Weight Enumerators and Weight Distribution of KM Codes

by

Keith Pickavance

A thesis submitted to
the Faculty of Science
at the University of Glasgow
for the degree of
Doctor of Philosophy

July 1997

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Preface

This thesis was submitted in accordance with the rules of the Faculty of Science at the University of Glasgow for the degree of Doctor of Philosophy.

I would like to express my deepest gratitude to my supervisor, Doctor Stuart G. Hoggar for his advice, assistance and encouragement during the course of this research.

I would also like to thank the then Head of Department Professor Ray W. Ogden and his successor Professor Ken A. Brown.

Thanks are also due to to the Science and Engineering Research Council (now renamed as the Engineering and Physical Science Research Council) for their financial support; further thanks being due to the Department of Mathematics, Computing Services and The Office of Information Technology all at the University of Glasgow.

Finally I would like to say how grateful I am to my parents for their support and encouragement throughout my time at the University and would like to dedicate my thesis to them.

Abstract

In this thesis we will attempt to classify a particular class of KM codes (named after H. Krishna and S.D. Morgera). The construction of these codes is based mainly on the multiplicative complexity problem of multiplying two polynomials together, as this relates directly to a particular type of bilinear form and the main theorem linking codes and bilinear forms (Theorem 3.11) can then be invoked.

We review work done by Winograd, Hopcroft and Musinski and Fiduccia on the general multiplicative complexity problem and work done by Lempel, Seroussi and Winograd on the topic of using the Chinese Remainder Theorem on this problem. We introduce a new method of looking for algorithms for polynomial multiplication, by way of diagrams. This will then enable us to produce many more related algorithms. We also develop two new bounds for the minimum number of multiplications needed to multiply two polynomials modulo u^η over $\mathbb{F}_2$ (which we denote by $M_2(\eta-1, u^\eta)$), namely one lower bound of $\frac{5}{2}(\eta-1)$ and an attainable upper bound of $\frac{\eta^2}{4}+\eta+1$ for η even and $\frac{\eta^2}{4}+\eta-\frac{5}{4}$ for η odd. This attainable bound is used in the construction of the KM codes with wraparound later in the thesis. We also develop an attainable upper bound for the number of multiplications needed to multiply two polynomials modulo $P(u)$ over $\mathbb{F}_2$ (which we denote by $M_2(\eta-1, P(u))$); again this turns out to be of the order η^2, but the algorithms are easily generalised for any η.

We fully classify KM codes for the parameter $N(=k+d-1) \leq 4$ using these results on complexity and algorithm formation, and attempt the case of $N=5$. This enables us to find that the optimal weight enumerator is obtainable by KM codes for $N \leq 5$.

Next we introduce *constant degree codes* and obtain tables of the weight enumerators

for the shunted (the idea of reducing/increasing k while increasing/decreasing d) families for w (the degree of the coprime polynomials used in the CRT) equals 2 and give the reader the proof of the generalising nature of our work. This last section is done in two separate ways, firstly we obtain results via the dual code weight enumerator and then dualise back to the primary code using the MacWilliams identities, and then we further develop the idea of using the dual code and look at the sets of the relations found and obtain results that use some complex linear algebra and operations on the vectors to express the weight distribution of the codes. The results obtained should serve as a basis for anybody using KM codes.

Contents

Chapter 1

Introduction

In this thesis we look further at KM codes in a more mathematical sense than the way they were first investigated by Krishna in his doctoral thesis and further in Krishna and Morgera (1987), Krishna (1987) and Krishna (1993).

In Chapter 2, we give a thorough background on where the KM codes are useful, in particular in a GH-ARQ scheme of coding. We explain some relevant coding theory, concerning mainly the weight enumerators and the relation between the dual codes and the primary codes. Some finite field theory is explained that will be used mostly when dealing with multiplying polynomials modulo another polynomial over a particular field. For the most part of the thesis we are concerned only with the finite field $\mathbb{F}_2$, but any results that are applicable for a larger field will be stated as so. Next we introduce the polynomial form of the Chinese Remainder Theorem and further an improved procedure for using the Chinese Remainder Theorem to multiply two polynomials together.

In Chapter 3 we introduce KM codes. Initially we look at systems of bilinear forms and what are known as computations and concern ourselves with the *multiplicative complexity* of these computations. This work was done by Hopcroft and Musinski (1973), Winograd (1977) and Fiduccia (1971). An important link by Krishna (1987) exposing the multiplicative complexity and bilinear forms is then explained. Further to the ideas in Krishna (1987), we look at polynomial multiplication and produce a bilinear form relating to this so that all the results of this chapter can be applied. So we have that from a computation $G(Ax \times By)$

of a system of bilinear forms $\boldsymbol{\theta} = X\boldsymbol{y}$, G generates a linear code. The parameters of the code are explained in the chapter.

In Chapter 4 we move away slightly from coding theory and investigate complexity theory, specifically that concerned with multiplying two polynomials together, both directly and modulo another polynomial. We use specifically the theory developed in Chapter 3 to investigate the problem of finding $\boldsymbol{m}$ from the coefficients of polynomial multiplication, $\boldsymbol{\Phi} = A^T\boldsymbol{m} = A^T(G\boldsymbol{x} \times B\boldsymbol{y})$. In Section 3 we give some already known bounds, on the degrees of the two polynomials, for the number of multiplications needed. In Section 4 we develop a new lower bound for $M_2(\eta - 1, u^\eta)$, the number of multiplications needed to multiply two degree $\eta - 1$ polynomials modulo u^η. The proof is based on the work in Winograd (1977), but the result is new. In Section 5 we develop a totally new way of looking at the idea of polynomials multiplication, that of a diagrammatic representation. This idea is developed so that we can develop new algorithms from old ones so as to eventually (in our case anyway) find the maximum number of KM codes for a given set of parameters. Indeed in Section 6 we develop a new infinite algorithm, known as the *square, P(u) algorithm* that is used to multiply two degree $\eta - 1$ polynomials. A quadratic bound is found for this attainable algorithm. We next look in Section 7 at some previously known algorithms and obtain the upper bounds from them. The LSW upper bound is directly from the paper of Lempel, Seroussi and Winograd (1983), but we extend this using techniques from the improved Chinese Remainder Theorem of Chapter 2, and also some known facts about the minimum number of multiplications needed for particular values of η. This gives a good upper bound over the values we choose but a closed form of the upper bound cannot be found. The Knuth algorithm and upper bound that we find is obtained by converting the multiplication of two numbers to that of multiplying two polynomials, giving another quadratic bound. A graphical comparison is then given to visually let the reader perceive the advantages and disadvantages of the algorithms explained. Finally we develop a new upper bound $M_2(\eta - 1, u^\eta)$ and associated algorithm which we will use later in the thesis for the wraparound part of the KM codes (that associated with the improved procedure of the Chinese Remainder Theorem).

In Chapter 5 we start to use all the techniques developed to classify KM codes for particular values of the parameter N $(= k + d - 1)$. First of all we explain a little about the structure of KM codes and some of the powerful attributes associated with them, in particular their ability to *shunt* (the idea of increasing/decreasing k in order to decrease/increase d). We then look at all the KM codes for $N \leq 4$ and tabulate all the possible cases. We concern ourselves mainly with the *optimal* codes in the way of having weight enumerator with least $A_{w_{min}}$, but also consider other obtainable weight enumerators. Next we look at the $N = 5$ problem and bring in the usage of the diagrammatic representation as developed in Chapter 4. This way enables us to find KM codes with optimal weight enumerator and some (but not all) of the non-optimal weight enumerators. We state a conjecture that says that we can never get all the possible weight enumerators from KM codes.

In Chapter 6 we are concerned with finding the weight enumerators of shunted families of constant degree KM codes (with $w = 2$). We first investigate the periodicity exhibited by the blocks associated with the coprime factors of $P(u)$ of the generator matrices. Next we develop some important results about where we are to find the $n - k$ required independent relations that we will use to form the dual code and then dualise back to the primary code using the MacWilliams transform. On the way we explain how instead of using the Chinese Remainder Theorem to obtain the relations it can be done directly from the generator matrix G, by removing certain columns. For the constant degree codes we take $P(u) = u^2(u^2 + 1)(u^2 + u + 1)$ with $s = 0, 1, 2$ and construct tables of the weight enumerators. The intention is both to construct tables and also to relate the weight enumerators for various values of k. This we do as best possible by finding important results on the sets of relations that we are to use. Finally we consider the larger case of $w = 3$, taking $P(u) = u^3(u^3 + 1)(u^3 + u + 1)(u^3 + u^2 + 1)$. The resulting codes are much bigger and so this is only given a brief overview, but the ideas developed previously are still applicable.

In Chapter 7 we work closely with the ideas of Hoggar (1997), but keeping our ideas of increasing s using the algorithm developed in Chapter 4. We introduce new vectors and their shifts and related operations and use these to further develop the theory needed to find the weight enumerators of shunted families of KM codes. Using the extra knowledge

we are easily able to contain the $s = 3$ case and obtain a table of the weight distribution.

It is useful to note that often in Chapters 6 and 7 results obtained for one part that appear to be overkill are essential for making a future problem much easier and so overall all results are necessary.

Finally in the thesis we look to the future and indicate where the work can move onto. It is hoped that all fields will be pursued further as the topic sheds light for many areas of mathematics.

