is defined as the ratio of the number of information digits transmitted to the total number of digits transmitted. The ratio of the differnce between the total number of digits and the number of information digits, to the total number of digits, is the redundancy of a code. The Shannon theory states that reliability in transmission is possible but does not suggest how to obtain such a system. Coding Theory shows that through the addition of non-information or redundant digits to a message, the theoretical degree of error-protection can be obtained but only with an associated decrease in the rate at which information can be transmitted.

The channel depicted in Figure 1.1.1 is a one-way transmission system. Two-way channels exist also and are used in situations where error-detection is suitable. In such an instance, a request for retransmission can be sent to the source and consequently the message retransmitted. Such systems will not be discussed in the following as their efficiency, in general, is limited since a short code is inefficient in correction of errors and a long code requires too much retransmission time. However, a combination of error-correction and error-detection holds much potential as a communication system and will be discussed later.

1.3 Types of Codes

Codes can be divided into two basic categories, block and tree codes.

Block codes are so named because the encoder accepts a block of k information digits at a time from the stream of information digits produced by the source. These digits are encoded into a block of n channel symbols, $n > k$. This block, called a codeword, is transmitted through the channel where it may be corrupted by noise. It is then decoded as a block of n digits by the decoder. The number n is called the block length.

The second kind of code, the tree code, does not break the information sequence into independent blocks, but rather, operates on the input as a continuous stream. Each semi-infinite information sequence is associated with a larger number of digits than were input. Based on each set of k_0 information digits, k_0 small, and all the previous information digits, a n_0-symbol section of code is emitted, $n_0 > k_0$. The term 'tree' refers to the convenient tree graph description of such codes. The most common and most researched tree code is the convolution code, which consists of shifts of a basic string of n_0 digits.

Both block and tree codes have the same basic error-correcting properties as well as limitations and restrictions imposed by rate and code complexity. Block codes have, with their more well defined mathematical structure, a correspondingly richer research history. This thesis is concerned with a subset of block codes.

1.4 Thesis Outline

The thesis consists of three distinct parts. The first of these provides the general Coding Theory and Mathematical background required in the rest of the thesis. Part II deals with Projective Geometry Codes and a new decoding method for these codes based on the mathematical structure of the null space. The final part considers further research questions and the conclusions.

More explicitly, in Part I, Chapter 2 presents a general discussion of Coding Theory, emphasizing good codes that have been developed, their decodability and the problems associated with them. Also, the general objectives of the thesis are detailed. Chapter 3 develops a particular subset of block codes, viz. Majority Logic Decodable Codes. A discussion of these codes, as presented by Reed[41] and Muller[35], is given. Also in this chapter, an extensive discussion of the mathematical properties of Projective Geometries and of their associated Majority Logic Decodable codes is presented. Part I is concluded with a discussion of several methods developed to simplify Majority Logic Decoding.

Part II, Chapters 4 through 7, presents a detailed study of the mathematical structure of the null space of Projective Geometry Codes, illustrating how knowledge of such structure can simplify decoding. In Chapter 4, a discussion of a 1966 paper by Yamamoto et al[58] provides the basis for the development of the orbit structures used for the proposed decoding method. Chapter 5 contains an extensive explanation of the orbits of, and the associated decoder for, the order-3 PG(5,2) codes. A comparison is made between this non-orthogonal decoder and the standard Majority Logic Decoder for the code. The next chapter consists of a similar discussion of the order-5 PG code over PG(7,2) emphasizing that it is a logical extension of the PG(5,2) structure presented in Chapter 5. The seventh chapter considers the feasibility of adding error-detection to the orbit decoder to detect more errors than are correctable by the standard Majority Logic Decoder and an analysis is made of the amount of information obtained from this addition.

Part III considers the many further research questions which have arisen as a result of this work. Finally, the conclusions which are suggested by this study are presented.

CHAPTER 2: BASIC THEORY AND FUNDAMENTAL CODES

2.1 Introduction

This chapter provides the reader with the basic algebraic Coding Theory background necessary for an understanding of the ideas and concepts presented in the thesis. The first section is a brief introduction to the algebra used in developing codes. Next follows a discussion of two subclasses of the class of block codes: 1) linear, or group, codes and 2) cyclic codes. The defining properties and outstanding features of these codes are discussed. Several parameters used in evaluating algebraic codes are examined. Some of the most common block codes are reviewed with emphasis being placed on error-correcting properties, ease of decodability and minimum distance of the codes. Finally, the objectives of the thesis are given.

We note that as most equipment today is binary, nonbinary results are not emphasized and that in Part II, the results refer only to the binary case.

2.2 Basic Algebra for Coding Theory

This section is a concise outline of the Algebra required in the thesis. It consists of a review of Group Theory, Vector Spaces,. Matrix Theory, Polynomial Rings and Galois Fields. A more detailed discussion of these topics is given by Birkhoff and MacLane in A Survey of Modern Algebra[3].

2.2.1 Group Theory

A set of elements a, b, c,... and an operation * is a group if it satisfies the following four axioms:

1) Closure: If a and b are elements of the set, then $a*b$ is an element.

2) Associativity: For a, b and c in the set,

$$a*(b*c) = (a*b)*c.$$

3) Identity: There is an element e, the identity element, such that for all a in the set,

$$a*e = e*a = a.$$

4) Inverse: Every element a of the set has an inverse a^{-1} such that

$$a*a^{-1} = a^{-1}*a = e.$$

A group is Abelian or commutative if, for all a and b in the group

$$a*b = b*a.$$

In the following, we represent the group operation as multiplication rather than use the operator *.

A subset H of elements of a group G is a subgroup of G if H satisfies the above axioms under the group operator. The order of a subgroup (group) is the number of elements in the subgroup (group).

If g is any element in a finite group G, a sequence $g, g^2, g^3 \ldots$ can be formed. Then, since G is finite, there exist integers j and i, $j > i$, such that

$$g^i = g^j = g^i g^{j-i}.$$

This implies that $g^{j-i} = e$. The order of g is the least positive integer m for which $g^m = e$. The set of elements $g, g^2, \ldots, g^m = e$ is a subgroup, specifically the cyclic subgroup generated by g.

A right coset (left coset) of a subgroup H of the group G is the subset of G obtained by taking any fixed element g of G and multiplying all the elements $h_1, h_2, \ldots$ of H by g on the right (left) to form h_1g, h_2g, ... (gh_1, gh_2,...). The number of elements in any coset is simply the order of the subgroup H. No element of G is in more than one coset of H. Two elements g and g' of G are in the same right coset of H if and only if $g'g^{-1}$ is an element of H. The number of distinct cosets of G formed from the subgroup H is the index of G over H. A subgroup H is normal if for any h in H and g in G, $g^{-1}hg$ is H. In a normal subgroup each left coset is a right coset and vice versa. We deal primarily with Abelian subgroups where a left coset is always a right coset and hence each subgoup normal. The factor group of the group G over the subgroup H, G/H, is formed by defining a multiplication operator for cosets. If $\{g\}$ represents the coset containing the element g, then, the factor group has for elements, cosets, and an operator such that for $\{g_1\}$, $\{g_2\}$ cosets,

$$\{g_1\}\{g_2\} = \{g_1g_2\}.$$

The identity of the factor group G/H is the subgroup H = $\{e\}$.

A transformation Z: S → T, from a non-empty set S to a set T is a rule which assigns to each element p in S a unique image element pZ in T. Thus, a transformation is merely a function from S to T.

A permutation is a one to one transformation of a finite set into itself. Permutations which give a circular arrangement of the symbols permuted are called cyclic permutations or cycles. For example if

$$1 \rightarrow 2,\ 2 \rightarrow 3,\ 3 \rightarrow 4,\ 4 \rightarrow 1,$$

then this can be written as the cycle (1 2 3 4). Any permutation Z can be written as a product of cycles acting on disjoint sets of symbols.

A binary relation R on a set S relates any two elements a and b of the set. Either a is in relation to b, aRb, or a is not in relation to b, $a \not R b$. A relation R which has the reflexive (aRa), symmetric (aRb implies bRa) and transitive (aRb, bRa imply aRc) properties for all a, b and c in the set S is called an equivalence relation. If a is any element of S, we denote by R(a) the set of all elements b equivalent to a, that is b is in R(a) if and only if bRa. Two such R-subsets are either identical or have no elements in common. A partition π of a set S is any collection of subsets A,B,... of S such that each element of S belongs to one and only one of the subsets of the collection. The collection of all the R-subsets form a partition of S.

If a group G of transformations of a set S exists, then G defines an equivalence relation and hence a partition on S by the rule $a \sim_G b$ (a is G equivalent to b) if b = g(a) for some g in G and a, b in S. The G-equivalence class determined by an element a of S is the set $Ga = \{g(a) | g \in G\}$ and this is called the G-orbit or orbit of a in S.

We proceed now to discuss a set with two operators. A ring R is a set of elements a, b, c,... with two operations, addition a+b and multiplication ab, satisfying the following axioms:

1) R is an Abelian group under addition.
2) Closure: For a, b in R, ab is in R.
3) Associativity: For a,b,c in R, a(bc) = (ab)c.
4) Distributivity: For a,b,c in R, a(b+c) = ab+ac and (b+c)a = ba+ca.

The ring R is communative if ab = ba for any two elements a and b in R.

A communative ring with a multiplicative identity and for which each non-zero element has a multiplicative inverse, is called a field. A non-communative ring with an inverse for each non-zero element is a division ring or skew field.

We return to the discussion of rings and fields later.

2.2.2 Vector Spaces

A set V of elements is a vector space over a field F for which the following axioms hold:

1) V is an Abelian group under addition.
2) For any vector $\underline{v}$ in V and c a field element or scalar, $c\underline{v}$ is defined and in V.
3) For $\underline{u}$ and $\underline{v}$ in V, c a scalar, $c(\underline{u}+\underline{v}) = c\underline{u} + c\underline{v}$.
4) For $\underline{v}$ in V, c and d scalars, $(c+d)\underline{v} = c\underline{v} + d\underline{v}$.
5) For $\underline{v}$ in V, c and d scalars, $(cd)\underline{v} = c(d\underline{v})$, $1\underline{v} = \underline{v}$.

A set A of elements over a field F is a linear associative algebra if:

1) A is a vector space over F.
2) Closure: For $\underline{u}$ and $\underline{v}$ in A, $\underline{uv}$ is in A.
3) Associativity: For $\underline{u}$,$\underline{v}$ and $\underline{w}$ in A, $(\underline{uv})\underline{w} = \underline{u}(\underline{vw})$.
4) Distributivity: For c and d in F, $\underline{u}$,$\underline{v}$,$\underline{w}$ in A, $\underline{u}(c\underline{v}+d\underline{w}) = c\underline{uv} + d\underline{uw}$ and $(c\underline{v}+d\underline{w})\underline{u} = c\underline{vu} + d\underline{wu}$.

An n-tuple of n field elements $a_1,a_2,\ldots,a_n$ is the ordered set $(a_1,a_2,\ldots,a_n)$. Addition of n-tuples is defined by $(a_1,a_2,\ldots,a_n) + (b_1,b_2,\ldots,b_n) = (a_1+b_1,\ldots,a_n+b_n)$. Multiplication by a scalar is as follows:

$$c(a_1,\ldots,a_n) = (ca_1,\ldots,ca_n).$$

With these two operations defined, it is easy to show that the set of all n-tuples over a field is a n-dimensional vector space. A linear combination of k vectors or n-tuples $\underline{v}_1$, $\underline{v}_2,\ldots,\underline{v}_k$ is the vector $\underline{u}$

$$\underline{u} = a_1\underline{v}_1 + \ldots + a_k\underline{v}_k,$$

where the a_i are scalars. The set of all linear combinations of a set of vectors $\underline{v}_1,\underline{v}_2,\ldots,\underline{v}_k$ of a vector space V is a subspace of V. The vectors $\underline{v}_1,\underline{v}_2,\ldots,\underline{v}_k$ are linearly dependent if and only if there are scalars $a_1,a_2,\ldots,$ a_k for which

$$a_1\underline{v}_1 + a_2\underline{v}_2 + \ldots + a_k\underline{v}_k = \underline{0}.$$

A set of vectors which are not linearly dependent are linearly independent. A set of vectors spans a subspace if every vector in the subspace is a linear combination of this set. The least number of independent vectors which spans a space is the dimension of the space. A set of k linearly independent vectors which span a k-dimensional subspace is a basis of the subspace. The inner product of two n-tuples is a scalar as follows:

$$(a_1,a_2,\ldots,a_n)\cdot(b_1,b_2,\ldots,b_n) = a_1b_1 + a_2b_2 + \ldots + a_nb_n.$$

If the inner product is zero the two vectors are orthogonal.

2.2.3 Matrix Theory

In the development of several of the codes to follow we refer to matrices. Consequently we now give some basic results from Matrix Theory. An (nxm) matrix $\underline{M}$ is an ordered set of nm field elements expressed as n rows and m columns:

$$\underline{M} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ & \vdots & & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} = \left[a_{ij}\right]$$

Each column or row can be thought of as a vector. The row (column) space of $\underline{M}$ is the set of all linear combinations of the row (column) vectors. They form a subspace of all the possible m (n)-tuples, the dimension of which is called the row (column) rank of $\underline{M}$. A matrix may be operated on by any or all of the following elementary row operations or their inverses:

1) interchange any two rows,

2) multiply any row by a non-zero field element,

3) add any multiple of one row to another.

If a matrix $\underline{M}'$ is obtained from the matrix $\underline{M}$ by elementary row operations, then $\underline{M}$ and $\underline{M}'$ have the same row space.

If, for a square (nxn) matrix, the rows are linearly independent, then the matrix is non-singular. The identity matrix has ones along the diagonal and zeros elsewhere. The transpose of a (nxm) matrix $\underline{M}$ is a (mxn) matrix, $\underline{M}^T$, with the rows and columns of $\underline{M}$ interchanged. Two matrices $[a_{ij}]$ and $[b_{ij}]$ are added element by element as follows:

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}].$$

By defining matrix multiplication of a (nxk) matrix $[a_{ij}]$ by a (kxm) matrix $[b_{ij}]$ as $[c_{ij}]$ where the element c_{ij} of the matrix is defined by

$$c_{ij} = \sum_{s=1}^{k} a_{is}b_{sj} ,$$

we can represent the elementary row operations as elementary matrices. With multiplication so defined the inverse of a matrix can be formed.

The set of all n-tuples orthogonal to a subspace V_1 of a matrix is a subspace V_2 and is called the null space of V_1. If the dimension of V_1 is k, then the dimension of the null space V_2 is (n-k).

2.2.4 Polynomial Rings

We now return to our treatment of rings. An ideal I is a subset of elements of a ring R such that:

1) I is a subgroup of the additive group of R.

2) For all a in I, r in R, ar and ra are in I. The cosets formed relative to the ideal I are called residue classes. They form a ring called the residue class ring.

The ideals below are from the algebra of all polynomials in one indeterminate over a field F. A monic polynomial has as a coefficient, one, for the highest power of X. An irreducible polynomial p(X) of degree n is not divisible by any polynomial of non-zero degree less than n. The greatest common divisor of two polynomials f(X) and g(X) is defined as the monic polynomial a(X) with greatest degree, such that a(X) divides f(X) and g(X). If a(X)=1, then f(X) and g(X) are relatively prime. Degree zero polynomials, or field elements, have inverses. However, no polynomial of positive degree has an inverse.

A subset of polynomials is an ideal if and only if every polynomial in the subset is a multiple of a fixed polynomial. The residue classes of polynomials modulo a polynomial f(X) of degree n, form a commutative linear algebra A_n of dimension n over a coefficient field. The algebra A_n plays an important role in the development of the codes discussed in Chapter 3. In this algebra, if g(X) is the monic polynomial of least degree such that the multiples of g(X) form an ideal $J = \{g(X)\}$ and p(X) is a polynomial of degree less than n and divisible by g(X), then p(X) is in J. Moreover, if p(X) is in J, then g(X) divides p(X).

Every monic polynomial g(X) which divides (X^n-1) forms an ideal with generator polynomial g(X). The null space of the ideal generated by g(X), is the ideal with generator polynomial h(X) where $g(X)\cdot h(X) = (X^n-1)$. If h(X) has degree k then the ideal generated by g(X) modulo (X^n-1) has dimension k.

2.2.5 Galois Fields

We conclude this section by dealing briefly with some properties of Galois Fields.

An extension field of degree m over a field F is formed by taking polynomials over a field F modulo an irreducible polynomial, p(X), of degree m. F is called the ground field of the extension field. For any prime number p, the residue classes of integers modulo p form a field called the Galois Field GF(p). The field of polynomials over GF(p) modulo a degree m irreducible polynomial is an extension field called the Galois Field of p^m elements, $GF(p^m)$.

Any finite field with q elements is isomorphic to GF(q). Galois fields represented as residue classes of polynomials modulo an irreducible polynomial over GF(p) have characteristic p. If α is an element of the extension field, the monic polynomial m(X) of least degree over the ground field F for which $m(\alpha) = 0$, is the minimal polynomial of α and is irreducible. If p(X) is a polynomial with coefficients in the ground field and $p(\alpha)=0$, then the minimal polynomial of α, m(X), divides p(X). Every element in the extension field of dimension m has a minimal polynomial of degree m or less. The polynomial $(X^{q-1}-1)$ has all (q-1) non-zero elements of GF(q) as roots. The polynomial (X^m-1) divides (X^n-1) if and only if m divides n. A primitive element of GF(q) has order (q-1) and every non-zero element of GF(q) can be written as a power of α. The multiplicative group of GF(q), consisting of powers of α, is cyclic. An extension field of GF(q) contains all the roots of $(X^{q^{m-1}}-X)$. Moreover, these roots form a subfield. If p(X) is an irreducible polynomial of degree m with coefficients over GF(q) and β is a root of p(X) in the extension field, then

$\beta, \beta^q, \ldots, \beta^{q^{m-1}}$ are all roots of p(X) with the same order. The order of the roots of an irreducible polynomial is the exponent to which the polynomial belongs. If a polynomial p(X) belongs to b, then p(X) divides (X^b-1) but no other polynomial of the form (X^r-1), $r<b$. An irreducible polynomial of degree m is primitive if it has a primitive element as a root. A degree m polynomial is primitive if and only if it belongs to (q^m-1) or, if and only if it divides (X^r-1) for no r less than (q^m-1).

2.3 Linear or Group Codes

All the codes, save one, discussed in the thesis are members of the class of linear codes. In the following we assume that the code symbols are elements of a finite field and that q is a power of a prime, where for most cases q is 2. An excellent and more detailed treatment of the remaining topics in this chapter can be found in both Algebraic Coding Theory by E. R. Berlekamp[1], and in Error-Correcting Codes by W. W. Peterson and E. J. Weldon[37].

A linear block code over GF(q) of block length n and dimension k is a k-dimensional subspace V of the n-dimensional vector space W of all n-tuples over GF(q). This subspace is called a (n,k) linear block code. These codes are also called group codes because the k-dimensional subspace V forms an Abelian subgroup over the prime field. Vector space theory provides a simple characterization of these codes. A k-dimensional subspace has a set of k basis vectors. These can be written as the k rows of a (kxn) matrix $\underline{G}$, called the generator matrix of the (n,k) code. The set of codewords is then the row space of the matrix $\underline{G}$. The number of such linear combinations is q^k, which is simply the number of ways of selecting the q possible coefficients of each of the k possible basis vectors.

An alternative description of the (n,k) code is given by the (n-k)-dimensional null space V' of the subspace V. A matrix $\underline{H}$ of rank (n-k) with row space V' can be formed. Then, the (n,k) code is defined as the set of vectors $\underline{v}$, orthogonal to $\underline{H}$, which is to say $\underline{v}$ and any row of $\underline{H}$ have an inner product of zero, or in terms of matrices,

$$\underline{vH}^T = \underline{0}.$$

This equality can be expressed as (n-k) independent equations:

$$\sum_j v_j h_{ij} = 0, \quad i=1,2,\ldots,(n-k)$$

where v_j is the j-th component in the n-tuple $\underline{v}$, and h_{ij} is the element from the i-th row and j-th column of $\underline{H}$. Since each of these equations is, in fact, a parity check on the codeword $\underline{v}$, the matrix $\underline{H}$ is called the parity-check matrix. V' is a subspace of W and hence is also a linear code, the (n,n-k) code with null space V. V is the dual code of V' and vice versa.

Decoding of these codes is accomplished by a rearrangement of the vectors of the generating matrix $\underline{G}$. Two decoders based on this process are presented here, the second being an improvement over the first.

If the q^k code vectors are ordered as $\underline{i}_1, \underline{i}_2, \ldots, \underline{i}_{q^k}$, where $\underline{i}_1$ is the all zero n-tuple, a decoding table can be formed. Row one consists of the q^k codewords, $\underline{i}_1, \ldots, \underline{i}_{q^k}$. Next, one of the remaining non-codeword n-tuples, say $\underline{m}_1$, is selected, where usually $\underline{m}_1$ is one of the most likely vectors to be received if $\underline{i}_1$, the identity, were sent. The second row of the decoding table is then $\underline{m}_1 + \underline{i}_j$, $j=1,\ldots,q^k$. This process is repeated until each possible n-tuple appears in the table, for a total of q^{n-k} rows. Each row of the table is a coset and the first element in each row, the coset leader. Although the table just formed could be used to decode, a simplified version requiring less storage is now defined. The product of the (n-k)xn parity-check matrix $\underline{H}$ and a codeword vector $\underline{v}$ is an (n-k) component vector $\underline{S} = \underline{vH}^T$ called the syndrome. A vector is a codeword if and only if it has a $\underline{0}$ syndrome. To decode in the binary case, a table is formed consisting only of the coset leader and syndrome for each of the 2^{n-k} cosets. When a vector is received, its syndrome is calculated and then found in the table. The coset leader which is associated with it is the most probable error. It is subtracted from the received codeword to give the decoder's estimate of the message sent.

A variation of the above gives step-by-step decoding. For this method, it is necessary to determine for each received vector the weight of the minimum weight element in its coset. This requires a larger table than for the previous method but can be accomplished without listing the whole table of cosets plus syndromes. The q field elements $f_1, f_2, \ldots, f_q$ are listed with zero appearing last. The vectors are ordered lexigraphically in the sense that if $(v_1, \ldots, v_n)$ and $(w_1, \ldots, w_n)$ are two vectors alike in the first (j-1) positions, but v_j follows w_j in the order of the field elements, then $(v_1, \ldots, v_n)$ follows $(w_1, \ldots, w_n)$. To decode the received vector $(v_1, \ldots, v_n)$, the weight associated with its coset is determined. The step-by-step process is then begun. The first element v_1 of the received vector is replaced by $v_1 - f_1$, $v_1 - f_2$ and so on until the weight of the coset of the altered vector is less than the original coset weight. The first element of the received vector is then replaced by $v_1 - f_i$, where f_i corresponds to a coset of lower weight. If no smaller coset weight is found v_1 remains the first element of the received vector. This procedure is repeated until the weight of the coset is zero, that is the vector represents a codeword. It is then assumed that this vector was the codeword sent. This decoding method decodes each received vector into the nearest code vector.

The above two methods are indeed valid decoders, however their drawbacks are significant. With both these decoders the storage requirements and look-up times are prohibitive for any reasonable length code. Moreover, step-by-step decoding can become a very lengthy process.

Although these methods are of interest on their own, we present them here primarily for the purpose of illustrating the relative effectiveness of the more sophisticated methods which are to follow.

2.4 Cyclic Codes

Cyclic codes are a very important class of block codes. This importance is due mainly to the high degree of mathematical structure possessed by the codes. We begin this section with a discussion of the defining properties of these codes.

If V is any subspace of the space of all n-tuples such that if $\underline{v} = (v_{n-1}, v_{n-2}, \ldots, v_0)$ is in V then $\underline{v}' = (v_0, v_{n-1}, \ldots, v_1)$, the vector resulting from cyclically shifting $\underline{v}$ one unit to the right, is also in V, then V is called a cyclic subspace. The corresponding code is a cyclic code.

The most common cyclic subspace considered in coding is a subsapce of the n-dimensional algebra A_n of polynomials modulo (X^n-1). The elements of this algebra are residue classes of polynomials, where the representative polynomials have degree less than n and each distinct polynomial of degree less than n represents a distinct residue class. For a polynomial $v(X) = v_{n-1}X^{n-1} + \ldots + v_1X + v_o$, the corresponding n-tuple is written $(v_{n-1},\ldots,v_1,v_o)$. Recalling the definition of an ideal given in Section 2.2, it is apparent that a subspace is cyclic if and only if it is an ideal. This is so because multiplication by X of a polynomial modulo (X^n-1) is equivalent to a cyclic shift of the n-tuple representation of the polynomial. Every ideal has a generator polynomial, g(X). Thus a code is specified by giving the generator polynomial g(X) where g(X) divides (X^n-1). The null space is then given by the parity-check polynomial $h(X) = (X^n-1)/g(X)$. The code generated by h(X) is equivalent to the dual code of the code generated by g(X). The polynomial f(X) is a codeword polynomial if and only if g(X) divides f(X).

An alternative description of a k-dimensional code is given by the roots $\alpha_1, \ldots, \alpha_{n-k}$ of the degree (n-k) generator polynomial g(X). For this definition, a polynomial f(X) is a code polynomial if and only if $\alpha_1, \ldots, \alpha_{n-k}$ are roots of f(X). These roots can be expressed as powers of a fixed primitive element α of order e, where $\alpha_i = \alpha^{\beta_i}$ for some β_i. If $m_i(X)$ is the minimal polynomial of α_i, then all the roots of $m_i(X)$ are given in the sequence α^{β_i}, $\alpha^{\beta_i q}$, $\alpha^{\beta_i q^2}$, ... Through an analysis of its roots, g(X) is factored into its minimal polynomials and the corresponding sets of roots are called cycle sets. We refer to this concept again in Part II.

We note briefly that these codes can easily be put into matrix form. If $g(X) = g_{n-k}X^{n-k} + \ldots + g_1X + g_o$ is the generator polynomial of a cyclic code, then the set of polynomials, $X^{k-1}g(X)$, $X^{k-2}g(X)$, ..., $Xg(X)$, $g(X)$ are all code vectors and when expressed as n-tuples, linearly independent. Thus, the (kxn) matrix

$$\underline{G} = \begin{bmatrix} g_{n-k} & g_{n-k-1} & \cdot\ \cdot\ \cdot & g_0 & 0 & \cdot\ \cdot\ \cdot & 0 \\ 0 & g_{n-k} & \cdot\ \cdot\ \cdot & g_1 & g_0 & \cdot\ \cdot\ \cdot & 0 \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ \cdot & & & & & & \cdot \\ 0 & 0 & \cdot\ \cdot\ \cdot & g_{n-k} & & \cdot\ \cdot\ \cdot & g_0 \end{bmatrix}$$

is the generating matrix of the code and the row space of $\underline{G}$ is the code.

One of the most important features of cyclic codes is their simple implementation. For this reason we present a succint exposition of the two methods used for encoding.

For both encoders, a k-symbol block of information digits forms the input and is sent through the channel. A delay then occurs as the (n-k) check digits are generated. The first encoder simply uses a k-stage shift register to produce the (n-k) check digits. The second method is somewhat more complicated but still not difficult to implement. If f(X) is a polynomial with the k information digits as coefficients of $X^{n-1},\ldots,X^{n-k}$, then $f(X)=g(X)q(X)+r(X)$, by the division algorithm, and the degree of r(X) is less than (n-k) for some q(X). Then, f(X)-r(X) is a codeword, and the coefficients in the low order (n-k) positions are the negative of the check digits. The encoder for this method is an (n-k)-stage shift register with premultiplication by X^{n-k}. The negative of the (n-k) digits remaining in the shift register are sent on to the channel.

The decoder for cyclic codes provides a simple method of error-detection. The received check digits are subtracted from the check digits calculated from the received information digits, using, as above, an (n-k)-stage shift register. If the result is $\underline{0}$, then the received vector is a codeword, otherwise an error is detected. Such a decoder is an excellent means of error-detection. However, the alterations required to allow for error-correction are complex and hence the resulting decoder usually is not a feasible method of error-correction.

In this section and in the previous one we presented two general classes of codes and their associated simplistic decoders. In later sections we introduce specific codes and their more complicated and hence more powerful decoders. However, first we disucss certain bounds used for comparison of codes.

2.5 Error and Distance Bounds

In this section we deal with several means of evaluating the performance, or describing the capabilities, of a given code relative to other codes.

Probably the most common metric used to characterize a code is the Hamming distance. The Hamming weight of a code vector $\underline{v}$, written $w(\underline{v})$, is the number of non-zero components in $\underline{v}$. The Hamming distance between two vectors $\underline{v}_1$ and $\underline{v}_2$ is defined as the number of positions in which they differ, or in the notation given, $w(\underline{v}_1-\underline{v}_2)$. For a linear block code, $(\underline{v}_1-\underline{v}_2)$ is a code vector. Thus, the distance between any two code words is in fact the weight of some codeword. The minimum distance of a linear code is the minimum weight of the non-zero code vectors.

Another distance metric, the Lee weight, exists but this is used only infrequently. For completeness we include its definition. The Lee weight of an n-tuple $(v_{n-1}, v_n, \ldots, v_o)$ the v_i from the set $(0,1,\ldots,q-1)$ is

$$w_L = \sum_{i=0}^{n-1} |v_i|, \quad \text{where } |v_i| = \begin{cases} v_i, & 0 \le v_i \le q/2 \\ q-v_i, & q/2 < v_i \le (q-1). \end{cases}$$

As for the Hamming distance, the Lee distance between two n-tuples is defined as the Lee weight of their difference. We note that for q=2 and q=3 the two metrics coincide.

We now examine some of the upper and lower bounds on the maximum minimum Hamming distance of a linear (n,k) code, with fixed n and k. We do not give a detailed study of these bounds but rather present only the results necessary for comparing codes.

The most general upper bound on the minimum distance d of an (n,k) linear code is the Plotkin bound of the average weight $nq^{k-1}(q-1)/(q^k-1)$. Further, if $n \ge (qd-1)/(q-1)$, the number of check digits necessary to attain the minimum weight d is at least $((qd-1)/(q-1))-1-\log_q d$.

The Hamming upper bound on $d \ge 2t+1$ is expressed as the following bound on the number of check symbols:

$$(n-k) \ge \log_q(1 + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \ldots + \binom{n}{t}(q-1)^t).$$

A third upper bound on the minimum distance of a linear (n,k) code is due to Elias. For large n this bound is tighter than either of the above bounds. The Elias bound is:

$$d \le 2t(1-\frac{t}{n})(\frac{k}{k-1}),$$

where t is any integer such that

$$\sum_{j=0}^{t} \binom{n}{j} 2^{n-k},$$

and k is the smallest integer such that

$$k \geq \sum_{j=0}^{t} \binom{n}{j} / \ (2^{n-k}).$$

These three bounds are the most common upper bounds on minimum distance.* We now give the Varsharmov-Gilbert lower bound on minimum distance. This bound states that there exists an (n,k) code with minimum distance at least d provided

$$\sum_{i=0}^{(d-2)} \binom{n}{i} (q-1)^i \geq (q^{n-k}).$$

As noted earlier, we are primarily interested in binary codes and hence with binary symmetric channels. We now consider some bounds on probability of error, P_e, for such channels. In the following Q is the probability of correct transmission, P the probability of an error, for the best binary code with given rate and length. Given a minimum distance of d, we have the rather trivial bound,

$$P_e \geq \sum_{i=[d/2]+1}^{d} \binom{d}{i} P^i Q^{(d-i)},$$

where [x] is the greatest integer in x. The sphere-packing lower bound on P_e is

$$P_e \geq \left(\binom{n}{t+1} - \alpha_{t+1}\right) P^{(t+1)} Q^{(n-t-1)} + \sum_{i=t+2}^{n} \binom{n}{i} P^i Q^{n-i},$$

t the greatest integer such that

$$\alpha_{t+1} = 2^{n-k} - 1 - \binom{n}{1} - \binom{n}{2} - \dots - \binom{n}{t} \geq 0.$$

We conclude with the random-coding upper bound on P_e which applies only to group codes and states

$$P_e \leq \sum_{j=[d_g/2]}^{d_g-1} \sum_{i=d_g-j}^{j} \sum_{h=d_g}^{j+1} \binom{n-j}{\frac{h+i-j}{2}} \binom{j}{\frac{h+j-i}{2}} \binom{n}{j} \frac{P^j Q^{n-j}}{2^{n-k-1}} + \sum_{j=d_g}^{n} \binom{n}{j} P^j Q^{n-j},$$

where d_g is the largest integer such that

$$2^{n-k} \geq 2 \sum_{i=0}^{d_g-1} \binom{n}{i}.$$

* However, the Greimer bound is considered a closer bound.

2.6 Hamming Codes

The Hamming code, which corrects all single errors, is a natural extension of the decoding table decoders discussed in Section 2.3. The basic binary Hamming code has length $n=2^m-1$ and m parity check digits. The decoder for this code makes use of the fact that there are 2^m-1 single error patterns, each of which can be made the coset leader of a distinct coset. Moreover, since there are m rows in the parity-check matrix $\underline{H}$, its columns can be written as the binary representations of the digits 1 to 2^m-1. Then if a codeword $\underline{v}$ is transmitted and a single error occurs, giving $\underline{v}'=\underline{v}+\underline{e}$, $\underline{v}$ can be correctly decoded by observing the syndrome $\underline{S} = \underline{v}'\underline{H}^T = \underline{e}\underline{H}^T$. This is a consequence of the error vector $\underline{e}$ being all zeros except for a single 1 in position i, for $1 \leq i \leq 2^m-1$. Thus, when $\underline{e}$ is multiplied by $\underline{H}^T$, the resulting syndrome is simply the i-th row of $\underline{H}^T$, that is the binary representation of the digit i. To correct the single error in $\underline{v}$, the i-th digit is inverted.

The code can be modified to any length n. To do so, the matrix $\underline{H}$ is constructed as above, using the smallest m such that $(2^m-1) \geq n$. Then, any selection of (2^m-1-n) columns are removed, leaving the required n columns. To decode a single error, the syndrome is calculated and if it corresponds to the j-th row of $\underline{H}^T$, then the j-th digit of the codeword is in error.

Detection as well as correction of errors is possible with the Hamming decoder if a single parity-check digit is added to the codeword. This increases to (m+1) the number of parity-check digits and increases the length of the code to 2^m. To decode, the syndrome is calculated. If it is $\underline{0}$, no errors are detected. If the last syndrome digit is a one, a single error is assumed to have occurred. The position in error is determined from the remaining syndrome digits, using the above decoder. If the syndrome is not the zero vector and the last digit is zero, then a non-correctable error pattern of at least two errors is detected. In a similar way, the modified Hamming decoder of any length n can be made to detect errors.

The binary Hamming codes have a fairly simple structure. All the vectors in the null-space of the single error-correcting code have the same Hamming weight. They are also in their basic form, one of the few examples of a perfect code. [illegible]

Overall, these codes are a marked improvement on single parity-check codes. However, they are still basically high rate codes unable to correct any pattern of two or more errors.

2.7 BCH Codes

The codes discussed in this section are a generalization of the Hamming Coaes. They are the best known of all non-random cyclic codes and possess powerful error-correcting properties and are relatively simple to decode. The BCH codes, developed independently by Bose and Chaudhuri in 1960 and Hocquenghem in 1959, are probably the single most important class of codes yet developed and have long served as a standard by which other codes are compared.

The definition of a BCH code over an extension field $GF(q^m)$ of $GF(q)$ is given in terms of the minimal polynomials of its roots. For α an element of $GF(q^m)$ with order e, r an arbitrary non-negative integer, d' an integer such that $2 \leq d' \leq e$, and $m_i(X)$ the minimal polynomial of α, $r \leq i \leq r+d-2$, the generator polynomial of the BCH code with parameters q,m,r,d and α, is defined as

$$g(X) = \mathrm{LCM}\ (m_r(X),\ \ldots\ ,\ m_{r+d-2}(X)\ ).$$

The block length n of the code is the product of the orders of the roots $\alpha^r, \alpha^{r+1}, \ldots , \alpha^{r+d-2}$. The binary BCH codes with $r=1$ and $d=2t+1$, t an integer and α a primitive element of $GF(2^m)$, are the most important subclass of this class of codes. The roots of the generator polynomial for this subclass are $\alpha, \alpha^2, \ldots , \alpha^{2t}$. Recalling the discussion of minimal polynomials, if α^i is the root of $m_i(X)$, then so are $\alpha^{2i}, \alpha^{4i}, \ldots$ etc. Thus, only the roots with an odd power yield a distinct minimal polynomial in the factorization of g(X), that is

$$g(X) = \mathrm{LCM}\ (m_1(X),\ m_3(X), \ldots,\ m_{2t-1}(X)).$$

As each $m_i(X)$ can have degree at most m, g(X) has degree at most mt_o, t_o the number of factors in g(X), which is tantamount to the code having mt_o parity-checks.

Binary BCH codes have a lower bound on minimum distance of $(d-1)$. Moreover, for any BCH code with fixed rate, the ratio of minimum distance to code length n tends to zero as n becomes large.

The BCH decoding algorithm consists of several steps, some of which involve terminology that must be defined before the decoder can be presented.

If the code word sent is given as the polynomial f(X) and errors occur which are described by the error polynomial e(X), then the received polynomial is $u(X)=f(X)+e(X)$. If the elements $\alpha^r, \alpha^{r+1}, \ldots, \alpha^{r+2t-1}$ are substituted sucessively into u(X), the resulting sequence is $e(\alpha^r), \ldots ,$ $e(\alpha^{r+2t-1})$ since $\alpha^r, \ldots , \alpha^{r+2t-1}$ are roots of the code vector u(X). The sequence listed is the basis of the BCH decoding algorithm. The parity-check calculations give the 2t equations:

$$e(\alpha_j) = \sum_{i=0}^{n-1} e_i\alpha_i^j = \sum_{i=1}^{v} Y_i X_i^j = S_j, \quad r \le j \le 2t-1,$$

where Y_i is the magnitude of the error, X_i is the error location number and v errors have occurred. Decoding is based on the observation that e_i is non-zero in the above set of equations for the positions in error in the received vector. In the binary case, as Y_i is one for each error, only the X_i need be found. In the general case, to correct any number $v \le t$ of errors, the pair (Y_i, X_i) must be obtained for each error. This is achieved by solving the set of equations:

$$S_j = \sum_i Y_i X_i, \quad r \le j \le r+2t-1.$$

To facilitate this, the quantities $\sigma_1, \sigma_2, \ldots, \sigma_v$, the elementary symmetric functions of the X_i are defined by the equation:

$$(X+X_1)(X+X_2)\ldots(X+X_v) = X^v + \sigma_1 X^{v-1} + \ldots \sigma_{v-1}X + \sigma_v.$$

From this equation the following set of linear equations relating the S_j and σ_j are formed:

$$S_j\sigma_v + S_{j+1}\sigma_{v-1} + \ldots + S_{j+v-1}\sigma_1 + S_{j+v} = 0, \quad r \le j \le 2t-1-v.$$

With the above definitions, the decoding can now be described as follows:

1) Calculate from the received vector u(X), the parity checks S_j, $r \le j \le r+2t-1$.
2) The maximum successive number v of these equations that are linearly independent is the number of errors which occurred.
3) Set $\sigma_{v+1}, \ldots, \sigma_t$ to zero and solve the first v equations for $\sigma_1, \sigma_2, \ldots, \sigma_v$.
4) Substitute each of the non-zero elements of $GF(q^m)$ in

   $$X^v - \sigma_1 X^{v-1} + \ldots + \sigma_v(-1)^v,$$

   the roots of which are the error location numbers.

5) In the non-binary case the error location numbers obtained in step 4) are substituted into the first v equations in

$$e(\alpha^{j}) = \sum_{i=1}^{v} Y_i X_i^j = S_j,$$

to solve for the Y_i.

Many simplifications of the basic algorithm have been found which reduce the complexity of the calculations used in the decoder. For the binary case several simplifications are possible. First, the binary circuitry is simpler to construct for step 1) and step 4). In step 3) the calculations themselves are simpler and moreover, step 5) is unnecessary as the magnitude of each error must be one.

We have discussed the decoder used for BCH codes in some detail for two reasons. First, as mentioned earlier, BCH codes are perhaps the best known and most powerful of all codes. Secondly, the decoder has illustrated what is considered to be a relatively easily decoded code.

2.8 Arithmetic Codes

We include these codes in our discussion as they are a practical method of encoding in a network using a computer.

These codes are unlike most codes in that, as their name suggests, all operations for encoding and decoding are arithmetic. Consequently, the standard error and distance definitions are not applicable.

A number N is represented as a polynomial in a radix-r system as:

$$N = N_{n-1}r^{n-1} + \dots + N_1 r^1 + N_o, \quad 0 \le N < r^n, \quad 0 \le N_i < r.$$

The length n codeword is written as $N_{n-1}N_{n-2}\dots N_1 N_o$. The minimum number of non-zero terms in the polynomial expression:

$$N = a_n r^n + \dots + a_1 r + a_o,$$

where the a_i can be positive or negative but the absolute value of each a_i is less than r, is the minimum weight of N. The arithmetic distance between two numbers N_1 and N_2 is the weight of the differnece N_1-N_2. If the number N_1 is sent and $N_2 \neq N_1$ received and the distance between N_1 and N_2 is d, then there is a d-fold error, or equivalently, a number of weight d has been added to N_1. If there is an arithmetic distance of at least d between all coded numbers, then all errors of magnitude d or less can be detected. To correct t or fewer errors, it is necessary for the minimum distance to be at least (2t+1). This definition of distance is especially applicable to a computer system as it considers each digit of the radix-r number to be possibly in error.

The actual coded form of the number N is the n-digit radix-r representation of the number AN, where A and r are relatively prime. With this definition there is a similarity between these codes and the class of cyclic codes. If there is a smallest n such that A divides (r^n-1) and n is the length of a codeword, then every cyclic shift of a codeword is a codeword, for if

$$N = a_{n-1}r^{n-1} + \dots + a_1r + a_o,$$

then a cyclic shift gives

$$a_{n-2}r^{n-1} + a_{n-3}r^{n-2} + \dots + a_or + a_{n-1} = rN - a_{n-1}(r^n-1)$$

which is, as required, a multiple of A.

To represent AN in the form described, the smallest number of digits required is at least

$$\log_r AN = N + \log_r A.$$

The constant $\log_r A$ is the redundancy of the code. The AN code is capable of correcting all combinations of t errors if and only if all numbers of weight at most t have distinct residues modulo A.

To encode the AN code, the number N is multiplied by A. To decode, the received code number is divided by A and if the remainder is zero, no errors are assumed. Otherwise, the remainder is taken to be the characteristic of the error number, which is the difference between the received number and the most likely code number transmitted. To determine this most likely code number, a search is made in a table of remainders and their corresponding most probable error number.

These codes are interesting as a class of codes which are easily implemented on a computer. However, they are restrictive in the sense that they basically perform a check on the arithmetic of a computer rather than act as a practical code for a communication network using a computer.

2.9 Conclusions and Thesis Objectives

In this chapter we have reviewed algebraic topics required for the development of codes discussed both in this chapter and in later chapters. Two important classes of codes, linear codes and cyclic codes, were discussed. Several bounds on distance were presented as a means of comparing codes. Hamming Codes, as the forerunner of generalized error-correcting codes, were reviewed. As the most well known and powerful of all algebraic codes, BCH Codes received a thorough treatment. The discussion on codes concluded with Arithmetic Codes. Notably, an important class of codes, the Majority Logic Decodable Codes, was omitted. The next chapter is devoted to these codes.

In this chapter we have given the reader the background necessary to appreciate the rest of the thesis both in terms of Mathematics and basic Coding Theory. By presenting several typical decoding methods, we emphasized the distinct need for much simpler decoding algorithms if error-correcting codes are to be a feasible addition to communication networks. We remark that computers, now a major constituent of many networks, should be considered as a factor when designing, or simplifying existing, decoders.

The need for simple decoding algorithms with correspondingly simple circuitry is the prime requirement for a practicable error-correcting communication system. It is this necessity to which we direct the thesis. The approach taken is to study in detail the mathematical structures of the null space of a class of codes already considered to possess a relatively simple decoder compared to those discussed in this chapter. To this end, Projective Geometry Codes, a subclass of the class of all Majority Logic Decodable Codes, were chosen for the study. In the analysis, emphasis is placed on finding symmetries in the null space which prove useful in simplifying the standard decoder for this class of codes.

Chapter 3 is an extensive review of Majority Logic Decoding and Projective Geometry Codes.

CHAPTER 3: MAJORITY LOGIC DECODABLE CODES

3.1 Introduction

In this chapter, Majority Logic Decoding and several related topics are discussed. We first review a class of binary linear codes, the Reed-Muller codes, and the associated decoder. The Majority Logic Decoding algorithm, which is based on the Reed-Muller decoding method, is considered in some detail. The best known class of codes which are Majority Logic Decodable are the finite geometry codes. We discuss a subclass of this class, the Projective Geometry Codes, giving a detailed mathematical description of Projective Geometries and the codes formed from them. The decoder used for this subclass is studied. The chapter is concluded with a summary of several modifications which have been made to the Projective Geometry Code Majority Logic Decoder in order to improve its performance.

3.2 Reed-Muller Codes

The class of codes discussed in this section are an ingenious alternative to Hamming Codes and BCH Codes. Unlike the latter codes, the Reed-Muller (R-M) Codes, [35], [41], can be decoded without the error digits being located and corrected as separate steps. The decoder for the R-M Codes depends on the majority testing of redundant digits within the code. As a result, the decoder can be very easily implemented. As the mathematical development of the R-M Codes is instructive in understanding the basis of the decoding method, we include a detailed exposition of it.

We note that Majority Logic Decoding is simply a variation of the decoding method presented here.

The R-M Codes are binary codes of length $n=2^m$ with codewords of the form:

$$\underline{f} = (f_o, f_1, \ldots, f_{n-1}),$$

$f_j = 0,1$, $j=0,1,\ldots,n-1$, from the space of all n-tuples over GF(2). If $\oplus$ denotes binary addition, then the sum of two codewords $\underline{f} = (f_o, f_1, \ldots, f_{n-1})$ and $\underline{g} = (g_o, g_1, \ldots, g_{n-1})$ is:

$$\underline{f} \oplus \underline{g} = (f_o, \ldots, f_{n-1}) \oplus (g_o, \ldots, g_{n-1})$$
$$= (f_o \oplus g_o, \ldots, f_{n-1} \oplus g_{n-1}),$$

where $f_j \oplus g_j$ is the modulo-2 sum of the binary digits f_j and g_j, $j=0,1,\ldots,n-1$. Multiplication by a binary scalar a is defined as:

$$a\underline{f} = a(f_o, \ldots, f_{n-1}) = (af_o, \ldots, af_{n-1}),$$

and multiplication of the vector $\underline{f}$ by the vector $\underline{g}$ as:

$$\underline{f}\underline{g} = (f_o, \ldots, f_{n-1})(g_o, \ldots, g_{n-1})$$
$$= (f_o g_o, \ldots, f_{n-1} g_{n-1}).$$

The complement $\underline{f}'$ of the vector $\underline{f}$ is:

$$\underline{f}' = \underline{f} \oplus \underline{e},$$

where $\underline{e}$ is the identity vector (1,1,...,1). The distance between any two vectors $\underline{f}$ and $\underline{g}$, is the Hamming distance $w(\underline{f}, \underline{g})$. Any code vector $\underline{f}$ can be expressed as:

$$\underline{f} = f_o \underline{I}_o \oplus f_1 \underline{I}_1 \oplus \ldots \oplus f_{n-1} \underline{I}_{n-1},$$

where $\underline{I}_j$ is the unit vector with a one in the j-th position and zeros elsewhere, $j=0,1,\ldots,n-1$. Moreover, each $\underline{I}_j$ can be written as a product of m vectors from the set of 2m vectors $\underline{X}_1, \ldots, \underline{X}_m, \underline{X}_1', \ldots, \underline{X}_m'$, where $\underline{X}_1$ is the vector consisting of the pair of digits 01 repeated 2^{m-1} times, $\underline{X}_2$ is the vector of digits 0011 repeated 2^{m-2} times and so on, as follows:

$$\underline{X}_1 = (010101\ldots0101)$$
$$\underline{X}_2 = (001100\ldots0011)$$
$$\vdots$$
$$\underline{X}_m = (000000\ldots1111),$$

and $\underline{X}_m$ consists of 2^{m-1} zeros followed by 2^{m-1} ones. To simplify the notation, $\underline{X}_k^{i_k}$ is defined as

$$\underline{X}_k^{i_k} = \begin{cases} \underline{X}_k' & \text{if } i_k=0 \\ \\ \underline{X}_k & \text{if } i_k=1. \end{cases}$$

Then, using Boolean algebra,

$$\underline{I}_j = \underline{X}_1^{i_1}\underline{X}_2^{i_2} \ldots \underline{X}_m^{i_m},$$

where the i_t, $t=1,\ldots,m$ are the binary coefficients of the radix-2 expansion of j:

$$j = \sum_{k=1}^{m} i_k 2^{k-1}, \quad i_k=0,1, \quad k=1,\ldots,m.$$

Hence, $\underline{f}$ can be expressed as:

$$\underline{f} = \sum_{j=0}^{2^{m-1}} f_j \underline{X}_1^{i_1}\underline{X}_2^{i_2} \ldots \underline{X}_m^{i_m},$$

with the summation taken modulo-2 and the i_t, $t=1,\ldots,m$ as above. This expression can be rewritten using the distributive law and the identity:

$$\underline{X}_j' = \underline{e} \oplus \underline{X}_j,$$

to give the following polynomial in the $\underline{X}_j$'s:

$$\underline{f} = g_o \oplus g_1\underline{X}_1 \oplus \ldots \oplus g_m\underline{X}_m \oplus g_{12}\underline{X}_1\underline{X}_2 \oplus \ldots \oplus g_{m-1,m}\underline{X}_{m-1}\underline{X}_m$$
$$\oplus \ldots \oplus g_{12\ldots m}\underline{X}_1\underline{X}_2\ldots\underline{X}_m.$$

The coefficients in this polynomial are the multiple partial differences and are defined as follows:

$$g_k = \overset{k}{\Delta} f(i_1,\ldots,i_m) = f(i_1,\ldots,i_k\oplus 1,i_{k+1},\ldots,i_m)$$
$$\oplus f(i_1,\ldots,i_k,\ldots,i_m)$$

$$g_{k_1k_2\ldots k_p} = \underset{k_1k_2\ldots k_p}{\overset{p}{\Delta}} f(i_1,\ldots,i_m)$$

$$= \underset{k_1\ldots k_{p-1}}{\overset{p-1}{\Delta}} f(i_1,\ldots,i_{k_p}\oplus 1,i_{k_{p+1}},\ldots,i_m)$$

$$\oplus \underset{k_1\ldots k_{p-1}}{\overset{p-1}{\Delta}} f(i_1,\ldots,i_{k_p},\ldots,i_m),$$

where Δ denotes the partial difference and the i_t's in $f(i_1,\dots,i_m)$ are the coefficients in the radix-2 expansion of j.

For the order-r R-M Code, the set of codewords consists of the set of polynomials of degree r or less, $r \leq m$, of the form:

$$g_o \oplus g_1 X_1 \oplus \dots \oplus g_m X_m \oplus \dots \oplus g_{12\dots r} X_1 \dots X_r \oplus$$
$$\dots \oplus g_{m-r+1,\dots,m} X_{m-r+1} \dots X_m.$$

The sum of any two polynomials from this set is another polynomial from the set. It can be shown [35] that the Hamming weight of any non-zero vector $\underline{f}$ in the set is:

$w(\underline{f}) \geq 2^{m-r}$, $m=0,1,\dots,$ $r \leq m$.

We introduce the decoding method of the R-M Codes through and example. For a code with r=1, m=3, any vector from the code space is given as:

$$g_o \underline{e} \oplus g_1 X_1 \oplus g_2 X_2 \oplus g_3 X_3.$$

The information digits are (g_o, g_1, g_2, g_3) and the generating vectors are:

$$X_1 = (01010101)$$
$$X_2 = (00110011)$$
$$X_3 = (00001111)$$
$$\underline{e} = (11111111).$$

Referring to the definition of multiple differences given above we have:

$$g_o = f(0,0,\dots,0) = f_o$$
$$g_1 = \underset{1}{\Delta} f(0,\dots) = f_o \oplus f_1$$
$$g_2 = \underset{2}{\Delta} f(0,\dots) = f_o \oplus f_2$$
$$g_3 = \underset{3}{\Delta} f(0,\dots) = f_o \oplus f_4$$

and,

$$\underset{12}{\Delta} f(0,\ldots) = f_0 \oplus f_1 \oplus f_2 \oplus f_3 = 0$$

$$\underset{13}{\Delta} f(0,\ldots) = f_0 \oplus f_1 \oplus f_4 \oplus f_5 = 0$$

$$\underset{23}{\Delta} f(0,\ldots) = f_0 \oplus f_2 \oplus f_4 \oplus f_6 = 0$$

$$\underset{123}{\Delta} f(0,\ldots) = f_0 \oplus f_1 \oplus f_2 \oplus f_3 \oplus f_4 \oplus f_5 \oplus f_6 \oplus f_7 = 0.$$

The last four equations are all zero as r=1 and hence no term is of degree two or more. From these equations, we have that g_1 satisfies:

$$\begin{aligned} g_1 &= f_0 \oplus f_1 \\ &= f_2 \oplus f_3 \\ &= f_4 \oplus f_5 \\ &= f_2 \oplus f_3 \oplus f_4 \oplus f_5 \oplus f_6 \oplus f_7. \end{aligned}$$

Substituting the second and third of these equations into the last relation, we obtain:

$$g_1 = f_6 \oplus f_7$$

and hence,

$$g_1 = f_0 \oplus f_1 = f_2 \oplus f_3 = f_4 \oplus f_5 = f_6 \oplus f_7.$$

These four relations on g_1 are disjoint in the sense that no two of them have any variables in common. Similarly we obtain four independent and disjoint relations on g_2 and g_3:

$$g_2 = f_0 \oplus f_2 = f_1 \oplus f_3 = f_4 \oplus f_6 = f_5 \oplus f_7$$

$$g_3 = f_0 \oplus f_4 = f_1 \oplus f_5 = f_2 \oplus f_6 = f_3 \oplus f_7.$$

If, in the received codeword $(f_0, f_1, \ldots, f_7)$, there are no errors, all the above relations hold. If there is a single error, then three of the four relations for each of g_1, g_2, g_3 hold. And if two errors occur, two of the four relations on g_j, j=1,23, hold. Consequently, g_1, g_2, g_3 can be determined correctly from a majority of the estimates on these variables if no more than one error occurs during transmission. Moreover, two errors can always be detected. If the four relations on g_j, $j=1,2,3$ are denoted by $r_{j1}, r_{j2}, r_{j3}, r_{j4}$, then the arithmetic sum of them, S_j, is given by:

$$S_j = \sum_{i=1}^{4} r_{ji}.$$

Then, the majority decision test for g_j is:

$$g_j \begin{cases} = 0 \text{ if } 0 \le S_j < 2, \\ \text{is indeterminate if } S_j = 2, \\ = 1 \text{ if } 2 < S_j \le 4, \qquad j=1,2,3. \end{cases}$$

Assuming less than two errors occur, g_1, g_2 and g_3 can be determined and used to find g_0. This is accomplished by adding g_1X_1, g_2X_2, g_3X_3 to the received vector to give $(m_0,\ldots,m_7)$. If no errors have occurred, the resulting vector is $g_0\underline{e}$, and if there were one error, $(m_0,\ldots,m_7)$ is a distance of one from $g_0\underline{e}$. A second majority test can now be applied to obtain g_0:

$$g_0 = 0 \text{ if } \sum_{i=0}^{7} m_i < 4,$$

$$g_0 = 1 \text{ if } \sum_{i=0}^{7} m_i > 4.$$

The method used to decode the above example can be generalized, such that, using the definition of the polynomial coefficients in terms of the multiple differences, all the information digits can be determined from a series of majority tests. This is due to the fact that each highest or r-th degree coefficient of any polynomial in the code satisfies exactly 2^{m-r} disjoint relations of the form:

$$\sum_{k=1}^{2^r} f_{i_k}, \quad i_k \text{ from } 0,\ldots,2^{m-1}, \ k=1,\ldots,2^r$$

and the summation taken modulo 2.

The number of information digits for the R-M Codes of order-r is

$$\sum_{i=0}^{r} \binom{m}{i}.$$

The order-(m-r-1) code is the dual of the order-r R-M code. Although for large n the rates of R-M Codes are significantly lower than those of the BCH Codes, the simple decoding method for the former still make them a competitive code. For low to medium length codes, the rates of these two classes of codes are more comparable, but still, in general, lower for the R-M Codes. However, as we are interested in this thesis in simple decoding methods, low rate is not considered a detrimental code characteristic.

The R-M Codes have a geometric interpretation. We include it here as it established the link between the method of decoding discussed above and the Majority Logic Decoding algorithm which is given later. For this description of the code we consider the space of dimension m over GF(2), consisting of 2^m points. Each of these points corresponds to a digit position j, $j=0,1,\ldots,n-1$, in the length n generating vectors $\underline{X}_i$ given above. A one appears in each position of the vector $\underline{X}_i$ for which the corresponding point has its i-th coordinate t_i equal to zero, that is for each point in an (m-1)-dimensional hyperplane through the origin. Thus $\underline{X}_i$ is the incidence vector of the hyperplane through the origin defined by $t_i=0$. The vector $\underline{X}_i\underline{X}_j$ has a one in each position which corresponds to a point with both its i-th and its j-th coordinates equal to zero. Hence, $\underline{X}_i\underline{X}_j$ is the incidence vector for an (m-2)-dimensional flat* through the origin. Similarly, each generating vector represents an incidence vector of a flat.

Each of the code symbols can be associated with a point of the m-dimensional space just described. Then, every parity-check rule is a check on those symbols associated with the points of a flat of dimension at least (r+1) through the

* See page 55.

origin. Each such flat describes an independent parity-check rule.

This interpretation of the code leads to the following geometric explanation of the decoding method. Each of the vectors used to generate the order-r code consists of a product of at most r of the vectors $X_1,\ldots,X_m$. There are 2^r points in an r-flat and hence a product of r vectors from the X_i has 2^{m-r} ones. For each of the 2^{m-r} points, p, in a generating vector, there exists a perpendicular flat of dimension r with 2^r points which passes through the point p. These perpendicular flats correspond to the parity-check rules used above to determine the information symbols. To see this correspondence we elaborate further. The parity-check flats intersect the generating flat whose coefficient is being determined, in one point, but intersects every other flat of dimension at least r either not at all or in at least a line. This implies the intersection has an even number of points, 2^t, where t is the dimension of the intersection. Thus, in the modulo-2 parity-check sum only the coefficient being determined will not cancel out, while all other coefficients will.

The geometrical description of the R-M Codes assists in the understanding of the associated Majority Logic Decoding algorithm, which is an extension of the decoding method given here.

The simplicity and elegance of this decoding method have made it the most feasible means of decoding a code which possesses the necessary independent redundancy relations. The remainder of this chapter deals exclusively with such codes and the Majority Logic Decoding of them.

3.3 Majority Logic Decoding

In this section we discuss Majority Logic Decoding, a decoding method renowned for its relative simplicity. Majority Logic Decoding (MLD), or Threshold Decoding (TD) in the binary case, is based on the existence of certain relationships among the parity-check equations of a code in much the same way as in the R-M Codes.

We introduce MLD by giving an example which illustrates the basic concept used in this decoding method. The binary repetition code of length n has two codewords, the all one n-tuple and the all zero n-tuple. Of the n codeword digits, the first is the message digit, with the remaining (n-1) being check digits. To decode, the number of ones and the number of zeros in the received codeword are counted. If the majority of the digits are ones, then the all one codeword is assumed to have been sent, otherwise the all zero codeword is assumed. If less than $[n/2]$ errors occurred, the decoder's decision is correct. The majority test used here is similar to the R-M majority tests. It is this concept of taking a majority test on which the MLD algorithm is based.

We return to the above example after several terms have been defined.

In the following discussion on MLD, the code to which we refer is assumed to the cyclic, linear (n,k) code with k information digits and (n-k) parity digits.

If $(c_{n-1},\dots,c_1,c_o)$ represents a codeword and $(e_{n-1},\dots,e_1,e_o)$ the corresponding error word, then the received word $(r_{n-1},\dots,r_1,r_o)$ can be written as $(c_{n-1}+e_{n-1},\dots,c_o+e_o)$. If $c_o,\dots,c_{k-1}$ are the information digits, the (n-k) parity digits are, in terms of the information digits:

$$c_j = \sum_{i=0}^{k-1} p_{ji}c_i, \quad j=k,k+1,\dots,n-1,$$

and the p_{ji} are from GF(p^s). The syndrome S_j is defined as the value of this equation when r_g is substituted for c_g:

$$S_j = \sum_{i=0}^{k-1} p_{ji}r_i - r_j$$

$$= \sum_{i=0}^{k-1} p_{ji}e_i - e_j, \quad j=k,k+1,\dots,n-1.$$

Linear combinations of the S_j are used to Majority Logic Decode.

We say that the sums $T_1,T_2,\dots,T_t$ of error digits are orthogonal on an error digit e_i if every sum T_s includes e_i and no other error digit occurs in more than one of the T_s. This definition can be extended to sums being orthogonal on a set of error digits weighted by coefficients from GF(p^s). For example, if $v_1,\dots,v_6$ are elements of GF(p^s), then the following sums are orthogonal on $v_1e_1 + v_2e_2$:

$$\begin{aligned}
T_1 &= v_1e_1 + v_2e_2 \\
T_2 &= v_1e_1 + v_2e_2 + v_3e_3 \\
T_3 &= v_1e_1 + v_2e_2 \qquad\quad + v_4e_4 \qquad\quad + v_6e_6 \\
T_4 &= v_1e_1 + v_2e_2 \qquad\qquad\qquad + v_5e_5.
\end{aligned}$$

If we look again at the binary repetition code of length $n=2^m-1$ with codeword $(c_{2^m-2},\dots,c_1,c_o)$ we note that:

$$\begin{aligned}
0 &= c_o + c_1 \\
0 &= c_o + \quad c_2 \\
&\vdots \\
0 &= c_o \qquad\qquad\qquad\qquad + c_{2^m-2}
\end{aligned}$$

and thus the following sums are orthogonal on e_o:

$$r_o + r_1 = e_o + e_1$$
$$r_o + r_2 = e_o + e_2$$
$$\vdots$$
$$r_o + r_{2m-2} = e_o + e_{2m-2}.$$

To decode this code using MLD, the 2^m-2 estimates on e_o are input to a threshold unit. If more than $(2^{m-1}-1)$ of the inputs are one, the threshold of $(2^{m-1}-1)$ is overcome and the output is a one, otherwise a zero. If less than $(2^{m-1}-1)$ errors occurred, the decoder correctly decodes the error digit e_o. The value of e_o is added to r_o giving the correct value of c_o. For the repetition codes there is only one information digit, so the remaining digits need not be decoded.

As a threshold unit is used to decode in the binary case, the corresponding decoding method is often referred[33], [37] to as Threshold Decoding rather than MLD.

For non-binary codes, a majority unit is used instead of a threshold unit. For such codes, the majority unit outputs the value receiving a clear majority of the sums orthogonal on it. If no value has a clear majority, then the error digit is assumed to be zero.

In general, once the first error digit is decoded in a cyclic code, the codeword is shifted and the next error digit decoded in the same manner. This process is repeated until the whole n-digit codeword is decoded.

Orthogonal estimates of a transmitted digit c_i, rather than sums orthogonal on an error digit e_i can be used to decode. For the binary repetition code discussed above we have the following orthogonal estimates of c_o:

$$c_o = \begin{cases} r_1 & \text{if } e_1=0 \\ r_2 & \text{if } e_2=0, \\ \vdots & \\ r^{2^{m-2}} & \text{if } e^{2m-2}=0. \end{cases}$$

To decode, using the orthogonal estimates, the received digits, rather than the syndrome digits, are used as input to the threshold unit. The output is then the cstimate of the code digit c_o, not the error digit e_o.

When there are an even number of parity-check sums which are orthogonal estimates of a digit c_i, a tie can occur. A more sophisticated decoder can overcome this difficulty. When one of the estimates is of the form:

$$c_o=r_o \quad \text{if } e_o=0,$$

the other estimates of c_o each contain two digits and the probability of an error in transmission is p, $p<\frac{1}{2}$, then the modified decoder can decode using error probabilities. If the probability that e_o is zero is:

$$\Pr(e_o=0) = (1-p)$$

and the probability that e_i+e_j is zero is:

$$\Pr(e_i+e_j=0) = (1-p)^2 + p^2 < (1-p), \quad i,j\neq 0,$$

then the estimate $c_o=r_o$ is more likely to be correct than any other single estimate, and hence can be given more weight when input to the majority unit. Thus, using probabilities, the tie can be broken.

We present some useful results concerning Majority Logic Decodable codes. If a linear code has at least (d-1) check sums orthogonal on each code digit, then the code has minimum distance at least d. This is a consequence of the following. If the i-th digit c_i has (d-1) orthogonal check sums $T_1,T_2,\ldots,T_{d-1}$, then each of these sums is zero since the word is in the code. Also, because each of these sums has a non-zero entry by construction, there must be at least one other non-zero code digit in each sum, which gives a total of d digits. Thus, the minimum distance is at least d.

A linear code with minimum distance d is said to be completely orthogonalizable if (d-1) parity-checks on each digit can be determined. Thus, any error pattern guaranteed correctable by the minimum distance of the code is correctable if the code is completely orthogonalizable. Further, if the code is cyclic, then it is completely orthogonalizable if there are (d-1) parity-checks orthogonal on e_0, the error digit in position zero.

If $\bar{d}$ is the minimum distance of the dual code of an (n,k) linear code, then the number of errors that can be corrected in the (n,k) code is t_1 and

$$t_1 \leq \frac{n-1}{2(\bar{d}-1)} .$$

To establish this we note that each word in the null space has weight at least $\bar{d}$ and thus there must be at least $\bar{d}$ digits in each of the orthogonal sums. One of these digits appears in each sum, while $(\bar{d}-1)$ appear in only one sum. There are (n-1) error digits in addition to the one on which the sums are orthogonal. Thus, the total number of orthogonal sums which can be constructed is no more than $(n-1)/(\bar{d}-1)$. Hence, t_1 must be less than or equal to half this number that is

$$t_1 \leq \frac{(n-1)}{2(\bar{d}-1)} .$$

The decoding process discussed thus far has required only one estimate of the outputs to determine an error or code digit. Obviously there are linear (n,k) codes for which it is not possible to construct (d-1) parity-checks orthogonal on each information digit. However, the process discussed above can be generalized so that after several steps, some of these codes can be decoded. Given the original set of parity-checks defined by the parity-check matrix $\underline{H}$, it is possible to form sets of at least (d-1) parity-checks orthogonal on certain sums of the information noise bits. It is then assumed that these sums are known because threshold estimates of their values have been obtained. These sums are now considered as additional parity-checks and added to the matrix $\underline{H}$ to give a new matrix $\underline{H}'$. The extended parity-check matrix is a true parity-check matrix if the sums added were correctly decoded. This process can be repeated L times until (d-1) parity-checks orthogonal on the error digit e_o are obtained. If this procedure can be carried out for each of the n error digits, then the code is said to be L-step orthogonalizable.

Formally, a t-error-correcting code is said to be L-step orthogonalizable if and only if the code contains sets of positions $P^{(1)}, P^{(2)}, \ldots,$ such that:

1) For all i, the code contains 2t parity-checks orthogonal on $P^{(i)}$.

2) The subcode of the code that satisfies the additional parity-checks:

$$\sum_{j \in P^{(i)}} c_j = 0, \quad \text{for all } i,$$

is (L-1)-step orthogonalizable.

With L-step decoding the bound given above for 1-step decoding can be improved. If $\overline{d}$ again denotes the minimum distance of the dual code of an (n,k) linear code, then the number of errors, t_L, that can be corrected with L-step MLD is bounded by:

$$t_L \leq \begin{cases} \frac{n}{\bar{d}} - \frac{1}{2}, & \bar{d} \text{ even,} \\ \frac{n+1}{\bar{d}+1} - \frac{1}{2}, & \bar{d} \text{ odd.} \end{cases}$$

This is established in much the same manner as the t_1 bound for the 1-step decoder. The t_L errors are corrected if there are at least $2t_L$ check sums orthogonal on a set B of digits common to each sum. Now B is at most $[\bar{d}/2]$, otherwise, when combining two such sets to give a set still in the null space, the new set would have weight less than $\bar{d}$, which is a contradiction. Consequently, the $2t_L$ equations have at most $[\bar{d}/2]$ digits in common and at least $\bar{d}$ digits altogether. Thus, $(\bar{d}-[\bar{d}/2])$ digits appear in one sum only and after selecting the $[\bar{d}/2]$ digits, there are $(n-[\bar{d}/2])$ digits left from which to choose the sets of $(\bar{d}-[\bar{d}/2])$ digits. Hence, there are at most:

$$\frac{(n-[\bar{d}/2])}{(\bar{d}-[\bar{d}/2])}$$

orthogonal equations, which implies:

$$2t_L(\bar{d}-[\bar{d}/2]) \leq (n-[\bar{d}/2]).$$

As an example of a cyclic 2-step MLD code, consider the binary (7,4) code with

$$\underline{H} = \begin{bmatrix} 1011100 \\ 1110010 \\ 0111001 \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}.$$

The received word is premultiplied by X^3 before decoding begins. The check sums corresponding to R_1 and R_1+R_2 are orthogonal on $S_1=e_6+e_o$, where $(e_6,\ldots,e_o)$ denotes the error word. These two sums are input to a threshold unit whose output is an estimate of S_1. The sums $(R_1+R_2+R_3)$ and $(R1+R_3)$ are orthogonal on $S_2=e_6+e_4$. These sums are input to a second threshold unit, whose output is an estimate of S_2. Then the estimates of S_1 and S_2, orthogonal on e_6, are input, at the second step to the threshold unit which outputs an estimate of the error digit e_6. Thus, the error digit e_6 can be determined in two steps, since the code is cyclic and hence 2-step orthogonalizable.

We remark here that with every added step in the orthogonalizing process, the complexity of the decoder increases. It is of interest to note though, that in L-step orthogonalization of a code, it is never necessary to make more than k threshold-decoding decisions. This follows since each decision is an estimate of a sum of the variables $e_0, \ldots, e_{k-1}$, and, there being only k such variables, there can be at most k linearly independent sums formed from them.

The Majority Logic Decoder is most applicable to binary codes. For non-binary alphabets the number of parity-checks that can be formed is roughly the same as for the binary alphabet. Thus, the advantage of a larger alphabet is lost. The Projective Geometry Codes analysed in Part II of the thesis are binary codes and hence attain the maximum power of this decoding method.

We conclude this section with an interesting example of the power of a t-error correcting Majority Logic Decoder. Unlike other t-error-correcting decoders, such as the BCH decoder, the Majority Logic Decoder corrects many errors of weight greater than t.

Consider the (1023,10) maximal length code with d=512. If this is transmitted over a Binary Symmetric Channel with transmission probability of error p_0=0.25, then the average number of errors per block is (1023)p_0 which is approximately 256. Now, $\left[(d-1)/2\right]$ is 255, so the probability of error, P_e, is nearly one half for an algorithm capable of correcting only errors of weight $\left[(d-1)/2\right]$ or less. However, if MLD is used to determine each of the ten information digits from the $(d-1)=511$ parity-checks orthogonal on each bit, the total probability of error is approximately $5x10^{-7}$. To see this, note that if $e_0=0$, then e_0 is incorrectly decoded only if more than 256 of the 511 parity-checks orthogonal on e_0 are one. Since each parity-check includes two other error bits, the probability that the sum is one is 0.375. The probability that more than 256 of the 511 parity-checks are one, is less than $3.1x10^{-8}$. Similarly, the probability of incorrect decoding if $e_0=1$ is less than $1.0x10^{-7}$. The average P_e in decoding e_0 is then:

$$(0.750)(3.1x10^{-8}) + (0.250)(10^{-7}) < 5.0x10^{-7}.$$

Although certainly all examples comparing MLD to other methods are not so dramatic, for-high rate codes there are many instances when MLD is the most effective decoding method. The main advantage of MLD, and certainly the reason that it is used in this thesis, is its ease of implementation. It is this characteristic which we emphasize in the thesis.

3.4 Projective Geometries

Part II of the thesis is devoted to a study of a subset of Projective Geometry Codes. These codes are based on a particular class of finite geometry, the finite projective geometry.

Formally, a finite geometry of dimension m is the set of elements (points) satisfying the following five conditions, given by Veblen and Bussey [51]:

1) The set contains a finite number of points and one or more subsets called lines, each of which contains at least three points.

II) If A and B are distinct points, there is one and only one line that contains both A and B.

III) If A, B and C are non-collinear points, and if a line l_o contains a point D of the line AB and a point E of the line BC but does not contain A or B or C, then the line l_o contains a point F of the line CA.

IV) If r is an integer less than m, not all the points of the geometry are in the same r-space.

V) If IV is satisfied, there is no (m+1)-space.

We now give the definition of a projective geometry, PG(m,p^s), of dimension m over GF(p^s), and later show that it does indeed satisfy the above postulates. A point is defined as a 0-space and a line as a 1-space and an r-space* is defined inductively as follows. Given (r+1) points $P_1,\ldots,P_r,P_{r+1}$ not all in the same (r-1)-space, the set of all points collinear with P_{r+1} and a point of the (r-1)-space $(P_1,\ldots,P_r)$ is the r-space $(P_1,\ldots,P_{r+1})$.

The most common means of representing a PG(m,p^s) is by GF($p^{(m+1)s}$). In the following we use this description without exception. We now outline this representation. A point of an m-dimensional finite PG can be described by a set of homogeneous coordinates $(\alpha_o,\alpha_1,\ldots,\alpha_m)$, the coefficients of (m+1) linearly independent points which define the GF($p^{(m+1)s}$) m-space. The α_i are elements of GF(p^s), such that at least one of them is non-zero. Any point $(\alpha\alpha_o,\ldots,\alpha\alpha_m)$ is equivalent to the point $(\alpha_o,\ldots,\alpha_m)$ for α one of the non-zero (p^s-1) elements of GF(p^s). The coefficients $\alpha_o,\ldots,\alpha_m$ which define a point, can be chosen in ($p^{(m+1)s}$ - 1) ways, where there are (p^s-1) representations of any given point. It then follows that there are a total of

$$(p^{(m+1)s}-1)/(p^s-1) = (p^{ms} + \ldots + p^s + 1)$$

* or r-flat

distinct points in $PG(m,p^s)$. Given any two distinct points $(\alpha_o,\dots,\alpha_m)$ and $(\beta_o,\dots,\beta_m)$, a line is defined as the set of points:

$$(\alpha\alpha_o+\beta\beta_o,\dots,\alpha\alpha_m,+\beta\beta_m)$$

where α and β are elements of $GF(p^s)$ such that both are not zero. The coefficients α,β can be chosen in $(p^{2s}-1)$ ways, where (p^s-1) of the choices give the same line. Thus a line contains

$$(p^{2s}-1)/(p^s-1) = (p^s+1)$$

distinct points. And, in r dimensions, $r\leq m$, given the $(r+1)$ points $(\alpha_{io},\alpha_{i1},\dots,\alpha_{im})$, $i=0,1,\dots,r$, not all in the same $(r-1)$-space, the r-space consists of the points

$$\left(\sum_{i=o}^{r}\alpha_i\,\alpha_{io},\dots,\sum_{i=0}^{r}\alpha_i\,\alpha_{im}\right)$$

where $\alpha_o,\alpha_1,\dots,\alpha_m$ are elements of $GF(p^s)$, not all simultaneously zero. Since there are $(p^{(r+1)s}-1)$ possible non-zero combinations of the α_i, where (p^s-1) combinations define the same point, there are

$$(p^{(r+1)s}-1)/(p^s-1) = p^{rs} + p^{rs-1} + \dots + p + 1$$

points in an r-space.

We now show that this formulation, based on $GF(p^{(m+1)s})$, satisfies the above postulates. Each line contains (p^s+1) points, which fulfills the requirements of postulate I. The definition of a line validates postulate II. If $(\alpha_o,\dots,\alpha_m)$, $(\beta_o,\dots,\beta_m)$ and $(\gamma_o,\dots,\gamma_m)$ are any three non-collinear points A,B and C, and t_o a line containing $D=(\alpha\alpha_o+\beta\beta_o,\dots,\alpha\alpha_m+\beta\beta_m)$, a point of AB and $E=(\rho\beta_o+\sigma\gamma_o,\dots,\rho\beta_m+\sigma\gamma_m)$, a point of BC, and t_o does not contain A,B or C, and α,β,ρ,σ different from zero, then we show that t_o consists of the points

$$(a\alpha\alpha_o+a\beta\beta_o+b\rho\beta_o+b\sigma\gamma_o,\dots,a\alpha\alpha_m+a\beta\beta_m+b\rho\beta_m+b\sigma\gamma_m)$$

such that a and b are elements of $GF(p^s)$, not both zero.

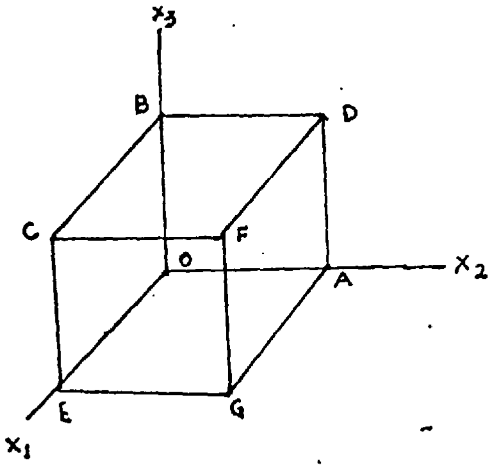
Now it is always possible to find β and ρ such that

$$a\beta + b\rho = 0.$$

This gives the point

$$(a\alpha\alpha_0+b\sigma\gamma_0,\dots,a\alpha\alpha_m+b\sigma\gamma_m)$$

on t_0 which is also a point on CA, and postulate III is validated. This is illustrated for m=2, s=1, p=2 in the following 3-dimensional diagram of PG(2,2).



O = (0,0,0)

A = (0,1,0)

B = (0,0,1)

C = (1,0,1)

D = (0,1,1)

E = (1,0,0)

F = (1,1,1)

G = (1,1,0)

Figure 3.4.1 An Example of PG(2,2)

In Figure 3.4.1, A, B and C are three non-collinear points and O is the origin (which is not allowed as a point). Obviously D is on line AB (i.e. ADB) and E is on BC (BCE) and D and E are on t_0=EFD and F is on CA (CFA).

The last two postulates are satisfied by the following argument. An r-dimensional geometry may be represented by $(\alpha_0,\alpha_1,\dots,\alpha_r)$, the α_i (r+1) linearly independent points. Then the r-dimensional geometry has the same description as the m-dimensional geometry containing it. Thus, for r<m, there are points not in the r-space and there is no (m+1)-space.

These arguments show that the formulation given for $PG(m,p^s)$ defines a valid finite geometry.

Using this representation, if α is an element of

$GF(p^{(m+1)s})$, then (α),

$$(\alpha) = \alpha, \alpha\beta, \alpha\beta^2, \ldots, \alpha\beta^{p^s-2},$$

β a primitive element of $GF(p^s)$, represents a point of the projective geometry.

Knowledge of the number of r-spaces contained in a given m-space of the projective geometry is very useful for the construction of projective geometry codes. Several steps are required in the calculation of this value. First the number of ways the (r+1) linearly independent points of the $PG(r,p^s)$ can be chosen, in order, from the points of $PG(m,p^s)$ so that they are not all in the same (r-1)-space is:

$$(1+p^s+\ldots+p^{ms})(p^s+p^{2s}+\ldots+p^{ms})(p^{2s}+p^{3s}+\ldots+p^{ms}) \ldots(p^{rs}+\ldots+p^{ms}).$$

The first term in this expression is the number of ways of selecting one point from the $PG(m,p^s)$, the second term the number of ways of choosing the second point distinct from the first point, the third point so that it is not in the line defined by the first two points and so on. We now determine the number of these bases which yield the same r-space. The number of ways the (r+1) base points of the given $PG(r,p^s)$ can be selected so that they do not all lie in the same (r-1)-space is, following the derivation above,

$$(1+p^s+\ldots+p^{rs})(p^s+\ldots+p^{rs})\ldots(p^{(r-1)s}+p^{rs})(p^{rs}).$$

The number of r-spaces in the $PG(m,p^s)$ is then:

$$\frac{(1+p^s+\ldots+p^{ms})(p^s+\ldots+p^{ms})\ldots(p^{rs}+\ldots+p^{ms})}{(1+p^s+\ldots+p^{rs})(p^s+\ldots+p^{rs})\ldots(p^{(r-1)s}+p^{rs})p^{rs}}$$

$$= \frac{(1+p^s+\ldots+p^{ms})(1+p^s+\ldots+p^{(m-1)s})\ldots(1+\ldots p^{(m-r)s})}{(1+p^s+\ldots+p^{rs})(1+p^s+\ldots+p^{(r-1)s})\ldots(1+p^s)\cdot 1}$$

$$= \frac{(p^{(m+1)s}-1)(p^{ms}-1)\ldots(p^{(m-r+1)s}-1)}{(p^{(r+1)s}-1)(p^{rs}-1)\ldots(p^{2s}-1)(p^s-1)} .$$

Similarly, a given t-space, $t<r<m$, is contained in

$$\lambda(t,r,m,p^s) = \frac{(p^{(m-t)s}-1)(p^{(m-t-1)s}-1)\dots(p^{(m-r+1)s}-1)}{(p^{(r-t)s}-1) \quad \dots \quad (p^{2s}-1)(p^s-1)}$$

distinct $PG(r,p^s)$'s in a given $PG(m,p^s)$.

An alternative description of a projective geometry can be given in terms of an (m+1)-dimensional vector space and its subspaces over $GF(p^{(m+1)s})$. A point is a 1-dimensional, non-affine (not through the origin) subspace of the vector space; a line a 2-dimensional subspace etc. A point lies on a line if it is contained in the 2-dimensional subspace representing the line. Then, if $(x_1,\dots,x_{m+1})$ is a point in the (m+1)-space, $(cx_1,\dots,cx_{m+1})$ defines the same point for c non-zero, c an element of $GF(p^s)$, because this is simply another member of the 1-dimensional subspace representing the point. In this description, the coordinates

$$(x_1,\dots,x_{m+1}) = (cx_1,\dots,cx_{m+1}),\ c\neq 0,$$

are called the homogeneous coordinates of the point. Formally, if the points of a linear subspace are represented as:

$$0 \cup \{\alpha^j\beta^k \mid k=0,1,\dots,(p^s-2),\ j \in A\},$$

with A a subset of the integers $(0,1,\dots,(p^{(m+1)s}-1)/(p^s-1))$, and β a primitive element of $GF(p^s)$, then the projective subspace of the linear subspace is the set:

$$\{\alpha^j \mid j\in A\}.$$

Thus, an r-dimensional projective space of $PG(m,p^s)$ is the set of all 1-dimensional vector subspaces in some (r+1)-dimensional vector subspace.

A hyperplane (a subspace of dimension (m-1) in $PG(m,p^s)$), is the locus of points given by:

$$a_1x_1 + \dots + a_mx_m + a_{m+1}x_{m+1} = 0,$$

the a_i not all simultaneously zero, where the a_i are the homogeneous coordinates of the hyperplanes. From the properties of linear vector spaces, it is obvious that taking the hyperplanes of $PG(m,p^s)$ as points, the dual projective space of $PG(m,p^s)$ can be formed. In the dual space, any two points (hyperplanes), intersect to give a line (an (m-2)-space). A space of dimension (m-r-1) results from the intersection of (r+1) hyperplanes. The set of points in the (m-r-1)-space are the solutions to a set of (r+1) linearly independent equations of the form:

$$a_1x_1 + \dots + a_{m+1}x_{m+1} = 0.$$

This corresponds to the definition of an r-space by (r+1) linearly independent points in the $PG(m,p^s)$. The notion of duality is of prime importance in defining the finite geometry codes.

Any non-singular linear transformation T carries a 1-dimensional vector subspace to another 1-dimensional vector subspace. Thus, T induces a one to one transformation of the points of the $PG(m,p^s)$ and hence projective subspaces are carried to projective subspaces. The induced transformation is called the projective transformation. These transformations are useful in decoding the finite geometry codes.

We conclude this section by referring again to Figure 3.4.1. Based on this representation of PG(2,2), we list the projective points and lines in Table 3.4.1. The number of points in this geometry is

$$(p^{(m+1)s}-1)/(p^s-1) = 7.$$

The number of lines is

$$\frac{(p^{(m+1)s}-1)(p^{ms}-1)}{(p^{(r+1)s}-1)(p^s-1)} = 7.$$

Projective Points	Projective Lines
OA	OBCE
OB	OBFG
OC	OBDA
OD	OEGA
OE	OEFD
OF	OAFC
OG	ODCG

Table 3.4.1 Projective Points and Lines of PG(2,2)

3.5 Projective Geometry Codes

Projective Geometry (PG) Codes are a generalization of R-M Codes. They are a cyclic, length $n=(p^{(m+1)s}-1)$ code over GF(p). The PG Codes are so named because each of the n digit positions in a codeword can be associated with a point from the projective geometry $PG(m,p^s)$. If (α^i), $i=0,\ldots,n-1$, represents a point from the geometry, then it corresponds to X^i in the polynomial interpretation of the n-tuple. Thus, an r-flat of the geometry can be associated with an n-tuple with ones in the positions corresponding to the points in the flat and zeros elsewhere. The n-tuple then represents a polynomial in the algebra A_n of polynomials modulo (X^n-1). A cyclic shift of the polynomial representation of an r-flat defines another r-flat. If $\alpha^{e_0}, \alpha^{e_1},\ldots,\alpha^{e_r}$ are the defining points of the original flat, then $\alpha^{e_0+1},\ldots,\alpha^{e_r+1}$ define the new flat. Also, since each point of a PG is a 1-dimensional linear subspace and an r-flat is a set of these points, the r-flats correspond to (r+1)-dimensional linear subspaces.

We now give the formal definition of the code. A Projective Geometry cyclic Code of order-r and length $n = p^{ms}+p^{(m-1)s}+\ldots+p^s+1$, over GF(p) is defined to be the largest cyclic code whose null space contains the polynomials corresponding to all r-flats of the $PG(m,p^s)$.

The PG Codes can also be characterized in terms of the roots of the parity-check and generator polynomials of the code. These roots are now determined. If f(X) is the polynomial associated with an r-flat of PG(m,p^s), α a primitive root of GF($p^{(m+1)s}$), then α^u is a root of f(X) provided u is a multiple of (p^s-1). We establish this with the following argument. If α is a primitive element of GF($p^{(m+1)s}$), then $(\alpha^{(p^{s(m+1)}-1)}) = 1$. Also, in the algebra of polynomials modulo (X^n-1),

$$(\alpha^u)^n = \alpha^{u(p^{s(m+1)}-1)\ /(p^s-1)} = 1.$$

Thus, u must be a multiple of (p^s-1), and so any root of f(X) is of the form $\alpha^{t(p^s-1)}$, that is

$$f(\alpha^{t(p^s-1)}) = \sum_j (\alpha^{t(p^s-1)})^j = 0,$$

the summation taken over the set R of the $(p^{s(r+1)}-1)/(p^s-1)$ points of the r-flat. The points in R are of the form:

$$\alpha^h = \beta^{i_o}\alpha^{e_o} + \dots + \beta^{i_r}\alpha^{e_r}, \quad i_a = 0,\dots,(p^s-1),\ a=0,\dots,r$$

the α^{ei} linearly independent elements of GF($p^{(m+1)s}$), β a primitive element of GF(p^s). Furthermore, each point α^j occurs (p^s-1) times in

$$\sum_{i_i i_1 \dots i_r} (\beta^{i_o}\alpha^{e_o} + \dots + \beta^{i_r}\alpha^{e_r}).$$

Consequently, this sum can be written as

$$\sum_{j \in R} (p^s-1)\alpha^j = (p^s-1) \sum_{j \in R} \alpha^j = (p^s-1)f(\alpha),$$

which is,

$$f(\alpha) = \frac{1}{(p^s-1)} \sum_{i_o \dots i_r} (\beta^{i_o}\alpha^{e_o} + \dots + \beta^{i_r}\alpha^{e_r}).$$

Then, $f(\alpha^{t(p^s-1)}) = 0$ if and only if

$$\sum_{i_o \dots i_r} (\beta^{i_o}\alpha^{e_o} + \dots + \beta^{i_r}\alpha^{e_r})^{t(p^s-1)} = 0 \qquad (1)$$

$$= \sum_{j \in R} (p^s-1)^{t(p^s-1)} \alpha^{jt(p^s-1)}$$

$$= (p^s-1)^{t(p^s-1)} \sum_{j\in R} \alpha^{jt(p^s-1)}.$$

Expanding (1) above, we have

$$\sum_i \sum_h \frac{(t(p^s-1))!}{h_o!\dots h_r!} (\beta^{i_o}\alpha^{e_o})^{h_o}\dots(\beta^{i_r}\alpha^{e_r})^{h_r},$$

and

$$\sum_{j=0}^{r} h_j = t(p^s-1).$$

This sum is zero unless $h_j=k_j(p^s-1)$ for $0\le j\le r$. Hence the equation can be written as:

$$\sum_{k_i} \frac{(t(p^s-1))!}{(k_o(p^s-1))!\dots(k_r(p^s-1))!}\alpha^{e_ok_o(p^s-1)}\dots\alpha^{e_rk_r(p^s-1)}.$$

This sum is zero unless $t(p^s-1)$ is the sum of at least $(r+1)$ multiples of (p^s-1). Thus $\alpha^{t(p^s-1)}$ is a root of the parity-check polynomial of the PG code of order-r and length $(p^{s(m+1)}-1)/(p^s-1)$ if and only if

$$w_s(t(p^s-1)) \le r, \quad t\ne 0,$$

where $w_s(x)$, the s-weight of x, is the largest number of multiples of (p^s-1) in the radix-p expansion of x. For p=2, s=1, this is simply the number of ones in the binary expansion of x. We note that α^o is not a root of the parity-check polynomial for if t=0, then

$$f(\alpha^{t(p^s-1)}) = \sum_{j\in R} (\alpha^{t(p^s-1)})^j$$

$$= \sum_{i=1}^{|R|}(1) \quad = \quad |R|$$

$$= (p^{s(r+1)}-1)/(p^s-1) \quad = \quad 1, \text{ mod } p,$$

where $|R|$ is the cardinality of the set R. Thus, α^o is not a root of h(X), the parity-check polynomial and so is a root of g(X), the generator polynomial.