Chapter 2

Background

2.1 Digital Communication Systems

In this section we very briefly describe the two fundamental methods for error control in digital communication systems, those of FEC (forward error correction) and ARQ (automatic repeat request), and their hybrid combinations, see McFarlane (1992), Lin and Costello (1983), Krishna and Morgera (1987). A generalised version of one of the hybrid combinations is given at the end of the section, see Krishna (1987).

2.1.1 FEC Systems

Using this scheme the receiver will make use of an error-correcting code. The transmitter will send a codeword over the channel and the receiver will, if an error has been detected, attempt to correct it. The problem with this scheme is that in some cases the receiver may not be able to correct the errors. In this case the erroneous word is sent to the user and it is assumed to be error free. This scheme uses one way traffic only along the channel. If we introduce some feedback we can get the following.

2.1.2 ARQ Systems

These schemes utilise a system that requires two-directional channels, to enable the transmitter to keep a record of what the receiver is doing. There are essentially two types of

ARQ: stop-and-wait ARQ, and continuous ARQ. No error correction is employed by these systems, only error-detection.

Stop-and-Wait ARQ

Here the transmitter sends a codeword to the receiver, then waits for a reply from the receiver in the form of ACK (acknowledge that the codeword has no detectable errors) or NACK (the receiver has detected errors). If ACK is received the transmitter then sends the next codeword. If NACK is received the transmitter resends the preceding codeword. This will go on until the codeword is received with no detectable errors.

Continuous ARQ

In this scheme the transmitter sends codewords continuously to the receiver, and the receiver ACKs/NACKs continuously (after the initial round trip delay, where $N-1$ codewords have been sent). Two methods exist for the retransmission of codewords that have resulted in NACK, as follows:

Go-back-N ARQ For this when a NACK is received for a particular codeword, the transmitter resends that codeword plus all the codewords that follow. So in all a further N codewords are sent regardless of whether the following $N-1$ were erroneous. The main advantage of this method is that the codewords are kept in order all the time, obviously the time spent sending the extra codewords is a waste and we may wish to use the following method.

Selective Repeat ARQ In this the transmitter only resends those codewords that result in a NACK. This method obviously will result in the ordering of the codewords being upset and so the receiver must have adequate buffering facilities.

Many drawbacks exist for each of the FEC and ARQs, and there exist schemes that incorporate parts of each of FEC and ARQ.

2.1.3 Hybrid ARQ Systems

By combining both FEC and ARQ we are able to build schemes that offer higher throughput than can be achieved by ARQ alone and a higher reliability than with FEC alone.

Type I Hybrid ARQ

In this system, when a received codeword is found to contain errors, the receiver first attempts to correct them. If this is possible (i.e. the number of errors is within the error-correcting capability of the code) they will be corrected, the decoded word then delivered to the user and ACK sent to the transmitter. The transmitter will then send the next codeword. If errors are not correctable, the receiver will send a NACK to the transmitter thus requesting the sending of the codeword again. The transmitter will then send the codeword, the receiver attempts to correct any errors and so on.

Type II Hybrid ARQ

This scheme is based on the parity check bits for error-correction being sent to the receiver only when required. Two linear codes are used for this scheme; one is a high rate (n, k) error-detecting code C_0, the other is a half-rate invertible[1] $(2k, k)$ code C_1.

First of all the message to be sent is encoded using C_0 then sent across the channel. At the receiver the codeword is checked for errors, if none are detected then the receiver assumes that none have occurred, ACK is sent to the transmitter and the receiver decodes the codeword for the user. If errors have been detected, but can't be corrected by C_0, the receiver sends NACK to the transmitter and stores the erroneous codeword in a buffer. This indicates to the transmitter that the receiver requires more information. The transmitter then encodes the original message using C_1 and sends only the parity check bits. Then combining the previous erroneous codeword and the new parity check bits a new codeword is formed and then checked for errors using C_1. If the number of errors is not within the error correcting capabilities of C_1, then a NACK is sent to the transmitter and a second

[1] An invertible code is one such that the original information word can be obtained from just the parity check bits of a codeword, see Krishna (1987).

retransmission occurs. This retransmission can be either the original codeword or again the parity check bits depending on the setup of the system. If the errors are within the capability of C_1, then the codeword is decoded for the user.

2.1.4 Generalised Type II Hybrid ARQ (GH-ARQ)

In Type II Hybrid ARQ schemes the second retransmission is either the same as the original codeword or the parity check bits following this. If it was possible to introduce more parity blocks then the performance of the system could be improved. For example, if instead of an invertible $(2k, k)$ code we had a $(3k, k)$ code then there would be the possibility of two extra parity blocks to assist in error-correction. If the $(3k, k)$ code was invertible this would enable an extra chance to decode the parity check bits if the previous codeword or parity checks had been totally destroyed by for example a burst of errors. This scheme can be generalised to any number of extra parity check blocks before repetition starts, and we can get GH-ARQ as follows. We again use two codes; one a high rate (n, k) error-detecting code, C_0, and a (maybe invertible) (mk, k) code C_1, which is used adaptively for error-correction. The quantity m here, is known as the *depth* of the code.

What we require is that the generator matrix of C_1 should be of the form

$$G = [G_1|\ G_2|\ \ldots|\ G_m]$$

with at least G_1 invertible. Also that if

$$G^{(i)} = [G_1|\ G_2|\ \ldots|\ G_i]$$

has minimum distance d_i then $d_i < d_j$ for all $1 \leq i < j \leq m$. The exact method of implementation is simply an extension of Type II Hybrid ARQ, and can be found in McFarlane (1992), Krishna and Morgera (1987), Krishna (1987).

2.2 Coding Theory

In this section we give a brief explanation of the types of codes we will be looking at and some aspects of the codes that will prove to be useful and interesting. For a more complete look at the sections see among others MacWilliams and Sloane (1977), Hill (1986), van Lint (1982) or Lin and Costello (1983).

2.2.1 Linear Codes

Primary and dual codes

Definition 2.1 (i) Let V_q denote the set of all all ordered n - tuples over $\mathbb{F}_q$. The value n will be obvious in the context.
(ii) By a *linear* (n, k) *code,* C, we shall mean a k - dimensional subspace of V_q.
(iii) By a *generator matrix* G of C we shall mean a $(k \times n)$ matrix such that the set of all linear combinations of the rows of G over $\mathbb{F}_q$ form the code C.
(iv) By the *dual code* $C^{\perp}$ of C we shall mean the set of all vectors in V_q orthogonal to each vector in C. It is also linear and so has a generator matrix, which we shall usually denote as H. This matrix H is known as the *parity check matrix* of the code C, because for any $\boldsymbol{c}$ in V_q, $\boldsymbol{c} \in C \Leftrightarrow H\boldsymbol{c}^T = 0$, i.e. we get parity check equations of the form

$$h_{i0}c_0 + \cdots + h_{i,n-1}c_{n-1} = 0 \tag{2.1}$$

from (row i of $H)\boldsymbol{c}^T$. These parity check equations are sometimes known as the *defining equations* for the codewords of C, as they define every codeword in C completely. H has rank $n - k$, the dimension of $C^{\perp}$ so we can get $n - k$ linearly independent parity check equations.

Minimum Distance

From now on we will refer to the n - vectors in a code C as *codewords* in C.

Definition 2.2 (i) The *(Hamming) distance* between two codewords $\boldsymbol{c_0} = c_{00} \ldots c_{0,n-1}$ and $\boldsymbol{c_1} = c_{10} \ldots c_{1,n-1}$ is the number of places where they differ, denoted by dist $(\boldsymbol{c_0}, \boldsymbol{c_1})$.

(ii) The *(Hamming) weight* of a codeword $\boldsymbol{c} = c_0 \ldots c_{n-1}$ is the number of non-zero components, and is denoted by wt $(\boldsymbol{c})$. Obviously dist $(\boldsymbol{c_0}, \boldsymbol{c_1})$ = wt $(\boldsymbol{c_0} - \boldsymbol{c_1})$
(iii) The *minimum distance* of a linear (n, k) code C is the minimum (Hamming) distance between distinct codewords of C:

$$\begin{aligned} d &= \min \{\text{dist } (\boldsymbol{c_0}, \boldsymbol{c_1})\} \\ &= \min \{\text{wt } (\boldsymbol{c_0} - \boldsymbol{c_1})\} \end{aligned}$$

with $\boldsymbol{c_0}, \boldsymbol{c_1} \in C, \boldsymbol{c_0} \neq \boldsymbol{c_1}$.

From now on a linear (n, k) code with minimum distance d will be called an (n, k, d) *code*. Finding d seems a daunting task for large codes but if some structure is known about the parity check matrix the the following two theorems may help.

Theorem 2.3 *The minimum distance of a linear code is the minimum weight of any non-zero codeword.*

Theorem 2.4 *If H is the parity check matrix of an (n, k, d) code C then every $d-1$ columns of H are linearly independent, but some d columns are linearly dependent.*

2.2.2 Weight Enumerators

In this section we introduce a well known way of classifying codes in a polynomial form (see MacWilliams and Sloane (1977), MacWilliams (1963)).

Definition 2.5 Let A_i denote the number of codewords of weight i in C. Then the polynomial

$$A(z) = \sum_{i=0}^{n} A_i z^i,$$

is called the *weight enumerator of C*. Note that this can also be written

$$A(z) = \sum_{\boldsymbol{c} \in C} z^{\text{wt } (\boldsymbol{c})}.$$

Similarly let B_i denote the number of codewords of weight i in $C^{\perp}$. The *weight enumerator of* $C^{\perp}$ is

$$B(z) = \sum_{i=0}^{n} B_i z^i.$$

The following theorem due to MacWilliams (1963) is a remarkable one relating the weight enumerator $B(z)$ of the dual code to the weight enumerator $A(z)$ of the primary code . The proof can be found in many texts, obviously in MacWilliams and Sloane (1977) and MacWilliams (1963), but a rather nice version for the binary case (which we are mainly interested in here) can be found in Hill (1986).

Theorem 2.6 (MacWilliams (1963)) *If C is a (n, k, d) code over $\mathbb{F}_q$ with dual $C^{\perp}$, then*

$$B(z) = \frac{1}{q^k}(1 + (q-1)z)^n A\left(\frac{1-z}{1+(q-1)z}\right).$$

The MacWilliams Identities are symmetric in the sense that we can find the weight enumerator of the primary code uniquely from the weight enumerator of the dual code from

$$A(z) = \frac{1}{q^{n-k}}(1 + (q-1)z)^n B\left(\frac{1-z}{1+(q-1)z}\right). \tag{2.2}$$

The usefulness of these identities is obvious when we have to calculate the weight enumerator of a primary (n, k, d) code where k is large. To enumerate all q^k codewords may require vast computation time. However $n - k$ may be small compared with k, then the dual code is smaller and so if we find the weight enumerator of the dual code, then use (2.2), the weight enumerator of the primary code can be found with less computation.

2.3 The Finite Fields $\mathbb{F}_q$

Here we give a brief look at some ideas we will be using throughout the thesis, concerning finite fields. The theory in this section holds for any field although generally we will be looking at the fields $\mathbb{F}_2$.

2.3.1 Polynomials over $\mathbb{F}_q$

Unless otherwise stated let $f(x)$, $g(x)$ and $h(x)$ be polynomials over $\mathbb{F}_q$. We give a few definitions just to make this section fully understandable.

Definition 2.7 (i) A polynomial $f(x)$ is *irreducible* if $f(x)$ has positive degree and $f(x) = g(x)h(x)$ implies that either $g(x)$ or $h(x)$ is a constant polynomial.
(ii) Let $f(x)$ be a non-constant polynomial. If $f(0) \neq 0$, then the least positive integer e for which $f(x)$ divides $x^e - 1$ is called the *order* of f, denoted ord(f). If $f(0) = 0$, then $f(x) = x^i g(x)$, where i is a positive integer and $g(0) \neq 0$. Here i and $g(x)$ are uniquely determined and we define ord (f) = ord (g).
(iii) Define the *reciprocal polynomial*, $\overline{f}(x)$ of $f(x)$ as

$$\overline{f}(x) = x^{\deg[f(x)]} f\left(\frac{1}{x}\right).$$

2.3.2 Reduction of Polynomials Modulo Another Polynomial

In the next chapter we must have a thorough understanding of the residue class ring $\mathbb{F}_q[x]/(f)$, where f is an arbitrary non-zero, non-constant polynomial so we give some theory here (see Lidl and Niederreiter (1993)).

Theorem 2.8 *The residue class ring $\mathbb{F}_q[x]/(f)$ is a field iff f is irreducible over $\mathbb{F}_q$.*

The residue class ring $\mathbb{F}_q[x]/(f)$ consists of residue classes

$$[g] = g + (f).$$

Two residue classes $[g_1]$ and $[g_2]$ are identical if $g_1 \equiv g_2 \pmod{f}$, i.e. if $g_1 - g_2$ is divisible by f. This is equivalent to requiring that g_1 and g_2 leave the same remainder after division

by f. Each residue class $[g]$ has a unique representative $r \in \mathbb{F}_q[x]$ with $\deg(r) < \deg(f)$, where r is the remainder on dividing g by f. The process of passing from g to r is known as reduction modulo f. So we can now see that the residue classes are distinct and are $r + (f)$ where r runs through all the polynomials in $\mathbb{F}_q[x]$ with $\deg(r) < \deg(f)$.

Once we look at the residue class ring $\mathbb{F}_q[x]/(f)$, which may be a field, a natural question to ask is whenever we have $g(x)$ and $f(x)$, when do we have an inverse of $g(x)$ in $\mathbb{F}_q[x]/(f)$ i.e. does there exist $h(x)$ such that

$$g(x)h(x) \equiv 1 \pmod{f(x)}.$$

This exists iff $\gcd(g(x), h(x)) = 1$ as $\mathbb{F}_q[x]$ is a Euclidean domain.

The residue class ring $\mathbb{F}_q[x]/(f)$, f irreducible, can easily be seen to represent a field in a vector form by taking the coefficients of the powers of x of the polynomials reduced modulo $f(x)$ as the elements of the vectors.

Example 2.9 We look at $x^2 + x + 1$ over $\mathbb{F}_2[x]$. This polynomial is irreducible over $\mathbb{F}_2[x]$, so $\mathbb{F}_2[x]/(x^2 + x + 1)$ is a field. Let α be a root of the polynomial $f(x)$ (so $\alpha \in \mathbb{F}_{2^2}$), then each element of the field $\mathbb{F}_{2^2}$ can be represented by a polynomial of degree less than 2 (the degree of $f(x)$) in α. Here we have $\alpha^2 = \alpha + 1$, so we can write the elements of the field as $\{0, 1, \alpha, \alpha + 1\}$.

Definition 2.10 Let $\beta = a_0\alpha^0 + \cdots + a_{m-1}\alpha^{m-1}$ be an element of a finite ring $\mathbb{F}_q[x]/(f(x))$, with $m = \deg(f(x))$, and α a root of $f(x)$. Then the *vector form* of the element β is

$$(a_0 \ldots a_{m-1}).$$

So we see that the field in Example 2.9 may be equivalently written as $\{00, 01, 10, 11\}$.

Definition 2.11 By the *matrix form of (a reduction polynomial)* $f(x)$, we shall mean the matrix R in the following. If we have $Y(x)$ of degree $h - 1$ and we reduce it modulo $f(x)$ of degree D, say, to give $a(x)$, then there are constants r_{ij}, such that

$$x^i \pmod{f(x)} = \sum_{j=1}^{D-1} r_{ij} x^j. \tag{2.3}$$

Hence

$$a(x) = \sum_{i=0}^{h-1} y_i \sum_{j=0}^{D-1} r_{ij} x^j. \tag{2.4}$$

We may write this in terms of the $h \times D$ matrix $R = [r_{ij}]$, where obviously the first D rows (if $h > 0$) form the identity matrix,

$$\boldsymbol{a} = \boldsymbol{y}R = \boldsymbol{y} \begin{bmatrix} I_D \\ \vdots \end{bmatrix}. \tag{2.5}$$

Example 2.12 Let $f(x) = x^3 + x^2 + x + 1$ be over $\mathbb{F}_2$. As $f(x)$ is not irreducible, the finite ring $\mathbb{F}_2[x]/(f(x))$ is not a field. If we let α be a root of $f(x)$ then $\alpha^3 + \alpha^2 + \alpha + 1 = 0$, and as we are working over $\mathbb{F}_2$ we get $\alpha^3 = \alpha^2 + \alpha + 1$, and all the powers of α (reduced modulo $\alpha^3 + \alpha^2 + \alpha + 1$) are α, α^2, $\alpha^2 + \alpha + 1$. In the matrix form as Definition 2.11 we get:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

Now the matrix R can easily be seen (when $f(x)$ is irreducible) to contain each non-zero element of the finite field $\mathbb{F}_q[x]/(f(x))$ in vector form. We now give an example to show how this matrix R can be used in practice.

Example 2.13 Taking the general degree 5 polynomial $g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + g_4 x^4 + g_5 x^5$ modulo $x^2 + x + 1$ we could work this out via

$$\begin{bmatrix} g_0 & g_1 & g_2 & g_3 & g_4 & g_5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \tag{2.6}$$

giving the result $\begin{bmatrix} g_0 + g_2 + g_3 + g_5 & g_1 + g_2 + g_4 + g_5 \end{bmatrix}$ as expected.

2.4 The Chinese Remainder Theorem

The main results in this section can be applied to any finite field $\mathbb{F}_q$, (q a prime power), but all the examples will be done over $\mathbb{F}_{2^n}$.

2.4.1 Setting the Scene

We are trying to find a way to find an algorithm for multiplying two polynomials, $Z(u)$ of degree $k-1$ and $Y(u)$ of degree $d-1$, together. If we take a polynomial $P(u)$ of degree $N = k+d-1$ then (multiplying $Z(u)$ and $Y(u)$ to give $\Phi(u)$,) reducing $\Phi(u)$ mod $P(u)$ will have no effect on $\Phi(u)$. This forms the basis of our reason for using the CRT. Define formally

$$\Phi(u) = Z(u)Y(u) \tag{2.7}$$

so $\Phi(u)$ of degree $N-1$ is unchanged if it is reduced modulo $P(u)$.

Now suppose

$$P(u) = \prod_{i=1}^{L} P_i(u), \quad \deg\left[P_i(u)\right] = \alpha_i \tag{2.8}$$

and $\gcd\left(P_i(u), P_j(u)\right) = 1$, $1 \leq i < j \leq L$.