The minimum distance of a PG code is at least the BCH bound on minimum distance for a code of length n. We now establish this. Any element of $GF(p^{(m+1)s})$ of the form $\alpha^{t(p^s-1)}$, with s-weight at least (r+1), and the root α^o, is a root of $g(X)$. The number v,

$$
\begin{aligned}
v &= (p-1)p^{s(m+2-r)-1} + \dots + (p-1)p^{s(m+1-r)+1} + (p-1)p^{s(m-r+1)} \\
&\quad + (p-1)p^{s(m-r+3)-1} + \dots + (p-1)p^{s(m-r+2)} \\
&\quad + \\
&\quad \vdots \\
&\quad + (p-1)p^{s(m+1)-1} + \dots + (p-1)p^{sm+1} + (p-1)p^{sm} \\
&= (p^s-1)p^{s(m-r+1)}(p^{s(r-1)} + \dots + p^s + 1) \\
&= p^{s(m-r+1)}(p^{sr}-1) \\
&= p^{s(m+1)} - p^{s(m-r+1)} \\
&= (p^{s(m+1)}-1) - (p^{s(m-r+1)}-1)
\end{aligned}
$$

is divisible by (p^s-1), has s-weight r and is a root of $h(X)$. If another multiple of (p^s-1) is added to v, then the s-weight becomes $(r+1)$, and hence α^v is the largest root of $h(X)$. So, for $v < i < p^{s(m+1)}$, i a multiple of (p^s-1), α^i is a root of $g(X)$. The number of successive roots is:

$$
\begin{aligned}
&(p^{s(m+1)}-1) - (\,(p^{s(m+1)}-1) - (p^{s(m-r+1)}-1)\,) \\
&\qquad = (p^{s(m-r+1)}-1).
\end{aligned}
$$

There are (p^s-1) repetitions of each root, so the number of distinct successive roots is

$$(p^{s(m-r+1)}-1)/(p^s-1).$$

Thus, as α^0 is also a root of $g(X)$, the minimum distance of an order-r PG code is at least:

$$\frac{p^{s(m-r+1)}-1}{(p^s-1)} + 1.$$

Using r-step MLD it is possible to correct

$$\left[\frac{1}{2}\left(\frac{p^{s(m-r+1)}-1}{(p^s-1)}\right)\right]$$

or fewer errors in an order-r PG Code. This is possible because the parity-check sums corresponding to the r-flats which intersect on a given $(r-1)$-flat, are orthogonal on the parity-check sum corresponding to the (r-1)-flat.

In order to decode, it is necessary to know the number of r-flats that intersect on a given (r-1)-flat L. We recall from the dicsussion on Projective Geometries in Section 3.4 that this quantity is:

$$J = (p^{s(m-r+1)}-1)/(p^s-1).$$

The decoding process is as follows. Initially all the parity-check sums corresponding to the r-flats are known to the decoder. As every point in the PG(m,p^s) is either in L or in precisely one of the r-flats that intersect on L, the J r-flats which contain L can be used to obtain a set of parity-checks orthogonal on L. Thus, with one level of majority logic the parity-check sums corresponding to the (r-1)-flats are determined, assuming $[J/2]$ or fewer errors occur. Similarly, the (r-2)-flat parity-check sums are obtained, and, after r steps, the 0-flats or error digits. The r-th order PG code is thus r-step orthogonalizable, r-step Majority Logic Decodable, and has distance J+1.

To illustrate the MLD of a PG code, we take the code with m=2,p=2 and s=1. This PG code of order r=1 has all the 1-flats of the projective geometry PG(2,2) in its null space. If α is a root of X^3+X+1, then the roots of h(X) are those α^i with $w_s(i) \leq 1$, $i\neq 0$, that is, $\alpha^1, \alpha^2, \alpha^4$. Thus,

$$\begin{aligned} h(X) &= (X-\alpha)(X-\alpha^2)(X-\alpha^4) \\ &= X^3 + X + 1, \end{aligned}$$

and

$$\begin{aligned} g(X) &= (X-\alpha^0)(X-\alpha^3)(X-\alpha^5)(X-\alpha^6) \\ &= X^4 + X^2 + X + 1. \end{aligned}$$

The null space has all the 1-flats of PG(2,2). These are given below in Table 3.5.1. Since the flats 1,2 and 4 are orthogonal on α^3, if zero or one errors occur, the majority of the estimates give the correct value of α^3.

Flat	0	1	2	3	4	5	6
1	1	1	0	1	0	0	0
2	0	1	1	0	1	0	0
3	0	0	1	1	0	1	0
4	0	0	0	1	1	0	1
5	1	0	0	0	1	1	0
6	0	1	0	0	0	1	1
7	1	0	1	0	0	0	1

Table 3.5.1 1-flats of PG(2,2)

In general, the PG codes have fewer information symbols than comparable BCH codes. For instance, in Chapter 5, we study the order-3, length 63, PG code over PG(5,2). This code has 41 information digits while the BCH code with the same length and error-correcting ability, has 45 information digits. We also discuss the order-5, length 255 PG code over PG(7,2) with 218 information digits. The corresponding BCH code has 231 information digits. However, the much simpler decoding method of the PG codes seems to outweigh this loss in information rate, which, for short block lengths, is small. Also, the PG codes do obtain the BCH lower bound on minimum distance.

In Part II of the thesis, these codes are analysed further and a simplification of their decoder found.

3.6 Majority Logic Decoder of an (n,k) Code

The most important feature of a Majority Logic Decoder is the simple circuitry required for its implementation. In this section, we briefly discuss the decoder in these terms.

Upon receiving a word from the linear (n,k) code, the Majority Logic Decoder multiplies it by $X^{(n-k)}$ and then divides the result by the generator polynomial g(X). The remainder, a shifted version of the syndrome, is stored in the register. This is shown in Figure 3.6.1. The next step in the decoding process is to form the (d-1) check sums orthogonal on the first error digit. This is done by the (d-1) $GF(p^s)$ adders and their scalar multipliers. Finally, the majority gate, with the (d-1) inputs from the $GF(p^s)$ adders, outputs the value assumed by majority of its inputs, or zero if there is no clear majority.

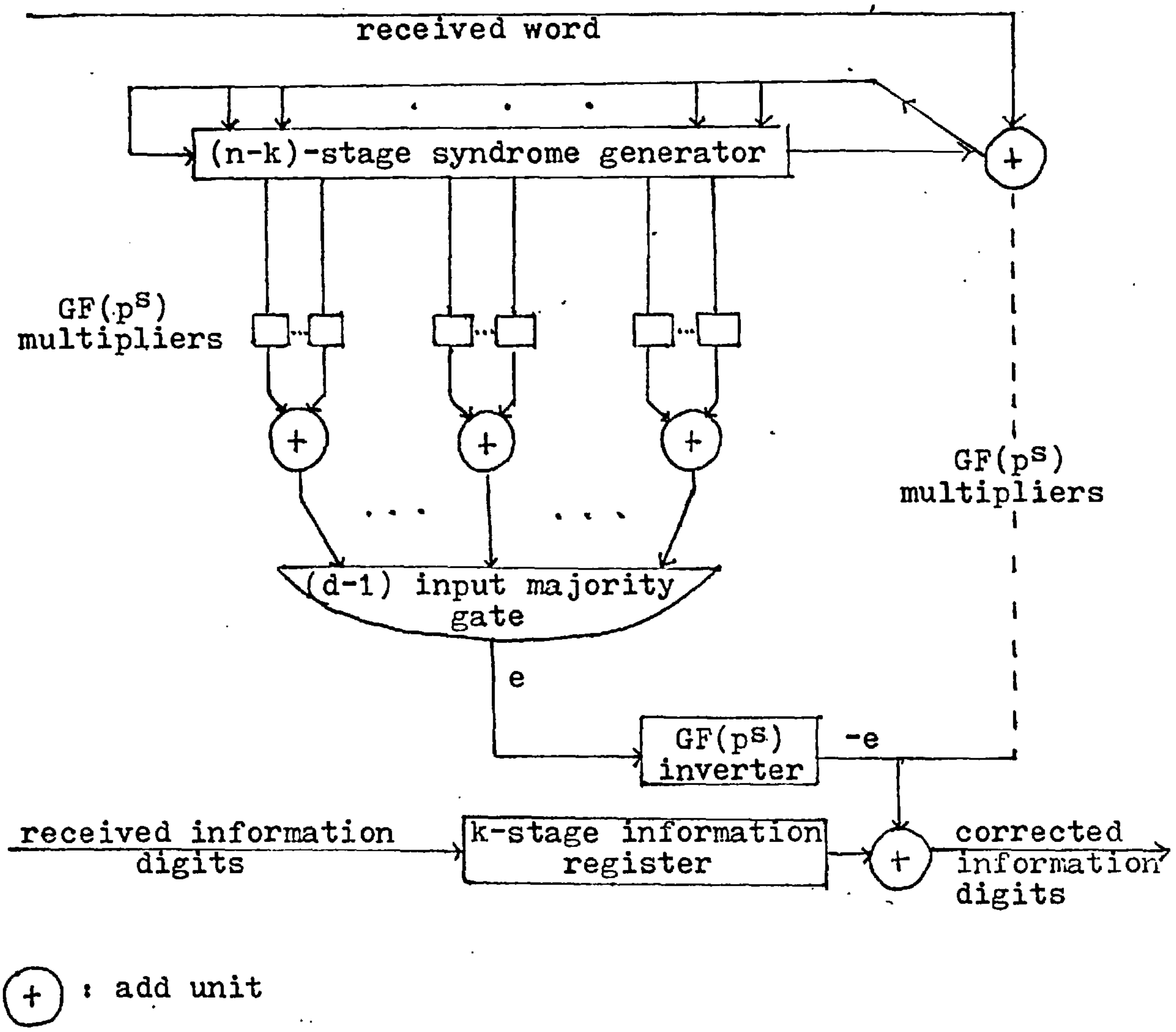

received word
(n-k)-stage syndrome generator
GF(p^s) multipliers
GF(p^s) multipliers
(d-1) input majority gate
e
GF(p^s) inverter
-e
received information digits
k-stage information register
corrected information digits
+ : add unit

Figure 3.6.1 One-step MLD of Cyclic (n,k) Code

In the binary case, which we study in Part II, the majority gate can be replaced by the much simpler threshold unit. Then, if at least half of the inputs are one, a one is output. The value output by the majority unit or threshold unit, is subtracted from the first information digit.

As the codes considered are cyclic, by shifting both the information register and the syndrome generator, the second information digit can be corrected in the same manner as the first. This process is repeated until all the digits are corrected. As the whole codeword can be decoded with one level of majority testing, the decoder is called a one-step Majority Logic Decoder.

As was noted earlier, each level of majority logic that is added to the one-step decoder increases the complexity of the decoder. We depict, in Figure 3.6.2, a 2-step decoder.

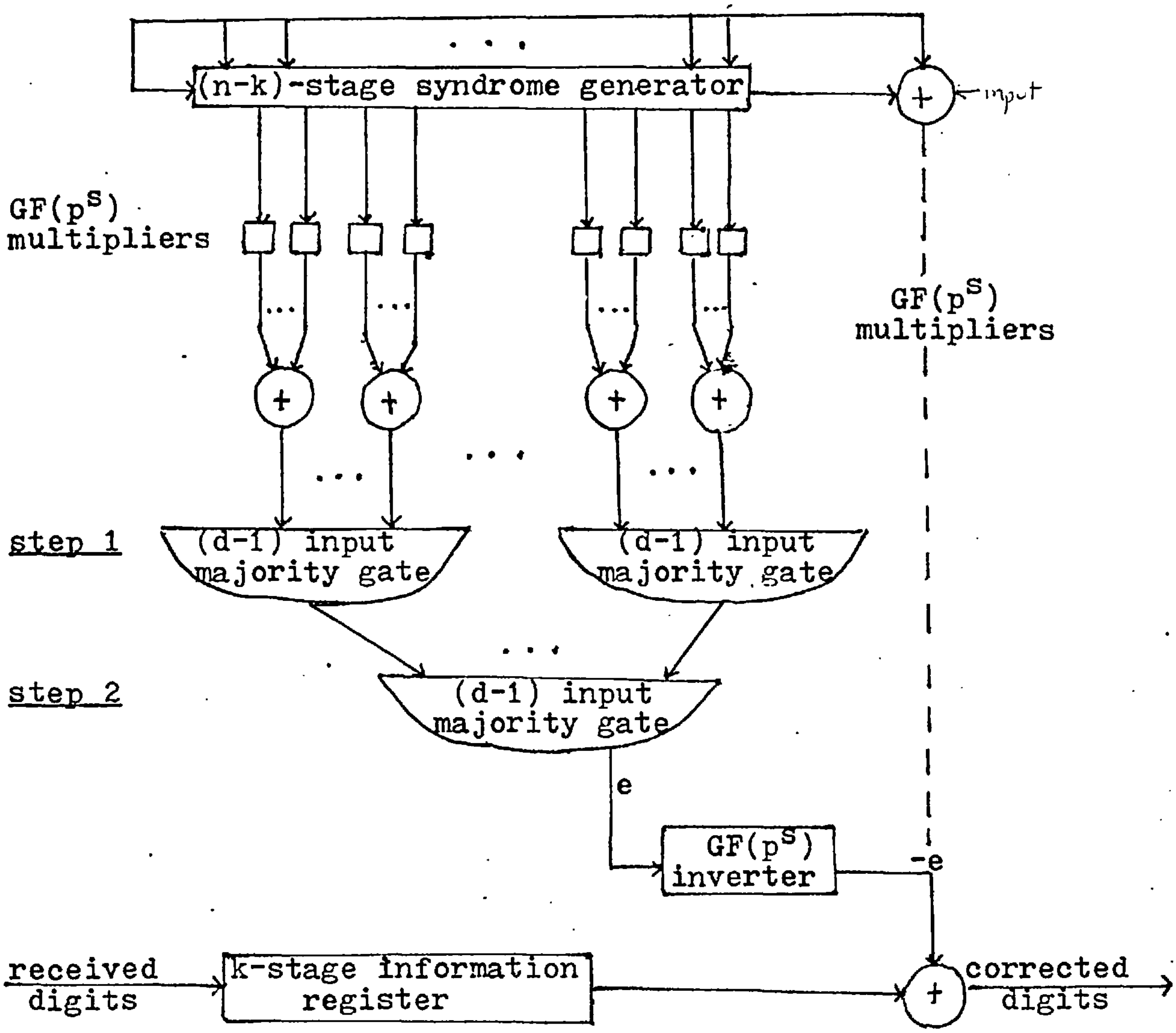

(n-k)-stage syndrome generator
input
GF(p^s) multipliers
GF(p^s) multipliers
step 1
(d-1) input majority gate
(d-1) input majority gate
step 2
(d-1) input majority gate
e
GF(p^s) inverter
-e
received digits
k-stage information register
corrected digits

Figure 3.6.2 Two-step MLD for Cyclic (n,k) Code

For the two-step decoder, the procedure for the first step is much the same as in the one-step decoder, the difference being that (d-1) check sums are orthogonal on two digits rather than one. Of these two digits, the first digit in each of the (d-1) digit pairs, is the digit e_0, in position zero, the second digit is distinct from every other digit of the (d-1) pairs. Each set of (d-1) parity-check sums orthogonal on a digit pair is input to a distinct majority unit. On the second step, (d-1) orthogonal estimates of the (d-1) digit pairs obtained as output from the majority units in step one, are input to a second level majority gate (or threshold unit in the binary case). The (d-1) estimates obtained from step one are orthogonal on the digit e_0. Thus, the output from the second level majority gate is an estimate of the error digit in position zero. As for the one-step decoder, the final output is subtracted from the first information digit. Again, as the codes are cyclic, the information register and syndrome generator are shifted and the second information digit corrected as the first was. By repeating this process n times, the whole codeword can be decoded.

The purpose of including a detailed description of the Majority Logic Decoder is two-fold. First, it illustrates the relative simplicity of this decoding method, both in terms of its circuitry and as compared to the BCH decoder discussed in Chapter 2. Secondly, in Chapter 4, we develop a simplified version of this decoder and in evaluating it, compare it to the standard decoder described here.

3.7 Modifications to the MLD of PG Codes

Since the introduction of MLD in 1954 [35],[41], there have been many attempts to simplify even further this decoding method. In this section, we review several of the most relevant of these.

The first of the improvements involves the concept of feed-back. If the basic decoder is t-error-correcting, then (t+1) errors can be corrected using the following method. Suppose there is an error in position e_0 of the codeword and that in one of the (d-1) check sums orthogonal on e_0, there are two additional errors. Then, e_0 can be corrected and through feed-back the correction fed into the received codeword. Then, the remaining t errors can be corrected using the t-error-correcting decoder.

Townsend and Weldon [50] suggest using a variable threshold level to decode binary MLD codes. In the standard decoder, the output from the threshold unit is a one if

$$T = [(d+1)/2]$$

or more of the inputs are one. The modification suggested in this paper initially sets the threshold to T=(d-1). An attempt is made to decode all n bits of the codeword. If the decoder is unsuccessful, the threshold of T is decreased by one to (T-1). The procedure is repeated and if again no changes are made, T is lowered by one a second time. However, if an error is corrected, then T is increased by one. This whole process is continued until T is set to $[(d+1)/2]$, at which point decoding ceases. Although this method corrects many more than $[(d-1)/2]$ errors, it does require considerably more time and more complex circuitry than the standard MLD.

Gallager [16] gives a decoding method applicable to low density codes, that is, codes with a large number of zero entries in the nullspace. The method is particularly suited to PG codes as it requires a fixed number j, $j \geq 3$, of ones in each column, and k, $k \geq j$, ones in each row, of the parity-check matrix. The parity-checks are calculated and then any digit that appears in more than a fixed number of unsatisfied parity-check equations is changed. The new value of the digit is used to recompute the parity-checks. The process is continued until all the parity-checks are satisfied. If e_o is the error digit to be corrected, then the term first level tier is used to describe all the parity-checks that include e_o. A second level tier contains the parity-checks on the digits involved in the level one tier. The relation to orthogonal checks is obvious. A variation on this decoder includes a posteriori probabilities on the channel outputs.

Although this decoder is capable of correcting more than the standard number of errors, its decoder is more complex and the time required for decoding is greater than for the standard MLD.

The MLD algorithm and the above modified decoders all are based on orthogonal check sums. We now discuss several variations on the Majority Logic Decoder which depend on non-orthogonal check sums.

The first of the non-orthogonal decoders of an order-r PG code requires only one majority gate, but the gate does however have a very large number of inputs. For this algorithm, the number

$$N = \frac{(p^{sm} + p^{s(m-1)} + \dots + p^{s}) \dots (p^{sm} + \dots + p^{sr})}{(p^{sr} + p^{s(r-1)} + \dots p^{s}) \dots (p^{sr} + p^{s(r-1)})p^{sr}}$$

of r-flats which pass through a given point, and

$$y = \frac{(p^{sm} + \dots + p^{s}) \dots (p^{sm} + \dots + p^{sr})(p^{s(r-1)} + \dots + p^{s} + 1)}{(p^{s} + p^{s(r-1)} + \dots + p^{s}) \dots (p^{s(r-1)} + p^{sr})p^{sr}(p^{s(m-1)} + \dots + p^{s} + 1)}$$

the number of r-flats passing through a given line, are needed. For a given point contained in N r-flats, each other digit in the geometry appears in y of the r-flats. If

$$\left[N/2y\right] = \left[(p^{sm}-1)/2(p^{sr}-1)\right]$$

or fewer errors occur, then the error digit contained in all the r-flats is given correctly by the majority of the r-flats.

The complexity of the standard Majority Logic Decoder increases with the number of decoding steps. Although this version of the decoder requires only one step, it is not feasible to use it as a decoding method for two reasons. First, N above is very large for reasonable size codes and hence the single majority gate has a very large number of inputs. Secondly, this method corrects fewer errors than the standard Majority Logic Decoder. The latter problem can be overcome by increasing the number of decoding steps to two, and correspondingly increasing the number of majority gates. For this decoder, the r-flats are used to determine the (r-1)-flats. Then the above non-orthogonal procedure can be applied to the (r-1)-flats, giving a decoder which corrects at least the number of errors that the standard Majority Logic Decoder does. However, the number of inputs to the second step remains large and hence it is questionable whether such a decoder is less complex than the standard decoder.

A second non-orthogonal decoder was originally presented by Rudolph[42] and modified by Ng [36]. Given a parity-check matrix $\underline{H}$ of the order-r PG Code over $PG(m,p^s)$, $\underline{H}_E$ the row space of $\underline{H}$, C a codeword and $\underline{R}$ the received vector, set

$$\underline{H}_E^T \cdot \underline{C} = \underline{0}.$$

Then, to decode r_i, the i-th digit in $\underline{R}$, a matrix

$$\underline{H}_E^i = \left[h_{p_j}^{\,i} \right]$$

is chosen for which each row has a non-zero element in the i-th position, to give J^i rows such that there are α_{ij} non-zero elements in column j, $i \neq j$. From the equation $\underline{H}_E \cdot \underline{R}^T = \underline{0}$, we obtain the J^i estimates c_i^* of c_i, the i-th digit of the codeword C,

$$c_i^* = -(h_{p_i}^{\,i})^{-1} \sum_{j \neq i} h_{p_j}^{\,i} r_j \,, \qquad p_j = 1,2,\ldots,J^i.$$

Each r_j, $j \neq i$ is in α_{ij} of the J^i estimates. In the more powerful version of this decoder, given by Ng[36], α^i_{max} is selected as the maximum of the α_{ij}, $j \neq i$. The α^i_{max} equations $c^*_i = r_i$ are added to the estimates to give $(J^i + \alpha^i_{max})$ estimates of c_i. The original decoder by Rudolph[42] added only one such estimate. Since each error in $\underline{R}$ can affect at most α^i_{max} of the estimates, MLD can be used to correctly decode the i-th digit if no more than

$$\left[(J^i + \alpha^i_{max} - 1)/2\alpha^i_{max}\right]$$

errors occur. The maximum number of errors is corrected when $(J^i + \alpha^i_{max} - 1)$ is maximized. The matrix $\underline{H}^i_E$ is selected on this basis. The decoding process is repeated for each of the code digits. The total number of errors that can be corrected is then

$$\min_i \left\{ \left[\frac{J^i + \alpha^i_{max} - 1}{2\alpha^i_{max}} \right] \right\},$$

which is a constant for PG codes, since they are cyclic.

This algorithm increases the distance and hence number of errors correctable by the decoder but with a corresponding increase in complexity.

Perhaps the most important improvement to the MLD algorithm is the one proposed by Chen[6],[7]. In developing his simplification, he shows that the minimum number of steps in which it is possible to decode an order-r PG code, using orthogonal MLD, is $N = 1 + \lceil \log_2(m/(m-r)) \rceil$, which is

$$N = \begin{cases} 1 & \text{if } r = 0 \\ 2 & \text{if } m/2 \geq r > 0 \\ i+1 & \text{if } (1-2^{-i})m \geq r > (1-2^{-(i-1)})m. \end{cases}$$

This is based on the observation that, since all r-flats are in the null space of the code, all the (r-1)-flats can be determined on the first decoding step. Then, if k is the least integer such that a set of at least J (r-1)-flats orthogonal on a given k-flat can be constructed, J the maximum number of r-flats orthogonal on any (r-1)-flat, then each of the k-flats can be determined from the (r-1)-flats orthogonal on it. This procedure can be repeated until the noise digits, or 0-flats, are determined. Further, Chen shows, using flats parallel to a given flat through the origin, that there exists a sufficient number of flats of the given dimension to guarantee this decoding method. Moreover, this method obtains the minimum complexity possible using the standard MLD algorithm as the basis.

The final modification discussed involves altering the Majority Logic procedure to obtain a decoding algorithm requiring fewer majority gates but more time and buffer storage. The algorithm makes use of certain relationships among the syndrome digits. It cyclically shifts these digits, stores them and then uses them for the next level of decoding.

If $\underline{c} = (c_0,\ldots,c_{n-1})$ is the transmitted codeword, and $\underline{e} = (e_0,\ldots,e_{n-1})$ the error vector, then the received vector is $\underline{c} + \underline{e} = \underline{r} = (r_0,\ldots,r_{n-1})$. The following algorithm decodes r_0 correctly if no more than t errors occur. To decode, the decoder solves for e_0 in the equation $\underline{e}\cdot\underline{H}^T = \underline{S}$, for $\underline{S}$ the syndrome, and $\underline{H}$ the parity-check matrix. There are 2^k solutions to this linear matrix equation. The non-linear constraint of $w(\underline{e}) \leq t$, where w(x) is the Hamming weight of x, reduces the number of solutions to one. The standard Majority Logic Decoder accomplishes this reduction by deriving new parity-checks from the old and adding them to the parity-check matrix $\underline{H}$, to increase its rank. In so doing, the number of solutions to the above equation is reduced. This process is repeated L times for a L-step MLD code, at which point enough new parity-checks have been added to assure that all the solutions give the same value of e_0. Then, as the code is cyclic, the received word is shifted and the process repeated. This decoder, the sequential code reduction decoder [44], uses the cyclic property at each step. This reduces the combinational complexity of the decoder by making it a linear function rather than exponential function, of the number of steps needed for decoding. At each level, the many majority units are replaced by one majority unit and extra storage space added. The single majority unit calculates, in the standard way, an estimate to be used on the next level. This estimate, and its (n-1) cyclic shifts are stored. On the next level the estimate, and a linear combination of its cyclic shifts, are used as input to another majority unit. The process is repeated until the error digit, e_0, is obtained. When this method is applied to PG codes of length $n \leq 2047$, it is possible to decode using, at each stage, 2t orthogonal parity-checks and one majority gate, and to correct the standard number of errors. This decoder requires that, at each level of the decoding procedure, there exists a polynomial flat which divides a set of 2t polynomial flats orthogonal on a flat of lower dimension at the next level.

This algorithm has illustrated that the complexity of the MLD algorithm can be significantly decreased by increasing the time and storage required for decoding.

In the next chapter we suggest a method of simplifying the MLD algorithm for a subclass of PG codes, based on the results of an analysis of the mathematical structure of the null space.

3.8 Conclusions

In this chapter, we have considered several topics relating to MLD. We began the chapter by discussing R-M Codes, a class of codes known for its simple decoding algorithm. It is this decoding method which forms the basis of the MLD algorithm. Together with examples, the MLD method was studied in detail. The mathematical properties of Projective Geometries were dealt with. PG Codes, a subclass of the class of all codes which are Majority Logic Decodable, were discussed. A subclass of these codes are examined in the next chapter. We presented a general Majority Logic Decoder for PG Codes, emphasizing the circuitry required to implement the decoder. This chapter was concluded with a survey of several modifications which can be made to simplify the standard Majority Logic Decoder for PG Codes.

With this background, the reader is now in a position to appreciate the aim of the thesis, the development of a simplified version of the Majority Logic Decoder for PG Codes, based on mathematical structures of the null space.

PART II
CHAPTER 4: ORBIT STRUCTURE OF PG(5,2)

4.1 Introduction

In this chapter a structural description of the flats of a finite Projective Geometry is presented. The background to this interpretation is the work of Rao [39], and Yamamoto, Fukuda and Hamada [58], concerning the compact representation of the flats of a projective geometry. The definition of a cycle of a flat, first introduced by Rao [39], is used to define another structure, the orbit of a set of flats. The decoding method introduced in the thesis is based on these orbits.

We begin the chapter with a review of the material from Rao [39] and Yamamoto et al [58] which is pertinent to our study. In this and the next chapter we refer exclusively to the order-3 (63,41) PG code over PG(5,2). The cycles of this geometry are analysed using the theorems of Yamamoto et al. Based on these cycles, the orbit structure is defined. A detailed investigation is made of this structure and of the symmetries which it exposes. Finally, it is established that the orbit structure is independent of the minimal polynomial used to define the geometry.

4.2 Cycle Description of Finite Geometry Flats

Rao [39] and Yamamoto et al [58] present a compact representation of a finite PG, based on the cycles of the flats. We review those results of their work which are of use in this study.

The concept of a cycle of a flat was introduced by Rao [39] to analyse the structure of a family of flats from a finite geometry. Yamamoto et al [58] found that some of Rao's conjectures were true in only certain cases. Consequently, we refer to the more generalized results of Yamamoto et al in the following.

Recalling the discussion of both Galois Fields and Projective Geometries, we note the following results which are useful in establishing certain theorems given below. We write q for p^s, p a prime, in the following. The projective geometry to which we refer is PG(m,q).

If (m+1)/(i+1) is integral for some non-negative integer i, then $(q^{(i+1)}-1)$ is the least integer u such that:

$$(\alpha^{\theta})^{u} = 1,$$

$$\theta = (q^{(m+1)}-1)/(q^{(i+1)}-1).$$

Hence, α^{θ} is a primitive element of $GF(q^{(i+1)})$ and can be used to give the following representation of $GF(q^{(i+1)})$:

$$GF(q^{(i+1)}) = \{0, \alpha^{o}, \alpha^{\theta}, \ldots, \alpha^{(q^{i+1}-2)\theta}\}.$$

The corresponding Projective Geometry, PG(i,q), is then:

$$PG(i,q) = \{(\alpha^{o}), (\alpha^{\theta}), \ldots, (\alpha^{(((q^{i+1}-1)/(q-1))-1)\theta})\}.$$

In particular,

$$GF(q) = \{0, \alpha^{o}, \alpha^{v}, \ldots, \alpha^{(q-2)v}\},$$

and,

$$PG(m,q) = \{(\alpha^{o}), (\alpha), \ldots, (\alpha^{v-1})\},$$

where, $v = (q^{(m+1)}-1)/(q-1)$, the number of points in PG(m,q). The first (i+1) points,

$$(\alpha^{o}), (\alpha^{\theta}), \ldots, (\alpha^{i\theta})$$

of PG(i,q) are linearly independent over GF(q). The set of all linear combinations of these points yields PG(i,q).

If $V_d(0)$ denotes a d-flat in PG(m,q) passing through the (d+1) linearly independent points:

$$(\alpha^{bo}), (\alpha^{b1}), \ldots, (\alpha^{bd}),$$

then $V_d(0)$ is the set of points given by:

$$V_d(0) = \{(a_o\alpha^{bo} + a_1\alpha^{b1} + \ldots + a_d\alpha^{bd})\},$$

and the d-flat $V_d(c)$ is

$$V_d(c) = \{(a_0\alpha^{b_0+c} + \dots + a_d\alpha^{b_d+c})\},$$

the a_i from GF(q). For some positive integer c, $V_d(c)=V_d(0)$. The integer c is called a cycle of the initial d-flat $V_d(0)$. This definition was introduced by Rao in [39]. In particular, v, the number of points in the geometry, is a cycle of any d-flat $V_d(0)$ because $V_d(0)=V_d(v)$. The minimum value of the cycles of a d-flat $V_d(0)$ is called the minimum cycle (m.c.) of $V_d(0)$. The following are consequences of the definition of a cycle:

i) If θ is the m.c., then it is a factor of any cycle c, and therefore a factor of v.

ii) All the points of a d-flat of m.c. θ can be listed as follows, in terms of powers of α:

$$\begin{array}{l} c_0,\ c_0+\theta,\ \dots,\ c_0+(r-1)\theta \\ c_1,\ c_1+\theta,\ \dots,\ c_1+(r-1)\theta \\ \vdots \\ c_s,\ c_s+\theta,\ \dots,\ c_s+(r-1)\theta, \end{array}$$

$c_i-c_j \not\equiv 0 \bmod \theta$, $i\neq j$, $i,j=0,1,\dots,s$, $r=v/\theta$. This representation follows if we note that c_i can be expressed as $c_i=c_0+k_i$, for some k_i. If the m.c. is v, then the above representation reduces to the $\phi(d,0,q)$ points, $c_0,c_1,\dots,c_{\phi(d,0,q)}$, where

$$\phi(m,d,q) = \frac{(q^{m+1}-1)(q^m-1)\dots(q^{m-d+1}-1)}{(q^{d+1}-1)\dots(q-1)},$$

the number of d-flats in PG(m,q).

iii) A necessary condition for the existence of a d-flat with m.c. θ, $\theta<v$, is that $v=\phi(m,0,q)$, the number of points in PG(m,q), and $\phi(d,0,q)$, the number of points in a d-flat, are not relatively prime. This is actually the requirement that a subgeometry of $\phi(d,0,q)$ points can be formed.

iv) If θ is the m.c. of a d-flat, then a d-flat consisting of points obtained by adding the integer k, k=1,2,..., θ-1, to all the powers of the α's in the original flat, has the same m.c. θ. Thus, for ease of notation, we assume that $c_0=0$ and that from the initial d-flat $V_d(0)$, the d-flats, $V_d(1)$, ..., $V_d(\theta-1)$ can be obtained.

We now present six theorems from Yamamoto et al [58] which we use in the development of the orbit structure and which are necessary to obtain the cycles of the flats of PG(5,2) and PG(7,2). Each theorem is followed by a brief explanation of its derivation.

Theorem 1: If θ_i is integral, $\theta_i = (q^{m+1}-1)/(q^{i+1}-1)$, then $V_i(0) = \{(a_0\alpha^0 + a_1\alpha^{\theta i} + \dots + a_i\alpha^{i\theta i})\}$ is an i-flat of m.c. θ_i.

Since θ_i is integral, $\alpha^{\theta i}$ is a primitive element of $GF(q^{i+1})$ and hence,

$$PG(i,q) = \{(\alpha^0),(\alpha^{\theta i}),\dots,(\alpha^{i\theta i}),\dots,(\alpha^{((q^{i+1}-1)/(q-1)\ -1)\theta})\}.$$

The first (i+1) of these points are linearly independent over GF(q) and hence the linear combination of these points can be used to form PG(i,q). Thus,

$$V_i(0) = \{(a_0\alpha^0 + a_1\alpha^{\theta i} + \dots + a_i\alpha^{i\theta i})\}$$

is an i-flat. That it has m.c. θ_i is a consequence of the fact that any power of α greater than $i\theta_i$ is necessarily a linear combination of the α's of lower power.

Theorem 2: If a d-flat has cycle less than v, then there exists a positive integer j such that (j+1) divides both (m+1) and (d+1), and $\theta=(q^{m+1}-1)/(q^{j+1}-1)$ is the m.c. of V_d. Further, V_d is composed of a particular set of $(q^{d+1}-1)/(q^{j+1}-1)$ j-flats from the set of θ j-flats, $V_j(0)$, $V_j(1)$,...,$V_j(\theta-1)$ generated from the initial j-flat

$$V_j(0) = \{(a_0\alpha^0 + a_1\alpha^\theta + \ldots + a_j\alpha^{j\theta})\}, \text{ m.c. } \theta.$$

The d-flat V_d with m.c. θ, by property ii) of the definition of a cycle, is written as:

$$\begin{array}{l} 0,\ \theta,\ \ldots\ j\theta,\ \ldots,\ (r-1)\theta \\ c_1, c_1+\theta, \ldots\ c_1+j\theta, \ldots,\ (r-1)\theta+c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_s, c_s+\theta, \ldots\ c_s+j\theta, \ldots\ (r-1)\theta+c_s, \quad r=v/\theta. \end{array}$$

For some integer j, $j \leq d$, the j points (α^0), $(\alpha^\theta), \ldots, (\alpha^{j\theta})$ are linearly independent and hence $(\alpha^{(j+1)\theta})$ is a linear combination of these points. Then,

$$V_j(0) = \{(a_0\alpha^0 + a_1\alpha^\theta + \ldots + a_j\alpha^{j\theta})\}$$

is a j-flat with cycle θ. This implies that if (α^c) is any point of $V_j(0)$, then so is $(\alpha^{c+k\theta})$ for any integer k. If (α^b) is any point in V_d, then so must be (α^{b+c}), as the points of V_d are of the form $k\theta+c_i$. So if $(\alpha^0), (\alpha^\theta), \ldots, (\alpha^{j\theta}), (\alpha^{b_1}), \ldots, (\alpha^{b_{d-j}})$ form a basis for V_d, then so also do $(\alpha^c), (\alpha^{c+\theta}), \ldots (\alpha^{c+j\theta}), (\alpha^{c+b_1}), \ldots, (\alpha^{c+b_{d-j}})$. This implies that c is a cycle of V_d since these points generate V_d as well. And, as θ is the m.c. of V_d, c must be a multiple of θ, which implies the points of $V_j(0)$ can be represented as $(\alpha^0), (\alpha^\theta), \ldots, (\alpha^{j\theta}), \ldots, (\alpha^{(r-1)\theta})$. Then, as the number of points in $V_j(0)$ is $(q^{j+1}-1)/(q-1)$, we can substitute this for r and hence

$$\theta = v/r = (q^{m+1}-1)/(q^{j+1}-1),$$

which implies (j+1) divides (m+1) since $(q^{m+1}-1)$ is divisible by $(q^{j+1}-1)$ if and only if (j+1) divides (m+1). Further, V_d consists of (s+1) j-flats, $V_j(0), V_j(c_1), \ldots, V_j(c_s)$ with m.c. θ, and hence

$$(s+1) = \emptyset(d,0,q)/\emptyset(j,0,q) = (q^{d+1}-1)/(q^{j+1}-1)$$

which proves that (j+1) is a divisor of (d+1).

A corollary of this theorem requires terminology which we now define. For $(i+1)$ a factor of both $(m+1)$ and $(d+1)$,

$$V_i(0) = \{(a_0\alpha^0 + a_1\alpha^{\theta_i} + \dots + a_i\alpha^{i\theta_i})\}$$

is an i-flat of m.c. $\theta_i=(q^{m+1}-1)/(q^{i+1}-1)$ from which the θ_i i-flats $V_i(0), V_i(1), \dots, V_i(\theta_i-1)$ having the same m.c. θ_i are obtained. From these θ_i i-flats it is possible to select

$$(d_i+1) = (d+1)/(i+1)$$

flats for which all the respective basis points are linearly independent. The linear combinations of these $(d+1)=(i+1)(d_i+1)$ points generate a d-flat with cycle θ_i. Such a flat is called a "d(i)-flat" generated from (d_i+1) linearly independent i-flats of m.c. θ_i. When the (d_i+1) generating flats are in fact $(d+1)$ points, the corresponding flat is a d(0)-flat. The following corollary is an extension of Theorem 2.

Corollary: A d-flat having m.c. θ less than v is a d(j)-flat for some positive integer j.

Theorem 3: There always exists a d-flat with m.c. v. If there exists a positive integer j such that $(j+1)$ divides both $(m+1)$ and $(d+1)$, then there exists a d-flat with m.c. θ_j,

$$\theta_j = (q^{m+1}-1)/(q^{j+1}-1) < v.$$

The first statement follows from the observation that (α^0), $(\alpha^1), \dots, (\alpha^m)$ are linearly independent points and hence

$$V_d = \{(a_0\alpha^0 + a_1\alpha^1 + \dots + a_d\alpha^d)\}$$

is a d-flat with m.c. v. The second part of the theorem is established in a similar manner. If we set $(m_j+1)=(m+1)/(j+1)$ and $(d_j+1)=(d+1)/(j+1)$, and let α be a primitive element of $GF(q^{m+1})$, then it is also a primitive element of $GF((q^{j+1})^{(m_j+1)})$. Hence, the first (m_j+1) points, (α^0), $(\alpha^1), \dots, (\alpha^{m_j})$ of $PG(m_j, q^{j+1})$ are linearly independent over

$GF(q^{j+1})$. By selecting a particular set of (d_j+1) flats, $V_j(0), V_j(1), \ldots, V_j(d_j)$, from the θ_j j-flats of m.c. θ_j, it can be shown that these flats are linearly independent and generate a d(j)-flat of m.c. θ_j.

Theorem 4: For $(j+1)$ a factor of both $(m+1)$ and $(d+1)$, if there is a d-flat V_d with m.c. $\theta_j=(q^{m+1}-1)/(q^{j+1}-1)$, then V_d is considered both a d(j)-flat and a d(i)-flat for any non-negative integer i such that $(i+1)$ divides $(j+1)$, or i is 0.

To establish this theorem, it is only necessary to examine the representation of the points of the d-flat V_d. A d(j)-flat V_d is generated from (d_j+1) linearly independent j-flats, $V_j(c_o), V_j(c_1), \ldots, V_j(c_{d_j})$, where the points of a component $V_j(c_s)$ flat are given, in terms of powers of α, as:

$$c_s,\ c_s+\theta_j,\ \ldots,\ c_s+(r-1)\theta_j, \qquad r_j=(q^{j+1}-1)/(q-1).$$

These points can be expressed in k groups as follows:

$$\begin{array}{llll} c_s, & c_s+\theta_i, & \ldots, & c_s+(r_i-1)\theta_i \\ c_s+\theta_j, & c_s+\theta_i+\theta_j, & \ldots, & c_s+(r_i-1)\theta_i+\theta_j \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ c_s+(k-1)\theta_j, & c_s+\theta_i+(k-1)\theta_j, & \ldots, & c_s+(r_i-1)\theta_i+(k-1)\theta_j, \end{array}$$

where i is given in the theorem, $\theta_i=(q^{m+1}-1)/(q^{i+1}-1)=k\theta_j$, $k=(q^{j+1}-1)/(q^{i+1}-1)$, and $r_i=(q^{i+1}-1)/(q-1)$. Each of the above k groups is an i-flat with m.c. θ_i. Thus, $V_j(c_s)$ is composed of k i-flats of m.c. θ_i, $V_i(c_s), V_i(c_s+\theta_j), \ldots, V_i(c_s+(k-1)\theta_j)$. And hence, V_d is a d(i)-flat for any i such that $(i+1)$ divides $(j+1)$, or i is zero.

The above theorems guarantee that the totality of d(i)-flats not only contains d(i)-flats of m.c. θ_i but as well, d(j)-flats of m.c. θ_j for any integer j such that θ_j divides θ_i. To obtain n_i^*, the number of d(i)-flats of m.c. θ_i, n_j^*, the number of d(j)-flats of m.c. θ_j, must be subtracted from n_i, the number of d(i)-flats. The next theorem gives the value of n_i.

Theorem 5: The number of d(i)-flats is $n_i=\emptyset(m_i,d_i,q^{i+1})$,

where $m_i=((m+1)/(i+1))-1$, $d_i=((d+1)/(i+1))-1$.