Let $Z_i(u)$ be the unique reduction of $Z(u)$ modulo $P_i(u)$, to a polynomial of degree less than that of $P_i(u)$. We can now form the $2L$ congruences

$$\begin{aligned} Z_i(u) &\equiv Z(u) \pmod{P_i(u)} \\ Y_i(u) &\equiv Y(u) \pmod{P_i(u)} \end{aligned} \qquad i = 1, \ldots, L, \tag{2.9}$$

and hence the L congruences

$$\Phi_i(u) \equiv Z_i(u)Y_i(u) \pmod{P_i(u)}, \quad i = 1, \ldots, L. \tag{2.10}$$

From these L congruences we can find a unique solution $\Phi(u)$ mod $P(u)$ using the Chinese Remainder Theorem.

2.4.2 The Chinese Remainder Theorem for Polynomials

Here we state and prove the version of the Chinese Remainder Theorem we will be using throughout this thesis. For a very general proof of the CRT see Lang (1993).

Theorem 2.14 (The CRT for polynomials) *Let $P(u) = P_1(u)P_2(u)\dots P_L(u)$ be pairwise relatively prime polynomials. Then the system of congruences*

$$\begin{aligned} \Phi(u) &\equiv \Phi_1(u) \pmod{P_1(u)} \\ \Phi(u) &\equiv \Phi_2(u) \pmod{P_2(u)} \\ &\vdots \\ \Phi(u) &\equiv \Phi_L(u) \pmod{P_L(u)} \end{aligned} \tag{2.11}$$

has a unique solution, modulo $P(u)$,

$$\Phi(u) = \Phi_1(u)Q_1(u)R_1(u) + \cdots + \Phi_L(u)Q_L(u)R_L(u) \tag{2.12}$$

where $Q_k(u) = P(u)/P_k(u)$, and $R_k(u)$ is an inverse of $Q_k(u)$ mod $P_k(u)$, i.e.

$$Q_k(u)R_k(u) \equiv 1 \pmod{P_k(u)}.$$

Proof. Let $Q_k(u) = P(u)/P_k(u)$. We know that gcd $[Q_k(u), P_k(u)] = 1$, since gcd $[P_i(u), P_j(u)] = 1$ whenever $i \neq j$. Hence we can find an inverse $R_k(u)$ of $Q_k(u)$ modulo $P_k(u)$, so that

$$Q_k(u)R_k(u) \equiv 1 \pmod{P_k(u)}.$$

We now form the sum

$$\Phi(u) = \Phi_1(u)Q_1(u)R_1(u) + \cdots + \Phi_L(u)Q_L(u)R_L(u). \tag{2.13}$$

The polynomial (2.13) is a simultaneous solution of the L congruences. To demonstrate this we must show that

$$\Phi(u) \equiv \Phi_k(u) \pmod{P_k(u)}, \quad k = 1, \dots, L.$$

Since $P_j(u) | Q_k(u)$ whenever $j \neq k$, we have

$$Q_j(u) \equiv 0 \pmod{P_k(u)}.$$

Therefore, in the sum (2.13), all the terms except the kth term are congruent to zero modulo $P_k(u)$. Hence

$$\Phi(u) \equiv \Phi_k(u)Q_k(u)R_k(u) \equiv \Phi_k(u) \pmod{P_k(u)},$$

since $Q_k(u)R_k(u) \equiv 1 \pmod{P_k(u)}$. □

Definition 2.15 When using the CRT with L pairwise relatively prime polynomials $P_1(u), \ldots, P_L(u)$, we say that we are using the CRT with L *blocks*. This then enables us, when using the CRT to section the work into blocks.

Example 2.16 Let deg $[Z(u)] = 3$ and deg $[Y(u)] = 1$. Therefore $k = 4$, $d = 2$ and $N = 5$. We need, in order for the product $Z(u)Y(u)$ to be unaffected by taking it modulo $P(u)$, to have deg $[P(u)] = 5$. For this example let $P(u) = u^2(u+1)(u^2+u+1)$, so we have $L = 3$ and hence three blocks.

Now

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 + z_3 u^3 \\ Y(u) &= y_0 + y_1 u \end{aligned} \tag{2.14}$$

and $P_1(u) = u^2$, $P_2(u) = u+1$ and $P_3(u) = u^2+u+1$.

We reduce $Z(u)$ and $Y(u)$ modulo $P_i(u)$, $i = 1, 2, 3$.

i) modulo u^2

$$\begin{aligned} Z_1(u) &= Z(u) \pmod{u^2} \\ &= z_0 + z_1 u \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_1(u) &= Y(u) \pmod{u^2} \\ &= y_0 + y_1 u. \end{aligned} \tag{2.15}$$

Then

$$\begin{aligned} \Phi_1(u) &\equiv Z_1(u)Y_1(u) \pmod{u^2} \\ &= (z_0 + z_1 u)(y_0 + y_1 u) \pmod{u^2} \\ &= z_0 y_0 + (z_0 y_1 + z_1 y_0)u + z_1 y_1 u^2 \pmod{u^2} \\ &\equiv z_0 y_0 + (z_0 y_1 + z_1 y_0)u \pmod{u^2}. \end{aligned} \tag{2.16}$$

ii) modulo $u+1$

$$\begin{aligned} Z_2(u) &= Z(u) \pmod{u+1} \\ &= z_0 + z_1 + z_2 + z_3 \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_2(u) &= Y(u) \pmod{u+1} \\ &= y_0 + y_1. \end{aligned} \tag{2.17}$$

Then

$$\begin{aligned} \Phi_2(u) &\equiv Z_2(u)Y_2(u) \pmod{u+1} \\ &\equiv (z_0 + z_1 + z_2 + z_3)(y_0 + y_1) \pmod{u+1}. \end{aligned} \tag{2.18}$$

iii) modulo u^2+u+1

$$\begin{aligned} Z_3(u) &= Z(u) \pmod{u^2+u+1} \\ &= (z_0 + z_2 + z_3) + (z_1 + z_2)u \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_3(u) &= Y(u) \pmod{u^2+u+1} \\ &= y_0 + y_1 u. \end{aligned} \tag{2.19}$$

Then

$$\begin{aligned}\Phi_3(u) &\equiv Z_3(u)Y_3(u) \pmod{u^2+u+1} \\ &= (z_0+z_2+z_3)y_0 \\ &\quad +((z_0+z_2+z_3)y_0+(z_1+z_2)y_1 \\ &\quad +(z_0+z_1+z_3)(y_0+y_1))u+(z_1+z_2)y_1u^2 \\ &\quad \pmod{u^2+u+1} \\ &\equiv (z_0y_0+z_2y_0+z_3y_0+z_1y_1+z_2y_1) \\ &\quad +(z_2y_0+z_0y_1+z_1y_0+z_1y_1+z_3y_1)u \\ &\quad \pmod{u^2+u+1}.\end{aligned} \tag{2.20}$$

Now we must find $R_i(u)$, $i=1,2,3$.

i) modulo u^2

$$\begin{aligned}R_1(u)P_2(u)P_3(u) &\equiv 1 \pmod{P_1(u)} \\ R_1(u)(u^3+1) &\equiv 1 \pmod{u^2}.\end{aligned} \tag{2.21}$$

Therefore $R_1(u)=1$.

ii) modulo $u+1$

$$\begin{aligned}R_2(u)P_1(u)P_3(u) &\equiv 1 \pmod{P_2(u)} \\ R_2(u)(u^4+u^3+u^2) &\equiv 1 \pmod{u+1}.\end{aligned} \tag{2.22}$$

Therefore $R_2(u)=1$.

iii) modulo u^2+u+1

$$\begin{aligned}R_3(u)P_1(u)P_2(u) &\equiv 1 \pmod{P_3(u)} \\ R_3(u)(u^3+u^2) &\equiv 1 \pmod{u^2+u+1}.\end{aligned} \tag{2.23}$$

Therefore $R_3(u)=u+1$.

Now we can form the sum, $\Phi(u)$, as follows

$$\begin{aligned}\Phi(u) &= \Phi_1(u)R_1(u)Q_1(u)+\Phi_2(u)R_2(u)Q_2(u) \\ &\quad +\Phi_3(u)R_3(u)Q_3(u) \\ &\vdots \\ &= z_0y_0+(z_0y_1+z_1y_0)u+(z_1y_1+z_2y_0)u^2 \\ &\quad +(z_2y_1+z_3y_0)u^3+z_3y_1u^4.\end{aligned} \tag{2.24}$$

What we are interested in is looking at the algorithms for finding the multiplication of two polynomials, so in order to do this we must introduce the notion of the multiplications needed to construct the algorithm. The reason we only need concern ourselves with the multiplications and not the additions will become clear later, but originates from the fact that the time for a computer to execute an addition is far less than the time for a multiplication. More on the multiplications needed to compute an algorithm can be found in the next chapter but here we will give a brief flavour so that the rest of this section can be understood to be useful.

Example 2.17 Let $Z(u) = z_0 + z_1 u$ and $Y(u) = y_0 + y_1 u$, then the polynomial product $Z(u)Y(u)$ is

$$Z(u)Y(u) = z_0 y_0 + (z_0 y_1 + z_1 y_0)u + z_1 y_1 u^2. \tag{2.25}$$

Written like this it takes four multiplications, but if we define:

$$\begin{array}{rcl|rcl} m_0 & = & z_0 y_0 & m_2 & = & (z_0 + z_1)(y_0 + y_1). \\ m_1 & = & z_1 y_1 & & & \end{array} \tag{2.26}$$

then we can write

$$Z(u)Y(u) = m_0 + (m_0 + m_1 + m_2)u + m_1 u^2 \tag{2.27}$$

which on a computer would take less time than the above method. This method can be implemented directly into the CRT, which as we will see converts the problem of multiplying two large degree polynomials into the problem of multiplication of many smaller degree polynomials.