That n_i is as given, can be established by an enumeration argument. The first of the linear independent i-flats can be chosen in $\theta_i=(q^{m+1}-1)/(q^{i+1}-1)$ ways, the second in (θ_i-1) ways, the third in

$$\theta_i - \frac{(q^{2(i+1)}-1)}{(q^{i+1}-1)}$$

ways, and so on, where this selection is simply ensuring the linear independency of the i-flats. The total number of ways of choosing the (d_i+1) linear independent i-flats is

$$T(\theta_i) = \theta_i(\theta_i-1)(\theta_i - \frac{(q^{2(i+1)}-1)}{(q^{i+1}-1)}) \ldots (\frac{\theta_i - (q^{d_i(i+1)}-1)}{(q^{i+1}-1)}).$$

Now, as each d(i)-flat is composed of $s_i=(q^{d+1}-1)/(q^{i+1}-1)$ i-flats which can be generated by any one of

$$T(s_i) = s_i(s_i-1)(s_i - \frac{(m^{2(i+1)}-1)}{(m^{i+1}-1)}) \ldots (s_i - \frac{(m^{d_i(i+1)}-1)}{(m^{i+1}-1)})$$

sets of (d_i+1) independent i-flats, the number of d(i)-flats with cycle θ_i is

$$n_i = T(\theta_i)/T(s_i) = \frac{(Q_i^{m_i+1}-1)(Q_i^{m_i}-1)\ldots(Q_i^{m_i-d_i+1}-1)}{(Q_i^{d_i+1}-1)(Q_i^{d_i}-1)\ldots(Q_i-1)}$$

$$= \emptyset(m_i,d_i,Q_i),$$

where $Q_i=q^{i+1}$.

The above five theorems establish the following theorem which we use to obtain the cycle structure of flats in PG(5,2) and PG(7,2).

Theorem 6: 1) If $(m+1)$ and $(d+1)$ are relatively prime, then all the d-flats of $PG(m,q)$ have m.c. v and can be generated from $\eta=\phi(m,d,q)/v$ initial d-flats.

2) If the Highest Common Factor (HCF) of $(m+1)$ and $(d+1)$ is

$$p_1^{\beta_1} p_2^{\beta_2} \cdots p_e^{\beta_e} > 1,$$

the p_i primes such that $p_i<p_{i+1}$, $i=1,\ldots,e-1$, then the number of distinct m.c.'s is

$$\prod_{i=1}^{e} (1+\beta_i).$$

Let

$$\theta(x_1,\ldots,x_e) = (q^{m+1}-1)/(q^{p_1^{x_1}p_2^{x_2}\cdots p_e^{x_e}} - 1)$$
$$m(x_1,\ldots,x_e) = ((m+1)/(p_1^{x_1}p_2^{x_2}\cdots p_e^{x_e}))-1)$$
$$d(x_1,\ldots,x_e) = ((d+1)/(p_1^{x_1}\cdots p_e^{x_e}))-1$$
$$q(x_1,\ldots,x_e) = (q^{p_1^{x_1}\cdots p_e^{x_e}}).$$

Then the number of $d(p_1^{x_1}\cdots p_e^{x_e})$-flats having the cycle $\theta(x_1,\ldots,x_e)$ and m.c. $\theta(x_1,\ldots,x_e)$ are respectively

$$n(x_1,\ldots,x_e) = \phi(m(x_1,\ldots,x_e),(d(x_1,\ldots,x_e),q(x_1,..x_e))$$
$$n^*(\beta_1,\ldots,\beta_e) = n(\beta_1,\ldots,\beta_e)$$
$$n^*(x_1,\ldots,x_e) = n(x_1,\ldots,x_e) - \sum_{\substack{x_j\le y_j\le\beta_j,\\ \exists j,\, x_j<y_j.}} n^*(y_1,\ldots,y_e).$$

The number of initial d-flats of any m.c. $\theta(x_1,\ldots,x_e)$ is $\eta(x_1,\ldots,x_e)=n^*(x_1,\ldots,x_e)/\theta(x_1,\ldots,x_e)$ from which the totality of d-flats having m.c. $\theta(x_1,\ldots,x_e)$ can be generated.

The theorems of Yamamoto et al.[58] provide a means of partitioning the flats of a finite projective geometry. In the following, we extend this concept and develop a non-orthogonal decoding method for order-$(m-2)$ PG codes over $PG(m,2)$.

4.3 Cycle Description of PG(5,2)

We now apply the results obtained in Section 4.2 to the PG(5,2). We begin by finding the cycles of the 3-flats of PG(5,2). These flats form the null space of the order-3 (63,41) PG code over PG(5,2), which, in Chapter 5, is decoded using a decoding algorithm related to the cycles of the flats. The null space 3-flats are obtained and partitioned using Yamamoto et al's [58] sixth theorem for m=5, d=3 and $p^s=q=2$.

The second part of Theorem 6 is applied for the 3-flats since the highest common factor of (m+1,d+1) is

$$\mathrm{HCF}(6,4) = 2 \neq 1.$$

Using this theorem, we obtain the number of distinct cycles, the values of the cycles, the number of 3-flats with a given cycle and the number of initial 3-flats (i3f) of PG(5,2). Since $p_1=2$ and $\beta_1=1$, the number of m.c.'s is (1+1)=2. From the third theorem above, we know that one of the m.c.'s is v=63. For $x_1=1$ and $x_1=0$ in Theorem 6 we have:

$x_1=1$	$x_1=0$
$\theta(1) = (2^6-1)/(2^2-1) = 21$	$\theta(0) = (2^6-1) = 63$
$m(1) = (6/2)-1 = 2$	$m(0) = (6-1) = 5$
$d(1) = (4/2)-1 = 1$	$d(0) = (4-1) = 3$
$q(1) = 2^2 = 4$	$q(0) = 2$
$n^*(1) = n(1) = \phi(2,1,4) = 21$	$n(0) = \phi(5,3,2) = 651$
$\lambda(1) = 21/21 = 1$	$n^*(0) = 651-21 = 630$
	$\lambda(0) = 630/63 = 10.$

These calculations show that there are ten i3f's of m.c. 63 and one i3f of m.c. 21. Each of the ten i3f's of m.c. 63 generates 63 distinct 3-flats. Recalling the discussion on Projective Geometries, we note that each 3-flat can be represented by a 63-tuple of zeros and ones. In the 63-tuple, each position refers to a point of the geometry. A one appears in each position which corresponds to a point in the 3-flat. The resulting 63-tuple is an incidence vector of the 3-flat. We let (α^i) refer to the point in position i, i=0, 1,...,62, α a primitive element of GF(2^6). In the remainder of the thesis a point is referrred to as i, rather than (α^i), i=0,1,...,62, and a flat as a set of these points listed in ascending order of magnitude. The incidence vectors of the 63 flats generated by an i3f of m.c. 63 are obtained by cyclically shifting the i3f incidence vector i positions to the left for i=0,1,...,62. The point representations of the flats are generated by subtracting i from each i3f mod 63, i=0,1,...,62. Similarly the 21 3-flats generated from the i3f of m.c. 21 are obtained by cyclically shifting the incidence vector of the i3f i positions to the left, or subtracting i, mod 63, from the point representation of the i3f, i=0,1,...,20.

The PG codes are cyclic and so, when decoding them, it is only necessary to consider those 3-flats in the null space which contain the point 0. This reduction of the null space is possible because once the digit in position 0 has been decoded, the codeword can be shifted once and the same procedure used to decode the next digit. Thus, for each of the ten i3f's of m.c. 63, we need generate only the fifteen 3-flats which intersect on 0. That there are fifteen such flats can be established by the following argument. Each 3-flat has $\phi(5,0,3)$=15 points, which can be represented by the integers $(i_0,i_1,\ldots,i_{14})$. By successively subtracting i_j, j=0,1,...,14, mod 63, from the point representation of the flat, fifteen distinct 3-flats through the point 0 are obtained.

Referring to Theorem 2 above, we note that the i3f of m.c. 21 is composed of five 1-flats, one of which contains the point 0. The other four flats are shifted versions of the first. In terms of initial flats, the first 3-flat through 0 can be written as $V_3(0)$ and hence the others as $V_3(c_i)$ for the four positive integers c_i, $i=1,\ldots,4$. After shifting the first five incidence ones to position 0, the digits in the incidence vector must again be in the same relative positions because the points not in $V_3(0)$ are simply shifted versions of these points. If we continue to shift the 63-tuple, the same five 3-flats are generated a second time. As a result, the i3f of m.c. 21 generates only five distinct 3-flats passing through the point 0.

In later sections, the cycle description of the 1-flats of PG(5,2) is required. Thus, we apply Theorem 6 above for d+1. In this case, (d+1)=2, so

$$HCF(m+1,d+1) = HCF(6,2) = 2$$

and again there are (1+1)=2 distinct cycles, one of which must be 63, by Theorem 3. Applying Theorem 6 for $x_1=1$ and $x_1=0$ we have:

$x_1=1$	$x_0=0$
$\theta(1) = (2^6-1)/(2^2-1) = 21$	$\theta(0) = (2^6-1)/(2-1) = 63$
$m(1) = ((5+1)/2)-1) = 2$	$m(0) = (6-1) = 5$
$d(1) = (2/2)-1 = 0$	$d(0) = (2-1) = 1$
$q(1) = 2^2 = 4$	$q(0) = 2$
$n^*(1) = n(1) = \emptyset(2,0,4) = 21$	$n(0) = \emptyset(5,1,2) = 651$
$\lambda(1) = 21/21 = 1$	$n^*(0) = 651-21 = 630$
	$\lambda(0) = 630/63 = 10$

Interpreting the above, there are ten i1f's of m.c. 63 and one i1f of m.c. 21. Again each i1f of m.c. 63 generates 63 1-flats and the i1f of m.c. 21 generates 21 1-flats. If we consider the 1-flats as those flats which are obtained after two levels of majority logic decoding, then, as for the 3-flats, it is only necessary to know the 1-flats which pass through 0. As there are $\emptyset(5,0,1)=3$ points in each 1-flat, we need only consider the three 1-flats generated from each i1f of m.c. 63 which contain the point 0. From Theorem 2 above, the i1f with m.c. 21 is the flat $(a_0\alpha^0 + a_1\alpha^{21})$, a_0 and a_1 from GF(2). Irrespective of the minimal polynomial chosen to represent PG(5,2) the points of this flat are 0, 21 and 42. That this is so, is established in Section 4.6. Thus, as for the 3-flat case, cyclically shifting this flat 21 positions to obtain a one in position 0 of the incidence vector, gives the i1f with which we began. As a result, the i1f of m.c. 21 has only one distinct flat which passes through the point 0.

In the following section we develop further the ideas presented here.

4.4 Orbits of PG(5,2)

In this section we introduce the orbits, the structures which form the basis of the simplified decoder of the Projective Geometry Codes. First, however, it is necessary to define the transformation Z. We denote by Z that transformation which takes the point representation of an i-flat, V_i, and subtracts 1 from each point, mod $(2^{m+1}-1)$, to give the i-flat $Z(V_i)$ in PG(m,2), where in this chapter m=5. This corresponds to dividing the polynomial representation $V_i(X)$ of the i-flat V_i by X, mod $(X^{63}-1)$, or, to cyclically shifting the incidence vector of V_i one position to the left. For V_i an i-flat, we define Z^n by

$$Z^n(V_i) = Z(Z^{n-1}(V_i)), \quad n>1.$$

The set

$$\{Z, Z^2, \ldots, Z^{63} = Z^o = e\}$$

forms a group over the set of all i-flats in PG(5,2). For i=3 and i=1, this group partitions the i-flats into orbits, where each orbit corresponds to one of the iif's and the i-flats generated from it. Thus, there are ten orbits with 63 members each and one orbit with 21 members. For decoding purposes, we are interested only in those flats containing the point 0. Therefore, we define a 0_i-orbit to be the subset of an orbit consisting of only those i-flats in which the point 0 occurs. The '0_i' in the term '0_i-orbit' refers to the fact that the constituent flats are i-flats through the point 0.

For i=3, each of the 0_3-orbits of flats with m.c. 63 has fifteen members. The 0_3-orbit corresponding to the i3f of m.c. 21 has five members. The 0_1-orbits with flats of m.c. 63 have three members each, the 0_1-orbit with the flat of m.c. 21 has one member. These observations follow from the discussion in Section 4.3 concerning the number of flats which pass through the point 0.

Thus, there is a one to one correspondence between the eleven orbits and the eleven iif's and between the 0_i-orbits and the iif's through the point 0, i=1,3.

To illustrate these concepts, we list the 0_1-orbits of PG(5,2) in Table 4.4.1, where the 1-flats are given using the exponential representation. The three 1-flats contained in any given 0_1-orbit are labelled 'a', 'b' and 'c', where the 'a' 1-flat is the i1f of the 0_1-orbit, and the 'b' and 'c' flats are obtained by applying the transformation Z successively to the i1f until two 1-flats through the point 0 are obtained. The 0_1-orbits are numbered $(1_1),(2_1),\ldots,(11_1)$, where the subscript denotes the dimension of the constituent flats. The 1-flats a, b and c of (t_1) are referred to as t_1a, t_1b and t_1c, respectively, $t=1,\ldots,10$. The single 1-flat in (11_1) is labelled 11_1a. The eleven 0_1-orbits are grouped into four distinct classes I_1, II_1, III_1, IV_1, the subscript denoting the dimension of the flats in the class. These classes are defined later.

I_1	(1_1)	a:	0	1	6
		b:	0	5	62
		c:	0	57	58
	(2_1)	a:	0	2	12
		b:	0	10	61
		c:	0	51	53
	(3_1)	a:	0	4	24
		b:	0	20	59
		c:	0	39	43
	(4_1)	a:	0	8	48
		b:	0	40	55
		c:	0	15	23
	(5_1)	a:	0	16	33
		b:	0	17	47
		c:	0	30	46
	(6_1)	a:	0	3	32
		b:	0	31	34
		c:	0	29	60
II_1	(7_1)	a:	0	7	26
		b:	0	19	56
		c:	0	37	44
	(8_1)	a:	0	14	52
		b:	0	38	49
		c:	0	11	25
	(9_1)	a:	0	28	41
		b:	0	13	35
		c:	0	22	50
III_1	(10_1)	a:	0	9	45
		b:	0	36	54
		c:	0	18	27
IV_1	(11_1)	a:	0	21	42

Table 4.4.1: 0_1-orbits of PG(5,2), minimal polynomial (0 1 6)

We note that the minimal polynomial used in the representation of PG(5,2) given in Table 4.4.1 is $m(X)=1+X+X^6$. Throughout the thesis this minimal polynomial is used to represent PG(5,2). We show in Section 4.6 that the representation of PG(5,2) is well-defined, that is independent of the minimal polynomial chosen to represent it. This implies that the 0_i-orbit structure is identical for each representation of PG(5,2).

The 1-flats of Table 4.4.1 are generated by selecting two independent points, one of which is 0, and forming the corresponding 1-flat. This flat is then cyclically shifted to form the 0_1-orbit. Once an ilf from each of the classes I_1, II_1 and III_1 is found, the remaining ilf's in each of these classes are formed by multiplying the exponent set of each of the ilf's by 2, mod 63, successively until all the initial flats are obtained. This process is explained in more detail below. Computer programs were written to generate the i-flats in the geometry and to partition them into 0_i-orbits, i=1,2,3.

We now exhibit a most interesting and useful correspondence between the 0_1-orbits and the 0_3-orbits.

In any 0_3-orbit of m.c. 63, the element 0, by construction, is present in all fifteen of the 0_3-orbit 3-flats. Inspection of the point sets of each of these 0_3-orbits shows that a set of six of the 62 non-zero points of the geometry occurs seven times and the remaining 56 elements each occur only three times. Moreover, the set of six points which occurs seven times in a given 0_3-orbit is distinct from the set which occurs in any other 0_3-orbit. An analysis shows that each such set consists of the six non-zero points of a particular 0_1-orbit. Thus, a one to one correspondence can be established between the 0_3-orbits of m.c. 63 and the 0_1-orbits of m.c. 63, where an orbit is said to have the m.c. of its constituent flats. The 0_3-orbits are therefore numbered $(1_3),(2_3),\ldots,(10_3)$ to exhibit this correspondence. The 0_3-orbit (t_3) contains the six non-zero points of the 0_1-orbit (t_1) seven times, t=1,...,10. The 0_3-orbit of m.c. 21 corresponds to the 0_1-orbit (11_1). It repeats the two non-zero points of (11_1) five times each, that is points 21 and 42 appear in each of the five 3-flats which compose (11_3). Every other non-zero point of the geometry appears once only.

Of the fifteen 3-flats in any 0_3-orbit (t_3) of m.c. 63, there are two 3-flats which do not contain a 1-flat from (t_1), t=1,...,9. For the purposes of this study we omit these two 3-flats and refer to the 0_3-orbit (t_3) as the thirteen 3-flats which intersect on 0 and contain at least one 1-flat from (t_1).

We now investigate the structure of the 0_3-orbits (1_3) through (9_3). The 3-flats of each 0_3-orbit (t_3), t=1,...,9, can be ordered so that, representing each 3-flat by the 1-flats a, b and c of (t_1) that it contains, the following description of the 3-flats in terms of the 1-flats is obtained:

a,a,b,b,c,c,ab,ab,ac,ac,bc,bc,abc.

In this representation, a single letter indicates that only one (t_1) 1-flat is present in the (t_3) 3-flat, a pair of letters that two (t_1) 1-flats are in the (t_3) 3-flat and abc that all three (t_1) 1-flats occur in the (t_3) 3-flat. More explicitly, the 1-flat t_1a is the only(t_1) 1-flat in two of the thirteen 3-flats of (t_3), appears with the 1-flat t_1b twice and with the 1-flat t_1c twice. All three 1-flats occur together in one of the thirteen 3-flats. Observing this, each 0_3-orbit (t_3) can be divided into three intersecting subsets, A_t, B_t, C_t, where, for example, the set A_t consists of the (t_3) 3-flats, in the representation above, which contain the (t_1) 1-flat a:

a,a,ab,ab,ac,ac,abc.

Thus, A_t is the subset which contains the seven 3-flats which intersect on t_1a. Flats t_1b and t_1c occur three times each in A_t. We refer to the elements which occur seven times as 7-repeats and those which occur three times as 3-repeats. The remaining points of a given subset are either 3-repeats or 1-repeats of points not in (t_1).

We note that the subsets A_t, B_t and C_t are not orthogonal on t_1a since there are three occurrences of the 1-flat t_1b and of the 1-flat t_1c in the seven 3-flats composing A_t. We illustrate later the advantages of the non-orthogonal orbit structure for decoding.

The thirteen 3-flats in (t_3) are labelled to reflect further the correspondence of (t_3) with (t_1), t=1,...,9. Each 3-flat in (t_3) contains at least one 1-flat from(t_1). We label the 3-flats of (t_3) with the label of these contained (t_1) 1-flats. In (t_3) there are two 3-flats which contain each of the 1-flat groups a,b,c,ab,ac,bc from (t_1). These two 3-flats of (t_3) are distinguished by the subscripts 'i' and 'ii'. For example, the 3-flats of (t_3) which contain both t_1a and t_1b are denoted as t_3ab_i and t_3ab_{ii}.

In the discussion of the O_3-orbits, we have omitted (10_3) and (11_3). This is due to the unique structures of both these orbits. These structures allow (10_3) and (11_3) to be used for a special purpose in the decoding algorithm. We now describe the structures of these two O_3-orbits. Although every non-zero point of (10_1) appears seven times and the other points of the geometry three times each in (10_3), the repeat pattern of the 1-flats a, b and c of (10_1) is, using the representation above:

abc,abc,abc,abc,abc,abc,abc,

that is, all six non-zero elements of (10_1) appear in seven of the 3-flats of (10_3) and do not appear at all in the remaining eight 3-flats of (10_3). We omit those 3-flats in (10_3) which do not contain a 1-flat of (10_1), to yield seven members only in (10_3). Consequently, in (10_3) there is only one subset, say A_{10}, consisting of seven 3-flats, all of which contain the 1-flats 10_1a, 10_1b and 10_1c. The same lettering method is used to label the 3-flats of (10_3) as was used for the (t_3) O_3-orbits, t=1,...,9. However, it is less illustrative in this case since all the seven 3-flats are labelled 10_3abc, with subscripts i,ii,...,vii.

In (11_3) we have the single 1-flat 11_1a occurring in each of the five 3-flats. Thus, each of the 3-flats is labelled 11_3a, with subscripts i,ii,iii,iv,v.

In the next section we analyse the O_3-orbits in terms of the O_1-orbits, illustrating the numerous symmetries made apparent by this arrangement of the 3-flats.

4.5 Symmetries in the PG(5,2) O_i-Orbits

The analysis of PG(5,2), based on the O_i-orbit structure, is completed below. A second transformation, which operates on the flats represented as point sets, is introduced, allowing for a further investigation of the symmetrical properties of the O_i-orbits.

We introduce the analysis of the O_i-orbits with Tables 4.5.1 and 4.5.2 which illustrate several symmetries of the O_i-orbits. The first Table, Table 4.5.1, consists of the 7-repeats and 3-repeats of each O_3-orbit. In Table 4.5.2, each O_3-orbit 3-flat is represented in terms of its constituent O_1-orbit 1-flats.

O_3-orbit	7-repeat subset	3-repeats (as 1-flats)
(1_3)	A_1	1b 1c 2a 3b 7a 8c
	B_1	1a 1c 2b 3a 7b 8c
	C_1	1a 1b 2c 3b 7b 8a
(2_3)	A_2	2b 2c 3a 4b 8a 9c
	B_2	2a 2c 3b 4a 8b 9c
	C_2	2a 2b 3c 4b 8b 9a

I_3	(3_3)	A_3	3b 3c 4a 5b 9a 7c
		B_3	3a 3c 4b 5a 9b 7c
		C_3	3a 3b 4c 5b 9b 7b
	(4_3)	A_4	4b 4c 5a 6b 7b 8c
		B_4	4a 4c 5b 6a 7a 8c
		C_4	4a 4b 5c 6b 7a 8b
	(5_3)	A_5	5b 5c 6a 1b 8b 9c
		B_5	5a 5c 6b 1a 8a 9c
		C_5	5a 5b 6c 1b 8a 9b
	(6_3)	A_6	6b 6c 1a 2b 9b 7c
		B_6	6a 6c 1b 2a 9a 7c
		C_6	6a 6b 1c 2b 9a 7a
II_3	(7_3)	A_7	7b 7c 8a 10a 2c 5a
		B_7	7a 7c 8b 10a 2a 5c
		C_7	7a 7b 8c 10c 2c 5c
	(8_3)	A_8	8b 8c 9a 10c 3c 6a
		B_8	8a 8c 9b 10c 3a 6c
		C_8	8a 8b 9c 10b 3c 6c
	(9_3)	A_9	9b 9c 7b 10b 4c 1a
		B_9	9a 9c 7a 10b 4a 1c
		C_9	9a 9b 7c 10a 4c 1c
			1-repeats
III_3	(10_3)	A_{10}	2b 5b 1c 4c
			3b 6b 2c 5c
			4b 1b 3c 6c
			1a 4a 8a 8b
			2a 5a 9a 9b
			3a 6a 7a 7b
			7c 8c 9c 11a
		5-repeat subset	
IV_3	(11_3)	A_{11}	1c 4c 2a 5a 7c 10b
			2c 5c 3a 6a 8c 10a
			3c 6c 1a 4a 9c 10c
			1b 3b 5b 7a 8b 9a
			4b 6b 2b 7b 8a 9b

Table 4.5.1: 7-repeats and 3-repeats of I_3, II_3
7-repeats and 1-repeats of III_3
5-repeats and 1-repeats of IV_3

Note: The subscript "1" has been omitted on the 3/1-repeat column as all the constituent flats are 1-flats.

O_3-orbit (t_3)	(t_3) 3-flat j	1-flats in j_i	1-flats in j_{ii}
(1_3)	a	1a 2a 4a 6a 7a 9b 10c	1a 3c 4b 7c 8b 9c 10a
	b	1b 2b 3a 4c 6a 6c 10a	1b 3c 4a 7c 8b 10b 11a
	c	1c 2c 4a 5b 5c 8a 10a	1c 3c 5a 6a 6b 7c 8b
	ab	1a 1b 2a 5b 6b 7a 8c	1a 1b 2b 3a 5a 5c 8c
	ac	1a 1c 2a 3b 6c 7a 11a	1a 1c 2c 3b 4c 8a 10b
	bc	1b 1c 2b 3a 4b 7b 10c	1b 1c 2c 7b 8a 9b 9c
	abc	1a 1b 1c 3b 7b 8c 9a	
(2_3)	a	2a 3a 5a 1a 8a 7a 10b	2a 4c 5b 8c 9b 7c 10c
	b	2b 3b 4a 5c 1a 1c 10c	2b 4c 5a 8c 9b 10a 11a
	c	2c 3c 5a 6b 6c 9a 10c	2c 4c 6a 1a 1b 8c 9b
	ab	2a 2b 3a 6b 1b 8a 9c	2a 2b 3b 4a 6a 6c 9c
	ac	2a 2c 3a 4b 1c 8a 11a	2a 2c 3c 4b 5c 9a 10a
	bc	2b 2c 3b 4a 5b 8b 10b	2b 2c 3c 8b 9a 7a 7c
	abc	2a 2b 2c 4b 8b 9c 7b	
(3_3)	a	3a 4a 6a 2a 9a 8a 10a	3a 5c 6b 9c 7a 8c 10b
	b	3b 4b 5a 6c 2a 2c 10b	3b 5c 6a 9c 7a 10c 11a
	c	3c 4c 6a 1b 1c 7b 10b	3c 5c 1a 2a 2b 9c 7a
I_3	ab	3a 3b 4a 1b 2b 9a 7c	3a 3b 4b 5a 1a 1c 7c
	ac	3a 3c 4a 5b 2c 9a 11a	3a 3c 4c 5b 6c 7b 10c
	bc	3b 3c 4b 5a 6b 9b 10a	3b 3c 4c 9b 7b 8a 8c
	abc	3a 3b 3c 5b 9b 7c 8b	
(4_3)	a	4a 5a 1a 3a 7b 9a 10c	4a 6c 1b 7c 8a 9c 10a
	b	4b 5b 6a 1c 3a 3c 10a	4b 6c 1a 7c 8a 10b 11a
	c	4c 5c 1a 2b 2c 8b 10a	4c 6c 2a 3a 3b 7c 8a
	ab	4a 4b 5a 2b 3b 7b 8c	4a 4b 5b 6a 2a 2c 8c
	ac	4a 4c 5a 6b 3c 7b 11a	4a 4c 5c 6b 1c 8b 10b
	bc	4b 4c 5b 6a 1b 7a 10c	4b 4c 5c 7a 8b 9a 9c
	abc	4a 4b 4c 6b 7a 8c 9b	
(5_3)	a	5a 6a 2a 4a 8b 7b 10b	5a 1c 2b 8c 9a 7c 10c
	b	5b 6b 1a 2c 4a 4c 10c	5b 1c 2a 8c 9a 10a 11a
	c	5c 6c 2a 3b 3c 9b 10c	5c 1c 3a 4a 4b 8c 9a
	ab	5a 5b 6a 3b 4b 8b 9c	5a 5b 6b 1a 3a 3c 9c
	ac	5a 5c 6a 1b 4c 8b 11a	5a 5c 6c 1b 2c 9b 10a
	bc	5b 5c 6b 1a 2b 8a 10b	5b 5c 6c 8a 9b 7b 7c
	abc	5a 5b 5c 1b 8a 9c 7a	
(6_3)	a	6a 1a 3a 5a 9b 8b 10a	6a 2c 3b 9c 7b 8c 10b
	b	6b 1b 2a 3c 5a 5c 10b	6b 2c 3a 9c 7b 10c 11a
	c	6c 1c 3a 4b 4c 7a 10b	6c 2c 4a 5a 5b 9c 7b
	ab	6a 6b 1a 4b 5b 9b 7c	6a 6b 1b 2a 4a 4c 7c
	ac	6a 6c 1a 2b 5c 9b 11a	6a 6c 1c 2b 3c 7a 10c
	bc	6b 6c 1b 2a 3b 9a 10a	6b 6c 1c 9a 7a 8b 8c
	abc	6a 6b 6c 2b 9a 7c 8a	
(7_3)	a	7a 3b 4b 4c 5a 8a 11a	7a 1b 2b 3a 3c 5b 10b
	b	7b 1b 1c 2a 6b 8b 11a	7b 2b 4b 5b 6a 6c 10b
	c	7c 1a 1b 4a 4b 8c 10c	7c 2b 5b 9a 9b 10b 11a
	ab	7a 7b 1a 2a 8b 9a 10a	7a 7b 4a 5a 8a 9b 10a
	ac	7a 7c 2c 6a 6b 8c 10c	7a 7c 1c 2c 5a 6c 8a
	bc	7b 7c 2a 3c 4c 5c 8b	7b 7c 3a 3b 5c 8c 10c
	abc	7a 7b 7c 2c 5c 9c 10a	

	(8_3)	a	8a 4b 5b 5c 6a 9a 11a	8a 2b 3b 4a 4c 6b 10a
		b	8b 2b 2c 3a 1b 9b 11a	8b 3b 5b 6b 1a 1c 10a
		c	8c 2a 2b 5a 5b 9c 10b	8c 3b 6b 7a 7b 10a 11a
		ab	8a 8b 2a 3a 9b 7b 10c	8a 8b 5a 6a 9a 7a 10c
		ac	8a 8c 3c 1a 1b 9c 10b	8a 8c 2c 3c 6a 1c 9a
		bc	8b 8c 3a 4c 5c 6c 9b	8b 8c 4a 4b 6c 9c 10b
II_3		abc	8a 8b 8c 3c 6c 7c 10c	
	(9_3)	a	9a 5b 6b 6c 1a 7b 11a	9a 3b 4b 5a 5c 1b 10c
		b	9b 3b 3c 4a 2b 7a 11a	9b 4b 6b 1b 2a 2c 10c
		c	9c 3a 3b 6a 6b 7c 10a	9c 4b 1b 8a 8b 10c 11a
		ab	9a 9b 3a 4a 7a 8b 10b	9a 9b 6a 1a 7b 8a 10b
		ac	9a 9c 4c 2a 2b 7c 10a	9a 9c 3c 4c 1a 2c 7b
		bc	9b 9c 4a 5c 6c 1c 7a	9b 9c 5a 5b 1c 7c 10a
		abc	9a 9b 9c 4c 1c 8c 10b	

Table 4.5.2: O_3-orbits expressed as 1-flats for classes I_3 and II_3.

Before analysing these Tables, we define the transformation g. We denote by g that transformation which assigns to any point (α^i) of the geometry, the point (α^{2i}), the exponents mod 63. Using the exponent representation mentioned earlier, we say that g takes any point i to the point 2i, mod 63, that is

$$g(i) = 2i$$

$$g^n(i) = g(g^{n-1}(i)), \quad n>1.$$

The group G of transformations is then

$$G = \{g, g^2, \ldots, g^6 = g^o = e\},$$

where g^i is 2^i. We note that $g^6 = 2^6$ is the identity because, for α a primitive element, $\alpha^{63}=1$ and hence $\alpha^{64} = 1$ or $2^6=1$, mod 63. The group G operates on the 3-flats and 1-flats by operating on each point of the flat. Thus, we can write g as a function of a flat. For example,

$$g^2(1_1a) = g(g(1_1a)) = g(g(0\ 1\ 6))$$
$$= g(0\ 2\ 12) = g(2_1a) = (0\ 4\ 24) = 3_1a.$$

We observe that the transformation g is a one to one transformation of the set of all 0_i-orbit i-flats to itself, that is g is a permutation. For example, g establishes the following mappings:

$1_1a \longrightarrow 2_1a$, $2_1a \longrightarrow 3_1a$, $3_1a \longrightarrow 4_1a$,
$4_1a \longrightarrow 5_1a$, $5_1a \longrightarrow 6_1a$, $6_1a \longrightarrow 1_1a$,

which can be written as the cycle $(1_1a\ 2_1a\ 3_1a\ 4_1a\ 5_1a\ 6_1a)$, of order six, where the order of a cycle is the number of elements in the cycle. We note that the term cycle used here is distinct from the cycle of Yamamoto et al [58]. The correct usage is determined by the context. By applying the transformation g to the 1-flats of each 0_1-orbit, the 1-flats can be divided into the four disjoint cycle classes, I_1, II_1, III_1 and IV_1, given in Table 4.5.3.

Class	1-flat Cycles	Order of Cycle
I_1	(1a 2a 3a 4a 5a 6a)	6
	(1b 2b 3b 4b 5b 6b)	6
	(1c 2c 3c 4c 5c 6c)	6
II_1	(7a 8a 9a 7b 8b 9b)	6
	(7c 8c 9c)	3
III_1	(10a 10b 10c)	3
IV_1	(11a)	1

Table 4.5.3: 1-flat Cycles

Similarly, g can be applied to the 3-flats of each 0_3-orbit to give the classes I_3, II_3, III_3, IV_3. We note that in I_3, the subscripts 'i', 'ii' on the 3-flats can be omitted as g takes 'i' ('ii') 3-flats only to 'i' ('ii') 3-flats. This is also true in the II_3 class for all but the 7_3ab_i, 7_3ac_i and 7_3bc_i cycles. For the three cycles mentioned, g takes an 'i' 3-flat to an 'ii' 3-flat and hence the subscripts must be included. For example, we have $(7_3ab_i\ 8_3ab_i\ 9_3ab_i\ 7_3ab_{ii}\ 8_3ab_{ii}\ 9_3ab_{ii})$. The 3-flat cycles are listed in Table 4.5.4. The seven 3-flats in (10_3) are distinguished by the subscripts i,ii,...,vii, where, in Table 4.5.1 they appear in this order. The five 3-flats of (11_3a) are subscripted similarly. In Table 4.5.1, the five 3-flats of A_{11} appear in the order i, ii,iii,iv,v.

Class	3-flat Cycles	Cycle Order
I_3	$(1_3a\ 2_3a\ 3_3a\ 4_3a\ 5_3a\ 6_3a)$	6
	$(1_3b\ 2_3b\ 3_3b\ 4_3b\ 5_3b\ 6_3b)$	6
	$(1_3c\ 2_3c\ 3_3c\ 4_3c\ 5_3c\ 6_3c)$	6
	$(1_3ab\ 2_3ab\ 3_3ab\ 4_3ab\ 5_3ab\ 6_3ab)$	6
	$(1_3ac\ 2_3ac\ 3_3ac\ 4_3ac\ 5_3ac\ 6_3ac)$	6
	$(1_3bc\ 2_3bc\ 3_3bc\ 4_3bc\ 5_3bc\ 6_3bc)$	6
	$(1_3abc\ 2_3abc\ 3_3abc\ 4_3abc\ 5_3abc\ 6_3abc)$	6
II_3	$(7_3a\ 8_3a\ 9_3a\ 7_3b\ 8_3b\ 9_3b)$	6
	$(7_3c\ 8_3c\ 9_3c)$	3
	$(7_3ab_i\ 8_3ab_i\ 9_3ab_i\ 7_3ab_{ii}\ 8_3ab_{ii}\ 9_3ab_{ii})$	6
	$(7_3ac_i\ 8_3ac_i\ 9_3ac_i\ 7_3bc_{ii}\ 8_3bc_{ii}\ 9_3bc_{ii})$	6
	$(7_3bc_i\ 8_3bc_i\ 9_3bc_i\ 7_3ac_{ii}\ 8_3ac_{ii}\ 9_3ac_{ii})$	6
	$(7_3abc\ 8_3abc\ 9_3abc)$	3
III_3	$(10_3abc_i\ 10_3abc_{ii}\ 10_3abc_{iii})$	3
	$(10_3abc_{iv}\ 10_3abc_v\ 10_3abc_{vi})$	3
	(10_3abc_{vii})	1
IV_3	$(11_3a_i\ 11_3a_{ii}\ 11_3a_{iii})$	3
	$(11_3a_{iv}\ 11_3a_v)$	2

Table 4.5.4: 3-flat Cycles

Referring to Table 4.5.3 and Table 4.5.4, we observe that the cycle structure can be represented at a gross level as:

I_g: (1 2 3 4 5 6)

II_g: (7 8 9)

III_g: (10)

IV_g: (11),

where the cycle element t refers to the O_i-orbit (t_i), i=1,3, t=1,2,...,11, and the subscript g denotes that these cycles are induced by the transformation g.

The structure of cycles and orbits forms the basis of the simplified decoding method presented in the next chapter. The set of 3-flats, expressed as O_1-orbit 1-flats, that were selected from the null space and arranged in the O_3-orbits, have many symmetric interpretations. We now discuss these symmetries in detail and later show that a knowledge of them greatly reduces the complexity of the Majority Logic Decoder for the PG code based on PG(5,2). Moreover, the O_i-orbits provide for a mathematically interesting analysis of the composition and structure of the flats of PG(5,2).

In order to provide a more précise analysis of the symmetrical properties of the O_3-orbits, we introduce more terminology. The O_3-orbit (10_3), as we noted earlier, plays an important role in the decoding process of Chapter 5. Also, it partitions the O_1-orbit 1-flats into distinct blocks of flats, such that no 1-flat, except from (10_1), appears in more than one block. We refer to these blocks as symmetry blocks because they are the basis of many of the symmetric distributions found in the O_3-orbits and because sets of errors which have the same symmetric block composition are treated similarly by the decoder. The symmetric blocks are the 1-repeat 1-flats of (10_3), plus the 1-flats of (10_1).

S_1:	2b	5b	1c	4c
S_2:	3b	6b	2c	5c
S_3:	4b	1b	3c	6c
S_4:	1a	4a	8a	8b
S_5:	2a	5a	9a	9b
S_6:	3a	6a	7a	7b
S_7:	7c	8c	9c	11a
S_8:	10a	10b	10c	

Table 4.5.5: Symmetry Blocks of PG(5,2)

In the following analyses, the symmetry blocks play an important role. In particular, those pairs of 1-flats in symmetry blocks S_1 to S_6 from the same cycle of I_1 or II_1, for example 1c 4c in S_1, are symmetrically distributed in the O_3-orbits. The members of S_7 represent the two distinct cycles, (7c 8c 9c) and (11a). As the symmetry blocks are of prime importance, both in the analysis of O_i-orbit symmetries and in the defining of the O_i-orbit decoder, we investigate them in more detail. The symmetry blocks bring to light the fact that each element of an order 6 cycle is symmetrically related to the element in the cycle three cycle positions away. If we examine the cycle in II_1 of order 6, (7a 8a 9a 7b 8b 9b), we observe that this can be considered to be composed of an 'a' semi-cycle plus a 'b' semi-cycle, each of order 3. Here, the element 7a, after 3 cyclic shifts does not reappear. However the element that does, 7b, is from the same O_1-orbit as 7a. We refer to the element t_1j' as the symmetric relative of t_1j, for t=7,8,9, j=a,b and j'=a(b) if j=b(a). To complete the definition for the II_1 class, we say that t_1c is the symmetric relative of t_1c, t=7,8,9. Any two symmetric relatives are separated by three cyclic shifts. The same terminology can be used to describe the I_1 cycles. If we look again at the symmetry blocks, we note the pairing of cycle members of I_1 which are separated by three cycle shifts, 1_1j and 4_1j, 5_1j and 2_1j, 3_1j and 6_1j, j=a,b,c. As above, we refer to these pairs, a distance of three cycle shifts apart, as symmetric relatives. In symmetry blocks S_1 to S_6, the symmetric relatives, t_1j and $(t+3)_1j$, appear as pairs, j=a,b,c, t=1,2,3. Also, the symmetric relatives t_1a and t_1b, t=7,8,9, appear in symmetry blocks S_4, S_5, S_6.

For consistency, we consider each orbit (t) of I_g to possess the symmetric relative (t+3), t=1,2,3. We note that there are no corresponding symmetry blocks or relatives defined on the 3-flats. In the following, we refer frequently to both the symmetry blocks and the symmetric relatives.

If we now examine Table 4.5.1, several examples can be found of the symmetric distribution of the 1-flats. We interpret these by referring to the symmetry blocks and symmetric relatives. In Table 4.5.1, the O_3-orbit (1_3) and its symmetric relative in I_g, (4_3), have the 1-flats 7a,7b,7b,8c, 8c,8a and 7b,7a,7a,8c,8c,8b, respectively. Thus, the O_3-orbit (4_3) contains the symmetric relatives of the II_1 1-flats which occur in (1_3). Moreover, the I_1 class 3-repeats in (1_3) and (4_3) are symmetric relatives as well. For instance, in (1_3), 2a,2b,2c and 3b,3a,3b appear as 1-flats, while in (4_3), we have 5a,5b,5c and 6b,6a,6b. In Table 4.5.1, more examples of the balanced distribution of symmetric relatives occur.

It is also possible to analyse Table 4.5.1 by referring to the transformation g which defines the cycles. The set of entries from a fixed column and row of each O_3-orbit of one cycle class, shows the effect of the transformation g. For instance, if we consider the set of 1-flats from the first row and column of the 3-repeats of each of the I_3 O_3-orbits, we have the cycle generated by 1-flat 1b, (1b 2b 3b 4b 5b 6b). The set of 1-flats in a fixed column and row of the O_3-orbits of both the I_3 and II_3 classes exhibits this same cyclic distribution of the 1-flats.

If we tabulate the number of times each 1-flat appears as a 3-repeat in Table 4.5.1, we find that each II_1 1-flat occurs seven times, while each I_1 1-flat occurs only five times. From this observation, should an error occur in a II_1 1-flat, more of the null space 3-flats would be effected than if the error were in a I_1 1-flat. This fact is reflected in the decoding algorithm given in Chapter 5.

Also of note is the fact that no (10_1) 1-flat appears as a 3-repeat in the I_3 0_3-orbits. Consequently, should errors occur in the II_1 or III_1 1-flats, the 3-flats in the null space from II_3 would be effected more than the I_3 3-flats. This too will be discussed in more detail when we present the 0_i-orbit decoder in Chapter 5.

We conclude the discussion of Table 4.5.1 by briefly referring to the III_3 and IV_3 0_3-orbits. Since each 1-flat of (10_1) appears in each of the (10_3) 3-flats, a single error in a 1-flat in (10_1) would cause all estimates obtained from the (10_3) null space 3-flats to be in error. The pairing of 1-flats determined by the symmetry blocks is reflected in the IV_3 0_3-orbit (11_3).

We now discuss Table 4.5.2. This Table provides many more examples of symmetries. The most obvious of these, as for Table 4.5.1, is the set of 1-flats which are the entries for a fixed row and column of each of the 0_3-orbits of I_3 or II_3. For example, the column two, row c_i entry of each 0_3-orbit in I_3 gives the cycle generated by the 1-flat 1c, (2c 3c 4c 5c 6c 1c). It is possible to find several examples of a set of 1-flats or their g-transformations repeating in distinct 3-flats. In each bc_{ii} 3-flat of I_3, the pair of 1-flats 7a,7c or a g-transformation of the pair, occurs. In the 3-flats t_3a_{ii}, t_3b_{ii} and t_3c_{ii}, t=1,...,6, the same set of three 1-flats appears, where one flat is from I_3 and two are from II_3. For example, the 1-flats 3c, 7c and 8b appear in each of 1_3a_{ii}, 1_3b_{ii} and 1_3c_{ii}. The corresponding g-transformations of this triple of 1-flats appear in the remaining 0_3-orbits of I_3. The same (10_1) 1-flat appears in t_3a_{ii}, t_3b_i, t_3c_i, t=1,...,6. For example, we have 10_1a in each of 1_3a_{ii}, 1_3b_i, 1_3c_i. Again, the g-transformations of 10_1a appear in the corresponding 3-flats of the other 0_3-orbits of I_3.

Examples of the symmetric relative pairs are found in the II_3 0_3-orbits. Here the 'semi-cycle' mentioned earlier becomes evident. In, for example, the 3-flat 7_3bc_i, the I_1 1-flats 4c and 5c occur. In 7_3ac_{ii}, the 3-flat symmetric relative of 7_3bc_i, we have the 1-flats 1c and 2c, the symmetric relatives of 4c and 5c, respectively. This pattern is repeated in the other II_3 0_3-orbit 3-flats. In 7_3a_i, the 0_1-orbit 1-flats 4b and 4c appear. The symmetric relative of 7_3a_i, 7_3b_i, contains the corresponding symmetric relatives of 4b and 4c, that is 1b and 1c, respectively. The g-transformations of these 1-flats appear in the associated 0_3-orbit 3-flats.

Many other such symmetries are found in Table 4.5.2. The few examples given here are sufficient to illustrate the way in which the 1-flats are distributed symmetrically among the 3-flats. The more frequent occurrence of the II_1 1-flats in the null space is an important consideration in the decoding algorithm introduced below. That in some 0_3-orbits an error in a 1-flat may appear as a singleton in one subset, but as a 3-repeat in another, proves useful for error-correction.

We note here that a single error is considered to occur in a particular 1-flat rather than in a point for the following reason. As mentioned above, a 1-flat refers to the non-zero points only. And, as the two non-zero points of a 1-flat must always occur together in a 3-flat, it is only necessary to analyse one of the two possible non-zero errors in the 1-flat, since it is irrelevant to the decoder which of the two points is actually in error.

We now present several Tables which further illustrate the symmetric distribution of the 1-flats in the 0_3-orbit 3-flats. We include them here to demonstrate that the selection of 3-flats for the null space provides a mathematically interesting distribution of the 0_1-orbit 1-flats. Later, it is shown that these symmetries are the basis of the 0_i-orbit decoder discussed in Chapter 5.