We have good algorithms for smaller polynomials (see the next chapter) so the recursive method proves very good. Let us do the example above (Example (2.16)) again to demonstrate this idea.

Example 2.18 We reduce $Z(u)$ and $Y(u)$ modulo $P_i(u)$, $i = 1, 2, 3$.
i) modulo u^2

$$\begin{array}{rcl|rcl} Z_1(u) & = & Z(u) \pmod{u^2} & Y_1(u) & = & Y(u) \pmod{u^2} \\ & = & z_0 + z_1 u & & = & y_0 + y_1 u. \end{array} \tag{2.28}$$

Let

$$\begin{array}{rcl|rcl} m_0 &=& z_0y_0 & m_2 &=& (z_0+z_1)(y_0+y_1). \\ m_1 &=& z_1y_1 & & & \end{array} \tag{2.29}$$

Then

$$\begin{aligned} \Phi_1(u) &\equiv Z_1(u)Y_1(u) \pmod{u^2} \\ &= m_0+(m_0+m_1+m_2)u+m_1u^2 \pmod{u^2} \\ &\equiv m_0+(m_0+m_1+m_2)u \pmod{u^2}. \end{aligned} \tag{2.30}$$

ii) modulo $u+1$

$$\begin{array}{rcl|rcl} Z_2(u) &=& Z(u) \pmod{u+1} & Y_2(u) &=& Y(u) \pmod{u+1} \\ &=& z_0+z_1+z_2+z_3 & &=& y_0+y_1. \end{array} \tag{2.31}$$

Let

$$m_3 = (z_0+z_1+z_2+z_3)(y_0+y_1). \tag{2.32}$$

Then

$$\begin{aligned} \Phi_2(u) &\equiv Z_2(u)Y_2(u) \pmod{u+1} \\ &\equiv m_3 \pmod{u+1}. \end{aligned} \tag{2.33}$$

iii) modulo u^2+u+1

$$\begin{array}{rcl|rcl} Z_3(u) &=& Z(u) \pmod{u^2+u+1} & Y_3(u) &=& Y(u) \pmod{u^2+u+1} \\ &=& (z_0+z_2+z_3)+(z_1+z_2)u & &=& y_0+y_1u. \end{array} \tag{2.34}$$

Let

$$\begin{array}{rcl|rcl} m_4 &=& (z_0+z_2+z_3)y_0 & m_6 &=& (z_0+z_1+z_3)(y_0+y_1). \\ m_5 &=& (z_1+z_2)y_1 & & & \end{array} \tag{2.35}$$

Then

$$\begin{aligned} \Phi_3(u) &\equiv Z_3(u)Y_3(u) \pmod{u^2+u+1} \\ &= m_4+(m_4+m_5+m_6)u+m_5u^2 \pmod{u^2+u+1} \\ &\equiv (m_4+m_5)+(m_4+m_6)u \pmod{u^2+u+1}. \end{aligned} \tag{2.36}$$

Now the $R_i(u)$, $i=1,2,3$ are the same as before and we can form the sum, $\Phi(u)$, as

follows

$$\begin{aligned}
\Phi(u) &= \Phi_1(u)R_1(u)Q_1(u) + \Phi_2(u)R_2(u)Q_2(u) + \Phi_3(u)R_3(u)Q_3(u) \\
&= \Phi_1(u)R_1(u)P_2(u)P_3(u) + \Phi_2(u)R_2(u)P_1(u)P_3(u) \\
&\quad +\Phi_3(u)R_3(u)P_1(u)P_2(u) \\
&\vdots \\
&= m_0 + (m_0 + m_1 + m_2)u + (m_3 + m_5 + m_6)u^2 \\
&\quad +(m_0 + m_3 + m_4 + m_6)u^3 \\
&\quad +(m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^4 \pmod{P(u)}.
\end{aligned} \tag{2.37}$$

This sum can be seen to use 7 multiplications, which is not known to be optimal, and indeed we can do better (i.e. less multiplications used) using an improved version of the CRT which can be found in the next section.

Definition 2.19 Further to the previous two examples, if we can find a set of multiplications $m_0, \ldots, m_{n-1}$ (either by the CRT or any other method), so that each coefficient of $\Phi(u)$ is a linear combination of the $m_i, i = 0, \ldots, n-1$, then we say that the $m_i, i = 0, \ldots, n-1$ *reconstruct* the polynomial $\Phi(u)$ or the polynomial $\Phi(u)$ can be *reconstructed* (using the $m_i, i = 0, \ldots, n-1$).

Note here that for each block in the CRT a new set of multiplications is introduced. For Example 2.18, the multiplications m_0, m_1, m_2 are introduced for the first block, m_3 is introduced for the second block and m_4, m_5, m_6 are introduced for the third block.

Definition 2.20 When a multiplication is introduced for a particular block, we say that the multiplication is *associated* with that particular block.

So for Example 2.18, m_5 is associated with block three.

2.5 An Improved Procedure

The above procedure can generally be improved if we permit intentional wraparound, that is if we obtain the top coefficients of the polynomial product $Z(u)Y(u)$ and then use the CRT to find the others. One way is to find the first coefficients of the polynomial product $\overline{Z}(u)\overline{Y}(u)$ modulo u^s, where s is called the *number of wraparound points.* The idea is as follows: it may require less multiplications if we take $P(u)$ of degree $N-1$ and allow a wraparound multiplication (found from $\overline{Z}(u)\overline{Y}(u)$ modulo u) $z_{k-1}y_{d-1}$. This can be generalised to taking $P(u)$ of degree $N-s$ and finding the other multiplications from $\overline{Z}(u)\overline{Y}(u)$ modulo u^s. Using this idea we can always do at least as good as the basic procedure by taking $s \geq 0$ (where $s=0$ is just the basic procedure with no wraparound). Formally put we have the following.

Definition 2.21 Let $Z(u)$ be of degree $k-1$ and $Y(u)$ of degree $d-1$. If $\deg[P(u)] \leq k+d-2$ then $\Phi(u) = Z(u)Y(u)$ is altered by taking it modulo $P(u)$, and we denote $\Phi(u)$ modulo $P(u)$ by $\hat{\Phi}(u)$. Of course if $\deg[P(u)] \geq k+d-1$ then $\Phi(u) = \hat{\Phi}(u)$ as in this case it is unaffected by taking it modulo $P(u)$.

Theorem 2.22 *If we have $Z(u)$ of degree $k-1$ and $Y(u)$ of degree $d-1$ and $P(u) = P_1(u)\dots P_L(u)$, deg $[P(u)] = k+d-1-s$, then if we use the CRT to find the multiplications corresponding to $\hat{\Phi}(u) = Z(u)Y(u) \pmod{P(u)}$, the other multiplications necessary to form $Z(u)Y(u)$ can be found from*

$$\overline{Z}(u)\overline{Y}(u) \pmod{u^s}.$$

Proof. Let the degree of $P(u)$ be D. Let $\Phi(u) = Z(u)Y(u) = \sum_{i=0}^{D+s-1} \phi_i u^i$. The reciprocal of $\Phi(u)$ is $\overline{\Phi}(u) = \sum_{i=0}^{D+s-1} \phi_{D+s-1-i}u^i$ with a reduction mod u^s of $\sum_{i=0}^{s-1} \phi_{D+s-1-i}u^i$. Thus $\phi_D, \dots, \phi_{D+s-1}$ are determined by the wraparound multiplications. Moreover by definition of $\hat{\Phi}(u)$, there are coefficients b_i such that

$$\Phi(u) = \hat{\Phi}(u) + (b_0 + b_1 u + \cdots + b_{s-1}u^{s-1})P(u). \tag{2.38}$$

Now let the $(s \times s)$ matrix

$$M = \begin{bmatrix} p_D & p_{D-1} & \cdots & p_0 & 0 & \cdots & 0 \\ 0 & p_D & \cdots & p_1 & p_0 & \cdots & 0 \\ \vdots & \vdots & & & & & \vdots \end{bmatrix}. \tag{2.39}$$

So equating coefficients of the top s powers of u from (2.38) gives

$$[\phi_{D+s-1} \ldots \phi_{D+1} \phi_D] = [b_{s-1} \ldots b_0]\, M; \tag{2.40}$$

but M is invertible, as $p_D = 1$, so the b_i can be determined by (2.40), as the $\phi_i, i = 0, \ldots, D+s-1$ are known, and as we can obtain $\hat{\Phi}(u)$ we can obtain all coefficients. □

Note when we use the wraparound in the CRT we are increasing the number of blocks by one (i.e. there is another set of multiplications associated with the wraparound part of the CRT).

Example 2.23 We attempt the same problem as Example 2.18 but using wraparound to show that it can be more efficient. Again let deg $[Z(u)] = 3$ and deg $[Y(u)] = 1$. Therefore $k = 4$, $d = 2$ and $N = 5$. Here we will have deg $[P(u)] = 4$ and $s = 1$. For this example let $P(u) = u(u+1)(u^2+u+1)$.