Table 4.5.1 and Table 4.5.6 can be combined to form Table 4.5.2. In Table 4.5.6, the same row and column entry of the 0_3-orbits of I_3 and II_3 reflect the g-transformations. For example, the 'ii' row of the second column of t_3a in I_3 is the cycle generated by the 1-flat 1a, (4a 5a 6a 1a 2a 3a). Several symmetries are made more obvious in this Table than in Table 4.5.2. For instance, in Table 4.5.6, in the t_3ab column of the I_3 0_3-orbits, the 'i' and 'ii' rows contain between them, all the 1-flats from a I_1 0_1-orbit. For example, the (3_3) 0_3-orbit has the three 1-flats 1a, 1c and 1b. In II_3, this same column illustrates the symmetric relative of the I_1 1-flats and of the II_1 'a' and 'b' 1-flats. For example, in the (8_3) row, we have the 1-flats 2a, 7b and 5a, 7a. This Table also shows that within the 0_3-orbits a given 1-flat is paired with several distinct 1-flats, a most useful property in error-correction.

The many other symmetries are simply more illustrations of the interrelations already mentioned, that is the effect of the transformation g and the distribution of 1-flats according to the symmetry blocks.

	t_3a	t_3b	t_3c	t_3ab	t_3ac	t_3bc	t_3abc
(1_3)							
i:	6a4a10c9b	6a6c4c10a	4a5b5c10a	5a5c	6c11a	4b10c	9a
ii:	3c4b10a7c8b9c	3c4a7c8b10b11a	5a6a6b3c7c8b	5b6b	4c10b	9b9c	
(2_3)							
i:	1a5a10b7a	1a1c5c10c	5a6b6c10c	6a6c	1c11a	5b10b	7b
ii:	4c5b10c8c9b7c	4c5a8c9b10a11a	6a1a1b4c8c9b	6b1b	5c10a	7a7c	
(3_3)							
i:	2a6a10a8a	2a2c6c10b	6a1b1c10b	1a1c	2c11a	6b10a	8b
ii:	5c6b10b9c7a8c	5c6a9c7a10c11a	1a2a2b5c9c7a	1b2b	6c10c	8a8c	
(4_3)							
i:	3a1a10c9a	3a3c1c10a	1a2b2c10a	2a2c	3c11a	1b10c	9b
ii:	6c1b10a7c8a9c	6c1a7c8a10b11a	2a3a3b6c7c8a	2b3b	1c10b	9a9c	
(5_3)							
i:	4a2a10b7b	4a4c2c10c	2a3b3c10c	3a3c	4c11a	2b10b	7a
ii:	1c2b10c8c9a7c	1c2a8c9a10a11a	3a4a4b1c8c9a	3b4b	2c10a	7b7c	
(6_3)							
i:	5a3a10a8b	5a5c3c10b	3a4b4c10b	4a4c	5c11a	3b10a	8a
ii:	2c3b10b9c7b8c	2c3a9c7b10c11a	4a5a5b2c9c7b	4b5b	3c10c	8b8c	
(7_3)							
i:	3b4b4c11a	6b1b1c11a	1a1b4a4b	1a9a	6c1c	4c3c	9c
ii:	1b2b5b3a3c10b	2b4b5b6a6c10b	2b5b9a9b10b11a	4a9b	6a6b	3a3b	
(8_3)							
i:	4b5b5c11a	1b2b2c11a	2a2b5a5b	2a7b	1c2c	5c4c	7c
ii:	2b3b6b4a4c10a	3b5b6b1a1c10a	3b6b7a7b10a11a	5a7a	1a1b	4a4b	
(9_3)							
i:	5b6b6c11a	2b3b3c11a	3a3b6a6b	3a8b	2c3c	6c5c	8c
ii:	3b4b1b5a5c10c	4b6b1b2a2c10c	4b1b8a8b10c11a	6a8a	2a2b	5a5b	

Table 4.5.6: 1-repeat 1-flats of I_3 and II_3

The remaining Tables in this section illustrate particular symmetrical or cyclical properties. Table 4.5.7 indicates the pairing of the 1-flats of the 0_1-orbits. It illustrates the symmetries, and the marked differences in the structures of the I_3 and II_3 cycle classes. The symmetry and cyclic nature of this distribution are obvious.

	O_3-orbit (t_3)	10_1a	10_1b	10_1c	11_1a
I_3	(1_3)	a,b,c	a,bc,X	ac,b,X	ac,b,X
	(2_3)	ac,b,X	a,b,c	a,bc,X	ac,b,X
	(3_3)	a,bc,X	ac,b,X	a,b,c	ac,b,X
	(4_3)	a,b,c	a,bc,X	ac,b,X	ac,b,X
	(5_3)	ac,b,X	a,b,c	a,bc,X	ac,b,X
	(6_3)	a,bc,X	ac,b,X	a,b,c	ac,b,X
II_3	(7_3)	ab,ab,abc	a,b,c	ac,bc,c	ac,b,X
	(8_3)	a,b,c	ac,bc,c	ab,ab,abc	bc,a,X
	(9_3)	ac,bc,c	ab,ab,abc	a,b,c	ab,c,X

Table 4.5.7: Summary of occurrence of (10_1),(11_1) 1-flats in I_3, II_3 3-flats; entries in row (t_3), column 10_1j or 11_1a are the (t_3) 3-flats containing the 1-flat 10_1j or 11_1a, j=a,b,c; X denotes that 10_1j or 11_1a appears in one of the two 3-flats excluded from (t_3).

Table 4.5.8 indicates the pairing of the 1-flats of the O_1-orbits. It illustrates the symmetries, and the marked differences,in the structures of the I_3 and II_3 cycle classes.

	O_3-orbit (t_3)	(1_1)	(2_1)	(3_1)	(4_1)	(5_1)	(6_1)	(7_1)	(8_1)	(9_1)
I_3	(1_3)	-	S	S	S	D_1	D_2	S	S	D_5
	(2_3)	D_2	-	S	S	S	D_1	D_4	S	S
	(3_3)	D_1	D_2	-	S	S	S	S	D_4	S
	(4_3)	S	D_1	D_2	-	S	S	S	S	D_4
	(5_3)	S	S	D_1	D_2	-	S	D_5	S	S
	(6_3)	S	S	S	D_1	D_2	-	S	D_5	S

	O_3-orbit (t_3)	(1_1)	(2_1)	(3_1)	(4_1)	(5_1)	(6_1)	(7_1)	(8_1)	(9_1)
II_3	(7_3)	D_6	S	D_7	D_8	S	D_9	-	S	D_{10}
	(8_3)	D_9	D_6	S	D_7	D_8	S	D_{10}	-	S
	(9_3)	S	D_9	D_6	S	D_7	D_8	S	D_{10}	-

D_1: a,c; b,c in $t_3$ab, t_3c (a,b)

D_2: a,b; a,c in t_3c, t_3b (b,c)

D_3: a,b in $t_3$bc (a,c; b,c)

D_4: a,c in $t_3$bc (a,b; b,c)

D_5: b,c in $t_3$bc (a,b; a,c)

D_6: a,b; b,c in t_3c, t_3b (a,c)

D_7: a,c; a,b in t_3a, $t_3$bc (b,c)

D_8: a,b; b,c in t_3a, t_3c (a,c)

D_9: a,c; a,b in t_3b, $t_3$ac (b,c)

D_{10}: a,b in t_3c (a,c; b,c)

Table 4.5.8: Occurrence of pairs of 1-flats in 3-flats; S denotes O_1-orbit 1-flats appear as singletons; '-' denotes that 1-flats are, by definition, in each 3-flat; D_i denotes that 1-flat pairs from the O_1-orbit of that column appear in the row O_3-orbit 3-flats, and the 1-flats in parentheses occur in the 3-flats omitted when defining the O_3-orbits.

Table 4.5.9 is an extended version of Table 4.5.1 illustrating precisely the occurrence of the 3-repeats in the O_3-orbits. Many symmetries of Table 4.5.1 are more clearly depicted.

We have presented these Tables as a concise means of describing the symmetric and cyclic properties of the selected subset of 3-flats of PG(5,2). We refer to these Tables when discussing the O_i-orbit decoder.

3- flat	(1_3)	(2_3)	(3_3)	(4_3)	(5_3)	(6_3)
A_t						
a	2a7a	3a8a	4a9a	5a7b	6a8b	1a9b
ab	1b2a7a8c 1b8c	2b3a8a9c 2b9c	3b4a9a7c 3b7c	4b5a7b8c 4b8c	5b6a8b9c 5b9c	6b1a9b7c 6b7c
ac	1c2a3b7a 1c3b	2c3a4b8a 2c4b	3c4a5b9a 3c5b	4c5a6b7b 4c6b	5c6a1b8b 5c1b	6c1a2b9b 6c2b
abc	1b1c3b8c	2b2c4b9c	3b3c5b7c	4b4c6b8c	5b5c1b9c	6b6c2b7c
B_t						
b	2b3a	3b4a	4b5a	5b6a	6b1a	1b2a
ab	1a2b3a8c 1a8c	2a3b4a9c 2a9c	3a4b5a7c 3a7c	4a5b6a8c 4a8c	5a6b1a9c 5a9c	6a1b2a7c 6a7c
bc	1c2b3a7b 1c7b	2c3b4a8b 2c8b	3c4b5a9b 3c9b	4c5b6a7a 4c7a	5c6b1a8a 5c8a	6c1b2a9a 6c9a
abc	1a1c7b8c	2a2c8b9c	3a3c9b7c	4a4c7a8c	5a5c8a9c	6a6c9a7c
C_t						
c	2c8a	3c9a	4c7b	5c8b	6c9b	1c7a
ac	1a2c3b8a 1a3b	2a3c4b9a 2a4b	3a4c5b7b 3a5b	4a5c6b8b 4a6b	5a6c1b9b 5a1b	6a1c2b7a 6a2b
bc	1b2c7b8a 1b7b	2b3c8b9a 2b8b	3b4c9b7b 3b9b	4b5c7a8b 4b7a	5b6c8a9b 5b8a	6b1c9a7a 6b9a
abc	1a1b3b7b	2a2b4b8b	3a3b5b9b	4a4b6b7a	5a5b1b8a	6a6b2b9a

	(7_3)	(8_3)	(9_3)
A_t			
a	8a5a	9a6a	7b1a
ab	7b8a10a5a 7b10a	8b9a10c6a 8b10c	9b7b10b1a 9b10b
ac	7c8a2c5a 7c2c	8c9a3c6a 8c3c	9c7b4c1a 9c4c
abc	7b7c10a2c	8b8c10c3c	9b9c10b4c

3-flat	(7_3)	(8_3)	(9_3)
B_t			
b	8b2a	9b3a	7a4a
ab	7a8b10a2a 7a10a	8a9b10c3a 8a10c	9a7a10b4a 9a10b
bc	7c8b2a5c 7c5c	8c9b3a6c 8c6c	9c7a4a1c 9c1c
abc	7a7c10a5c	8a8c10c6c	9a9c10b1c
C_t			
c	8c10c	9c10b	7c10a
ac	7a8c10c2c 7a2c	8a9c10b3c 8a3c	9a7c10a4c 9a4c
bc	7b8c10c5c 7b5c	8b9c10b6c 8b6c	9b7c10a1c 9b1c
abc	7a7b2c5c	8a8b3c6c	9a9b4c1c

Table 4.5.9: Expansion of Table 4.5.1
1-flats in column (t_3) are 3-repeats in their corresponding subset A_t, B_t or C_t; each entry in a given column and row must appear 3 times within its subset, e.g. 2a is in rows a, ab, and ac of A_1, column (1_3)

4.6 Independence of O_i-Orbit Structure on Minimal Polynomial

In the discussion of the PG(5,2) given in this chapter, we have defined the geometry using the minimal polynomial m(x) of the primitive element α, where

$$m(X) = 1 + X + X^6.$$

The fact that any primitive element and its associated minimal polynomial yields a valid representation of PG(5,2) assures that any such interpretation is structurally equivalent to any other. That is, the O_i-orbit structure given in Table 4.5.2 is independent of the minimal polynomial chosen to represent the geometry where the six possible polynomials for this geometry, expressed as powers are, (0 1 6), (0 5 6), (0 1 2 5 6), (0 2 3 5 6), (0 1 4 5 6), (0 1 3 4 6). The 1-flats of the geometries generated from the five miminal polynomials, other than (0 1 6), can be labelled such that the resulting isomorphic geometry has a 0_i-orbit structure identical to the one given for (0 1 6). In the following, it is shown how the isomorphic labelling of the 1-flats is accomplished.

First, we indicate how the 0_i-orbits, given any minimal polynomial, are constructed. We then illustrate the one to one correspondence between the (0 1 6) labelling of the 1-flats and the minimal polynomial m'(X) labelling for m'(X) not (0 1 6).

Before forming the 0_i-orbits we recall that the null space consists only of those flats which intersect on position 0, and hence that when forming a 1-flat, one of the two linearly independent points required to form a 1-flat, must be 0.

As the IV_1 0_1-orbit is fixed for any representation, we obtain it first. It has m.c. 21 and consists of one 1-flat only, which is (0 21 42). This follows from the second theorem of Yamamoto et al [58] given above.

We noted earlier that the (10_1) 0_1-orbit was distinct from the other orbits and because of this uniqueness would play a special role in decoding. Correspondingly, the structure of the III_1 0_1-orbit is unlike the structure of the other 0_1-orbits. Every non-zero point in the (10_1) 0_1-orbit is a multiple of 9. Thus, the initial 1-flat of (10_1) is always formed by selecting, as the two linearly independent points, 0 and a multiple of 9. The remaining 1-flats in the 0_1-orbit are obtained by subtracting, mod 63, the first non-zero point of the initial flat from each of the points in the flat. By repeating the subtraction a second time, using the third point, the three 1-flats of (10_1) are found.

To obtain the I_1 and II_1 1-flats, two linearly independent points, the first of which is 0, are selected. The second point can not be a multiple of 9 or 21 as these points are present in the III_1 and IV_1 0_1-orbits. We note that the totality of the non-zero points of the 0_1-orbit 1-flats exhaust the 62 non-zero points of PG(5,2). Thus, no non-zero point can appear in more than one 0_1-orbit 1-flat.

The process of obtaining the initial 1-flats for I_1 and II_1 0_1-orbits is simplified by selecting the second linearly independent point from the set of linearly independent points $\{1,2,3,4,5\}$, although any one of the non-zero points not already present in a 1-flat can be chosen. A 1-flat is formed using a point from this set and the point 0 as the two linearly independent points. This initial 1-flat is then successively multiplied by 2^i, $i=1,\ldots,5$, that is the transformation g is applied five times. If six distinct 1-flats are obtained, these are the class I_1 initial 1-flats. If only three distinct 1-flats are generated, they are the initial 1-flats of II_1. The 0_1-orbit for each distinct initial 1-flat is produced by the subtraction algorithm used for the class III_1.

To obtain the initial 1-flats of the remaining class, any point not occurring in the 1-flats already generated, is selected. The corresponding 1-flat is formed. The remaining initial 1-flats and their corresponding 0_1-orbit 1-flats are obtained as for the previous class of 0_1-orbits.

This procedure generates the 0_1-orbits irrespective of the minimal polynomial chosen. The 0_3-orbits are obtained by a similar process using four independent points rather than two.

We now proceed to establish the isomorphism between the 1-flats of the geometry represented by the minimal polynomial m(X)=(0 1 6) and the geometry represented by the minimal polynomial m'(X) where m'(X) is one of (0 5 6), (0 1 2 5 6), (0 2 3 5 6), (0 1 4 5 6), (0 1 3 4 6). The set of minimal polynomials can be partitioned into two sets according to the number of points in the minimal polynomial, that is (0 1 6), (0 5 6) and (0 1 2 5 6), (0 2 3 5 6), (0 1 4 5 6), (0 1 3 4 6). The procedure for establishing the 1-flat isomorphism is the same for minimal polynomials within the same set. We denote the geometry formed with minimal polynomial m(X)=(0 1 6) as PG(5,2) with O_j-orbits (t_j), j=1,3, 1-flats t_1, t=1,...,11 and cycle classes I_1,II_1,III_1,IV_1. For the geometry formed by any other minimal polynomial m'(X)≠(0 1 6), we refer to the geometry as PG'(5,2), with 1-flats t_1', t=1,...,11, O_j'-orbits (t_j'), j=1,3, t=1,...,11 and cycle classes I_1',II_1',III_1',IV_1'. First, the 1-flat isomorphism for m'(X)=(0 5 6) is established, as the correspondence is quite simple. Each O_1'-orbit (t_1') in PG'(5,2) consists of the same point set as the O_1-orbit (t_1) in PG(5,2), but with a different arrangement of the points among the 1-flats, t=1,...,9. Consequently the labelling of the 1-flats of (t_1'), t=1,...,9, is immediately obvious. The two non-zero points in each 1-flat of (t_1') are from two distinct 1-flats in (t_1). We label a given 1-flat j in (t_1') by the 1-flat letter from (t_1) not represented in j. For instance, (1_1) in PG(5,2) is:

a: 0 1 6
b: 0 5 62
c: 0 57 58.

The three 1-flats of $(1_1')$ are:

a: 0 57 62
b: 0 1 58
c: 0 5 6.

As the first of these, (0 57 62), has no representative from 1_1a, it is called $1_1'$a in $(1_1')$. By similar reasoning, $1_1'$b and $1_1'$c are labelled. The III_1 and IV_1 0_1-orbits are identical for the minimal polynomials (0 1 6) and (0 5 6).

Once the 1-flats have been labelled using this algorithm, the 3-flats can be labelled using the same method given in Section 4.4 for the geometry with minimal polynomial (0 1 6).

The assignment of labels to form the isomorphism of the 1-flats for the second set of minimal polynomials is slightly more complicated. To determine the isomorphism, we first define the set Q_t as the set of three 0_1-orbits of PG(5,2) of which the non-zero points of the 0_1-orbit (t_1') are members, t=1,...,9. For example, if m'(X)=(0 1 4 5 6), then a $0_1'$-orbit (t_1') of the I_1' class is:

0 1 39
0 38 62
0 24 25,

and the Q_t subset is formed as follows. Referring to Table 4.4.1, the points 1 and 62 are in (1_1), the points 24 and 39 are in (3_1) and the points 25 and 38 are in (8_1). Thus the set Q_t is (1),(3),(8). Each Q_t consists of two 0_1-orbits from I_1 and one from II_1, t=1,...,9. If we consult Table 4.5.1, we note that there is one and only one 0_3-orbit, (t_3), which has 1-flats from each orbit of Q_t in every 7-repeat subset A_t, B_t, C_t. We use this association to number the $0_1'$-orbits. Consequently the $0_1'$-orbit of PG'(5,2) listed above is the $0_1'$-orbit $(1_1')$. The labelling of the 1-flats of I_1' is as follows. The 1-flat consisting of an a and c 1-flat from two of the O_1-orbits of Q_t, $t=1,\dots,6$, is labelled $t_1'c$. The 1-flat with its two non-zero points from two b 1-flats of the Q_t O_1-orbits is $t_1'b$. The remaining 1-flat is labelled $t_1'a$. In the above example, the point 1 in the first 1-flat is from 1_1a and the point 39 from 3_1c. Thus, this 1-flat is labelled $1_1'c$. The three 1-flats given above are labelled:

a: 0 24 25
b: 0 38 62
c: 0 1 39.

The labelling of the 1-flats of the II_1' O_1'-orbits is slightly different than the I_1' case, but the assignment of the O_1-orbit number is as for the I_1' class. The $t_1'a$ 1-flat in (t_1') is the 1-flat with both its non-zero points from b 1-flats of the Q_t O_1-orbits, $t=7,8,9$. The $t_1'c$ flat is the one which consists of two non-zero points from the c 1-flats of the I_1 O_1-orbits in Q_t. The remaining 1-flat is labelled $t_1'b$. The labelling of $(7_1')$ is thus:

a: 0 20 49
b: 0 14 34
c: 0 29 43,

where the points 20 and 49 are from 3_1b and 8_1b, respectively. The points 29 and 43 are from 6_1c and 3_1c, respectively.

Each of the $(10_1')$ 1-flats consists of the same points as the 1-flats of (10_1) but $10_1j \neq 10_1'j$, $j=a,b,c$. These labels can be assigned by referring to the labels given to the I_1' and II_1' classes and the 3-flats of the O_3-orbits. In Table 4.6.1 we list the 1-flats of the O_1'-orbits of the PG'(5,2) with minimal polynomial (0 1 4 5 6).

The O_3'-orbits are labelled as they were in Section 4.4 for the case when the minimal polynomial was (0 1 6). Thus, the results given in the Tables of this chapter concerning the symmetrical properties of the O_i-orbit structure of PG(5,2) describe exactly the symmetries of PG'(5,2) with minimal polynomial m'(X)≠(0 1 6).

I'	$(1'_1)$	$(2'_1)$	$(3'_1)$
	a: 0 24 25	a: 0 48 50	a: 0 33 37
	b: 0 38 62	b: 0 13 61	b: 0 26 59
	c: 0 1 39	c: 0 2 15	c: 0 4 30
	Q_1:(1),(3),(8)	Q_2:(2),(4),(9)	Q_3:(3),(5),(7)
	$(4'_1)$	$(5'_1)$	$(6'_1)$
	a: 0 3 11	a: 0 6 22	a: 0 12 44
	b: 0 52 55	b: 0 41 47	b: 0 19 31
	c: 0 8 60	c: 0 16 57	c: 0 32 51
	Q_4:(4),(6),(8)	Q_5:(5),(1),(9)	Q_6:(6),(2),(7)
II'	$(7'_1)$	$(8'_1)$	$(9'_1)$
	a: 0 20 49	a: 0 35 40	a: 0 7 17
	b: 0 14 34	b: 0 5 28	b: 0 10 56
	c: 0 29 43	c: 0 23 58	c: 0 46 53
	Q_7:(3),(6),(8)	Q_8:(1),(4),(9)	Q_9:(2),(5),(7)
III'	$(10'_1)$	IV'	$(11'_1)$
	a: 0 36 54		a: 0 21 42
	b: 0 18 27		
	c: 0 9 45		

Table 4.6.1: O'_1-orbits of PG'(5,2)

In this section we have shown that it is possible to establish an isomorphism between the flats of PG(5,2) and the flats of PG'(5,2). Consequently, the O_i-orbit structure defined is independent of the minimal polynomial chosen to represent the geometry and the results concerning decoding and error-correction hold for any representation of PG(5,2).

4.7 Basis of the O_i-orbit Symmetry

We conclude this chapter with a brief mathematical explanation of the O_i-orbit structure.

In the previous section, it was shown that the O_i-orbit structure was independent of the minimal polynomial chosen to represent PG(5,2). As a consequence, it is possible to express the irf's, r=1,2,3, without explicitly referring to a minimal polynomial, as was necessary in Table 4.4.1 and Table 4.6.1. In the following, the parameter d represents the non-zero point which, together with 0, generates the first i1f in I_1. All other irf's, r=1,2,3 can be expressed in terms of d. This is illustrated in Table 4.7.1.

	r=1		r=2		r=3
I_1	0,d	I_2	0,d,2d	I_3	0,d,2d,3d
	0,2d		0,2d,4d		0,2d,4d,6d
	0,4d		0,4d,8d		0,4d,8d,12d
	0,8d		0,8d,16d		0,8d,16d,24d
	0,16d		0,16d,32d		0,16d,32d,48d
	0,32d		0,32d,d		0,32d,d,33d
II_1	0,7d	II_2	0,7d,14d	II_3	0,7d,14d,21d
	0,14d		0,14d,28d		0,14d,28d,42d
	0,28d		0,28d,56d		0,28d,56d,21d
III_1	0,9d	III_2	0,9d,18d	III_3	0,9d,18d,27d

Table 4.7.1: Initial r-flats, r=1,2,3, in terms of the parameter d.

We note that for d=1 this reduces to the representation of PG(5,2) with minimal polynomial (0 1 6) given in Table 4.4.1 for r=1. For d=25, r=1, this generates Table 4.6.1, the representation with minimal polynomial (0 1 4 5 6).

Table 4.7.1 is formed by taking the two linearly independent points 0 and d as the defining points of the first i1f in I_1. The first i2f in II_1 is defined by the three linearly independent points 0,d,2d and the first i3f by 0,d,2d,3d. That these points are in fact linearly independent, is a consequence of the geometry being defined over GF(2). Thus, instead of taking the point (α) as a primitive element, the point (α^d) can be used and hence the points given are linearly independent. Moreover, this justifies the process, given in previous sections, of multiplying any given set of independent points by 2, to obtain another set of independent points. The choice of 0 and 7d as the two linearly independent points for II_1 i1f, reflects the special structure of this class. The coefficient 7 is a proper divisor of the number of points in the geometry. The interrelation of cylce members of this class is a result of this property. Similarly, the coefficient of d in the II^T and IV classes is a proper divisor of 63.

By referring to Table 4.7.1, we can account for the following:

i) the one to one correspondence between the 0_3-orbits and the 0_1-orbits,

ii) the occurrence of the 1-flat symmetric relatives in the 0_3-orbits, and

iii) the 1-flat t+1 appearing as a 3-repeat in the 0_3-orbit (t_3), t=1,...,9.

The first of these observations is immediately obvious from an inspection of the linearly independent points of the 1-flats and corresponding 3-flats. To generate the corresponding i3f of a given i1f t_1, the set $K = \{0, kd\}$, $k>0$, is augmented to the set $K^* = \{0,kd,2kd,3kd\}$, where these points represent the geometry based on the primitive element α^{kd} rather than α, and hence are linearly independent. The set of all linear combinations of 0 and kd must be contained in the set of all linear combinations of K^*. Hence, the correspondence between the two sets follows. Further, that each point of the 1-flat appears seven times in the 0_3-orbit generated from the 3-flat t_3 formed from K^*, can now easily be established. The points in the 1-flat t_1 are 0, kd, 0+kd. The points in the 3-flat t_3 are, in terms of the linearly independent points of t_1,

0,kd,2kd,3kd,0+kd,0+2kd,0+3kd,kd+2kd,kd+3kd,2kd+3kd,
0+kd+2kd,0+kd+3kd,0+2kd+3kd,kd+2kd+3kd,0+kd+2kd+3kd.

We say the two points i and j in t_3 differ by kd if both i and j, expressed terms of the linearly independent points of t_1, have the same number of components, and each component i_s of i is j_s+kd for j_s a linearly independent point in j. If there are two points, i and j, i<j, in t_3, differing by kd, subtracting i from the point set representation of t_3, gives a 3-flat in the 0_3-orbit (t_3) containing the points 0 and kd, and hence, the 1-flat t_1. The following are all the possible pairs of points from K* which differ by kd:

0,kd
kd,2kd
2kd,3kd
0+kd,kd+2kd
kd+2kd,2kd+3kd
0+kd+2kd,kd+2kd+3kd
0+2kd,kd+3kd.

Thus, the ilf t_1 appears seven times in the 0_3-orbit (t_3). Similarly, it can be shown that the other members of the (t_1) 0_1-orbit appear seven times, as required.

A further examination of the points in t_3 shows that neither of the two points, p_1=0+kd+3kd or p_2=0+2kd+3kd, differ by kd from any other point in t_3. Thus, subtracting p_i, i=1,2 from each point in the point set representation of t_3 gives two 3-flats through the point 0, but, such that they do not contain any of the points of t_1. We recall that the fifteen 3-flats through 0 are obtained from successively subtracting each point of t_3 from the point representation of t_3. Two of these flats are omitted from the 0_3-orbit of t_3. These two flats correspond to the points p_1 and p_2. The omission of eight of the 3-flats generated from 10_3 is similarly explained, noting that each linear combination of points in 10_1a forms a point which is again a multiple of 9d.

The occurrence of symmetric relatives in the 0_3-orbit 3-flats is also explained by studying Table 4.7.1. If we consider the non-zero point of two symmetric relative 1-flats in I_1, the same pair of points occurs in the corresponding I_3 3-flats separated by 3 cycle positions. For instance, in I_1, d and 8d are the independent points for the first and fourth flats, respectively. In the first and fourth flats of I_3, d and 8d, as extensions of the I_1 1-flats, appear. But, as well, d and 8d appear in the sixth and third I_3 3-flat point sets, respectively, that is in a second set of 3-flats separated by 3 cycle positions. Such a separation is simply the multiplication of the non-zero independent point of the first 1-flat, by 2, three times. This is half the number of multiplications required to give the identity 2^6. Hence, we have the term 'semi-cycle' used above. Thus, if kd is the non-zero independent point of a 1-flat, 2^3(kd) is the non-zero independent point in its symmetric relative 1-flat.

We observe that the non-zero independent point of the 1-flat t+1 always appears as a non-zero independent point in the i3f t_3. This occurs because the non-zero independent point in the i1f t+1 is formed by multiplying the independent point of t by 2, while the second non-zero independent point in the point set of t_3 is twice the first, that is the same point as the non-zero independent point of t+1. Thus, the 1-flat t+1 appears as a 3-repeat in the 0_3-orbit (t_3).

We have explained here the major symmetric properties of the 0_3-orbits. The many other symmetries which are presented in Section 4.5 are also explained by further reference to Table 4.7.1.

4.8 Conclusions

In this chapter we have introduced a most interesting mathematical structure, defined on the flats of PG(5,2), by extending the definitions of Yamamoto et al[58]. This structure exhibits a well-defined set of symmetrical properties which are of interest when analysing the geometry mathematically. Moreover, the structure allows for a simplification of the standard Majority Logic Decoder of the PG code defined on PG(5,2). The details of this decoder are given in Chapter 5.

In particular, we began this chapter by giving a detailed description of the results on Finite Geometries as outlined in the 1966 paper by Yamamoto et al[58]. Based on these results, an analysis of the cycles of the PG(5,2) was given. The O_i-orbit structure, used as the basis of the decoder in Chapter 5, was developed. The symmetrical distribution of flats in the O_i-orbits of PG(5,2) was described, both in the text and in the extensive set of Tables presented. Finally, it was shown that the structure introduced was independent of the minimal polynomial chosen to represent the geometry. Hence, the decoding method developed in Chapter 5 does not depend on a particular representation of PG(5,2) by a minimal polynomial. Several symmetric properties of the O_i-orbit structure were explained by referring to a representation of the flats which is independent of a minimal polynomial.

CHAPTER 5: 0_i-ORBIT DECODER OF THE ORDER-3 (63,41) PG CODE

5.1 Introduction

The decoder presented here is defined in terms of the concepts introduced in the previous chapter.

We begin this chapter with a detailed study of the standard Majority Logic Decoder of the order-3 (63,41) PG code. This is followed by the definition of the 0_i-orbit decoder of the code. Reference is made to the Tables and terminology of Chapter 4 in the analysis of the decoder. It is shown that all possible 1,2 and 3-errors are correctable using this decoder and that certain sets of i-errors are related in such a way that the decoding algorithm treats them identically, i=1,2,3. The simplicity of the 0_i-orbit decoder, as compared to the standard Majority Logic Decoder of this code, is emphasized with reference to the circuitry and decoding time required by each.

5.2 Order-3 (63,41) PG Code Standard Majority Logic Decoder

Order-r PG Codes are Majority Logic Decodable codes requiring r steps of Majority Logic. Several modifications to the original decoder have been suggested, several of which are discussed in Chapter 3. As the original Majority Logic Decoder is the most common method used to decode PG codes, we refer to it as the standard PG code decoder and compare the 0_i-orbit decoder with it. In this section we investigate in detail the Majority Logic Decoder for the order-3 PG code based on PG(5,2).

In Chapter 3 we discussed MLD in general. We now make this specific for the (63,41) order-3 PG code. As mentioned in the previous chapter, this code is cyclic and hence it is only necessary to consider as members of the null space, those 3-flats which pass through zero. There are, referring to Chapter 3,

$$\lambda(0,3,5,2) = \frac{(2^5-1)(2^4-1)(2^3-1)}{(2^3-1)(2^2-1)(2-1)} = 155$$

such 3-flats. Similarly, there are

$$\lambda(0,2,5,2) = \frac{(2^5-1)(2^4-1)}{(2^2-1)(2-1)} = 155$$

2-flats which intersect on 0, and

$$\lambda(0,1,5,2) = \frac{(2^5-1)}{(2-1)} = 31$$

1-flats which intersect on 0. The decoding process is based on the orthogonality of these flats. The 3-flats through 0 are initially known to the decoder. Those 3-flats which intersect on a given 2-flat through 0 are used to obtain an estimate of the 2-flat. Similarly, an estimate for each 2-flat through 0 is determined. These estimates then provide estimates of the 1-flats, since each 1-flat through 0 has an associated set of 2-flats which intersect on it. Finally, using the 1-flats which intersect on 0, an estimate of the error digit in position 0 is obtained. If no more than three errors have occurred, this estimate is correct. We now discuss the circuitry necessary to implement the decoder.

Before the decoding process can begin, a preliminary step is necessary in which the received word is multiplied by $X^{n-k}=X^{63-41}=X^{22}$ and then divided by the generating polynomial

$$\begin{aligned} g(X) &= (X+1)(X^6+X^5+1)(X^3+X+1)(X^6+X^5+X^4+X^2+1)(X^6+X^5+X^4+X+1) \\ &= X^{22}+X^{20}+X^{19}+X^{18}+X^{15}+X^9+X^7+X^5+X^4+X^3+X+1. \end{aligned}$$

The remainder, r(X), a shifted version of the syndrome, is stored in a register. Circuitry is required for the multiplication (simply a shift of the received word) and for the division by g(X).

On step 1, estimates of the 155 2-flats through 0 are obtained from the 3-flats known to the decoder. There are seven 3-flats which intersect on each 2-flat through 0. The appropriate bits in the syndrome register are tapped to obtain the values corresponding to each of these 3-flats. These are input to GF(2) adders to obtain the binary sum of each 3-flat. The seven binary sums which correspond to the seven 3-flats orthogonal on a given 2-flat, are input to the threshold unit, with threshold 4, which corresponds to the 2-flat. The output is the estimate of the 2-flat on which the seven input 3-flats are orthogonal.

The circuitry required for step 1 can be broken down into two parts. First, taps on the syndrome register and GF(2) adders for these taps, are required for a total of 155x7=1085 binary sums. Secondly, each set of seven sums which correspond to the 3-flats intersecting on a 2-flat, is input to a threshold unit, for a total of 155 7-input threshold units.

The second step is somewhat simpler. For each 1-flat through 0, there are 15 2-flats which intersect on it. The output from the 15 threshold units in step 1 which correspond to the 15 2-flats which intersect on a given 1-flat, are input to a threshold unit of threshold eight. The output is an estimate of the 1-flat through 0. The circuitry for the second step consists of 31 15-input threshold units, one for each of the 31 1-flats through 0.

On the third step, the 31 outputs from the second level threshold units, corresponding to the 31 1-flats which intersect on 0, are input to a single threshold unit. The output from this is the decoder's estimate of the error digit in position 0.

Therefore, the standard threshold decoder for the order-3 (63,41) PG code requires:

i) circuitry to multiply the received word by X^{22} and divide the result by $g(X)=X^{22}+X^{20}+X^{19}+X^{18}+X^{15}+X^{9}+X^{7}+X^{5}+X^{4}+X^{3}+X+1$,

ii) 155x7=1085 sets of taps on the syndrome register and corresponding sets of GF(2) adders,

iii) 155 7-input threshold units,

iv) 31 15-input threshold units,

v) one 31-input threshold unit.

The circuitry for the standard decoder can be reduced somewhat by the following observation. On step 1, the total number of errors which can be corrected is determined, that is $\lfloor 7/2 \rfloor = 3$. Thus, it is only necessary to input seven estimates to each threshold unit at any level, since no more than three errors can be corrected. This in turn reduces the number of 2-flat and 1-flat estimates required on steps 2 and 3, respectively. So, on step 3, only estimates of seven 1-flats are needed as input to the threshold unit. Consequently in step 2 only seven threshold units are necessary, one for each of the 1-flats required in step 3. Each of these seven threshold units needs only seven inputs, instead of the previous fifteen, for a total of 49 2-flat estimates. Hence, on step 1, 49, rather than 155, threshold units are necessary. Correspondingly, 49x7=343 sets of taps and GF(2) adders are required to form the inputs to the 49 threshold units. As a result, the version of the standard Majority Logic Decoder used most commonly to decode the order-3 (63,41) PG code requires:

i) circuitry to multiply the received word by X^{22} and divide the result by $g(X) = X^{22}+X^{20}+X^{19}+X^{18}+X^{15}+X^{9}+X^{7}+X^{5}+X^{4}+X^{3}+X+1$,

ii) 49x7=343 sets of taps on the syndrome register and corresponding sets of GF(2) adders,

iii) 49+7+1 = 57 7-input threshold units.

In the next section we present a decoding algorithm for the order-3 (63,41) PG code which is based on the O_i-orbit structures introduced in Chapter 4.

5.3 O_i-Orbit Decoder of the Order-3 (63,41) PG Code

The O_i-orbit non-orthogonal decoder of the order-3 (63,41) PG code is a simplification of the standard Majority Logic Decoder of the code. Fewer threshold units, simpler circuitry and fewer decoding steps are required for this decoder.

The first step of the O_i-orbit decoder involves obtaining non-orthogonal estimates of the 1-flats of the O_1-orbits (1_1) through (9_1), from the O_3-orbit 3-flats which are known to the decoder. Also, dependent on the errors in the (10_3) and (11_3) 3-flats, certain binary flags may be set. The estimates from the first step are orthogonal on the point 0. These estimates are input to a counter on the second step. Assuming no more than three errors have occurred, the error digit in position 0 is correctly determined by the output of the counter and, in a small number of cases, by the setting of the flags. The circuitry for the decoder is now described.

On the first step, for each O_3-orbit 3-flat, taps on the register positions corresponding to the points of a 3-flat are input to a binary adder, the output of which is the sum known to the decoder for the 3-flat. Associated with each of the subsets A_t, B_t and C_t, t=1,...,9, is a 7-input threshold unit of threshold 4. The seven inputs are the binary sums corresponding to the seven 3-flats comprising the subsets A_t, B_t, C_t, respectively. The seven sums corresponding to the seven 3-flats of (10_3) are input to a counter unit. The output of the 27 threshold units are orthogonal estimates on the point 0. A flag f_1 is set if one and only one of the seven inputs to the (10_3) counter unit is a one, and a flag f_2 if five or seven of the inputs are one. A counter determines the number of the (11_3) 3-flats, 11_3a_i, 11_3a_{ii}, 11_3a_{iii}, which have a binary sum of one. A flag f_3 is set if either two or three of these binary sums are one. These three flags are simply binary flip-flops, set if the output of a counter is a given value.

The first step of the O_i-orbit decoder requires a total of (13x9)+7+3=127 sets of calculations on the taps of the storage register vs. 1085 (343 in the simplified version) for the standard decoder. Three flags may have to be set. A total of 27 threshold units with seven inputs each are necessary, plus 2 counters for the flags. The standard decoder has 155 (49 in the simplified version) 7-input threshold units.

On the second step, the 27 outputs from step 1 are input to a counter. These outputs correspond to 27 1-flats orthogonal on the point 0. The decoder's decision as to the value of the error digit in position 0 is dependent on the value c output by the counter and, in some cases, the flags f_1, f_2 and f_3. We make this explicit in the following, where if e_o denotes the error digit in position 0, then, if

$c<14$ or $c=15$, then $e_o=0$,

$c>16$, then $e_o=1$

$c=16$, f_1 or f_2 is set, then $e_o=1$; otherwise $e_o=0$

$c=14$, both f_2 and f_3 are set, then $e_o=1$; otherwise $e_o=0$.

Thus, on the second step a simple counter is required and logical units to test the flags. It is shown in the next section, that the flags must be checked for less than $\frac{1}{4}\%$ of the possible correctable error patterns, that is, over 99.75% of the correctable error sets have a count of 15, or less than 14, or more than 16.

This decoding method does not require that the received word be premultiplied by X^{22} and the result divided by g(X). The tapped values need only be fed into binary flip-flops and the output then directly input into a threshold unit. The O_i-orbit decoder circuitry consists of a total of 27 7-input theshold units, two counters and three binary flags. The standard Majority Logic Decoder requires circuitry for premultiplication, division, GF(2) adders, 155 7-input threshold units, 31 15-input threshold units and a 31-input threshold unit, or, in the simplified version, 57 7-input theshold units. The reduction in circuitry and complexity for the O_i-orbit decoder is significant. Moreover, should the decoder consist of a front-end mini-computer, the counting and flag testing of the O_i-orbit decoder becomes even more simple.

5.4 O_i-orbit Decoder Error Analysis

An analysis of the correctable error patterns of the O_i-orbit decoder reflects the O_i-orbit structures used to define the decoder. This section contains an extensive investigation of these error patterns and the method the decoder uses to correct them.

The standard Majority Logic Decoder corrects all 1, 2, 3-errors and some errors of higher weight. The O_i-orbit decoder also corrects all 1, 2 and 3-errors. Some higher weight errors are corrected, however, we discuss those in Chapter 7.

A computer simulation of this decoder was written. All possible 1, 2 and 3-error patterns were shown to be correctable by the 0_i-orbit decoder through the use of this model. The symmetries of the 0_i-orbit structure allow for certain reductions in the space of errors which the decoder model must analyse. First, every 3-flat in the 0_i-orbit null space consists of the non-zero points of seven 1-flats from the 0_1-orbits. Thus, one of the non-zero points of a 0_1-orbit 1-flat present in a 3-flat, implies the other non-zero point is as well. Hence, it is only necessary to consider one of the two non-zero points in each 1-flat as a possible single error. Consequently, in this analysis, we refer to a 1-flat error as an error in one of the non-zero points of the 1-flat. This simplification also applies to the 2 and 3-error patterns. If two errors occur in the non-zero points of a 1-flat, that is each non-zero point is in error, then these errors, in effect, cancel out. As far as the decoder is concerned, there are no errors, for these two points always occur together in each null space 3-flat. Consequently, only non-zero 2-error patterns such that each single error is in a distinct 1-flat need be considered. For non-zero 3-error patterns, if two of the errors are in non-zero points of the same 1-flat, then these two errors, as in the 2-error case, 'cancel out', and this reduces to a single error. Thus, for the non-zero 3-error patterns, only error triples such that each point is from a distinct 1-flat, are analysed. In the following we refer to a non-zero error as t_j, t=1,...,10, j=a,b,c, or 11a, that is by the 1-flat label, since it is irrelevant which of the non-zero points is in error. For instance, we refer to the non-zero error triple 1a7a8a and the zero error triple

01a4a. The former error pattern consists of a single error in each of the 1-flats 1a, 7a and 8a. Thus, 1a7a8a refers to 2x2x2=8 possible point error triples, all of which are treated identically by the decoder. Similarly, the error triple 01a4a consists of an error in 0 and a single error in one of the non-zero points of both the 1-flat 1a and the 1-flat 4a, for a total of 1x2x2=4 possible point error triples. Again, the decoder treats each of the four error triples identically. The decrease in the error-space size is quite considerable. The simulated decoder need only test one-eighth of all possible non-zero 3-error sets, one-quarter of the non-zero 2-error sets, one-half of the non-zero single errors, one-quarter of the zero error 3-error sets and one-half of the zero 2-error sets.

A second reduction in the set of error patterns which must be tested by the decoder model is possible because of the 0_i-orbit structure and its cycles. Due to the structure of the cycles of the classes I_1, II_1, III_1 and IV_1, given in Table 4.5.3, it is only necessary to test one element of each cycle as a single error. Thus, the set of single errors 0,1a,1b,1c,7a,7c,10a,11a represents all possible single error patterns. For instance, 1a represents each of the single errors in the cycle (1a 2a 3a 4a 5a 6a) because, as shown in Table 4.5.2, each of the 1-flats ta, t=2,...,6, has the exact same distribution in the 3-flats as 1a. This is true for the representatives of the other cycles. This simplification can be extended to the 2 and 3-error sets. For instance, the non-zero pair 1a7a represents the pairs of errors, 2a8a, 3a9a, 4a7b, 5a8b, 6a9b. Similarly, the triple 1a7a10a represents the error triples 2a8a10c, 3a9a10b, 4a7b10a, 5a8b10c, 6a9b10b. Similar representations are used if one of the errors is 0. With this second reduction of the error space, we discuss the results of the computer simulation of the decoder.

We begin with the possible single errors. If 0 is in error, the count is 27, and hence the decoder correctly determines that e_0, the error digit in position 0, is a 1. If 1a, 1b, 1c, 7a or 7c is in error, then the count is 1 and the decoder makes the decision that $e_0=0$, that is that the digit in position 0 is correct. All other errors in the same cycle as these, have the same count and hence are correctly decoded. If 10a or 11a is in error, the count is 0 and the decoder determines correctly that $e_0=0$.

The 2-error patterns with one error in 0 and the other error any single error except 11a, have a count of 26. Consequently, the decoder determines correctly that there is an error in position 0 and hence that $e_0=1$. For the error pair 011a, the count is 27, and again the decoder decides that $e_0=1$. Every non-zero pair of errors has a count less than 14 and hence the decoder decides that $e_0=0$, that is that no error has occurred in position 0.

The sets of error triples can be divided into three distinct groups. The first group consists of the non-zero error triples with a count of 15, or less than 14. Each error triple of the second group has a count of at least 17 and consists of the 0 error plus two non-zero errors. The third group has both zero and non-zero error triples and a count of either 14 or 16. We discuss each group separately.

The first group is comprised of all possible non-zero error triples excepting those given in Table 5.4.1. The count for these error triples ranges from 1 to 13 or is 15.

Hence, the decoder decides correctly that the digit in position 0 is correct, and therefore that $e_0=0$.

The second group consists of all zero error triples except those listed in Table 5.4.1. The count for this group ranges from 17 to 24 and hence the decoder makes the decision that the digit in position 0 is in error and e_0 is set to 1.

We note that none of the 1, 2 or 3-errors discussed so far have required the testing of the flags f_1, f_2 or f_3.

It is of interest to note at this point that it is possible to obtain from the decoder, information about the composition of the error triples. For instance, if the count is 24, we know from the simulation that the error triple is one of the six error sets, 011ata, t=1,...,6. If the count is 23, then the error triple is 0ta11a or 0tb11a, t=7,8,9. If the count is 18, then one of the errors is 0 and one is 10a, 10b or 10c. More examples are given in Chapter 7 of the added information concerning the error sets that it is possible to gain from the decoder.

The third group is the set of 3-errors which have a count of either 14 or 16. We list these in Table 5.4.1. These error triples are broken down into sets such that each set is the cycle associated with the first point error triple of the set. Hence, each triple is treated identically by the decoder. The error triples followed by f_i, i=1,2,3, have the flag f_i set. That the decoding algorithm corrects the digit in position 0 when the error triples given in Table 5.4.1 occur, follows from the definition of the decoder and Table 4.5.1 (for the setting of the flags). We note that each triple in Table 5.4.1 which is followed by a flag consists of II_1 1-flat errors and/or the zero error only.

count c=14		count c=16		
1a9a7b	7a7b0;f_2,f_3	1a4b7c	1a8b0;f_2	7c10b0;f_1
2a7b8b	8a8b0;f_2,f_3	2a5b8c	2a9b0;f_2	8c10a0;f_1
3a8b9b	9a9b0;f_2,f_3	3a6b9c	3a7a0;f_2	9c10c0;f_1
4a9b7a		4a1b7c	4a8a0;f_2	
5a7a8a	7a7c0;f_2,f_3	5a2b8c	5a9a0;f_2	7c8c0;f_2
6a8a9a	8a8c0;f_2,f_3	6a3b9c	6a7b0;f_2	8c9c0;f_2
	9a9c0;f_2,f_3			9c7c0;f_2
1b1c7b		1b2c9b	1a9b0;f_2	
2b2c8b	7b7c0;f_2,f_3	2b3c7a	2a7a0;f_2	1c8b0;f_2
3b3c9b	8b8c0;f_2,f_3	3b4c8a	3a8a0;f_2	2c9b0;f_2
4b4c7a	9b9c0;f_2,f_3	4b5c9a	4a9a0;f_2	3c7a0;f_2
5b5c8a		5b6c7b	5a7b0;f_2	4c8a0;f_2
6b6c9a		6b1c8b	6a8b0;f_2	5c9a0;f_2
				6c7b0;f_2
1a2c4c		1c9b9c	1b8b0;f_2	
2a3c5c		2c7a7c	2b9b0;f_2	
3a4c6c		3c8a8c	3b7a0;f_2	
4a5c1c		4c9a9c	4b8a0;f_2	
5a6c2c		5c7b7c	5b9a0;f_2	
6a1c3c		6c8b8c	6b7b0;f_2	
7a7b10a;f_2				
8a8b10c;f_2				
9a9b10b;f_2				

Table 5.4.1: Error triples with c=14, 16;
flag f_1: 1 input to (10_3) counter is 1,
flag f_2: 5 or 7 inputs to (10_3) counter are 1
flag f_3: 2 or 3 inputs to (11_3) counter are 1.

It is possible to determine information from the decoder as to the composition of the error triples. For instance, if the count is 14 and both flag f_2 and f_3 are set, then 0 is in error plus one of the pairs tatb, tatc, tbtc, t=7,8,9. If the count is 16 and the f_1 flag set, then the error triple is 07c10b, 08c10a or 09c10c. If no flags are set, and the count is 14 or 16, then it is known that one of the non-flagged error triples from Table 5.4.1 is in error. Such knowledge can be useful if an analysis of the channel errors is being made. Moreover, the decoding process can be shortened, since it is unnecessary to decode the positions known to be correct. Only those positions which correspond to possible errors need be decoded.

As was noted in the definition of the decoder, if there is a computer associated with the channel, which is frequently the case, the testing and setting of the flags is a trivial task. However, even if this must be done using circuitry, the few additional flip-flops required to implement the flags is not costly. Moreover, we now show that the number of error triples which require the testing of flags, that is the error triples of Table 5.4.1, is very small. The calculation of the total number of possible correctable errors and the percentage of these which appear in Table 5.4.1 follows. The following error sets can be chosen, where each error is distinct, that is no two non-zero errors occur in the same 1-flat:

3 non-zero errors in 62x60x58 = 215760 ways,
0 + 2 non-zero errors in 1x62x60 = 3720 ways,
2 non-zero errors in 62x60 = 3720 ways,
0 + a non-zero error in 1x62 = 62 ways,
1 error in 63 = 63 ways,

for a total of 223325 correctable error patterns. Now the number of these patterns occurring in Table 5.4.1 is as follows:

3 non-zero errors (2x2x2)x39 = 312,
0 + 2 non-zero errors (2x2)x39 = 156,

for a total of 468 error patterns. Thus, the flag checking is necessary for only 468/223325 < 0.21% < $\frac{1}{4}$% of all possible correctable error patterns. If a decoder which corrected only 99.75% of all possible correctable error patterns were acceptable, then the resulting modified 0_i-orbit decoder would be very simple. It would require 27 7-input threshold units. If a count of 14 or 16 were flagged, then retransmission could be used for the $\frac{1}{4}$% of the error patterns that the decoder could not correct.

The simplicity and power of the defined decoder is obvious.

5.5 Conclusions

We have defined and analysed the 0_i-orbit decoder of the order-3 (63,41) PG code in this chapter. The simplicity of the decoder, requiring only 27 threshold units, two counters and three flags, was emphasized.

We began the chapter with a detailed discussion of the standard Majority Logic Decoder of the order-3 (63,41) PG code. MLD was discussed in Chapter 3 in general. In this chapter the discussion was made specific with a study of the circuitry required for the decoder of the (63,41) PG code. This was followed by the definition of the 0_i-orbit decoder of the code, comparing it to the standard Majority Logic Decoder. The reduction in circuitry for the 0_i-orbit decoder vs. the Majority Logic Decoder was significant. The former, unlike the latter, needs no division or multiplication circuitry. Even the simplified version of the Majority Logic Decoder requires more than twice the number of threshold units necessary for the 0_i-orbit decoder. By referring to the results of a computer simulation of the decoder, the various i-error sets, i=1,2,3, were analysed. Finally, it was shown that the testing of the 0_i-orbit decoder flags is necessary for less than $\frac{1}{4}$% of all possible correctable error patterns. Moreover, if it is acceptable to correct only 99.75% of all correctable error patterns, the 0_i-orbit decoder can be simplified even further.

CHAPTER 6: O_i-ORBIT DECODER OF ORDER-5 (255,218) PG CODE

6.1 Introduction

The order-5 (255,218) PG code and its O_i-orbit decoder are studied in this chapter. The analysis is similar to, but less detailed than, that given the order-3 (63,41) PG code. This is due to the fact that the O_i-orbit structures of PG(7,2) are merely extensions of those of PG(5,2). The objective of this chapter is then not to emphasize the mathematical symmetries as in Chapter 4, but to illustrate the dramatic decrease in the circuitry requirements if the O_i-orbit decoder is used to decode the order-5 (255,218) PG code instead of the Majority Logic Decoder.

6.2 Standard Majority Logic Decoder of Order-5 (255,218) PG Code

As the order of a Majority Logic Decodable code increases, so does the complexity of the decoder. With each added level of Majority Logic, more circuitry is required. In this section we discuss the circuitry for MLD an order-5, rather than order-3, code, in particular, the order-5 (255,218) PG code. As the theory of the decoder was given in Chapter 5, we simply present the details of the circuitry here.

In order to calculate the number of threshold units necessary to decode using MLD, the following information is required. Referring the Chapter 3, the number of 5-flats orthogonal on the position 0 is:

$$\lambda(0,5,7,2) = \frac{(2^7-1)(2^6-1)(2^5-1)(2^4-1)(2^3-1)}{(2^5-1)(2^4-1)(2^3-1)(2^2-1)} = 2667,$$

the number of 4-flats orthogonal on the position 0 is:

$$\lambda(0,4,7,2) = \frac{(2^7-1)(2^6-1)(2^5-1)(2^4-1)}{(2^4-1)(2^3-1)(2^2-1)} = 11811,$$

the number of 3-flats orthogonal on the position 0 is:

$$\lambda(0,3,7,2) = \frac{(2^7-1)(2^6-1)(2^5-1)}{(2^3-1)(2^2-1)} = 11811,$$

the number of 2-flats orthogonal on position 0 is:

$$\lambda(0,2,7,2) = \frac{(2^7-1)(2^6-1)}{(2^2-1)} = 2667,$$

and the number of 1-flats orthogonal on the position 0 is:

$$\lambda(0,1,7,2) = (2^7-1) = 127.$$

Also, the number of 5-flats orthogonal on a given 4-flat is:

$$\lambda(4,5,7,2) = (2^3-1) = 7,$$

the number of 4-flats orthogonal on a given 3-flat is:

$$\lambda(3,4,7,2) = (2^4-1) = 15,$$

the number of 3-flats orthogonal on a given 2-flat is:

$$\lambda(2,3,7,2) = (2^5-1) = 31,$$

the number of 2-flats orthogonal on a given 1-flat is:

$$\lambda(1,2,7,2) = (2^6-1) = 63,$$

and the number of 1-flats orthogonal on a point is:

$$\lambda(0,1,7,2) = (2^7-1) = 127.$$

Given these quantities, it is now possible to describe the circuitry required to Majority Logic Decode the order-5 (255,218) PG code.

The process begins with a preliminary step of multiplying the received word by $X^{255-218}=X^{37}$ and dividing the result by the generating polynomial g(X). The remainder after division is stored in the syndrome register. On the first step, as there are seven 5-flats orthogonal on each 4-flat, 11811x7=82677 sets of taps and GF(2) adders are necessary to obtain the inputs to the 11811 7-input threshold units. On the second step, there are 11811 15-input threshold units which output estimates of the 3-flats. These estimates are input to the 2667 31-input threshold units of step 3. The outputs, estimates of the 2-flats, are input to the 127 63-

input threshold units of step 4. On the final step, these estimates are input to a single 127-input threshold unit whose output is the estimate of the error digit in position 0.

Thus, the circuitry required for MLD the order-5 (255, 218) PG code involves:

i) circuitry for the premultiplication of the received word by X^{37} and division of the result by the generating polynomial g(X),

ii) 11811x7=82677 sets of taps on the syndrome register and corresponding GF(2) adders,

iii) 11811 7-input threshold units,

iv) 11811 15-input threshold units,

v) 2667 31-input threshold units,

vi) 127 63-input threshold units,

vii) one 127-input theshold unit.

A simplification similar to that made for the Majority Logic Decoder of the (63,41) PG code can be made to the (255,218) PG code Majority Logic Decoder. Since, on the first step, the total number of errors correctable, $[7/2] = 3$, is determined, no more than 3 errors can be corrected on any step. Thus three errors are corrected if only seven estimates are input to the threshold units at each level. As a result, the following circuitry will also decode the order-5 (255,218) PG code:

i) circuitry for the premultiplication of the received word by X^{37} and division of the result by the generating polynomial g(X); the remainder is stored in a register,

ii) 2401x7=16807 sets of taps on the syndrome register and corresponding GF(2) adders

iii) 2401+343+49+7+1=2801 7-input threshold units.

We refer to both the standard and simplified versions of the Majority Logic Decoder for the order-5 (255,218) PG code when we discuss the 0_i-orbit decoder of the code.

6.3 PG(7,2) 0_i-orbit Structure

0_i-orbits can be defined on PG(7,2). To do so, we begin by applying Yamamoto et al's [58] sixth theorem to the flats of dimension (m-2)=(7-2)=5, and of dimension 1 in PG(7,2). First, however, we calculate the total number of 5-flats and the total number of 1-flats in PG(7,2). From Chapter 3, there are:

$$\emptyset(7,5,2) = \frac{(2^8-1)(2^7-1)(2^6-1)(2^5-1)(2^4-1)(2^3-1)}{(2^6-1)(2^5-1)(2^4-1)(2^3-1)(2^2-1)} = 10795$$

5-flats in PG(7,2) and

$$\emptyset(7,1,2) = \frac{(2^8-1)(2^7-1)}{(2^2-1)} = 10795$$

1-flats in PG(7,2). We recall that the number of (m-2)=3-flats is the same as the number of 1-flats of PG(5,2). Similarly, the number of (m-2)=5-flats is the same as the number of 1-flats in PG(7,2). We now apply Theorem 6 for d=5,1.

The highest common factor of (m+1,d+1) is HCF(8,2)=2^1. Thus the number of cycles is (1+1)=2, one of which, from Theorem 3, is v=255, the number of points in the geometry. From Theorem 6, we have:

$x_1=1$	$x_1=0$
$\theta(1) = (2^8-1)/(2^2-1) = 85$	$\theta(0) = (2^8-1)/(2-1) = 255$
$m(1) = (8/2)-1 = 3$	$m(0) = (8/2^0)-1 = 7$
$d(1) = (6/2)-1 = 2$	$d(0) = (6/2^0)-1 = 5$
$q(1) = 2^2 = 4$	$q(0) = 2^{2^0} = 2$
$n(1) = \emptyset(3,2,4) = 85$	$n(0) = \emptyset(7,5,2) = 10795$
$\tau(1) = 85/85 = 1$	$n^*(0) = 10795 - 85 = 10710$
	$\tau(0) = 10710/255 = 42.$

These calculations show that there are 42 i5f's of m.c. 255

and one i5f of m.c. 85. Each of the 42 i5f's of m.c. 255 generates 255 distinct 5-flats. The point representations of these flats are obtained by subtracting j mod 255 from the point set of each i5f, j=0,1,...,254. Similarly, the 85 5-flats generated from the i5f of m.c. 85 are obtained by subtracting j, mod 255, from the point representation of the i5f, j=0,1,...,84.

The order-5 (255,218) PG code has all the 5-flats of PG(7,2) in its null space. This code is cyclic, so when decoding, it is only necessary to consider those 5-flats in the null space which contain the point 0. Thus, only the 5-flats in PG(7,2) which contain the point 0 are considered as null space 5-flats. Using the arguments of Section 4.3, we need generate for each i5f of m.c. 255, only the 63 5-flats which contain the point 0.

From Theorem 2 above and the discussion in Section 4.3, the i5f of m.c. 85 is composed of 21 1-flats, one of which is (0 85 170). The other 20 1-flats are $(0+c_i\ 85+c_i\ 170+c_i)$, addition mod 255, for positive integers c_i, i=1,...,20. As a result, the i5f of m.c. 85 generates only 21 distinct 5-flats through the point 0.

As in Chapter 4, we now apply Theorem 6 to the d=1-flats. In this case (d+1)=2, so HCF(m+1,d+1)=HCF(8,2)=2^1, and hence there are (1+1)=2 distinct cycles, one of which must be 255, by Theorem 3. Applying Theorem 6, we have that there are 42 i1f's of m.c. 255 and one i1f of m.c. 85. Each i1f of m.c. 255 generates 255 1-flats and the i1f of m.c. 85 generates 85 1-flats. As in Chapter 4, it is only necessary to know the 1-flats which pass through 0. As there are $\emptyset(7,0,1)=3$ points in each 1-flat, there are 3 1-flats generated from each i1f of m.c. 255 which contain the point 0. From Theorem

2 above, the i1f with m.c. 85 is the flat $(a_0\alpha^0 + a_1\alpha^{85})$, a_0, a_1 from GF(2), and thus, the points of the flat are (0 85 170), irrespective of the polynomial chosen to represent the geometry. As for the 11_1a 1-flat (0 21 42) of PG(5,2), the i1f of m.c. 85 of PG(7,2) generates only one distinct flat which contains the point 0. The calculations for Theorem 6 are given below.

$\underline{x_1=1}$

$\theta(1) = (2^8-1)/(2^2-1) = 85$

$m(1) = (8/2^1)-1 = 3$

$d(1) = (2/2)-1 = 0$

$q(1) = 2^2 = 4$

$n(1) = \phi(3,0,4) = 85$

$\eta(1) = 85/85 = 1$

$\underline{x_1=0}$

$\theta(0) = (2^8-1)/(2-1) = 255$

$m(0) = (8/2^0)-1 = 7$

$d(0) = (2/2^0)-1 = 1$

$q(0) = 2^1 = 2$

$n(0) = \phi(7,1,2) = 10795$

$n^*(0) = 10795 - 85 = 10710$

$\eta(0) = 10710/255 = 42.$

As in the PG(5,2) case, the 1-flats and the (m-2)-flats have the same distinct cycles and number of initial flats of these minimal cycles. Consequently we are able to define the orbits and 0_i-orbits, and hence a simplified decoder.

The transformation Z was defined in Chapter 4 as the transformation which takes the point j to the point (j-1) mod $(2^{m+1}-1)=255$. The transformations $\{Z, Z^2, \ldots, Z^{255}=Z^0=e\}$ form a group over the set of all i-flats in PG(7,2). For i=5 and i=1, this group partitions the i-flats into orbits, where each orbit corresponds to one of the iif's and i-flats generated from it. Thus, there are 42 orbits with 255 members each and one orbit with 85 members. As we are interested only in those flats containing the point 0, we use the term 0_i-orbit, introduced in Chapter 4, to refer to the subset of an orbit consisting of only those i-flats in which the point

0 occurs. For i=5, each O_5-orbit with flats of m.c. 255, has 63 members. The O_5-orbit corresponding to the i5f of m.c. 85 has 21 members. The O_1-orbits with flats of m.c. 255 have three members each and the O_1-orbit with the flat of m.c. 85 has one member. We say that the O_i-orbit has the m.c. of its constituent i-flats.

There is a one to one correspondence between the 43 orbits and the 43 iif's, and between the O_i-orbits and the iif's through 0, i=1,5. We recall a similar correspondence in PG(5,2). We list the O_1-orbits of PG(7,2) in Table 6.3.1, using the point representation.

(1_1)	(2_1)	(3_1)	(4_1)	
a: 0 1 25	a: 0 2 50	a: 0 4 100	a: 0 8 200	
b: 0 24 254	b: 0 48 253	b: 0 96 251	b: 0 192 247	
c: 0 230 231	c: 0 205 207	c: 0 155 159	c: 0 55 63	I_1
(5_1)	(6_1)	(7_1)	(8_1)	
a: 0 16 145	a: 0 32 35	a: 0 64 70	a: 0 128 140	
b: 0 129 239	b: 0 3 223	b: 0 6 191	b: 0 12 127	
c: 0 110 126	c: 0 220 252	c: 0 185 249	c: 0 115 243	
(9_1)	(10_1)	(11_1)	(12_1)	
a:0 5 138	a: 0 10 21	a: 0 20 42	a: 0 40 84	
b:0 133 250	b: 0 11 245	b: 0 22 235	b: 0 44 215	
c:0 117 122	c: 0 234 244	c: 0 213 233	c: 0 171 211	II_1
(13_1)	(14_1)	(15_1)	(16_1)	
a:0 80 168	a: 0 160 81	a: 0 65 162	a: 0 130 69	
b:0 88 175	b: 0 176 95	b: 0 190 97	b: 0 125 194	
c:0 87 167	c: 0 174 79	c: 0 158 93	c: 0 61 186	
(17_1)	(18_1)	(19_1)	(20_1)	
a:0 13 99	a:0 26 198	a:0 52 141	a:0 104 27	
b:0 86 242	b:0 172 229	b:0 80 203	b:0 178 151	
c:0 156 169	c:0 57 83	c:0 114 166	c:0 228 77	III_1
(21_1)	(22_1)	(23_1)	(24_1)	
a:0 208 54	a:0 161 108	a:0 67 216	a:0 134 177	
b:0 101 47	b:0 202 94	b:0 149 188	b:0 43 121	
c:0 201 154	c:0 147 53	c:0 39 106	c:0 78 212	
(25_1)	(26_1)	(27_1)	(28_1)	
a:0 19 92	a:0 38 184	a:0 76 113	a:0 152 226	
b:0 73 236	b:0 146 217	b:0 37 179	b:0 74 103	IV_1
c:0 163 182	c:0 71 109	c:0 142 218	c:0 29 181	

(29_1)	(30_1)	(31_1)	(32_1)	
a:0 49 197	a:0 98 139	a:0 196 23	a:0 137 46	IV$_1$
b:0 148 206	b:0 41 157	b:0 82 59	b:0 164 118	
c:0 58 107	c:0 116 214	c:0 232 173	c:0 209 91	
(33_1)	(34_1)	(35_1)	(36_1)	
a:0 7 112	a:0 14 224	a:0 28 193	a:0 56 131	V$_1$
b:0 105 248	b:0 210 241	b:0 165 227	b:0 75 199	
c:0 143 150	c:0 31 45	c:0 62 90	c:0 124 180	
(37_1)	(38_1)	(39_1)	(40_1)	
a:0 111 246	a:0 222 237	a:0 189 219	a:0 123 183	VI$_1$
b:0 9 120	b:0 18 240	b:0 36 225	b:0 72 195	
c:0 135 144	c:0 15 33	c:0 30 66	c:0 60 132	
(41_1)	(42_1)		(43_1)	
a:0 17 68	a:0 34 136	VII$_1$	a:0 85 170	VIII$_1$
b:0 51 238	b:0 102 221			
c:0 187 204	c:0 119 153			

Table 6.3.1: 0_1-orbits of PG(7,2), minimal polynomial (0 2 3 4 8)

The labelling of the flats is as in Chapter 4. The 0_1-orbits are numbered (1_1), (2_1), ..., (43_1), with constituent 1-flats ta, tb, tc, t=1,...,42. The subscript 1 on t is used only if it is not obvious that 1-flats are being discussed. The single 1-flat of (43_1) is labelled 43_a. The 0_1-orbits are divided into the eight classes I_1, II_1,..., $VIII_1$, defined later.

The 1-flats of Table 6.3.1 are formed as the 1-flats of Table 4.4.1 were, taking two linearly independent points, one of which is 0.

The correspondence exhibited between the 0_1-orbits and the 0_3-orbits of PG(5,2) has a counterpart in PG(7,2). Here, there is a one to one correspondence between the 0_1-orbits and the 0_5-orbits which is merely an extension of the PG(5,2) case. Each 0_5-orbit contains the six non-zero points of one and only one 0_1-orbit (1_1),...,(42_1), 31 times, and the remaining non-zero points 15 times each. Consequently, the

O_5-orbits are numbered $(1_5),\ldots,(42_5)$, to reflect this, where the O_5-orbit (t_5) contains the non-zero points of (t_1) 31 times. The O_5-orbit of m.c. 85 corresponds to the O_1-orbit (43_1) and is numbered (43_5). We note that the number of times the non-zero points of the O_1-orbit 1-flats repeat in a (m-2)-flat is equivalent to the number of points in a (m-3)-flat.

Of the 63 5-flats in any O_5-orbit (t_5) of m.c. 255, there are eight 5-flats which do not contain a 1-flat from (t_1), $t=1,2,\ldots,40$. For this study, we omit these eight 5-flats and refer to the O_5-orbit (t_5) as the 55 5-flats which intersect on 0 and contain at least one 1-flat from (t_1).

The three subsets A_t, B_t, C_t, $t=1,2,\ldots,40$, are formed to correspond to the similar subsets in PG(5,2). In each (t_5) O_5-orbit, $t=1,2,\ldots,40$, there are eight 5-flats containing only the 1-flat t_1a from (t_1). These are labelled t_5a_i, $\ldots$, t_5a_{viii}. Similarly, there are eight 5-flats which contain t_1b, eight containing t_1c, eight containing both t_1a and t_1b, eight containing t_1a and t_1c, eight containing t_1b and t_1c, and seven containing all three 1-flats t_1a, t_1b, t_1c. The labelling of these is consistent with the labelling of the 5-flats containing t_1a only. The subset A_t is defined as the subset of 5-flats from (t_5) which contains the 1-flat t_1a, that is the 31 5-flats, t_5a_j, t_5ab_j, t_5ac_j, t_5abc_s, $j=i,\ldots,viii$, $s=i,\ldots,vii$. B_t and C_t are defined similarly. As in the PG(5,2) case, the subsets A_t, B_t and C_t are non-orthogonal on their defining 1-flats a, b, c, respectively. These non-orthogonal subsets are the basis of the O_i-orbit decoder.

In the PG(7,2) there are certain O_5-orbits which have unique structures. These O_5-orbits, (41_5), (42_5), (43_5), are now described. They play a special role in the decoding process, just as (10_3) and (11_3) did in PG(5,2).

In (41_5) and (42_5), fifteen of the 5-flats contain all three 1-flats 41a, 41b, 41c, and 42a, 42b, 42c, respectively. The (41_5) 5-flats which do not contain all three of 41a, 41b, 41c, are omitted, leaving the 15 5-flats, $41_5abc_i,\ldots,41_5abc_{xv}$. The (42_5) 0_5-orbit is defined similarly. The subsets A_{41} and A_{42} are defined as the whole 0_5-orbit (t_5), t=41,42, respectively. It is interesting to note that 41a, 41b, 41c, 43a appear in each of the 15 5-flats of (42_5) and that 42a, 42b, 42c, 43a appear in each of the 15 5-flats of (41_5). These thirty 5-flats play an important role in decoding. Some of the 48 5-flats omitted from (41_5) and the 48 omitted from (42_5) have a special use in decoding.

In (43_5), the single 1-flat 43a occurs in each of the 21 5-flats. These flats are labelled $43_5a_i,\ldots,43_5a_{xxi}$. This 0_5-orbit also has a special role in the decoding algorithm. The 5-flats of these two classes appear in Appendix A.

The transformation g, introduced in Chapter 4, which takes the point j to the point 2j, mod 255, establishes symmetries in the PG(7,2) 0_i-orbits similar to the ones in the PG(5,2) 0_i-orbits. The cycles of the 0_1-orbits, defined by g are given in Table 6.3.2. Similarly, g can be applied to the 5-flats of each 0_5-orbit to give the classes $I_5,\ldots,$ $VIII_5$. We do not include them here as they follow directly from the 1-flat cycles and are similar to the 3-flat cycles of Table 4.5.4. The cycles at a gross level of 0_i-orbits are more illustrative and are listed in Table 6.3.3. The subscript g on each cycle class denotes that the cycles are induced by the transformation g.

Class	1-flat Cycles	Cycle Order
I_1	(1a 2a 3a 4a 5a 6a 7a 8a)	8
	(1b 2b 3b 4b 5b 6b 7b 8b)	8
	(1c 2c 3c 4c 5c 6c 7c 8c)	8
II_1	(9a 10a 11a 12a 13a 14a 15a 16a)	8
	(9b 10b 11b 12b 13b 14b 15b 16b)	8
	(9c 10c 11c 12c 13c 14c 15c 16c)	8
III_1	(17a 18a 19a 20a 21a 22a 23a 24a)	8
	(17b 18b 19b 20b 21b 22b 23b 24b)	8
	(17c 18c 19c 20c 21c 22c 23c 24c)	8
IV_1	(25a 26a 27a 28a 29a 30a 31a 32a)	8
	(25b 26b 27b 28b 29b 30b 31b 32b)	8
	(25c 26c 27c 28c 29c 30c 31c 32c)	8
V_1	(33a 34a 35a 36a)	4
	(33b 34b 35b 36b 33c 34c 35c 36c)	8
VI_1	(37a 38a 39a 40a)	4
	(37b 38b 39b 40b 37c 38c 39c 40c)	8
VII_1	(41a 42a)	2
	(41b 42b 41c 42c)	4
VIII_1	(43a)	1

Table 6.3.2: 1-flat Cycles

Class	0_i-orbit Cycle	Order
I_g	(1 2 3 4 5 6 7 8)	8
II_g	(9 10 11 12 13 14 15 16)	8
III_g	(17 18 19 20 21 22 23 24)	8
IV_g	(25 26 27 28 29 30 31 32)	8
V_g	(33 34 35 36)	4
VI_g	(37 38 39 40)	4
VII_g	(41 42)	2
VIII_g	(43)	1

Table 6.3.3: Cycles of the 0_i-orbits

The 0_5-orbits (41_5) and (42_5) partition the 0_1-orbit 1-flats, excepting the (41_1) and (42_1) 1-flats, into distinct blocks, such that no 1-flat appears in more than one block. This partition defines the symmetry blocks of PG(7,2). These blocks are the basis of many of the symmetric distributions of 1-flats in the O_5-orbits. Moreover, sets of errors which have the same symmetry block composition are treated similarly by the decoder. The 18 symmetry blocks are listed in Table 6.3.4, where, from symmetry block $S_j(0)$, the remaining h symmetry blocks generated from it, are obtained by applying the transformation g, a total of h times.

Symmetry Block	Order
$S_1(0)$: 1a 2b 5a 6b 26b 30b 35b 35c	4
$S_1(1)$: 2a 3b 6a 7b 27b 31b 36b 36c	
$S_1(2)$: 3a 4b 7a 8b 28b 32b 33c 33b	
$S_1(3)$: 4a 5b 8a 1b 29b 25b 34c 34b	
$S_2(0)$: 1c 5c 9b 13b 17b 18c 21b 22c	4
$S_2(1)$: 2c 6c 10b 14b 18b 19c 22b 23c	
$S_2(2)$: 3c 7c 11b 15b 19b 20c 23b 24c	
$S_2(3)$: 4c 8c 12b 16b 20b 21c 24b 17c	
$S_3(0)$: 9a 12c 13a 16c 27c 31c 40b 40c	4
$S_3(1)$: 10a 13c 14a 9c 28c 32c 37c 37b	
$S_3(2)$: 11a 14c 15a 10c 29c 25c 38c 38b	
$S_3(3)$: 12a 15c 16a 11c 30c 26c 39c 39b	
$S_4(0)$: 17a 19a 21a 23a 25a 27a 29a 31a	2
$S_4(1)$: 18a 20a 22a 24a 26a 28a 30a 32a	
$S_5(0)$: 33a 34a 35a 36a 37a 38a 39a 40a	1
$S_6(0)$: 41a 42a 41b 42b 41c 42c 43a	1

Table 6.3.4: Symmetry Blocks of PG(7,2)

A given member of a symmetry block appears in a 5-flat of (41_5) or (42_5) if and only if all other members of that symmetry block also appear in the 5-flat. A concise representation of (41_5) and (42_5) is given in Table 6.3.5, where $S_j(h)$ represents the symmetry block obtained from h applications of g on $S_j(0)$, that is, $S_j(h)=g^h(S_j(0))$. We note that there is a one to one correspondence between the 5-flats of (41_5) and (42_5). For V_i a 5-flat of (41_5), $g(V_i)$ is a 5-flat of (42_5).

The symmetry blocks are of prime importance in defining the 0_i-orbit decoder of the PG code over PG(7,2), just as the symmetry blocks of PG(5,2) were in the 0_i-orbit decoder of the PG code over PG(5,2).

(41_5) 0_5-orbit			
$S_2(0)$	$S_3(3)$	$S_4(1)$	$S_6(0)$
$S_2(1)$	$S_3(2)$	$S_5(0)$	$S_6(0)$
$S_2(2)$	$S_3(1)$	$S_4(1)$	$S_6(0)$
$S_2(3)$	$S_3(0)$	$S_5(0)$	$S_6(0)$
$S_1(0)$	$S_1(3)$	$S_4(0)$	$S_6(0)$
$S_1(1)$	$S_1(2)$	$S_4(0)$	$S_6(0)$
$S_3(1)$	$S_3(3)$	$S_4(0)$	$S_6(0)$
$S_1(0)$	$S_3(2)$	$S_3(3)$	$S_6(0)$
$S_1(2)$	$S_3(0)$	$S_3(1)$	$S_6(0)$
$S_2(1)$	$S_2(3)$	$S_4(1)$	$S_6(0)$
$S_1(3)$	$S_1(1)$	$S_5(0)$	$S_6(0)$
$S_1(3)$	$S_2(2)$	$S_3(0)$	$S_6(0)$
$S_1(1)$	$S_2(0)$	$S_3(2)$	$S_6(0)$
$S_1(0)$	$S_2(2)$	$S_2(1)$	$S_6(0)$
$S_1(2)$	$S_2(0)$	$S_2(3)$	$S_6(0)$

(42_5) 0_5-orbit			
$S_2(1)$	$S_3(0)$	$S_4(0)$	$S_6(0)$
$S_2(2)$	$S_3(3)$	$S_5(0)$	$S_6(0)$
$S_2(3)$	$S_3(2)$	$S_4(0)$	$S_6(0)$
$S_2(0)$	$S_3(1)$	$S_5(0)$	$S_6(0)$
$S_1(1)$	$S_1(0)$	$S_4(1)$	$S_6(0)$
$S_1(2)$	$S_1(3)$	$S_4(1)$	$S_6(0)$
$S_3(2)$	$S_3(0)$	$S_4(1)$	$S_6(0)$
$S_1(1)$	$S_3(3)$	$S_3(0)$	$S_6(0)$
$S_1(3)$	$S_3(1)$	$S_3(2)$	$S_6(0)$
$S_2(2)$	$S_2(0)$	$S_4(0)$	$S_6(0)$
$S_1(0)$	$S_1(2)$	$S_5(0)$	$S_6(0)$
$S_1(0)$	$S_2(3)$	$S_3(1)$	$S_6(0)$
$S_1(2)$	$S_2(1)$	$S_3(3)$	$S_6(0)$
$S_1(1)$	$S_2(3)$	$S_2(2)$	$S_6(0)$
$S_1(3)$	$S_2(1)$	$S_2(0)$	$S_6(0)$

Table 6.3.5: 5-flats of (41_5), (42_5) in terms of Symmetry Blocks

The detailed study of the 0_i-orbit structure of PG(5,2) emphasized the mathematical symmetries of the structure. The size of PG(7,2) makes a comparable representation unwieldy. Consequently, we stress only the dramatic reduction in the amount of circuitry required to decode the order-5 (255,218) PG code when the 0_i-orbit decoder is used to decode rather than the Majority Logic Decoder.

6.4 0_i-orbit Decoder of Order-5 (255,218) PG Code

The 0_i-orbit non-orthogonal decoder of the (255,218) PG code is a logical extension of the 0_i-orbit decoder of the (63,41) PG code. Much less circuitry and fewer decoding steps are required for this decoder than for the Majority Logic Decoder of the code. The description of the 0_i-orbit follows directly from the discussion of the 0_i-orbit decoder of the order-3 (63,41) PG code. There are, just as for the

PG(5,2) decoder, two decoding steps. We now discuss these.

On the first step, non-orthogonal estimates of the 1-flats of the 0_1-orbits (1_1) through (40_5) are obtained from the 0_5-orbit 5-flats which are initially known to the decoder. Also, dependent on the errors associated with the (41_5) and (42_5) 5-flats, certain binary flags may be set. The estimates obtained from the first step correspond to 1-flats orthogonal on the point 0. These estimates are input to a counter on the second step. Assuming no more than three errors have occurred, the error digit in position 0 is correctly determined by the output of the counter. In a few cases, the flags set in step 1 must also be consulted and sums obtained corresponding to certain 5-flats not used previously. The circuitry for the decoder is now described.

The received word is stored in a register. On the first step, for each 0_5-orbit 5-flat, taps on the register positions corresponding to the points of the 5-flat are input to a binary adder, the output of which is the sum known to the decoder for the 5-flat. Associated with each of the subsets A_t, B_t, C_t, $t=1,\ldots,40$, is a 31-input threshold unit with threshold 16. The 31 inputs are the binary sums corresponding to the 31 5-flats comprising the subsets A_t, B_t, C_t, respectively. The 15 sums corresponding to the 15 5-flats of (41_5) and the 15 sums corresponding to the 15 5-flats of (42_5) are input to two 15-input counter units, respectively. The output of the 120 threshold units are orthogonal estimates of the point 0. A flag f_1 is set if the number of ones input to the (41_5) and (42_5) counters is 11,9 or 9,9 or 9,11 or 3,3, respectively. A second flag, f_2, is set if the inputs are 9,7 or 7,9 or 9,5, respectively. A third flag, f_3, is set if all 30 inputs to the two counters are ones. The three flags are simply binary flip-flops, set if the number of inputs to a counter is a given value.

The circuitry required for this step involves a total of (55x40)+15+15=2230 sets of calculations on the taps of the storage register vs. 82677 (or 16807 in the simplified version) for the standard Majority Logic Decoder. The (55x40) in the above equation refers to the 55 5-flats in each of the 40 O_5-orbits. Three flags may have to be set. A total of 120 31-input threshold units and two counters for the flags, are necessary. The standard decoder requires on the first step, 11811 (2401 in the simplified version) 7-input threshold units.

On the second step the 120 outputs from step 1 are input to a counter. These outputs correspond to 120 1-flats orthogonal on the point 0. The decoder's decision as to the value of the error digit in position 0 is dependent on the value c output by the counter, and in some cases, the flags f_1, f_2, f_3. In the less than 0.02% of cases that f_3 is set, certain 5-flats not previously used in decoding, must be examined. If e_o denotes the error digit in position 0, then the decoder is defined by:

if $c < 100$, then $e_o=0$,
$c > 108$, then $e_o=1$,
$c=108$, f_3 set, then $e_o=0$; f_3 not set, then $e_o=1$,
$100 \leq c \leq 107$, f_1 set, then $e_o=1$,
f_2 set, then $e_o=0$,
f_3 set, $c=101$, then $e_o=0$,
f_3 set, $c \neq 101$, then consult decoding table.

We note that in the last case, that is when the count is in the range 100 to 107 and f_3 is set, $c \neq 101$, that the error triples consist of either the 0 error or one of the members of $S_6(0)$ and two non-zero errors from the same symmetry block.

In the next section it is shown that the flag f_3 is set for less than 0.02% of all the correctable 1, 2, and 3-error patterns. If the f_3 flag is set and $c \neq 101$, sums corresponding to certain 5-flats not previously used must be obtained. We discuss these seven cases where the f_3 flag is set and the count c is in the range 100 to 107, $c \neq 101$, in the next section.

Thus, on the second step, a simple counter, logical units to test the flags and seven further sets of 5-flat sums and logical units are required.

This decoding method does not require that the received word be premultiplied by X^{37} and the result divided by the generating polynomial g(X). The tapped values need only be fed into binary flip-flops and the output then directly input into a threshold unit. The O_i-orbit circuitry consists of a total of 120 31-input threshold units, three counters, three binary flags, logical units to test the flags and a small decoding table of 35 entries. The standard Majority Logic Decoder requires circuitry for multiplication, division, GF(2) adders, 11811 7-input threshold units, 11811 15-input threshold units, 2667 31-input theshold units, 127 63-input threshold units and one 127-input threshold unit, or, in the simplified version 2801 7-input threshold units. The savings in circuitry achieved by the use of the O_i-orbit decoder are significant. If, as suggested for the PG(5,2) case, a mini-computer is associated with the channel, the flag setting and testing step of the O_i-orbit decoder, becomes trivial. Specific 1, 2 and 3-error sets are discussed in the next section.

6.5 Error Analysis of the Decoder

In this section, it is shown that the O_i-orbit decoder of the order-5 (255,218) PG code is capable of correcting all 1, 2 and 3-errors. Hence, the 0_i-orbit decoder possesses the same error-correcting capablilities as the Majority Logic Decoder but requires only a fraction of the circuitry. In Chapter 7, the decoder's ability to correct some errors of higher weight is illustrated.

A computer simulation of the 0_i-orbit decoder was used to test all possible 1, 2 and 3-errors of the order-5 (255, 218) PG code. Reductions to the size of the error space were made using the same techniques as were used to decrease the size of the order-3 (63,41) PG code error space.

We begin the analysis by examining all possible single errors. We observe that it is only necessary to test the 0 error and one member of each cycle of Table 6.3.2. If 0 is in error, the output from the counter in step 2 is 120 and hence the decoder correctly determines that e_o, the error digit in position 0, is a one. If one of 1a, 1b, 1c, 9a, 9b, 9c, 17a, 17b, 17c, 25a, 25b, 25c, 33a, 33b, 37a or 37b is in error, then the count is 1 and the decoder makes the decision that e_o=0, that is that the digit in position 0 is correct. If one of 41a, 41b or 43a is in error, the count is 0, and the decoder determines correctly that e_o=0. All other single errors in the same cycle as one of the above errors have the same count and hence are correctly decoded.

The 2-error patterns consisting of the 0 error and any non-zero error, have a count of 119. Thus, the decoder determines correctly that there is an error in position 0 and hence that e_o=1. Every non-zero pair of errors has a count between 4 and 20 and hence the decoder decides that e_o=0, that is no error has occurred in position 0.

The sets of error triples can be divided into four distinct groups. The first group consists of the non-zero error triples with a count less than 100. Each error triple of the second group has a count of at least 108 and consists of the 0 error plus two non-zero errors. We also include the non-zero triples 41a41b41c, 42a42b41b, 41a41c42b and 42a42c41b, with count 108, in the second group. Error triples with a count between 100 and 107 such that one error is either 0 or a member of $S_6(0)$ and the other two are non-zero errors from two distinct symmetry blocks, are in the third group. The fourth group has both zero and non-zero error triples and a count between 100 and 107. Each triple in this group consists of those error triples such that two of the errors, e_1 and e_2, are from the same symmetry block and the other error is either 0 or the member of $S_6(0)$ in which e_1 and e_2 appear together in Table 6.5.1. The three triples of 1-flats from $S_6(0)$ listed in the S_6 section of Table 6.5.1 are also in this group.