Now

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 + z_3 u^3 \\ Y(u) &= y_0 + y_1 u \end{aligned} \tag{2.41}$$

and $P_1(u) = u$, $P_2(u) = u+1$ and $P_3(u) = u^2+u+1$.

We reduce $Z(u)$ and $Y(u)$ modulo $P_i(u)$, $i = 1, 2, 3$.

i) modulo u

$$\begin{aligned} Z_1(u) &= Z(u) \pmod{u} \\ &= z_0 \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_1(u) &= Y(u) \pmod{u} \\ &= y_0. \end{aligned} \tag{2.42}$$

Let $m_0 = z_0 y_0$, then

$$\begin{aligned} \Phi_1(u) &\equiv Z_1(u) Y_1(u) \pmod{u} \\ &\equiv m_0 \pmod{u}. \end{aligned} \tag{2.43}$$

ii) modulo $u+1$

$$\begin{array}{rcl|rcl} Z_2(u) & = & Z(u) \pmod{u+1} & Y_2(u) & = & Y(u) \pmod{u+1} \\ & = & z_0+z_1+z_2+z_3 & & = & y_0+y_1. \end{array} \tag{2.44}$$

Let $m_1 = (z_0+z_1+z_2+z_3)(y_0+y_1)$, then

$$\begin{array}{rcl} \Phi_2(u) & \equiv & Z_2(u)Y_2(u) \pmod{u+1} \\ & \equiv & m_1 \pmod{u+1}. \end{array} \tag{2.45}$$

iii) modulo u^2+u+1

$$\begin{array}{rcl|rcl} Z_3(u) & = & Z(u) \pmod{u^2+u+1} & Y_3(u) & = & Y(u) \pmod{u^2+u+1} \\ & = & (z_0+z_2+z_3)+(z_1+z_2)u & & = & y_0+y_1u. \end{array} \tag{2.46}$$

Let $m_2 = (z_0+z_2+z_3)y_0$, $m_3 = (z_1+z_2)y_1$ and $m_4 = (z_0+z_1+z_3)(y_0+y_1)$, then

$$\begin{array}{rcl} \Phi_3(u) & \equiv & Z_3(u)Y_3(u) \pmod{u^2+u+1} \\ & = & m_2+(m_2+m_3+m_4)u+m_3u^2 \\ & & \pmod{u^2+u+1} \\ & \equiv & (m_2+m_3)+(m_2+m_4)u \\ & & \pmod{u^2+u+1}. \end{array} \tag{2.47}$$

Now we must find $R_i(u)$, $i = 1,2,3$.

i) modulo u

$$\begin{array}{rcl} R_1(u)P_2(u)P_3(u) & \equiv & 1 \pmod{P_1(u)} \\ R_1(u)(u^3+1) & \equiv & 1 \pmod{u}. \end{array} \tag{2.48}$$

Therefore $R_1(u) = 1$.

ii) modulo $u+1$

$$\begin{array}{rcl} R_2(u)P_1(u)P_3(u) & \equiv & 1 \pmod{P_2(u)} \\ R_2(u)(u^3+u^2+u) & \equiv & 1 \pmod{u+1}. \end{array} \tag{2.49}$$

Therefore $R_2(u) = 1$.

iii) modulo u^2+u+1

$$\begin{array}{rcl} R_3(u)P_1(u)P_2(u) & \equiv & 1 \pmod{P_3(u)} \\ R_3(u)(u^2+u) & \equiv & 1 \pmod{u^2+u+1}. \end{array} \tag{2.50}$$

Therefore $R_3(u) = 1$.

Now we can form the sum, $\hat{\Phi}(u)$, as follows

$$\begin{aligned}
\hat{\Phi}(u) &= \Phi_1(u)R_1(u)Q_1(u) + \Phi_2(u)R_2(u)Q_2(u) + \Phi_3(u)R_3(u)Q_3(u) \\
&= \Phi_1(u)R_1(u)P_2(u)P_3(u) + \Phi_2(u)R_2(u)P_1(u)P_3(u) \\
&\quad +\Phi_3(u)R_3(u)P_1(u)P_2(u) \\
&= m_0 + (m_1 + m_2 + m_3)u + (m_1 + m_3 + m_4)u^2 \\
&\quad +(m_0 + m_1 + m_2 + m_4)u^3.
\end{aligned} \tag{2.51}$$

We now need to use the wraparound, with $s = 1$. We have

$$\begin{aligned}
\overline{Z}(u) &= z_3 + z_2u + z_1u^2 + z_0u^3 \\
\overline{Y}(u) &= y_1 + y_0u,
\end{aligned} \tag{2.52}$$

so

$$\begin{aligned}
\overline{Z}(u) &\equiv z_3 \pmod{u} \\
\overline{Y}(u) &\equiv y_1 \pmod{u}.
\end{aligned} \tag{2.53}$$

Let $m_5 = z_3y_1$, then we have according to the theorem

$$\begin{aligned}
\Phi(u) &= \hat{\Phi}(u) + m_5P(u) \\
&= m_0 + (m_1 + m_2 + m_3 + m_5)u + (m_1 + m_3 + m_4)u^2 \\
&\quad +(m_0 + m_1 + m_2 + m_4)u^3 + m_5u^4 \\
&\vdots \\
&= z_0y_0 + (z_0y_1 + z_1y_0)u + (z_1y_1 + z_2y_0)u^2 \\
&\quad +(z_2y_1 + z_3y_0)u^3 + z_3y_1u^4
\end{aligned} \tag{2.54}$$

using only six multiplications.

This example shows that the improved version of the CRT may be more efficient in calculating the number of multiplications necessary to multiply two polynomials.

Chapter 3

KM Codes

3.1 Introduction

This section will introduce KM Codes. These are expressed as codes formed from noting some linkages between bilinear forms and linear coding theory. See Krishna (1987), Krishna (1993), Krishna and Morgera (1987) and McFarlane (1992).

For the following, unless otherwise stated let the field, F, we shall be working over be $\mathbb{F}_q$, (q a prime power). The work in the section is based on many papers, stated when necessary although in Section 3.4 we have tried to approach the theory from a different angle than previous. The resulting notion is then hopefully clearer enabling the other areas to be considered.

3.2 Bilinear Forms

Consider a vector $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{k-1})^T$ of k bilinear forms over F. The most general form of the bilinear forms θ_i is (by looking at the coefficients of y_i)

$$\begin{aligned} \theta_i &= (a_{00i}x_0 + a_{01i}x_1 + \cdots + a_{0,r-1,i}x_{r-1})y_0 + \cdots \\ & \quad (a_{d-1,0,i}x_0 + a_{d-1,1,i}x_1 + \cdots + a_{d-1,r-1,i}x_{r-1})y_{d-1}, \end{aligned} \tag{3.1}$$

and we may then express $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta} = X\boldsymbol{y} = \begin{bmatrix} X_{0,0} & X_{0,1} & \cdots & X_{0,d-1} \\ X_{1,0} & X_{1,1} & \cdots & X_{1,d-1} \\ \vdots & \vdots & \ddots & \vdots \\ X_{k-1,0} & X_{k-1,1} & \cdots & X_{k-1,d-1} \end{bmatrix} \begin{bmatrix} y_0 \\ \vdots \\ y_{d-1} \end{bmatrix} \tag{3.2}$$

where

$$X_{i,j} = \sum_{m=0}^{r-1} a_{jmi}x_m, i = 0, 1, \ldots, k-1, j = 0, 1, \ldots, d-1. \tag{3.3}$$

Definition 3.1 Define a *computation of* $\boldsymbol{\theta}$ (with n multiplications) to be an expression of the form

$$\boldsymbol{\theta} = G(A\boldsymbol{x} \times B\boldsymbol{y}) \tag{3.4}$$

where A, B and G are matrices of dimension $(n \times r)$, $(n \times d)$ and $(k \times n)$ respectively over F, $\boldsymbol{x} = (x_0, x_1, \ldots, x_{r-1})^T$, $\boldsymbol{y} = (y_0, y_1, \ldots, y_{d-1})^T$ and $\times$ represents component-by-component multiplication of vectors.

It is easy to see that a computation is simply deciding on how the terms of the linear forms shall be grouped and on the order of the operations required to compute the bilinear forms. Hence it is not unique. Let us illustrate this with an example.

Example 3.2 Let the field we shall be working over be $\mathbb{F}_2$. Let us consider

$$\boldsymbol{\theta} = \begin{bmatrix} x_0y_0 + x_1y_1 \\ x_0y_1 + x_1y_0 \\ x_1y_1 \end{bmatrix}. \tag{3.5}$$

This can be written on the one hand as

$$\boldsymbol{\theta} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \right) \tag{3.6}$$

involving four component-by-component multiplications, but if we write it as

$$\boldsymbol{\theta} = \begin{bmatrix} x_0y_0 + x_1y_1 \\ (x_0 + x_1)(y_0 + y_1) + x_0y_0 + x_1y_1 \\ x_1y_1 \end{bmatrix} \tag{3.7}$$

then it can be written as

$$\boldsymbol{\theta} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \right) \tag{3.8}$$

involving only three component-by-component multiplications. This example illustrates that n, the number of component-by-component multiplications, is not fixed for the computation and we get the following.

Definition 3.3 The smallest n, denoted n_{min}, such that a computation exists is known as the *(minimum) multiplicative complexity* of the computation.

Definition 3.4 (Hopcroft and Musinski (1973)) Let $\boldsymbol{z} = (z_0, z_1, \ldots, z_{k-1})^T$ and let $G(A\boldsymbol{x} \times B\boldsymbol{y})$ be a computation of $\boldsymbol{\theta}$, a system of k bilinear forms. The *P-dual* of the computation of $\boldsymbol{\theta}$ is the computation $A^T(G^T\boldsymbol{z} \times B\boldsymbol{y})$.

Do not confuse the P-dual of a computation with the dual code $C^{\perp}$ of C as defined in Definition 2.1 (iv).

Remark 3.5 The word 'dual' in the above definition is valid in that the dual of the dual is the original. If we take the three tuple (G, A, B) as the computation $G(A\boldsymbol{x} \times B\boldsymbol{y})$ then

$$(G, A, B) \overset{\text{P-dual}}{\longleftrightarrow} (A^T, G^T, B) \overset{\text{P-dual}}{\longleftrightarrow} (G, A, B)$$

as expected. The prefix P is used as other types of dual computations do exist although they will not be used in this thesis.

The following theorem links the multiplicative complexity of the original computation and its P-dual.

Theorem 3.6 (Hopcroft and Musinski (1973)) *There is a computation for the system of expressions represented by $G(A\boldsymbol{x} \times B\boldsymbol{y})$ having n multiplications iff there is a computation having n multiplications for its P-dual $A^T(G^T\boldsymbol{z} \times B\boldsymbol{y})$.*

Theorem 3.7 (Winograd (1970)) *Let $X\boldsymbol{y}$ be a system of bilinear forms over F, where X is an $(k \times d)$ matrix whose elements are linear forms in the indeterminants $\{x_0, \ldots, x_{\tau-1}\}$ over F and $\boldsymbol{y} = (y_0, \ldots, y_{d-1})$. If the column rank of X is α, then any algorithm computing $X\boldsymbol{y}$ requires at least α multiplications.*

Theorem 3.8 (Fiduccia (1971)) *Let $X\boldsymbol{y}$ be a system of bilinear forms over F. If X has an $(\alpha \times \beta)$ submatrix S such that*

$$\begin{gathered} \boldsymbol{u}^T S\boldsymbol{v} \in F \quad \textit{iff} \quad \boldsymbol{u} = \boldsymbol{0} \quad \textit{or} \quad \boldsymbol{v} = \boldsymbol{0} \\ \forall \boldsymbol{u} \in F^{\alpha} \quad \textit{and} \quad \boldsymbol{v} \in F^{\beta} \, . \end{gathered} \tag{3.9}$$

Then, the multiplicative complexity of $X\boldsymbol{y}$ is at least $\alpha + \beta - 1$.

Finally in this section we give the details about another way of writing the computation, as defined in Definitions 3.1 and 3.4. If we have

$$\boldsymbol{\theta} = G(A\boldsymbol{x} \times B\boldsymbol{y}) \tag{3.10}$$

then we can expand the right hand side to give

$$\boldsymbol{\theta} = G\boldsymbol{m} \tag{3.11}$$

where $\boldsymbol{m} = (A\boldsymbol{x} \times B\boldsymbol{y})$ and consists of elements of bilinear forms of the form

$$m_i = m_{i_0}(\boldsymbol{x}) m_{i_1}(\boldsymbol{y}) \tag{3.12}$$

and $m_{i_0}(\boldsymbol{x})$ is a linear form in $x_0, \ldots, x_{r-1}$, and similarly for m_{i_1}.

This form is introduced as it will be used quite extensively throughout the thesis to prove some interesting and useful results.

Example 3.9 Looking again at Example 3.2 we can write $\boldsymbol{\theta}$ in the form $G\boldsymbol{m}$ as follows. For (3.6) we have

$$\boldsymbol{\theta} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 y_0 \\ x_1 y_1 \\ x_0 y_1 \\ x_1 y_0 \end{bmatrix} \tag{3.13}$$

and for (3.8) we have

$$\boldsymbol{\theta} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 y_0 \\ x_1 y_1 \\ (x_0 + x_1)(y_0 + y_1) \end{bmatrix} \tag{3.14}$$

3.3 The Relationship Between Bilinear Forms and Linear Codes

Definition 3.10 Denote the following

(i) the multiplicative complexity of the system (or expression) $\boldsymbol{\theta}$ by $M(\boldsymbol{\theta})$, and

(ii) the row rank of X by $\rho(X)$.

Theorem 3.11 (Krishna (1987)) *Let $\boldsymbol{\theta} = X\boldsymbol{y}$ be a linearly independent set of bilinear forms. The $(k \times n)$ matrix G in the computation (3.4) is the generator matrix of a linear (n, k, d') code over F where*

$$d' \geq d = \min\{\rho(\boldsymbol{u}^T X) : \boldsymbol{u} \in F^k, \ \boldsymbol{u} \neq 0\}. \tag{3.15}$$

Proof. We know that for any integer $n \geq M(\boldsymbol{\theta})$, $\boldsymbol{\theta}$ may be computed as (3.4).

If $\rho(X) = k$ then by Theorem 3.7, any computation of $\boldsymbol{\theta}$ will require at least k multiplications, i.e. $n \geq M(\boldsymbol{\theta}) \geq k$.

Further, $(A\boldsymbol{x} \times B\boldsymbol{y})$ involves n component-by-component multiplications. So the computation (3.4) has n multiplications provided all k rows of G are linearly independent. If G had any dependent rows, this computation would require fewer multiplications, hence the rank of G, $\rho(G)$ must be k.

So here we have shown that G is a $(k \times n)$ matrix where $n \geq k$ and all k rows of G are linearly independent. Thus G may be considered the generator matrix of a linear (n, k) code over F.

Now a typical codeword is $\boldsymbol{c}^T = (c_0, c_1, \ldots, c_{n-1})$ where

$$\boldsymbol{c}^T = \boldsymbol{u}^T G \quad \text{and} \quad \boldsymbol{u}^T = (u_0, u_1, \ldots, u_{k-1}).$$

We have

$$\boldsymbol{\theta} = X\boldsymbol{y} = G(A\boldsymbol{x} \times B\boldsymbol{y})$$

which implies

$$\boldsymbol{u}^T X\boldsymbol{y} = \boldsymbol{u}^T G(A\boldsymbol{x} \times B\boldsymbol{y}) \tag{3.16}$$

i.e.

$$\boldsymbol{u}^T X \boldsymbol{y} = \boldsymbol{c}^T (A\boldsymbol{x} \times B\boldsymbol{y}). \tag{3.17}$$

Consider any entry c_i of a non-zero codeword $\boldsymbol{c}$ $(i = 0, 1, \ldots, n-1)$. Observe if

i) $c_i \neq 0$ then i^{th} component-by-component multiplication of $(A\boldsymbol{x} \times B\boldsymbol{y})$ is necessary, while if

ii) $c_i = 0$ then i^{th} multiplication need not be done as it will disappear when multiplied by c_i.

It follows that the weight of any non-zero codeword $\boldsymbol{c}$, cannot be less than the multiplicative complexity of $\boldsymbol{c}^T(A\boldsymbol{x} \times B\boldsymbol{y})$. i.e.

$$w(\boldsymbol{c}) \geq M(\boldsymbol{c}^T(A\boldsymbol{x} \times B\boldsymbol{y})). \tag{3.18}$$

Further, by (3.17) the multiplicative complexity of $\boldsymbol{c}^T(A\boldsymbol{x} \times B\boldsymbol{y})$ is equal to the multiplicative complexity of $\boldsymbol{u}^T X \boldsymbol{y}$, i.e.

$$M(\boldsymbol{c}^T(A\boldsymbol{x} \times B\boldsymbol{y})) = M(\boldsymbol{u}^T X \boldsymbol{y}). \tag{3.19}$$

Now by Theorem 3.7, the number of multiplications necessary to compute $\boldsymbol{u}^T X \boldsymbol{y}$ is greater than or equal to the row rank of $\boldsymbol{u}^T X$. i.e.

$$M(\boldsymbol{u}^T X \boldsymbol{y}) \geq \rho(\boldsymbol{u}^T X) \tag{3.20}$$

Hence we have

$$w(\boldsymbol{c}) \geq M(\boldsymbol{c}^T(A\boldsymbol{x} \times B\boldsymbol{y})) = M(\boldsymbol{u}^T X \boldsymbol{y}) \geq \rho(\boldsymbol{u}^T X). \tag{3.21}$$

□

3.4 A Form Directly Related to Polynomial Multiplication

Consider the multiplication $\Phi(u)$ of two polynomials, $Z(u)$ of degree $k-1$ and $Y(u)$ of degree $d-1$, i.e.

$$\begin{aligned} \Phi(u) &= Z(u)Y(u) \\ &= z_0y_0 + (z_1y_0 + z_0y_1)u + \cdots + z_{k-1}y_{d-1}u^{k+d-2}. \end{aligned} \tag{3.22}$$

If we look at the coefficients of $\Phi(u)$ as the vector:

$$\mathbf{\Phi} = [\phi_0\phi_1 \ldots \phi_{k+d-2}]^T \tag{3.23}$$

then

$$\begin{aligned} \phi_0 &= z_0y_0 \\ \phi_1 &= z_1y_0 + z_0y_1 \\ &\vdots \\ \phi_i &= \textstyle\sum_{j+k=i} z_jy_k \\ &\vdots \\ \phi_{k+d-2} &= z_{k-1}y_{d-1} \end{aligned} \tag{3.24}$$

and we can write this is the matrix form

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{k+d-2} \end{bmatrix} = Z\boldsymbol{y} = \begin{bmatrix} z_0 & 0 & \cdots & 0 \\ z_1 & z_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ z_{k-1} & z_{k-2} & z_{k-3} & \cdots \\ 0 & z_{k-1} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & z_0 & 0 \\ \vdots & \vdots & \vdots & z_0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & z_{k-1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{d-1} \end{bmatrix}. \tag{3.25}$$

The following, based on Lempel and Winograd (1987), will expose the link between polynomial multiplication and bilinear forms, and further to linear codes.