<u>S_1 pairs</u>

41a:	1a26b	2b35c	5a30b	6b35b	2a6a	3b7b	27b31b	36b36c
	3a28b	4b33b	7a32b	8b33c	4a8a	5b1b	29b25b	34c34b
41b:	1a6b	2b5a	26b35b	30b35c	2a36b	3b31b	6a36c	7b27b
	3a33b	4b28b	7a33c	8b32b	4a5b	8a1b	25b34c	29b34b
41c:	1a35c	5a35b	2b26b	6b30b	2a3b	6a7b	27b36c	31b36b
	3a8b	7a4b	28b33c	32b33b	4a34c	8a34b	1b29b	5b25b
42a:	1a5a	2b6b	26b30b	35b35c	2a27b	3b36c	6a31b	7b36b
	3a7a	4b8b	28b32b	33b33c	4a29b	5b34b	8a25b	1b34c
42b:	1a2b	5a6b	26b35c	30b35b	2a7b	3b6a	27b36b	31b36c
	3a33c	4b32b	7a33b	8b28b	4a34b	5b29b	8a34c	1b25b
42c:	1a35b	5a35c	2b30b	6b26b	2a36c	3b27b	7b31b	6a36b
	3a4b	7a8b	28b33b	32b33c	4a1b	8a5b	29b34c	25b34b
43a:	1a30b	2b35b	5a26b	6b35c	2a31b	3b36b	6a27b	7b36c
	3a32b	4b33c	7a28b	8b33b	4a25b	5b34c	8a29b	1b34b

S_2 pairs

41a: 1c9b 5c13b 17b22c 21b18c 2c19c 6b23c 10b22b 14b18b
3c11b 7c15b 19b24c 23b20c 4c21c 8c17c 12b24b 16b20b

41b: 1c13b 5c9b 17b18c 21b22c 2c6c 10b14b 18b22b 19c23c
3c24c 7c20c 11b19b 15b23b 4c20b 9c16b 12b17c 16b21c

41c: 1c22c 5c18c 9b17b 13b21b 2c18b 6c22b 10b23c 14b19c
3c15b 7c11b 19b20c 23b24c 4c8c 12b16b 20b24b 17c21c

42a: 1c18c 5c22c 9b21b 13b17b 2c10b 6c14b 18b23c 22b19c
3c20c 7c24c 11b23b 15b19b 4c12b 8c16b 20b17c 24b21c

42b: 1c17b 5c21b 9b22c 13b18c 2c14b 6c10b 18b19c 22b23c
3c7c 11b15b 19b23b 20c24c 4c17c 8c21c 12b20b 16b24b

42c: 1c5c 9b13b 17b21b 18c22c 2c23c 6c19c 10b18b 14b22b
3c19b 7c23b 11b24c 15b20c 4c16b 8c12b 20b21c 24b17c

43a: 1c21b 5c17b 9b18c 13b22c 2c22b 6c18b 10b19c 14b23c
3c23b.7c19b 11b20c 15b24c 4c24b 8c20b 12b21c 16b17c

S_3 pairs

41a: 9a40c 13a40b 12c27c 16c31c 10a9c 14a13c 28c37c 32c37b
11a38b 15a38c 10c25c 14c29c 12a11c 16a15c 26c39b 30c39c

41b: 9a27c 13a31c 12c40c 16c40b 10a13c 14a9c 28c37b 32c37c
11a15a 10c14c 25c29c 38b38c 12a39c 16a39b 11c30c 15c26c

41c: 9a13a 12c16c 27c31c 40b40c 10a37c 14a37b 9c28c 13c32c
11a29c 15a25c 10c38c 14c38b 12a15c 16a11c 26c39c 30c39b

42a: 9a16c 13a12c 27c40b 31c40c 10a37b 14a37c 13c28c 17c32c
11a10c 15a14c 29c38c 25c38b 12a39b 16a39c 11c26c 15c30c

42b: 9a40b 13a40c 12c31c 16c27c 10a28c 14a32c 13c37b 17c37c
11a14c 15a10c 29c38b 25c38c 12a16a 11c15c 26c30c 39b39c

42c: 9a12c 13a16c 27c40c 31c40b 10a14a 13c9c 28c32c 37b37c
11a38c 15a38b 10c29c 14c25c 12a30c 16a26c 11c39c 15c39b

43a: 9a31c 12c40b 13a27c 16c40c 10a32c 13c37c 14a28c 9c37b
11a25c 14c38c 15a29c 10c38b 12a26c 15c39c 16a30c 11c39b

S_4 pairs

41a: 17a27a 19a29a 21a31a 23a25a 18a32a 20a26a 22a28a 24a30a

41b: 17a29a 19a27a 21a25a 23a31a 18a32a 20a26a 27a28a 24a30a

41c: 17a25a 19a31a 21a29a 23a27a 18a24a 20a22a 26a28a 30a32a

42a: 17a31a 19a25a 21a27a 23a29a 18a28a 20a30a 22a32a 24a26a

42b: 17a23a 19a21a 25a27a 29a31a 18a30a 20a28a 22a26a 24a32a

42c: 17a19a 21a23a 25a31a 27a29a 18a26a 20a32a 22a30a 34a28a

43a: 17a21a 19a23a 25a29a 27a31a 18a22a 20a24a 26a30a 28a32a

S_5 pairs

41a: 33a35a 37a39a 34a38a 36a40a
41b: 33a38a 34a35a 36a39a 37a40a
41c: 33a36a 34a37a 38a39a 35a40a
42a: 33a37a 34a36a 35a39a 38a40a
42b: 33a40a 34a39a 35a36a 37a38a
42c: 33a34a 35a38a 36a37a 39a40a
43a: 33a39a 34a40a 35a37a 36a38a

S_6 triples

41a42a43a 41b42b43a 41c42c43a

Table 6.5.1: Pairs and Triples of E

The pairs in Table 6.5.1 are those which always occur together in 41_5a, 41_5b, 41_5c, 42_5a, 42_5b, 42_5c and 43_5a 5-flats given in Appendix A. The fourth group is referred to as the set E. An example of an error triple in E is 42c1c5c. The two errors 1c and 5c, both from the symmetry block $S_2(0)$, occur together in the S_2 pairs 42c set in Table 6.5.1. We note that associated with each error triple in E is a cycle of errors in E such that each error in the cycle is treated identically by the decoder. For instance, the cycle set associated with 42c1c5c, that is (41b2c6c, 52b3c7c, 41c4c8c), are all in E and treated identically by the decoder. The error triple 01c5c is in E. The (41_5) and (42_5) f_3 flag is set for this triple and the triple 42c1c5c, since 15 inputs to the (41_5), and the 15 to the (42_5), counters are all ones for both these triples.

The first group of error triples is comprised of all possible non-zero error triples excepting those in the set E. The count for these triples ranges from 0 to 99. Hence, the decoder decides correctly that the digit in position 0 is correct, and therefore that $e_0=0$.

When discussing the remaining groups we refer to the number of ones input to the (41_5) and (42_5) counter units of step 1 as i/j, where a total of i of the inputs to (41_5) are ones, and j of the inputs to (42_5) are ones. If the input is 11/9, 9/9 or 3/3 the f_1 flag is set, if the input is 9/7, 7/9 or 9/5 the f_2 flag is set and if the input is 15/15 the f_3 flag is set.

The second group consists of all 0 error triples except those in E (see Table 6.5.2). The count for this group ranges from 108 to 116. If the count is 108 and the f_3 flag set, then e_o is 0, otherwise the decoder makes the decision that the digit in position 0 is in error and e_o is set to 1.

Just as in the PG(5,2) case, information can be obtained from the decoder concerning the composition of the error triples. For instance, if the count is 116, the error triple is in one of the cycles generated from 09a43a, 09b43a, 09c43a, 041a43a, 041b43a. More examples are given in Chapter 7 of the added information concerning the error sets which it is possible to gain from the decoder.

For the third group, either f_1 or f_2 is set. By referring to Table 6.3.5 or Appendix A, the decoding rule that if f_1 is set, 0 is in error and that if f_2 is set, 0 is not in error, is verified. We note that the input to (41_5) and (42_5) is 3/3 if and only if one error is 0, one error is from $S_6(0)$ and the third error is from any symmetry block excepting $S_6(0)$. In this case the 0 and $S_6(0)$ errors appear together in each 5-flat in (41_5) and (42_5) and hence cancel with each other. The remaining error gives a count of 3 in each of (41_5) and (42_5) since each point not in (41_1) or (42_1) appears three times in (41_5) and (42_5). The other counts for which f_1 or f_2 is set are explained similarly.

The error triples of E, the fourth group, are given below in Table 6.5.2.

count=100

1a30b43a
2a31b43a
3a32b43a
4a25b43a
5a26b43a
6a27b43a
7a28b43a
8a29b43a

9a12c42c
10a13c41b
11a14c42b
12a15c41c
13a16c42c
14a9c41b
15a10c42b
16a11c41c

25b34c41b
26b35c42b
27b36c41c
28b33b42c
29b34b41b
30b35b42b
31b36b41c
32b33c42c

33a34a42c
34a35a41b
35a36a42b
36a33a41c

37a40a41b
38a37a42b
39a38a41c
40a39a42c

37a39a41a
38a40a42a

33a40a42b
34a37a41c
35a38a42c
36a39a41b

33a35a0
34a36a0

count=101

33b33c42a
34b34c41a
35b35c42a
36b36c41a

count=102

1a26b41a
2a27b42a
3a28b41a
4a29b42a
5a30b41a
6a31b42a
7a32b41a
8a25b42a

1c18c42a
2c19c41a
3c20c42a
4c21c41a
5c22c42a
6c23c41a
7c24c42a
8c17c41a

1b34b43a
2b35b43a
3b36b43a
4b33c43a
5b34c43a
6b35c43a
7b36c43a
8b33b43a

9b21b42a
10b22b41a
11b23b42a
12b24b41a
13b17b42a
14b18b41a
15b19b42a
16b20b41a

17a25a41c
18a26a42c
19a27a41b
20a28a42b
21a29a41c
22a30a42c
23a31a41b
24a32a42b

17b22c41a
18b23c42a
19b24c41a
20b17c42a
21b18c41a
22b19c42a
23b20c41a
24b21c42a

25b34b42c
26b35b41b
27b36b42b
28b33c41c
29b34c42c
30b35c41b
31b36c42b
32b33b41c

1b34b0
2b35b0
3b36b0
4b33c0
5b34c0
6b35c0
7b36c0
8b33b0

```
count=103

1b29b41c    9c37c42b   25b29b41a        1a35b0   33a34a0
2b30b42c   10c38c41c   26b30b42a        2a36b0   34a35a0
3b31b41b   11c39c42c   27b31b41a        3a33c0   35a36a0
4b32b42b   12c40c41b   28b32b42a        4a34c0   36a33a0
5b25b41c   13c37b42b                    5a35c0
6b26b42c   14c38b41c                    6a36c0   33b33c0
7b27b41b   15c39b42c                    7a33b0   34b34c0
8b28b42b   16c40b41b                    8a34b0   35b35c0
                                                 36b36c0

count=104

1c5c42c     9a27c41b   17a27a41a         1a5a0   1a30b0   1a35c0
2c6c41b    10a28c42b   18a28a42a         2a6a0   2a31b0   2a36c0
3c7c42b    11a29c41c   19a29a41a         3a7a0   3a32b0   3a33b0
4c8c41c    12a30c42c   20a30a42a         4a8a0   4a25b0   4a34b0
           13a31c41b   21a31a41a                 5a26b0   5a35b0
41a42a43a  14a32c42b   22a32a42a         1b5b0   6a27b0   6a36b0
41b42b43a  15a25c41c   23a25a41a         2b6b0   7a28b0   7a33c0
41c42c43a  16a26c42c   24a26a42a         3b7b0   8a29b0   8a34c0
                                         4b8b0

                  1a26b0   25a29a0   25b34b0   33a39a0
                  2a27b0   26a30a0   26b35b0   34a40a0
                  3a28b0   27a31a0   27b36b0   35a37a0
                  4a29b0   28a32a0   28b33c0   36a38a0
                  5a30b0             29b34c0
                  6a31b0   25c29c0   30b35c0   33a40a0
                  7a32b0   26c30c0   31b36c0   34a37a0
                  8a25b0   27c31c0   32b33b0   35a38a0
                           28c32c0             36a39a0
count=105

37b37c42c      1a2b0   1a6b0   1b25b0   25c38c0
38b38c41b      2a3b0   2a7b0   2b26b0   26c39c0
39b39c42b      3a4b0   3a8b0   3b27b0   27c40c0
40b40c41c      4a5b0   4a1b0   4b28b0   28c37b0
               5a6b0   5a2b0   5b29b0   29c38b0
               6a7b0   6a3b0   6b30b0   30c39b0
               7a8b0   7a4b0   7b31b0   31c40b0
               8a1b0   8a5b0   8b32b0   32c37c0
count=106

33a37a42a      1c18c0   1c21b0   1b34c0   17a25a0   17a31a0
34a38a41a      2c19c0   2c22b0   2b35c0   18a26a0   18a32a0
35a39a42a      3c20c0   3c23b0   3b36c0   19a27a0   19a25a0
36a40a41a      4c21c0   4c24b0   4b33b0   20a28a0   20a26a0
               5c22c0   5c17b0   5b34b0   21a29a0   21a27a0
               6c23c0   6c18b0   6b35b0   22a30a0   22a28a0
               7c24c0   7c19b0   7b36b0   23a31a0   23a29a0
               8c17c0   8c20b0   8b33c0   24a32a0   24a30a0

                 25a27a0   25c38c0   25c38b0   33a37a0   33a38a0
                 26a28a0   26c39c0   26c39b0   34a38a0   34a39a0
                 27a29a0   27c40c0   27c40b0   35a39a0   35a40a0
                 28a30a0   28c37b0   28c37c0   36a40a0   36a37a0
                 29a31a0   29c38b0   29c38c0
                 30a32a0   30c39b0   30c39c0
                 31a25a0   31c40b0   31c40c0
                 32a26a0   32c37c0   32c37b0
```

```
count=107

 1a35b42c    1b29b0  1c13b0  1c22c0  17a27a0  17a29a0
 2a36b41b    2b30b0  2c14b0  2c23c0  18a28a0  18a30a0
 3a33c42b    3b31b0  3c15b0  3c24c0  19a29a0  19a31a0
 4a34c41c    4b32b0  4c16b0  4c17c0  20a30a0  20a32a0
 5a35c42c    5b25b0  5c9b0   5c18c0  21a31a0  21a25a0
 6a36c41b    6b26b0  6c10b0  6c19c0  22a32a0  22a26a0
 7a33b42b    7b27b0  7c11b0  7c20c0  23a25a0  23a27a0
 8a34b41c    8b28b0  8c12b0  8c21c0  24a26a0  24a28a0

             25c38b0  1c5c0  25b29b0
             26c39b0  2c6c0  26b30b0
             27c40b0  3c7c0  27b31b0
             28c37c0  4c8c0  28b32b0
             29c38c0
             30c39c0
             31c40c0
             32c37b0
```

Table 6.5.2: Set E Error Triples, f_3 set for each triple

The fourth group, E, requires the extra calculations mentioned in the last section. For these error triples, since the non-zero (or non-$S_6(0)$) errors are in the same symmetry block, a 5-flat of (41_5) or (42_5) either contains all three of the errors or only the one error, 0 or a $S_6(0)$ error. Thus the corresponding binary sum for each 5-flat is one and the input to (41_5) and (42_5) is thence 15/15. We recall that it was necessary to refer to certain (11_3) 0_3-orbit 3-flats to determine e_0 when all inputs to (10_3) were ones in the PG(5,2) case. Similarly, certain of the unused 5-flats must be consulted to determine whether 0 is in error for the error triples from E when all inputs to the comparable (41_5) and (42_5) 0_5-orbits of PG(7,2) are ones. The 5-flats which are used to determine the value of e_0 are from (43_5) or from the 5-flats of 41_5a, 41_5b, 41_5c, 42_5a, 42_5b, 42_5c, given in Appendix A. Corresponding to each of the counts 100, 102,103,...,107, is a set of one or more 5 to 16 bit storage words. For a count c in this range, the value of the error digit in position 0 is determined by whether or not the pattern of binary sums of the 5-flats associated with the count c matches the stored word or words for c. For example, if the count is 102, the stored word is the 8 bit word consisting of all ones. If the 5-flat sums of 43_5a_i, ..., 43_5a_{viii} are also all ones, then e_o is 1, otherwise it is 0. Four of the counts require only the one pattern word consisting of all ones to determine the value of e_o. The remaining three have 8, 15 and 8 pattern words, respectively. A simple binary compare is all that is required to test the matching of the calculated pattern with these. The complete set of decoding patterns for the fourth group, in the form of a decoding table, is given in Table 6.5.3.

Count	Associated 5-flats	Decoding Patterns and Rule
100	41_5a_{viii}, 41_5b_{xii}, 41_5c_{xv}, 42_5b_{xv}, 42_5a_v, 42_5c_{xv}	-if all 1's, then e_o=1; otherwise e_o=0
102	43_5a_i, ..., 43_5a_{viii}	-if all 1's, e_o=1; else e_o=0
103	41_5b_{xi}, 41_5b_{xii}, 41_5c_{xv}, 41_5c_{xvi}, 42_5b_{xv}, 42_5b_{xvi}, 42_5c_{xv}, 42_5c_{xvi}	-if all 1's, e_o=1; otherwise if less than 6 1's, e_o=0; if 6 1's and pattern is 11101011,11011101,01111110, 10111011,11010111,11101110, 10111101,01110111, e_o=1; else e_o=0
104	41_5b_{xi}, 41_5b_{xii}, 41_5c_{xv}, 41_5c_{xvi}, 42_5b_{xv}, 42_5b_{xvi}, 42_5c_{xv}, 42_5c_{xvi}, 43_5a_i, ..., 43_5a_{viii}	-if last 8 bits all 1's, e_o=0; if first 8 bits all 1's, e_o=0, else if pattern one of 01111111,10111111,11011111, 11101111,11110000,00001111, 00000000 then e_o=1; if 6 1's in patterns of 11100111,11111100, 00111111,11001111, e_o=1; else e_o=0; if 4 1's and pattern 00111100,00110011,11000011, 11001100, e_o=1; if in last 8 bits there are 4 1's, e_o=0; else e_o=1
105	41_5b_{ix}, 41_5c_{xiii}, 42_5b_{xiii}, 42_5b_{xiv}, 42_5c_{xiii}	-if all 1's, e_o=0, else e_o=1

Count	Associated 5-flats	Decoding Patterns and Rule
106	41_5a_{ii},41_5a_{vi},41_5a_{viii}, 42_5a_i,42_5a_{iii},42_5a_{iv}	-if all 1's, $e_o=0$; else $e_o=1$
107	41_5b_{xi},41_5b_{xii},41_5c_{xv}, 41_5c_{xvi},42_5b_{xv},42_5b_{xvi}, 42_5c_{xv},42_5c_{xvi}	-if pattern one of 10001011,11100100,00011101, 10110010,01000111,11011000, 00101110,01110001 $e_o=0$; else $e_o=1$

Table 6.5.3: Decoding Table for Error Triples in E

The addition of a decoding table for the E error triples adds decoding time rather than complexity to the decoder, for simple binary comparisons are all that are necessary to correctly determine e_o for these cases. However, for most of the correctable error patterns, the decoding table is not needed and hence no extra time added to the decoding process. In the following we show, assuming all errors equally probable, that the decoding table is consulted for less than 0.02% of the possible correctable error patterns.

We first calculate the number of error triples in E. The number of 0 error triples, from Table 6.5.2, is 250. Each of these triples represents four error triples, for a total of 250x4=1000 0 error triples. For the non-zero case, we have 154 triples, each of which represents eight point error triples for a total of 154x8=1232 error triples. Thus, there are 1000+1232=2232 error triples in E. We now obtain the total number of correctable error patterns. The total number of 1-errors is 255, zero 2-errors is 1x254=254, non-zero 2-errors is 254x252=64008, zero 3-errors is 1x254x252=64008, and non-zero 3-errors is 254x252x250=16002000, for a total of 16,130,525 correctable error patterns. Thus, the the error triples of E comprise only

$$2232/16130525 < 0.02\%$$

of all possible correctable error patterns. If a decoder correcting 99.98% of all correctable error patterns were acceptable, then the resulting 0_i-orbit decoder is extremely simple compared to the Majority Logic Decoder. It would require 120 31-input threshold units, three counters, and three flags. If retransmission were possible, then if the count were in the range 100 to 108, the 0.02% of the uncorrectable errors could be corrected using retransmission.

6.6 Conclusions

In this chapter the 0_i-orbit decoder of the order-5 (255,218) PG code has been defined and analysed. The objective of the discussion presented was to emphasize the enormous difference in the circuitry required for the 0_i-orbit decoder and the Majority Logic Decoder of the code. The standard Majority Logic Decoder requires circuitry for premultiplication and division, GF(2) adders and 26479 threshold units (2801 in the simplified version). The 0_i-orbit decoder does not require division or multiplication circuitry and only a total of 120 threshold units, three counters, three flags and a 35 entry decoding table and associated logical units.

The chapter began with a summary of the circuitry used in MLD the order-5 (255,218) PG code. The 0_i-orbit structure of PG(7,2) was presented. Based on these structures, the 0_i-orbit decoder was defined. That all 1, 2, 3-error patterns are correctable using the defined decoder was established by referring to the results of a simulation model of the decoder. Finally, it was shown that the error patterns for which the decoding table must be consulted comprise less than 0.02% of all correctable error patterns.

CHAPTER 7: MODIFICATIONS TO THE O_i-ORBIT DECODER

7.1 Introduction

In this chapter we analyse further and make modifications to, the O_i-orbit decoders introduced previously. The O_i-orbit structure provides an interpretation of the Projective Geometries which yields considerable information concerning the distribution and composition of the correctable error patterns. Investigation of the errors of higher weight indicates that the decoder can be modified to detect some such errors.

For both the O_i-orbit decoders studied, we discuss information which can be obtained concerning the composition of the 1, 2 and 3-errors. Also, several modifications to the two decoders are suggested which allow for the detection of some 4 and 5-errors.

First, the O_i-orbit decoder of the order-3 (63,41) PG code is considered. Then a similar discussion of the order-5 (255,218) PG code is given.

7.2 Order-3 (63,41) PG Code

7.2.1 Composition of Errors from Knowledge of Decoder's Output

If the composition of the error pattern which has occurred is known, it is possible to shorten the decoding process. As it is only necessary to correct those positions in the received word which can have been in error, only a fraction of the $(2^{m+1}-1)$ digit positions need to be decoded. In this section we show that it is possible, for some error sets, to determine the subset of error sets in which the given error pattern occurs. This requires that the decoder know the count c output from the counter of step 2, the value of the flags f_1, f_2, f_3, and whether the number of inputs to (10_3) is odd or even. In the following we assume that no more than three errors have occurred.

We begin by discussing the single errors. If the count c is 27, then it is immediately known that 0 is in error. If the f_3 flag is set, no other error has occurred and thus, decoding may cease. If the count c is 0, then there is one error in one of the non-zero points of the 1-flat 11a. Hence, the decoder needs to decode only the two positions 21 and 42, as these are the only digits possibly in error. No 2 or 3-error pattern has a count of 0. If the count is one and the f_3 flag not set, the decoder knows that a single non-zero error has occurred in a 1-flat other than 11a.

Similar information is available when two errors have occurred. If the count is 27 and the f_3 flag is not set, then the two errors are 0 and one of the non-zero points of 11a. Consequently only the digit in position 0 and the 11a digits need be decoded. Thus, at most three positions need to be decoded. If the count is 26, the decoder knows that two errors have occurred, one of which is 0. Once the 0 error and the second error have been corrected, decoding may cease. If the two errors are non-zero, then the decoder is able to determine, for some pairs, the error subsets in which the errors occurred. For instance, if the count is 13 and the number of ones input to (10_3) is even, then the two errors are in the cycles generated by 7a7b or 7a7c. If the count is three and the input to (10_3) even, then the error pair is a member of the cycle generated by 1a11a. If the count is four and the input to (10_3) even, then the error pair is a member of the cycle of 1b11a. For these cases only the positions corresponding to possible errors need be decoded, rather than all 63 positions.

The composition of some error triples can be ascertained from knowledge of the count c associated with the error triple. If the count is 24, then the three errors are from the cycle generated by 01a11a or 010a11a. The error triple is a member of the cycle of 01b11a, 01a10b, 01c10b or 07a10c, if the count is 23. If the count is 22, one of the cycles generated from 01c4c or 01c11a contains the triple of errors. If the count is 13 and the input to (10_3) odd, then the non-zero error triple is a member of one of the cycles generated from 1a8b10a, 1a1b8c, 1a2b5c or 7c8c10c. If the flag f_3 is not set and the count is 1, the non-zero error triple is from the cycle generated by 1a5b11a. We recall that if the f_3 flag is set, only one error has occurred.

These 1, 2 and 3-error examples have shown how information as to the composition of the errors can be obtained from the decoder. For certain counts only those positions that may be in error need to be decoded. Decoding time can be shortened if a table is added to the decoder which, for a set of counts, contains the possible digits in error if one of the counts occurs. Only those counts with a predetermined number of associated error positions are included to make storage requirements economical. If the count obtained is in the table, the corresponding positions are decoded. If the count is not in the table, normal decoding is continued until a count in the table appears. Once the positions given in the table for the count have been decoded, decoding terminates.

7.2.2 Errors of Weight Four

Applying the techniques of the previous sub-section, the 0_i-orbit decoder can be modified to detect many errors of weight four. We note that already many errors of weight four are corrected by the decoder defined. For instance, all 4-errors containing 0 an error in the first position with a count greater than 16 are automatically corrected, as are all non-zero 4-errors with a count less than 14. We let w represent the number of ones input to (10_3). In this chapter we denote as S_8', the symmetry block S_8 augmented by the element 0, that is S_8' = 10a10b10c0.

The results presented in this sub-section are based on the following observations. First, if a non-S_8' error triple occurs, the number w, of ones input to (10_3) must be 1 or 3. This follows if we consider the three possible distributions of the error triples in symmetry blocks. If all three errors are in the same symmetry block, then the sum in (10_3) corresponding to that symmetry block is one and all other (10_3) sums are 0. If the three errors occur in three distinct symmetry blocks, then the three corresponding sums in (10_3) each are one and the remaining four are zero. If two errors occur in the same symmetry block, their binary sum is zero. The third error, in a distinct symmetry block, gives a sum of one. All other sums are zero and hence only one of the inputs to (10_3) has a non-zero value.

If the error triple contains one member of S_8', the number of ones input to (10_3) is 5 or 7. We determine this as for the non-S_8' case. If the two non-S_8' errors are in the same symmetry block, the associated binary sum is zero. The S_8' error occurs in each 3-flat of (10_3) and hence each has an associated binary sum of one, for a total of w=7 ones input to (10_3). If the two non-S_8' errors are in distinct symmetry blocks, each sums to zero with the S_8' error. The remaining five inputs correspond to the 3-flats which contain only the S_8' error. Thus there are a total of w=5 inputs to (10_3) which are ones. If two errors are in S_8', w=1 and if all three are in S_8', w=7.

If there are four non-S_8' errors, then w is 0, 2 or 4. If all four errors are in the same symmetry block, or if two errors are in one symmetry block and two in another, then w=0. If two errors are in the same symmetry block and the remaining two in two distinct symmetry blocks, then w=2. If all four errors are in four distinct symmetry blocks, then w=4.

If one error is in S_8', then w can be 4 or 6, depending on the arrangement of errors in symmetry blocks. If each non-S_8' error is in a distinct symmetry block, w=4. If either two or three of the non-S_8' errors are in the same symmetry block, then w=6.

If two of the errors are from S_8', w is 0 or 2 depending on whether the non-S_8' errors are in the same or distinct symmetry blocks, respectively. If there are three S_8' errors, w=6. If there are four S_8' errors, then w=0.

Consequently, if there are three errors, w is odd and if there are four errors, w is even.

The following observations form the basis of the statements concerning 4-error patterns. First, we recall that if two errors occur, one of which is 0, then the count is 26 or 27. Secondly, if two non-zero errors occur, then the count is at most 13. If the count is 13, then the errors are either from the cycle of 7a7b or 7a7c, w equal to 0 or 2 respectively. If the count is 12, the error is in the cycle of 7c10b and w=6. If the count is 11, the error pair is from the cycle of 1a8b or 7c8c, with w=0, or from the cycle of 1a9b, 1b8b or 1c8b, with w=2. Finally, from the above discussion on 4-errors, we recall that w is 0, 2, 4 or 6 if there are four errors. And, from the simulation, if one of the errors is 0, the count is at least 12. We now suggest modifications to the 0_i-orbit decoder to allow for the consideration of 4-errors.

If the count is at least 14, but no more than 26 and w is even, then a 4-error is detected. From the discussion on the assignment of w, if w is 0, 2 or 6, then one of the errors is in S_8'. If the count is 13 and w is 4 or 6, or if w is 0 and f_3 is set, then the decoder detects a 4-error. If the count is 12 and w is even, then there are four errors. Finally, if the count is at most 11 and w is even, then 0 is not in error and there are either two or four errors.

Information concerning the composition of the four error sets can be obtained. For instance, if 0 is in error and the other three errors are in three distinct symmetry blocks, then the count is unusually low, between 12 and 16. Typically a 0 error 4-tuple has a count of at least 18. Similarly, if there is a non-zero 4-error with two errors from I_1 and two from II_1, then for a count of at least 18, the two II_1 errors and one of the I_1 are in distinct symmetry blocks. For example, 1b5b7ax has a high count for x not the 0 error. The three 1-flats 1b, 5b and 7a are in distinct symmetry blocks.

These few examples are included to illustrate the type of information which can be obtained concerning the 4-error patterns.

7.2.3 Errors of Weight Five

We briefly discuss the analysis possible when five errors have occurred. The situation becomes very complex, so only a few cases are presented. In the following we assume that at most five errors have occurred.

We begin by listing in Table 7.2.1, the value of w, the weight associated with the (10_3) inputs, for each possible distribution of the givenerrors among the symmetry blocks. In Table 7.2.1, each error 5-tuple is described by a sum of digits. Each digit represents the number of errors appearing in a single symmetry block. The digit 0 denotes an error from S_8'. For instance, 3+1+1 denotes that three of the errors are in the same symmetry block and the two remaining errors are in two distinct symmetry blocks.

non-S_8' 5-errors	w	S_8' 5-error	w
4+1	1	4+0	7
3+2	1	3+1+0	5
3+1+1	3	2+2+0	7
2+2+1	1	2+1+1+0	5
2+1+1+1	3	1+1+1+1+0	3
1+1+1+1+1	5		

Table 7.2.1: Weight w of (10_3) Inputs

From Table 7.2.1 and the discussion in the previous subsection, we know that if w=7, then a member of S_8' is in error. If w=1 and the count is at least 17, then no member of S_8' is in error and hence $e_0=0$. This follows from Table 7.2.1 and the discussion of 1 and 3-errors given earlier.

Certain information concerning the distribution of the errors is available. For instance, if all five non-S_8' errors are from five distinct symmetry blocks, then an unusually high count results. For example, the 5-error set 1c3c5c2a4a has a count of 25, while most non-S_8' 5-errors have a count of at most 15. However, if two of the errors are in the same symmetry block, then the count is as expected, less than 15. Similar problem cases arise if one of the errors is 0.

If three of the four remaining errors are in three distinct symmetry blocks and are in class II_1, then the associated count is unusually low. For example, the error 5-tuple 7a8a9a8b0 has a count of 3.

When 1, 2 or 3-errors occur, distinct numerical boundaries between the non-zero and zero error counts occurred. However, in the case of 4 and 5-errors, the boundaries are no longer distinct. It is possible to detect many 4 and 5-errors, however, only the 4 and 5-errors for which the count falls into the defined ranges of the 0_i-orbit decoder of Chapter 5, are corrected.

7.3 Order-5 (255,218) PG Code

7.3.1 Composition of Errors from Knowledge of Decoder's Output

The analysis below follows closely to that given in Section 7.2.1 for the order-3 code. Again we assume that no more than three errors have occurred. The analysis is based on the count c output by the counter of step 2, the number, w_1, of ones input to the counter associated with (41_5), the number, w_2, of ones input to the counter for (42_5), and the binary sums associated with certain 5-flats of 41_5a, 41_5b, 41_5c, 42_5a, 42_5b, 42_5c and (43_5), given in Appendix A.

We begin by discussing the single errors. If the count is 120 and w_1 and w_2 both 15, then 0 is in error and no other error has occurred. Thus, the decoding may cease after position 0 is corrected since no other digit position is in error. If the count is one and if exactly three of the sums associated with the 5-flats 43_5a_i, 43_5a_{ii}, ..., 43_5a_{xvi} are one, then a single non-zero error has occurred in one of the 1-flat classes I_1 through VI_1. If the count is 0 and if zero or one of the sums associated with 41_5a_i, 41_5b_i, 41_5c_i, 42_5a_i, 42_5b_i, 42_5c_i is one, then there is a single non-zero error from (41_1), (42_1) or (43_1). In this case, only the 14 positions corresponding to the points of these 0_1-orbits need to be decoded as the error cannot occur in any of the other 241 positions. For all three cases of a single error, once the error has been corrected, decoding can cease, as the received word is then correct. This can result in considerable savings in decoding time.

Similar information is available for the two errors. If the count is 119, then 0 and a single non-zero point from one of the classes I_1 through VI_1 are in error. If the count is 120 and w_1 and w_2 both 0, then 0 and one of the non-zero points of (41_1), (42_1) or (43_1) are in error. Once 0 and one of the positions associated with the points of (41_1), (42_1), (43_1) have been corrected, decoding may cease. At most 15 digit positions need to be corrected, rather than the standard 255. If the two errors are non-zero, the count is at most 20. If the count is 20, then the error pair is a member of the cycle of 33a35a. We note that the two errors are from the same symmetry block. Only the positions corresponding to the 1-flats of the symmetry block $S_5(0)$ need to be decoded, as the error pair is amongst these positions. If the count is 4 and w_1 and w_2 both 0, or both 12, then the error pair is from the cycle of 41a43a or 41b43a. Only the positions corresponding to these error pairs need to be decoded. If the count is 5 and w_1 and w_2 both even, then the error pair is from one of the cycles generated by 9a9b, 9a9c, 9a10a, 9a16a, 9b9c, 9b10b, 9b16b, 9c10c, 9c16c. If the count is 6 and w_1 and w_2 both 12, then the error pair is from one of the cycles of 9a42b, 17a43a, 17b43a, 17c43a, 37a43a, 37b42c, 37b43a.

The 3-errors can also be analysed and information obtained concerning the composition of certain error triples. For example, if the count is 101, and both w_1 and w_2 15, then the error triple is from the cycle of 33b33c42a. If w_1 and w_2 are 15, and the count 100, 102,..., 107, the error triples are as given in Table 6.5.2. If the count is 1 and w_1 and w_2 both 15, then the error triple is one of the members of the cycles of 33b33c41c, 33a33b33c or 37b37c41c. If the count is 0 and w_1 and w_2 both 15, then the errors are from the cycles generated by 41a42a41b, 41a41b42b, 41a42b43a, 41b42b41c or 41b42b43a. If the count is 116, then the error triples are from the cycle generated from 9a43a0, 9b43a0, 9c43a0 or 41b43a0. If the count is 114, then the error triple is from the cycle generated from 9a42b0, 37b42c0, 37b43a0, 17a43a0, 17b43a0 or 17c43a0. If the count is 108 and w_1 and w_2 are both 15, then the error triple is from the cycle of 41a41b42c. Finally, if the count is 104 and w_1 and w_2 both 15, then the error triple is one of the members of the cycle of 41a42a43a.

Many more such examples can be found for the 1, 2 and 3-errors. For these, only the positions corresponding to possible errors need to be decoded rather than all 255 positions. A table consisting of the positions possibly in error for certain counts could be added to the decoder. Given the count output on the second step of decoding, a simple look up would determine the positions to be decoded. Only those counts with an associated number of possible error positions less than a given value would be stored, in order to keep the storage requirements reasonable. For counts not in the table, decoding would continue as normal until a count was obtained which was in the table. Then, only the positions in the table associated with the count would be decoded. Once these positions were decoded, the decoding process would cease. It is obvious from the examples given, that this modification would greatly reduce the required decoding time.

7.3.2 Errors of Weight Four

An analysis of the count output on step 2 of the decoding process and the inputs w_1 and w_2 to the (41_5) and (42_5) counters, respectively, provides information concerning the 4-errors. As a result, some 4-errors can be detected. Those zero 4-errors with a count greater than 108 and the non-zero 4-errors with a count less than 100 are corrected by the decoder defined in Chapter 6. In the following we assume that no more than four errors occurred. We write $S_6'(0)$ to denote the set $S_6(0)$ augmented by the point 0.

If two or four errors occur, then the values of w_1 and w_2 are always even. This follows from an analysis of the distribution of the 2 and 4-errors among the symmetry blocks. If three errors occur, the values of w_1 and w_2 are both odd. Using these two facts and the results of the simulation of the decoder when four errors have occurred, we are able to make the following statements.

If the count is in the range 60 to 80 and w_1 and w_2 even, then e_0=0. If the count is high (at least 80), and w_1 and w_2 are not 8,6 or 6,8, and the four non-zero errors are from distinct symmetry blocks, then three of the four errors are in the same 5-flat of (41_5) or (42_5). If w_1, w_2 are 8,6 or 6,8 and the count is high, then one of the errors is from $S'_6(0)$ and the remaining three from the same flat of (41_5) or (42_5). If the count is high, then the values of w_1, w_2 are 8,6 or 12,12 or 6,6 or 8,8 or 10,6 or 10,8. If two of the errors are from the same symmetry block and two are from $S'_6(0)$, then both w_1 and w_2 are 0. If the count is less than 30, then w_1 and w_2 are either both 0, both 6 or one is 6 and one is 8.

As for the 4-errors in the order-3 (63,41) PG code, the 4-errors for this code do not have well-defined boundaries between the 0 and non-zero error sets. The comments given indicate some of the information which it is possible to obtain concerning the 4-errors.

7.3.3 Errors of Weight Five

We now discuss briefly the 5-errors and the way in which the 0_i-orbit decoder treats them. Even less information is available than was for the 4-error sets. We assume that no more than five errors have occurred.

As for the 4-error case, all zero 5-errors with a count greater than 108 or non-zero 5-errors with a count less than 100 are corrected by the 0_i-orbit decoder defined in Chapter 6. As for the 3-errors, the values of w_1 and w_2 are odd for all 5-error sets. In the following analysis, we say that the count is ordinary (vs. high or low) if for the non-zero 5-errors the value is between 20 and 40, and for the zero error sets between 40 and 60.

If the count is very high for a non-zero 5-error, or very low for a zero 5-error, then two non-distinct sets of single errors, each consisting of three errors, can be selected such that the three errors of each set appear in a distinct 5-flat of (41_5) or (42_5). If two such sets can not be formed, then the count is ordinary. If all 5-errors are from the same symmetry block, but not from $S_6'(0)$, then the count is very low and w_1 and w_2 are both 3. If the five errors are all from $S_6(0)$, then the count is very high and w_1 and w_2 are both 15. If one of the errors is 0 and the rest from $S_6(0)$, then the count is very low and both w_1 and w_2 are 15. Also, if four errors are from the same symmetry block, other than $S_6(0)$, and 0 is in error, then the count is very high and both w_1 and w_2 are 15. If the five errors occur in two distinct symmetry blocks, neither of which is $S_6(0)$, in the ratio 3 to 2, or 4 to 1, then the count is ordinary unless the two symmetry blocks appear together in Table 6.3.5.

The information available concerning the 5-errors is of interest primarily as an analysis of the distribution of the counts. The distinct boundary between the counts associated with zero and non-zero error sets that result when 1, 2 or 3 errors occur, is not present for the 5-errors. Consequently, detection of the higher weight errors, rather than correction is more feasible.

7.4 Conclusions

The results of this chapter indicate that it is possible to modify the decoder to allow for significant savings in the time required for decoding, and for the detection of some errors of higher weights. The symmetry blocks are of prime importance in the analysis presented.

The chapter began with a discussion of the information which it is possible to obtain from the decoder concerning the composition of the 1, 2 and 3-errors of the order-3 (63,41) PG code. It was suggested that significant decreases in the time required for decoding can be obtained by decoding only those codeword positions which, from knowledge of the composition of the errors, can possibly be in error. A short analysis of the distribution of the 4-errors and 5-errors in terms of the count obtained from step 2 of the decoding process was presented.

The second part of the chapter contained corresponding results for the order-5 (255,218) PG code. As the code length was much longer for this code, the decrease in decoding time was even more significant. The 4 and 5-error analysis gave an indication of the symmetries present in the distribution of these errors. As the demarcation between the counts of the zero and non-zero error sets was not well defined, the information was of use primarily for detection of errors rather than correction.

PART III
CHAPTER 8: SUMMARY OF RESULTS, CONCLUSIONS AND FURTHER RESEARCH TOPICS

8.1 Summary of Results and Conclusions

In this thesis a mathematical analysis of the two Projective Geometries, PG(5,2) and PG(7,2), led to the development of a decoding algorithm for the order-3 PG code over PG(5,2) and the order-5 PG code over PG(7,2) which required only a small fraction of the circuitry used to Majority Logic Decode the codes.

The first three chapters of the thesis were devoted to a concise presentation of the basic algebra and fundamental Coding Theory necessary to the understanding of the ideas and concepts presented in the thesis. In particular, Majority Logic Decoding and Projective Geometries were discussed. The fourth chapter introduced the work of Yamamoto et al [58] concerning the cycles of a Finite Geometry. The results from this were used to define the 0_i-orbits. An extensive analysis of the PG(5,2) 3-flats and 1-flats was presented based on the 0_i-orbits. The numerous symmetric properties of this structure provided an important mathematical interpretation of PG(5,2). After further investigation of this structure, the 0_i-orbit decoder was defined in Chapter 5. It was shown that this decoder corrected the same number of errors as the Majority Logic Decoder of the code but required only a fraction of the circuitry. While the Majority Logic Decoder consisted of 187 threshold units (57 in the simplified version) and circuitry for multiplication and division, the 0_i-orbit decoder had only 27 threshold units, 2 counters and no division or multiplication circuitry. In Chapter 6, the results of Yamamoto et al [58] were used to define the

O_i-orbits of PG(7,2). Owing to the size of the geometry, the analysis of the O_i-orbit structure was not as detailed as for PG(5,2). A decoder of the order-5 PG code over PG(7,2), based on the O_i-orbits, was defined and shown to correct the same number of errors as the Majority Logic Decoder of the code. The decrease in the circuitry required for the O_i-orbit decoder from that required for the Majority Logic Decoder, was far more significant for this case. The Majority Logic Decoder required division and multiplication circuitry and 26,479 threshold units (2,801 in the simplified version), while the O_i-orbit decoder required only 120 threshold units, three counters and a 35 entry decoding table and associated logical decision units. In Chapter 7 it was shown that significant decreases in decoding time for both decoders could be obtained as a result of a further analysis of the decoder. For certain outputs of the decoder's step 2 counter, it was possible to determine the subset of positions in which the errors occurred. Thus, it was only necessary to correct those positions, as all other digits in the received word were known to be correct and hence it was unnecessary to decode them. For both decoders, some 4 and 5-errors were correctable and many others detectable.

In this thesis we have shown that by analysing the mathematical structure of the null space of the order-3 PG code over PG(5,2) and the order-5 PG code over PG(7,2), a simplified decoder can be defined. The decoder presented requires only a fraction of the circuitry needed for MLD the code. Thus, we have significantly simplified a decoding algorithm which is already considered relatively simple. Moreover, if a small computer is associated with a communication channel, as is frequently the case, the implementation of the O_i-orbit decoder is trivial.

We have shown in this thesis that it is possible for a particular set of codes, to overcome the problem of a decoder being too complex to warrent its implementation.

8.2 Further Research Topics

We discuss in this section several topics related to the work in the thesis. First we consider the feasiblity of generalizing the O_i-orbit decoding method so that any order PG code over PG(m,2) can be decoded. To this end, we investigate the structure of the 2-flats of PG(5,2), that is the null space flats of the order-2=(m-3) PG code over PG(5,2). We recall that only order-(m-2) codes were studied in previous chapters. Secondly, we suggest further study of the structure of the null space with the goal of increasing the power of the O_i-orbit decoder. The section is concluded with several questions concerning the algebraic interpretation of the results presented in the thesis.

In the following, we present several interesting results concerning the structure of the 2-flats of PG(5,2) which suggest that a generalization of the O_i-orbit decoder is possible. To begin the analysis of the order-2 PG code we apply Yamamoto et al's [58] sixth theorem to PG(5,2) for d=2. This provides the following information:

$x_1=1$	$x_1=0$
$\theta(1) = (2^6-1)/(2^3-1) = 9$	$\theta(0) = (2^6-1)/(2-1) = 63$
$m(1) = (6/3)-1 = 1$	$m(0) = (6/1)-1 = 5$
$d(1) = (3/3)-1 = 0$	$d(0) = (3/1)-1 = 2$
$q(1) = 2^3 = 8$	$q(0) = 2$
$n^*(1) = n(1) = \phi(1,0,8) = 9$	$n(0) = \phi(5,2,2) = 1395$
$\lambda(1) = n^*(1)/\theta(1) = 1$	$n^*(0) = 1395-9 = 1386$
	$\lambda(0) = 1386/63 = 22.$

Thus we have that there are 22 i2f's of m.c. 63 and one i2f of m.c. 9. Each of the 22 i2f's of m.c. 63 generates 63 2-flats and the i2f of m.c. 9 generates 9 2-flats. The transformation Z, introduced in Chapter 4, can be applied to the 2-flats, with the result that the 2-flats can be partitioned into orbits such that each orbit corresponds to one of the i2f's and the 2-flats generated from it. The 0_2-orbits are the subsets of the orbits consisting of only those 2-flats in which the point 0 occurs. Thus, each 0_2-orbit of m.c. 63 has seven members and the 0_2-orbit of m.c. 9 has one member. We recall the one to one correspondence between the 0_3-orbits and the 0_1-orbits in Chapter 4. There, the six non-zero points of the 0_1-orbit (t_1) were repeated seven times in the 0_3-orbit (t_3). Certain points were repeated three times each and the remaining points once each. The number of times a point repeats, 7,3 or 1, corresponds to the number of points in a 2, 1 or 0-flat, respectively. We select the nine 0_2-orbits of m.c. 63, each of which repeats the six non-zero points of one and only one of the 0_1-orbits three times. These 0_2-orbits are numbered as the 0_3-orbits were to reflect the correspondence. The 2-flats of each 0_2-orbit (t_2), t=1,...,9, can be ordered so that, representing each 2-flat by the 1-flats a, b and c of (t_1) that it contains, the following description is obtained, a, b, c, ab, ac, bc. One of the seven 2-flats does not contain a 1-flat from (t_1). This 2-flat is omitted and the 0_2-orbit (t_2) is said to consist of the six 2-flats listed above. As for the 3-flats, three intersecting subsets A_t, B_t, C_t can be formed, where, for example, the set A_t consists of the three (t_2) 2-flats, in the representation above, which contain the (t_1)

1-flat a, that is a, ab, ac. The 1-flats t_1b and t_1c occur once each in A_t. The 2-flats are labelled as the 3-flats were, to show the correspondence with the 1-flats. For example, the 2-flat containing both the 1-flats t_1a and t_1b is labelled t_2ab. The 2-flat which consists of the six non-zero points of (10_1) is labelled (10_2). Corresponding to (11_1), there are five 0_2-orbits which repeat three times on the non-zero points of (11_1). As for the order-3 code, we propose that these flats be used strictly for setting flags.

We note that the number of times a point repeats in a 0_2-orbit is 3, 1 or 0, that is the number of points in a 1-flat, 0-flat; or, for continuity, a null flat. This is exactly 1 dimension less than the order-3 code.

We list in Table 8.1.1 the 2-flats of the 0_2-orbits (1_2) through (10_2), in terms of their constituent 1-flats.

	(1_2)		(2_2)		(3_2)	
I_2	a:	1a 2a 7a	a:	2a 3a 8a	a:	3a 4a 9a
	b:	1b 3a 2b	b:	2b 4a 3b	b:	3b 5a 4b
	c:	1c 2c 8a	b:	2c 3c 9a	c:	3c 4c 7b
	ab:	1a 1b 8c	ab:	2a 2b 9c	ab:	3a 3b 7c
	ac:	1a 1c 3b	ac:	2a 2c 4b	ac:	3a 3c 5b
	bc:	1b 1c 7b	bc:	2b 2c 8b	bc:	3b 3c 9b
	(4_2)		(5_2)		(6_2)	
	a:	4a 5a 7b	a:	5a 6a 8b	a:	6a 1a 9b
	b:	4b 6a 5b	b:	5b 1a 6b	b:	6b 2a 1b
	c:	4c 5c 8b	c:	5c 6c 9b	c:	6c 1c 7a
	ab:	4a 4b 8c	ab:	5a 5b 9c	ab:	6a 6b 7c
	ac:	4a 4c 6b	ac:	5a 5c 1b	ac:	6a 6c 2b
	bc:	4b 4c 7a	bc:	5b 5c 8a	bc:	6b 6c 9a
	(7_2)		(8_2)		(9_2)	
II_2	a:	7a 5a 8a	a:	8a 6a 9a	a:	9a 1a 7b
	b:	7b 2a 8b	b:	8b 3a 9b	b:	9b 4a 7a
	c:	7c 10c 8c	c:	8c 10b 9c	c:	9c 10a 7c
	ab:	7a 7b 10a	ab:	8a 8b 10c	ab:	9a 9b 10b
	ac:	7a 7c 2c	ac:	8a 8c 3c	ac:	9a 9c 4c
	bc:	7b 7c 5c	bc:	8b 8c 6c	bc:	9b 9c 1c
	(10_2)					
III_2	abc:	10a 10b 10c				

Table 8.1.1: 2-flats of (1_2) - (10_2)

We see from Table 8.1.1 that the cycles induced by the transformation g and given in Table 4.5.3 are present in the 2-flats. Moreover, if we were to form a Table of the subsets A_t, B_t, C_t, listing the 1-repeats, rather than the 3-repeats, the results would be identical to Table 4.5.1, with the title '3-repeat' replaced by '1-repeat'. All the cycle and repeat properties would simply be reduced by one dimension. For example, the II_1 1-flats appear as 3-repeats seven times and the I_1 1-flats five times each in Table 4.5.1. In the corresponding Table of O_2-orbits, the 1-flats of II_1 would appear as singletons seven times and the I_1 1-flats as singletons five times each.

Based on these observations, we propose that the decoding method used above can be adapted to the order-2 code to correct any number of errors less than 8, that is the number of errors correctable with MLD. Assuming that the order-2 code can be decoded as suggested, we propose further that this decoding method can be generalized to higher dimension PG codes. The fact that Yamamoto et al's [58] sixth theorem is applicable to any PG over GF(2) strongly supports such a generalization. As only two decoding steps and an associated decoding table, would be required, the savings over the MLD method would increase with an increase in the dimension of the code.

Before leaving the 2-flats of PG(5,2), we remark on another interesting property which may lead to an alternative decoding method for codes of order other than (m-2). We discussed above the nine O_2-orbits of m.c. 63 that correspond to the O_1-orbits $(1_1),\ldots,(9_1)$, and mentioned the five O_2-orbits which repeated three times on the O_1-orbit (11_1). The remaining eight O_2-orbits are listed in terms of 1-flats in Table 8.1.2. These 0_2-orbits are non-perfect difference sets, that is every point that appears, does so once only, except of course the point 0 which is in each 2-flat. Also, twenty of the points of PG(5,2) do not appear in each of the 0_2-orbits of Table 8.1.2. The 20 points for the 0_2-orbits in the first three columns are the non-zero points of two symmetric relative 0_1-orbits from I_1, of a 0_1-orbit from II_1 and of (11_1). The remaining two 0_2-orbits in column four have the non-zero points of (7_1), (8_1), (9_1) and (11_1) as omitted points. In Table 8.1.2 the 0_2-orbits are given in four columns, where the two 0_2-orbits in each column have the same set of omitted points. The omitted points, listed as 0_1-orbits, are given at the top of each column.

$(3_1),(6_1),$ $(7_1),(11_1)$	$(1_1),(4_1),$ $(8_1),(11_1)$	$(2_1),(5_1),$ $(9_1),(11_1)$	$(7_1),(8_1),$ $(9_1),(11_1)$
1a 8a 10b	2a 9a 10a	3a 7b 10c	2a 3b 6c
1b 2c 9b	2b 3c 7a	3b 4c 8a	4a 5b 2c
1c 4a 5c	2c 5a 6c	3c 6a 1c	6a 1b 4c
4b 8b 9c	5b 9b 7c	6b 7a 8c	1c 2b 10c
5a 9a 10c	6a 7b 10b	1a 8b 10a	3c 4b 10a
2a 5b 8c	3a 6b 9c	4a 1b 7c	5c 6b 10b
2b 4c 10a	3b 5c 10c	4b 6c 10b	1a 3a 5a
4a 8b 10b	5a 9b 10a	6a 7a 10c	3a 4b 1c
4b 5c 9a	5b 6c 7b	6b 1c 8b	5a 6b 3c
4c 1a 2c	5c 2a 3c	6c 3a 4c	1a 2b 5c
1b 8a 9c	2b 9a 7c	3b 7b 8c	2c 3b 10b
2a 9b 10c	3a 7a 10b	4a 8a 10a	4c 5b 10c
5a 2b 8c	6a 3b 9c	1a 4b 7c	6c 1b 10a
5b 1c 10a	6b 2c 10c	1b 3c 10b	2a 4a 6a

Table 8.1.2: 2-flats of m.c. 63 in terms of 1-flats

For the 0_2-orbits of the first three columns, if V_i is any 2-flat in one of the 0_2-orbits of a given column, then $g^3(V_i)$ is the corresponding 2-flat in the other 0_2-orbit of the column. In the fourth column, if V_i is any 2-flat in one of the 0_2-orbits, then $g(V_i)$ is the corresponding 2-flat in the other

O_2-orbit. Each of the O_2-orbits in column 4 can be divided into the three subsets indicated in Table 8.1.2.

The structure of these eight O_2-orbits assures that within a given column, no two 1-flats that appear in the same 2-flat in one of the O_2-orbits, appear together in the second O_2-orbit. It is interesting to note that, within the eight O_2-orbits, the II_1 1-flats appear less frequently than the I_1 1-flats, four times vs. six times each. We recall that the II_1 1-flats appear more frequently in the O_2-orbits (1_2), ..., (9_2). The following four facts concerning the O_2-orbits of Table 8.1.2 may prove useful in designing a decoder based on these O_2-orbits:

i) (11_1) does not appear in any of the O_2-orbits,

ii) three other O_1-orbits are missing in each O_2-orbit; in particular, the points of a symmetric relative pair of O_1-orbits are missing for each of the O_2-orbits in the first three columns,

iii) the two O_2-orbits in a given column are such that any two 1-flats which appear together in one of the O_2-orbits do not appear together in the other,

iv) no point, other than 0, is repeated more than once in any of the O_2-orbits, that is the 2-flats are orthogonal on 0.

We now present several other topics which require investigation. A further study of the structure of PG(m,2) is necessary in order to determine if the null space flats can give enough additional information to the decoder for errors of weight greater than $\left[(2^{m-r+1}-1)/2\right]$ to be corrected in the order-r PG code over PG(m,2). We recall that certain of the O_r-orbit r-flats were omitted in defining the O_i-orbit decoders. The use of some of these flats as checks may provide enough information to the decoder for errors of higher weight to be corrected.

The presentation of the codes that we have given does not refer to the algebraic interpretation of the PG codes. To relate the two descriptions could be quite complex, but would prove interesting. If such a study were done, the following points should be considered:

i) How are the 0_i-orbits related algebraically?

ii) Is there a partitioning of the roots of the parity check polynomial induced by the 0_i-orbit structure.

iii) Does a knowledge of the algebraic representation assist in obtaining the roots necessary to generate the individual classes $I_1, II_1 \ldots.$?

iv) How are the polynomials corresponding to the 1-flats in the 0_1-orbit (t_1) related to those of the r-flats in the 0_r-orbit (t_r)?

v) The distinct iif's of a class are obtained by successively multiplying the point representation of a given iif by 2 until all iif's are obtained. What is the comparable algebraic operation?

vi) Within a given 0_1-orbit, there are 6 non-zero points. These can be considered as 3 non-zero points and their mod $(2^{m+1}-1)$ inverses. Is there an algebraic interpretation of this?

vii) What is the algebraic explanation of the many symmetries present in the 0_i-orbit structures?

It is hoped that the investigation of the questions presented in this section will provide a generalized 0_i-orbit decoder for the order-r PG code over PG(m,2) requiring only a fraction of the circuitry needed to Majority Logic Decode the code.

A.1 O_5-orbit (41_5) 5-flats

```
1c5c 9b11c12a13b15c16a 17b18a18c20a21b22a22c24a 26a26c28a30a30c32a 39b39c 41a41b41c42a42b42c43a
3c7c 11b13c14a15b9c10a 19b20a20c22a23b24a24c18a 28a28c30a32a32c26a 37c37b 41a41b41c42a42b42c43a

2c6c 10b10c11a14b14c15a 18b19c22b23c 25c29c 33a34a35a36a 37a38a38b38c39a40a 41a41b41c42a42b42c43a
4c8c 12b12c13a16b16c9a  20b21c24b17c 27c31c 35a36a33a34a 39a40a40b40c38a37a 41a41b41c42a42b42c43a

1a2b5a6b 10c11a11c12a14c15a15c16a 25c26b26c29c30b30c 35b35c 38b38c39b39c 41a41b41c42a42b42c43a
3a4b7a8b 12c13a13c14a16c9a9c10a   27c28b28c31c32b32c 33c33b 40b40c37c37b 41a41b41c42a42b42c43a

2c4c6c8c 10b12b14b16b 17c18a18b19c20a20b21c22a22b23c24a24b 26a28a30a32a 41a41b41c42a42b42c43a

1c2a3b5c6a7b 9b10c11a13b14c15a 17b18c21b22c 25c27b29c31b 36b36c 38b38c 41a41b41c42a42b42c43a
3c4a5b7c8a1b 11b12c13a15b16c9a 19b20c23b24c 27c29b31c25b 34c34b 40b40c 41a41b41c42a42b42c43a

1a1b2b4a5a5b6b8a 17a19a21a23a 25a25b26b27a29a29b30b31a 34b34c35b35c 41a41b41c42a42b42c43a
3a3b4b6a7a7b8b2a 19a21a23a17a 27a27b28b29a31a31b32b25a 36b36c33c33b 41a41b41c42a42b42c43a

1b2a3b4a5b6a7b8a 25b27b29b31b 33a34a34b34c35a36a36b36c 37a38a39a40a 41a41b41c42a42b42c43a

1a2b2c3c5a6b6c7c 10b11a14b15b 18b19b19c20c22b23b23c24c 26b30b 35b35c 41a41b41c42a42b42c43a
3a4b4c5c7a8b8c1c 12b13b16b9b  20b21b21c22c24b17b17c18c 28b32b 33c33b 41a41b41c42a42b42c43a

9c10a11c12a13c14a15c16a 17a19a21a23a25a26c27a28c29a30c31a32c 37b37c39b39c 41a41b41c42a42b42c43a
```

A.2 O_5-orbit (42_5) 5-flats

```
2c6c 10b12c13a14b16c9a  18b19a19c21a22b23a23c17a 27a27c29a31a31c25a 40b40c 41a41b41c42a42b42c43a
4c8c 12b14c15a16b10c11a 20b21a21c23a24b17a17c19a 29a29c31a25a25c27a 38c38b 41a41b41c42a42b42c43a

3c7c 11b11c12a15b15c16a 19b20c23b24c 26c30c 34a35a36a33a 38a39a39b39c40a37a 41a41b41c42a42b42c43a
5c1c 13b13c14a9b9c10a   21b22c17b18c 28c32c 36a33a34a35a 40a37a37b37c38a39a 41a41b41c42a42b42c43a

2a3b6a7b 11c12a12c13a15c16a16c9a 26c27b27c30c31b31c 36b36c 39b39c40b40c 41a41b41c42a42b42c43a
4a5b8a1b 13c14a14c15a9c10a10c11a 28c29b29c32c25b25c 34c34b 37c37b38c38b 41a41b41c42a42b42c43a

3c5c7c1c 11b13b15b9b 18c19a19b20c21a21b22c23a23b24c17a17b 27a29a31a25a 41a41b41c42a42b42c43a

2c3a4b6c7a8b 10b11c12a14b15c16a 18b19c22b23c 26c28b30c32b 33c33b 39b39c 41a41b41c42a42b42c43a
4c5a6b8c1a2b 12b13c14a16b9c10a  20b21c24b17c 28c30b32c26b 35c35b 37c37b 41a41b41c42a42b42c43a

2a2b3b5a6a6b7b1a 18a20a22a24a 26a26b27b28a30a30b31b32a 35b35c36b36c 41a41b41c42a42b42c43a
4a4b5b7a8a8b1b3a 20a22a24a18a 28a28b29b30a32a32b25b26a 33c33b34c34b 41a41b41c42a42b42c43a

2b3a4b5a6b7a8b1a 26b28b30b32b 35a35b35c36a33a33c33b34a 38a39a40a37a 41a41b41c42a42b42c43a

2a3b3c4c6a7b7c8c 11b12b15b16b 19b20b20c21c23b24b24c17c 27b31b36b36c 41a41b41c42a42b42c43a
4a5b5c6c8a1b1c2c 13b14b9b10b  21b22b22c23c17b18b18c19c 29b25b34c34b 41a41b41c42a42b42c43a

10c11a12c13a14c15a16c9a 18a20a22a24a 26a27c28a29c30a31c32a25c 38b38c40b40c 41a41b41c42a42b42c43a
```

A.3 41_5a, 41_5b, 41_5c 5-flats

41_5a

```
   i: 1a1b1c4b5b 9b9c10a10c12b14b16c  18b19a19b20a24b24c 25c26a26b29a30c31c 33a33b35a36b36c 39c 41a
  ii: 3a3b3c6b7b 11b11c12a12c14b16b10c 20b21a21b22a18b18c 27c28a28b31a32c25c 35a35b33a34b34c 37b 41a
 iii: 5a5b5c8b1b 13b13c14a14c16b10b12c 22b23a23b24a20b20c 29c30a30b25a26c27c 33a33c35a36b36c 39b 41a
  iv: 7a7b7c2b3b 15b15c16a16c10b12b14c 24b17a17b18a22b22c 31c32a32b27a28c29c 35a35c33a34c34b 37c 41a

   v: 2a2b3a6a7c 9c10a10b13a15a15b16b  18c19a20b21b22b24a 25b28b29a29b30a30c 35c 37a38c39a39c40b 41a
  vi: 4a4b5a8a1c 11c12a12b15a9a9b10b   20c21a22b23b24b18a 27b30b31a31b32a32c 33b 39a40c37a37b38c 41a
 vii: 6a6b7a2a3c 13c14a14b9a11a11b12b  22c23a24b17b18b20a 29b32b25a25b26a26c 35b 37a38b39a39b40c 41a
viii: 8a8b1a4a5c 15c16a16b11a13a13b14b 24c17a18b19b20b22a 31b26b27a27b28a28c 33c 39a40b37a37c38b 41a

  ix: 1c2b3c4c6c8b 9b11b12c13c14a15a 17a20a21c23c 26a27a27b27c30c31b 33c34a34b34c35c 38a38c39c 41a
   x: 3c4b5c6c8c2b 11b13b14c15c16a9a 19a22a23c17c 28a29a29b29c32c25b 35c36a36b36c33b 40a40c37b 41a
  xi: 5c6b7c8c2c4b 13b15b16c9c10a11a 21a24a17c19c 30a31a31b31c26c27b 33b34a34c34b35b 38a38b39b 41a
 xii: 7c8b1c2c4c6b 15b9b10c11c12a13a 23a18a19c21c 32a25a25b25c28c29b 35b36a36c36b33c 40a40b37c 41a

xiii: 1a1b2a2c3a4c5b6a 11a12c15c16a 17b18a19a19c20c21c22c23b 26b27c28b29a32a32c 34a37b38a38b 41a
 xiv: 3a3b4a4c5a6c7b8a 13a14c9c10a  19b20a21a21c22c23c24c17b 28b29c30b31a26a26c 36a 39b40a40b 41a
  xv: 5a5b6a6c7a8c1b2a 15a16c11c12a 21b22a23a23c24c17c18c19b 30b31c32b25a28a28c 34a 37c38a38c 41a
 xvi: 7a7b8a8c1a2c3b4a 9a10c13c14a  23b24a17a17c18c19c20c21b 32b25c26b27a30a30c 36a 39c40a40c 41a
```

41_5b

```
  i: 3c4b7a 9a10b10c12a14b14c16b  17b18c21c22a23a24a24c 26b27b27c28b29b31a32c 34a34b35a35b 37c39c 41b
 ii: 7c8b3b 13a14b14c16a10b10c12b 21b22c17c18a19a20a20c 30b31b31c32b25b27a28c 34a34c35a35c 37b39b 41b

iii: 1c4b6a8c 9a11b13b15c 19a19b19c23c24b 25b25c26c27a27c28a28b29c30a30b32c 34c35c36a36c 37c39a 41b
 iv: 5c8b2a4c 13a15b9b11c 23a23b23c19c20b 29b29c30c31a31c32a32b25c26a26b28c 34b35b36a36b 37b39a 41b

  v: 1b3a4c7b8a 10a10c11b13c14c16a16c 17b18c19b19c20b21a23c 25a26a27b30b32a 33b35c 37a39b40a40b 41b
 vi: 5b7a8c3b4a 14a14c15b9c10c12a12c  21b22c23b23c24b17a19c 29a30a31b26b28a 33c35b 37a29c40a40c 41b
```

```
  vii: 1a1c2c3a6b6c7b 9c11a12b13a13b14a15a15b15c 17c21a23b 25a26c27b28a29b30a31c 33b34a34b35a 41b
 viii: 5a5c6c7a2b2c3b 13c15a16b9a9b10a11a11b11c  21c17a19b 29a30c31b32a25b26a27c 33c34a34c35a 41b

   ix: 1a2c4a4b5b6a6b6c 12c15b16a16b 17b18c19a21c23b 26a27a28b28c32a 33a36c 37b38a38b38c39b40c 41b
    x: 5a6c8a8b1b2a2b2c 16c11b12a12b 21b22c23a17c19b 30a31a32b32c28a 33a36b 37c38a38c38b39c40b 41b

   xi: 1a3b3c4a4c5b5c6b8b 9b11a15a15c16c  18b19a20b22a22b24a24c 26c27a31b32b32c 37a37c40a40b 41b
  xii: 5a7b7c8a8c1b1c2b4b 13b15a11a11c12c 22b23a24b18a18b20a20c 30c31a27b28b28c 37a37b40a40c 41b

 xiii: 1b2a5c7a7c8a 9b9c10b12c14a14b15c16b 20c21a21c22a24a25a25c26c29c30b 33a33c35c36b38a40c 41b
  xiv: 5b6a1c3a3c4a 13b13c14b16c10a10b11c12b 24c17a17c18a20a 29a29c30c25c26b 33a33b35b36c38a40b 41b

   xv: 2b3a4c5a6a7c 9c12a13a14a 17a17b18b18c20b20c22a22b24a 25b29a31c 33b34c36a36c 38b38c39a39c 41b
  xvi: 6b7a8c1a2a3c 13c16a9a10a 21a21b22b22c24b24c18a18b20a 29b25a27c 33c34b36a36b 38c38b39a39b 41b
```

$\underline{41}_{5}\underline{c}$

```
    i: 1c2b5b 10a12b12c14b15a16b16c 19c20a21a22a22c23b24c 25b25c26b27b29a30c32b 33a33b36a36c 37c39b 41c
   ii: 5c6b1b 14a16b16c10b11a12b12c 23c24a17a18a18c19b20c 29b29c30b31b25a26c28b 33a33c36a36b 37b39c 41c

  iii: 2b4a6c7c 9b11b13c15a 17a17b17c21c22b 25a25c26a26b27c28a28b30c31b31c32c 33c34a34c36b 37a39b 41c
   iv: 6b8a2c3c 13b15b9c11a 21a21b21c17c18b 29a29c30a30b31c32a32b26c27b27c28c 33b34a34b36c 37a39c 41c

    v: 1a2c5b6a7b 9b11c12c14a14c16a16c  17b17c18b19a21c23b24c 25b28b30a31a32a 33c35c37b38a38b39a 41c
   vi: 5a6c1b2a3b 13b15c16c10a10c12a12c 21b21c22b23a17c19b20c 29b32b26a27a28a 33b35b 37c38a38c39a 41c

  vii: 1a1c2a3b4a7c 9c10b11b11c12b14c16a16b  18a22c23a23c24a 27a27c28c31c32b 33b34c35a35c 38b40a 41c
 viii: 5a5c6a7b8a3c 13c14b15b15c16b10c12a12b 22a18c19a19c20a 31a31c32c27c28b 33c34b35a35b 38c40a 41c

   ix: 1a2c3a4a5c8b 10a11a12a15c 18b18c20a20b22a23a23b24b24c 27a29c31b 34a34c35c36b 37a37c40b40c 41c
    x: 5a6c7a8a1c4b 14a15a16a11c 22b22c24a24b18a19a19b20b20c 31a25c27b 34a34b35b36c 37a37b40c40b 41c

   xi: 1a4b4c5b7a7c8c 9a10b11a11b12a13a13b13c15c 19a21b23c 25b26a27b28a29c31a32c 33a35c36a36c 41c
  xii: 5a8b8c1b3a3c4c 13a14b15a15b16a9a9b9c11c   23a17b19c 29b30a31b32a25c27a28c 33a35b36a36b 41c
```

```
xiii: 2a2b3b4a4b4c7a8c 10c13b14a14b 17a19c21b23b24c 25a26b26c30a32a 34c35a37b38c39c40a40b40c 41c
 xiv: 6a6b7b8a8b8c3a4c 14c9b10a10b  21a23c17b19b20c 29a30b30c26a28a 34b35a37c38b39b40a40c40b 41c

  xv: 1b1c2a2c3b3c4b6b7a 9a13a13c14c15b 17a18b20a20b22a22c24b 25a27b30b30c32c 38a38b39a39b 41c
 xvi: 5b5c6a6c7b7c8b2b3a 13a9a9c10c11b  21a22b24a24b18a18c20b 29a25b26b26c28c 38a38c39a39c 41c
```

A.4 $\underline{42_5a,\ 42_5b,\ 42_5c\ \ 5\text{-flats}}$

$\underline{42_5a}$

```
    i: 3b4b8a8b8c 9a9c11b13b15c16b16c 17b18a18b19a23b23c 25a25b28a29c30c32c 34a35b35c36a36c 38c 42a
   ii: 5b6b2a2b2c 11a11c13b15b9c10b10c 19b20a20b21a17b17c 27a27b30a31c32c26c 36a33c33b34a34b 40c 42a
  iii: 7b8b4a4b4c 13a13c15b9b11c12b12c 21b22a22b23a19b19c 29a29b32a25c26c28c 34a35c35b36a36b 38b 42a
   iv: 1b2b6a6b6c 15a15c9b11b13c14b14c 23b24a24b17a21b21c 31a31b26a27c28c30c 36a33b33c34a34c 40b 42a

    v: 1a1b2a5a6c 9a9b12a14a14b15b16c   17c18a19b20b21b23a 27b28a28b29a29c32b 34c 37c38a38c39b40a 42a
   vi: 3a3b4a7a8c 11a11b14a16a16b9b10c   19c20a21b22b23b17a 29b30a30b31a31c26b 36c 39c40a40c37c38a 42a
  vii: 5a5b6a1a2c 13a13b16a10a10b11b12c 21c22a23b24b17b19a 31b32a32b25a25c28b 34b 37b38a38b39c40a 42a
 viii: 7a7b8a3a4c 15a15b10a12a12b13b14c 23c24a17b18b19b21a 25b26a26b27a27c30b 36b 39b40a40b37b38a 42a

   ix: 1b2c3c5c7b8c 10b11c12c13a14a16b 19a20c22c24a 25a26a26b26c29c30b 33a33b33c34c36b 37a37c38c 42a
    x: 3b4c5c7c1b2c 12b13c14c15a16a10b 21a22c24c18a 27a28a28b28c31c32b 35a35b35c36c34c 39a39c40c 42a
   xi: 5b6c7c1c3b4c 14b15c16c9a10a12b  23a24c18c20a 29a30a30b30c25c26b 33a33c33b34b36c 37a37b38b 42a
  xii: 7b8c1c3c5b6c 16b9c10c11a12a14b  17a18c20c22a 31a32a32b32c27c28b 35a35c35b36b34b 39a39b40b 42a

 xiii: 1c2b3a6a6b7a7c8a 9c12c13a16a  17c18c19c20b22b23a24a24c 25b26a29a29c31b32c 35a 38c39a39c 42a
  xiv: 3c4b5a8a8b1a1c2a 11c14c15a10a 19c20c21c22b24b17a18a18c 27b28a31a31c25b26c 33a 40c37a37b 42a
   xv: 5c6b7a2a2b3a3c4a 13c16c9a12a  21c22c23c24b18b19a20a20c 29b30a25a25c27b28c 35a 38b39a39b 42a
  xvi: 7c8b1a4a4b5a5c6a 15c10c11a14a 23c24c17c18b20b21a22a22c 31b32a27a27c29b30c 33a 40b37a37c 42a
```

$\underline{42_5 b}$

```
    i: 1b4b8c 9a11b11c13o14a15b15c 18c19a20a21a21c22b23c 25b26b28a29c31b32b32c 35a35c36a36c 38b40b 42b
   ii: 5b8b4c 13a15b15c9b10a11b11c 22c23a24a17a17c18b19c 29b30b32a25c27b28b28c 35a35b36a36b 38c40c 42b

  iii: 1b3a5c6c 10b12c14a16b 20c21b24a24b24c 25a25b26c27a27b29c30b30c31c32a32c 33a33c35b36b 38b40b 42b
   iv: 5b7a1c2c 14b16c10a12b 24c17b20a20b20c 29a29b30c31a31b25c26b26c27c28a28c 33a33b35c36c 38c40a42b

    v: 1a2b4a5c8b 9a9c11a11c12b14c15c  18b19c20b20c21b22a24c 25a26a27a28b31b 34b36c 37a37c38a40b 42b
   vi: 5a6b8a1c4b 13a13c15a15c16b10c11c 22b23c24b24c17b18a20c 29a30a31a32b27b 34b36b 37a37b38a40c 42b

  vii: 1c2a3a4c7b8a 9a10a11a14c  17b17c19a19b21a22a22b23b23c 26a28c30b 33a33c34c35b 39b39c40a40b 42b
 viii: 5c6a7a8c3b4a 13a14a15a10c 21b21c23a23b17a18a18b19b19c 30a32c26b 33a33b34b35c 39c39b40a40c 42b

   ix: 1a2b3a6c8a8c 9b10b10c11b13c15a15b16c 17a21c22a22c23a 26a26c27c30c31b 33c34a34c36c 37b39a 42b
    x: 5a6b7a2c4a4c 13b14b14c15b9c11a11b12c 21a17c18a18c19a 30a30c31c26c27b 33b34a34b36b 37c39a 42b

   xi: 2a2c3c4a7b7c8b 10c12a13b14a14b15a16a16b16c 18c22a24b 26a27c28b29a30b31a32c 34b35a35b36a 42b
  xii: 6a6c7c8a3b3c4b 14c16a9b10a10b11a12a12b12c 22c18a20b 30a31c32b25a26b27a28c 34c35a35c36a 42b

 xiii: 1a1b2b3a3b3c6a7c 9c12b13a13b 18c20b22b23c24a 25b25c29a31a32a 33c34a 37a38c39a39b39c40c 42b
  xiv: 5a5b6b7a7b7c2a3c 13c16b9a9b  22c24b18b19c20a 29b29c25a27a28a 33b34a 37b38b39a39c39b40b 42b

   xv: 1a1c2b2c3b5b6a8b8c 12a12c13c14b16a 17b19a19b21a21c 23b24a 28b29b29c31c32a 37a37b38a38b 42b
  xvi: 5a5c6b6c7b1b2a4b4c 16a16c9c10b12a  21b23a23b17a17c19b20a  32b25b25c27c28a 37a37c38a38c 42b
```

$\underline{42_5 c}$

```
    i: 2b6c7b 9b9c11b12a13b13c15a 17a18a19a19c20b21c24c 26a27c29b30b30c31b32b 33a33c34a34c 38b40c 42c
   ii: 6b2c3b 13b13c15b16a9b9c11a 21a22a23a23c24b17c20c 30a31c25b26b26c27b28b 33a33b34a34b 38c40b 42c

  iii: 1a3c4c7b 10c12a14b16b 18c19b22a22b22c 25a25b27c28b28c29c30a30c31a31b32c 33b34b35a35b 38a40c 42c
   iv: 5a7c8c3b 14c16a10b12b 22c23b18a18b18c 29a29b31c32b32c25c26a26c27a27b28c 33c34c35a35c 38a40b 42c
```

```
     v: 2a3c6b7a8b 9a9c10b12c13c15a15c  17c18b18c19b20a22c24b 25a26b29b31a32a 34c36c 38b39a39b40a 42c
    vi: 6a7c2b3a4b 13a13c14b16c9c11a11c 21c22b22c23b24a18c20b 29a30b25b27a28a 34b36b 38c39a39c40a 42c

   vii: 1a4c6a6c7a8b 9b11c13a13b14c15b16b16c 19c20a20c21a23a 25c28c29b32a32c 34c35b36a36b 37a39c 42c
  viii: 5a8c2a2c3a4b 13b15c9a9b10c11b12b12c  23c24a24c17a19a 29c32c25b28a28c 34b35c36a36c 37a39b 42c

    ix: 1a2c5b6a7c8a 9a12c15a16a  17a17b19a20a20b21b21c23b23c 26c28b32a 33b35a35b36b 37b37c38a38b 42c
     x: 5a6c1b2a3c4a 13a16c11a12a 21a21b23a24a24b17b17c19b19c 30c32b28a 33c35a35c36c 37c37b38a38c 42c

    xi: 1b1c2b4a4c5c6a 9a10a10b10c12c14a15b16a16b  18b20c24a 25a26c28a29c30b31a32b 33a33c34a36b 42c
   xii: 5b5c6b8a8c1c2a 13a14a14b14c16c10a11b12a12b 22b24c20a 29a30c32a25c26b27a28b 33a33b34a36c 42c

  xiii: 1a1b1c4a5c7a7b8b 10b11a11b15c 18b20b21c22a24c 27a29a30a31b31c 35b36a 37a37b37c38c39b40b 42c
   xiv: 5a5b5c8a1c3a3b4b 14b15a15b11c 22b24b17c18a20c 31a25a26a27b27c 35c36a 37a37c37b38b39c40b 42c

    xv: 1b3b4a6b6c7a7c8b8c 10a10c11c12b14a 17a17b19a19c21b22a23b 26b27b27c29c30a 39a39c40a40c 42c
   xvi: 5b7b8a2b2c3a3c4b4c 14a14c15c16b10a 21a21b23a23c17b18a19b 30b31b31c25c26a 39a39b40a40b 42c
```

A.5 0_5-orbit (43_5) 5-flats

```
     i:1a2a4a5c6c8b8c  11a12c13c15b15c 17b18a18b20b22a24c 25b25c27a30b31a31b 33b35a 37a37c39c40b 43a
    ii: 2a3a5a6c7c1b1c 12a13c14c16b16c 18b19a19b21b23a17c 26b26c28a31b32a32b 34b36a 38a38c40c37c 43a
   iii: 3a4a6a7c8c2b2c 13a14c15c9b9c    19b20a20b22b24a18c 27b27c29a32b25a25b 35b33a 39a39c37b38c 43a
    iv: 4a5a7a8c1c3b3c 14a15c16c10b10c 20b21a21b23b17a19c 28b28c30a25b26a26b 36b34a 40a40c38b39c 43a
     v: 5a6a8a1c2c4b4c 15a16c9c11b11c  21b22a22b24b18a20c 29b29c31a26b27a27b 33c35a 37a37b39b40c 43a
    vi: 6a7a1a2c3c5b5c 16a9c10c12b12c  22b23a23b17b19a21c 30b30c32a27b28a28b 34c36a 38a38b40b37b 43a
   vii: 7a8a2a3c4c6b6c 9a10c11c13b13c  23b24a24b18b20a22c 31b31c25a28b29a29b 35c33a 39a39b37c38b 43a
  viii: 8a1a3a4c5c7b7c 10a11c12c14b14c 24b17a17b19b21a23c 32b32c26a29b30a30b 36c34a 40a40b38c39b 43a

    ix: 1a1b8b4b5c 10b11b12a12c14a15a16b 17b17c19c20a20c24a 25a26c28c29a29c30b 33a33c34b36b 39a40b 43a
     x: 2a2b1b5b6c 11b12b13a13c15a16a9b  18b18c20c21a21c17a 26a27c29c30a30c31b 34a34c35b33c 40a37c 43a
    xi: 3a3b2b6b7c 12b13b14a14c16a9a10b  19b19c21c22a22c18a 27a28c30c31a31c32b 35a35c36b34c 37a38c 43a
   xii: 4a4b3b7b8c 13b14b15a15c9a10a11b  20b20c22c23a23c19a 28a29c31c32a32c25b 36a36c33c35c 38a39c 43a
  xiii: 5a5b4b8b1c 14b15b16a16c10a11a12b 21b21c23c24a24c20a 29a30c32c25a25c26b 33a33b34c36c 39a40c 43a
   xiv: 6a6b5b1b2c 15b16b9a9c11a12a13b   22b22c24c17a17c21a 30a31c25c26a26c27b 34a34b35c33b 40a37b 43a
    xv: 7a7b6b2b3c 16b9b10a10c12a13a14b  23b23c17c18a18c22a 31a32c26c27a27c28b 35a35b36c34b 37a38b 43a
   xvi: 8a8b7b3b4c 9b10b11a11c13a14a15b  24b24c18c19a19c23a 32a25c27c28a28c29b 36a36b33b35b 38a39b 43a

  xvii: 1a1c2b5a5c6b 9a9b12c13a13b16c    17b18c21b22c 26b27c30b31c 35b35c 40b40c 41a41b41c42a42b42c43a
 xviii: 2a2c3b6a6c7b 10a10b13c14a14b9c   18b19c22b23c 27b28c31b32c 36b36c 37b37c 41a41b41c42a42b42c43a
   xix: 3a3c4b7a7c8b 11a11b14c15a15b10c 19b20c23b24c 28b29c32b25c 33c33b 38b38c 41a41b41c42a42b42c43a
    xx: 4a4c5b8a8c1b 12a12b15c16a16b11c 20b21c24b17c 29b30c25b26c 34c34b 39b39c 41a41b41c42a42b42c43a

   xxi: 17a18a19a20a21a22a23a24a 25a26a27a28a29a30a31a32a 33a34a35a36a37a38a39a40a 41a41b41c42a42b42c43a
```

REFERENCES

1. Berlekamp, E. 1968. Algebraic Coding Theory. Toronto: McGraw-Hill Book C o.

2. Berman, G. 1952. "Finite Projective Geometries". Can. J. Math., 4, p. 302-313.

3. Birkhoff, G., and MacLane, S. 1965. A Survey of Modern Algebra, third edition. New York: The Macmillan Co.

4. Blake, I., and Mullin, R. 1975. The Mathematical Theory of Coding. London: Academic Press.

5. Blumenthal, L. 1961. A Modern View of Geometry. San Francisco: W. H. Freeman.

6. Chen, C. L. 1971. "On MLD of Finite Geometry Codes". IEEE Trans., IT-17, p. 332-336.

7. ______. 1972. "A Note on MLD of Finite Geometry Codes". IEEE Trans., IT-18, p.539-541.

8. ______. 1972. "On Shortened Finite Geometry Codes". Inf. and Control, 20, p.216-221.

9. Chen, C. L., Peterson, W. W. and Weldon, E. J. 1969. "Some Results On Quasi-Cyclic Codes". Inf. and Control 15, p.407-423.

10. Chen, C. L., and Warren, M. 1973. "Note on 1-step Majority Logic Decodable Codes". IEEE Trans., IT-19, p.135-137.

11. Chien, R. T., Hong, M., Preparata, F. P. 1971. "Results in the Theory of Arithmetic Codes". Inf. and Control, 19, p. 246-264.

12. Delsarte, P. 1969. "Geometric Approach to a Class of Cyclic Codes". J. Comb. Theory, 6, p.340-358.

13. Duc, N. G. 1971. "Pseudostep Orthogonalization: A New Threshold-Decoding Algorithm". IEEE Trans., IT-17, p.766-767.

14. Forney, G. D. 1966. "Generalized Minimum Distance Decoding". IEEE Trans., IT-12, p.125-131.

15. Gallager, R. 1963. "Low Density Parity-Check Codes". IEEE Trans., IT-8, p.21-28.

16. Gallager, R. 1968. Information Theory and Reliable Communication. Toronto: John Wiley and Sons, Inc.

17. Goethals, J. M. and Delsarte, P. 1968. "On a Class of MLD Cyclic Codes". IEEE Trans., IT-14, p.182-188.

18. Goethals, J. M., Delsarte, P., MacWilliams, F. J. 1970. "On Generalized Reed-Muller Codes and Their Relatives". Inf. and Control, 16, p.403-442.

19. Gore, W. E. 1969. "Generalized Threshold Decoding of Linear Codes". IEEE Trans., IT-15, p. 590-592.

20. ______. 1969. "The Equivalence of L-step Orthogonalization and Reed Decoding Procedure". IEEE Trans., IT-15, p.184-186.

21. Hall, M. 1943. "Projective Planes". Trans. Am. Math. Soc., 54, p.229-277.

22. ______. 1947. "Cyclic Projective Planes". Duke Math. J., 14, p. 1079-1090.

23. ______. 1967. Combinatorial Theory. London: Blaisdell Publishing Co.

24. Hartman, C. R. P.,Ducey, J. B. , Rudolph, L. D. 1974. "On the Structure of Finite Geometry Codes". IEEE Trans., IT-20, p.240-252.

25. Kasami, T., and Lin, S. 1971. "Decoding for the Duals of Primitive Polynomial Codes". IEEE Trans., IT-17, p.322-330.

26. Kasami, T., Lin, S., Peterson, W. 1968. "New Generalizations of the Reed-Muller Codes, Part I: Primitive Codes". IEEE Trans., IT-14, p.189-199.

27. ______. 1968. "Polynomial Codes". IEEE Trans., IT-14, no. 6, p.307-314.

28. Kasami, T., Takura, N., Azumi, S. 1976. "On the Weight Enumeration of Weights Less Than 2.5d of Reed-Muller Codes". Inf. and Control, 30, p.380-395.

29. Lin, S. 1973. "Multifold Euclidean Geometry Codes". IEEE Trans., IT-19, p.537-548.

30. MacWilliams, F. J. 1965. "Binary Cyclic Alphabets". Bell Systems Tech. J., 44, p.303-332.

31. Mandelbaum, D. 1967. "Arithmetic Codes With Large Distances". IEEE Trans., IT-13, p.237-242.

32. Mann, H. B., editor. 1968. Error-Correcting Codes. Proceedings of Symposium Organized by the Mathematics Research Center, U.S. Army, U. of Wisconsin, New York: Wiley.

33. Massey, J. L. 1963. Threshold Decoding. M.I.T. Press Research Monograph 20. Cambridge, Mass.: M.I.T. Press.

34. ______. 1968. "Advances In Threshold Decoding". Advances in Communications Systems. A. V. Balakrishnan, Ed., New York: Academic Press, Inc.

35. Muller, D. E. 1954. "Application of Boolean Algebra to Switching Circuit Design and Error Detection". IRE Trans., EC-3, p.6-12.

36. Ng, M. 1970. "On Rudolph's MLD Algorithm". IEEE Trans., IT-16, p.651-652.

37. Peterson, W., and Weldon, E. 1972. Error-Correcting Codes, second edition. Cambridge, Mass.: The M.I.T.Press.

38. Rahman, M., and Blake, I. 1975. "MLD Using Combinatorial Designs". IEEE Trans., IT-21, p.585-587.

39. Rao, C. R. 1944. "Finite Geometries and Certain Derived Results in the Theory of Numbers". Proc. of Nat. Inst. of Sci. of India, 10-11, p.136-149.

40. _____. "Difference Sets and Combinatorial Arrangements Derivable from Finite Geometries". Nat. Inst. Sci. of India, 12, p.123-135.

41. Reed, I. S. 1954. "A Class of Multiple Error-Correcting Codes and the Decoding Scheme". IRE Trans., PGIT-4, p.38-49

42. Rudolph, L. D. 1967. "A Class of MLD Codes". IEEE Trans., IT-13, p.305-307

43. _____. 1968. "Threshold Decoding of Cyclic Codes". IEEE Trans., IT-15,p.414-418.

44. Rudolph, L. D., and Hartman, C. R. P. 1973. "Decoding By Sequential Code Reduction". IEEE Trans., IT-19, p.549-555.

45. Shannon, C. E. and Weaver, W. 1949. Mathematical Theory of Communication. Urbana: U. of Illinois Press.

46. Shiva, S. G. S., and Tavares, S. E. 1974. "On Binary Majority Logic Decodable Codes". IEEE Trans., IT-20, p.131-133.

47. Smith, K. J. C. 1969. "On p-rank of the Incidence Matrix of Points and Hyperplanes in Finite Projective Geometries". J. Comb Th., p.122-129.

48. Tavares, S. E., Allard, P. E., Shiva, S. G. S. 1971. "On the Decomposition of Cyclic Codes into Cyclic Classes". Inf. and Control, 18, p.342-354.

49. _____. 1973. "A Note on the Decomposition of Cyclic Codes into Cyclic Classes". Inf. and Control, 22, p.100-106.

50. Townsend R. L. and Weldon, E. J. 1967. "Self-Orthogonal Quasi-Cyclic Codes". IEEE Trans., IT-13, p.183-195.

51. Veblen, O., and Bussey, M. 1906. "Finite Projective Geometries". Trans. Am. Math. Soc., 7, p.241-259.

52. Veblen, O., and Young, J. W. 1910. *Projective Geometry*, Vols. I and III. New York: Blaisdel Publishers.

53. Weldon, E. J. 1966. "Difference Set Cyclic Codes". *Bell Systems Tech J.*, 45, p.1045-1055.

54. _____. 1967. "Euclidean Geometry Cyclic Codes". *Proceedings of Symposium of Combinatorial Mathematics at the University of North Carolina, Chapel Hill*, N.C.

55. _____. 1968. "New Generalizations of the Reed-Muller Codes, Part II: Non-Primitive Codes". *IEEE Trans.*, *IT-14*, p.199-205.

56. Willet, M. C. 1975. "Cycle Representatives for Minimal Cyclic Codes". *IEEE Trans.*, *IT-21*, p.716-717.

57. Wolf, J. K. 1973. "A Survey of Coding Theory, 1967-72". *IEEE Trans.*, *IT-19*, p.381-389.

58. Yamamoto, S., Fukuda, T., Hamada, N. 1966. "On Finite Geometries and Cyclically Generated Incomplete Block Designs". *J. Sci. Hiroshima Univ. Ser. A-I*, p.137-149.