We have here a system of bilinear forms $\Phi = Zy = A^T m$, say, where A^T is a $((k+d-1) \times n)$ matrix with elements in F, and m is an n-vector, (n here is the same multiplicative complexity as we introduced in Definition 3.3) with elements of the form $m_i = \pi_i \sigma_i$, such that π_i is a linear form in the entries of $z = (z_0, \ldots, z_{k-1})^T$ and σ_i is a linear form in the entries of $y = (y_0, \ldots, y_{d-1})^T$. We can thus write

$$\begin{aligned} \pi_i &= P_i z, \quad P_i \in F^k \\ \sigma_i &= S_i y, \quad S_i \in F^d. \end{aligned} \tag{3.26}$$

Now, let $\Delta(z)$ denote the $(n \times n)$ diagonal matrix with $(\Delta(z))_{ii} = P_i z$, and let S denote the $(n \times d)$ matrix whose ith row is S_i. Then the algorithm $A^T m$ can be written as

$$A^T m = A^T \Delta(z) S y, \tag{3.27}$$

and since this holds for every y it follows that we have the following decomposition for Z

$$Z = A^T \Delta(z) S. \tag{3.28}$$

Now let $\phi_0, \ldots, \phi_{k-1}$ denote the k bilinear forms of the system $\Phi = Zy$, let $x = (x_0, \ldots, x_{k-1})$, and form the trilinear form (also known as the *defining polynomial*)

$$x^T Z y = \sum_{i=0}^{k-1} x_i \phi_i. \tag{3.29}$$

From this form we can extract a dual system $\theta = Xy$ of k bilinear forms in the entries of x and y which satisfy

$$x^T Z y = \sum_{i=0}^{k-1} z_i \theta_i = z^T X y \tag{3.30}$$

and where X is a $(k \times d)$ matrix whose entries are linear forms of the elements of x.

We can form a similar decomposition to the Z in (3.28) for the X as follows. From (3.30) we see that

$$x^T Z = z^T X \tag{3.31}$$

i.e.

$$x^T A^T \Delta(z) S = z^T X \tag{3.32}$$

Looking at the full version of $\Delta(\boldsymbol{z})$ we see that

$$\boldsymbol{x}^T U \begin{bmatrix} \boldsymbol{P}_0\boldsymbol{z} & 0 & \cdots & 0 \\ 0 & \boldsymbol{P}_1\boldsymbol{z} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & \boldsymbol{P}_{n-1}\boldsymbol{z} \end{bmatrix} S = \boldsymbol{z}^T X, \tag{3.33}$$

i.e. if $A^T = [\boldsymbol{A}_0 \ldots \boldsymbol{A}_{n-1}]$ then

$$\begin{bmatrix} (\boldsymbol{x}^T\boldsymbol{A}_0)(\boldsymbol{P}_0\boldsymbol{z}) & (\boldsymbol{x}^T\boldsymbol{A}_1)(\boldsymbol{P}_1\boldsymbol{z}) & \ldots & (\boldsymbol{x}^T\boldsymbol{A}_{n-1})(\boldsymbol{P}_{n-1}\boldsymbol{z}) \end{bmatrix} S = \boldsymbol{z}^T X, \tag{3.34}$$

i.e.

$$\boldsymbol{z}^T P \begin{bmatrix} \boldsymbol{x}^T\boldsymbol{A}_0 & 0 & \ldots & 0 \\ 0 & \boldsymbol{x}^T\boldsymbol{A}_1 & \ldots & 0 \\ \ldots & & \ddots & \ldots \\ 0 & 0 & \ldots & \boldsymbol{x}^T\boldsymbol{A}_{n-1} \end{bmatrix} S = \boldsymbol{z}^T X, \tag{3.35}$$

i.e.

$$X = PD(\boldsymbol{x})S, \tag{3.36}$$

where $\boldsymbol{P}_i$ is the ith column of P, and $(D(\boldsymbol{x}))_{ii} = \boldsymbol{x}^T\boldsymbol{A}_i$.

As X is uniquely defined, we can see that the multiplicative complexity of $\boldsymbol{\theta} = X\boldsymbol{y}$ is the same as the multiplicative complexity of $\Phi = Z\boldsymbol{y}$. Looking again at the matrix Z in (3.25) we would like to obtain the decomposition of Z in the form (3.28), i.e.

$$Z = A^T\Delta(\boldsymbol{z})S. \tag{3.37}$$

to obtain the decomposition of X and hence X. We can alternatively find X itself from the defining function as follows. We know

$$\phi_i = \sum_{j=0}^{i} z_j y_{i-j}, \quad i = 0, \ldots, k+d-2 \tag{3.38}$$

so

$$\begin{aligned} x^T Z y &= \textstyle\sum_{i=0}^{k+d-2} x_i \phi_i \\ &= \textstyle\sum_{i=0}^{k+d-2} x_i \sum_{j=0}^{i} z_j y_{i-j} \\ &= \textstyle\sum_{i=0}^{k+d-2} z_i \sum_{j=0}^{k+d-2} x_{i+j} y_j, \end{aligned} \tag{3.39}$$

where $z_i = 0$, $i > k-1$ and $y_j = 0$, $j > d-1$. This can be written as

$$x^T Z y = \sum_{i=0}^{k-1} z_i \theta_i, \tag{3.40}$$

where

$$\theta_i = \sum_{j=0}^{d-1} x_{i+j} y_j \tag{3.41}$$

and we write $\boldsymbol{\theta} = X\boldsymbol{y}$ where X is the following matrix,

$$X = \begin{bmatrix} x_0 & x_1 & \dots & x_{d-1} \\ x_1 & x_2 & \dots & x_d \\ \vdots & & & \vdots \\ x_{k-1} & x_k & \dots & x_{k+d-2} \end{bmatrix}. \tag{3.42}$$

So we have (using the decomposition of Z as in (3.28))

$$\begin{aligned} \boldsymbol{\Phi} = Z\boldsymbol{y} = A^T \boldsymbol{m} &= A^T(\Delta(\boldsymbol{z})S\boldsymbol{y}) \\ &= A^T(P^T\boldsymbol{z} \times S\boldsymbol{y}) \end{aligned} \tag{3.43}$$

and the 'dual' we have just worked out as (using the decomposition of X as in (3.36))

$$\boldsymbol{\theta} = X\boldsymbol{y} = P(D(\boldsymbol{x})S\boldsymbol{y}) = P(A\boldsymbol{x} \times S\boldsymbol{y}) \tag{3.44}$$

and so by Definition 3.4 on comparing (3.43) and (3.44) we see that $\boldsymbol{\Phi}$ is the P-dual of $\boldsymbol{\theta}$.

The following theorem together with Theorem 3.11 will prove to be foundational in linking bilinear forms and coding theory.

Theorem 3.12 *For the above form of X, $\forall \boldsymbol{a} \in F^k, \boldsymbol{a} \neq \mathbf{0}$ and $\forall \boldsymbol{b} \in F^d, \boldsymbol{b} \neq \mathbf{0}$, we have $\boldsymbol{a}^T X \boldsymbol{b} \neq \mathbf{0}$. Moreover $\rho(\boldsymbol{a}^T X) = d$ for all $\boldsymbol{a} \in F^k$, $\boldsymbol{a} \neq \mathbf{0}$.*

Proof. Look at $\boldsymbol{a}^T X \boldsymbol{b}$. This can be written

$$\boldsymbol{a}^T X \boldsymbol{b} = \sum_{j=0}^{d-1} \left(\sum_{i=0}^{k-1} a_i x_{i+j} \right) b_j = \sum_{i,j} a_i b_j x_{i+j} = \sum_{r=0}^{k+d-2} \left(\sum_{i+j=r} a_i b_j \right) x_r \ . \tag{3.45}$$

Now consider the polynomial product

$$\left(\sum a_i y^i \right) \left(\sum b_j y^j \right) = \sum_{r=0}^{k+d-2} \left(\sum_{i+j=r} a_i b_j \right) y^r \ . \tag{3.46}$$

Notice the coefficient of x_i in (3.45) is the same as the coefficient of y^i in (3.46), for $i = 0, \ldots, k+d-2$.

Now if $\boldsymbol{a} \neq \boldsymbol{0}$ and $\boldsymbol{b} \neq \boldsymbol{0}$, then we have from (3.46) the polynomial product of two non-zero polynomials which equals a non-zero polynomial, and so at least one of its coefficient is non-zero and $\boldsymbol{a}^T X \boldsymbol{b} \neq 0$.

But $\boldsymbol{a}^T X \boldsymbol{b}$ is an arbitrary linear combination of the columns of $\boldsymbol{a}^T X$ with not all coefficients zero, so $\boldsymbol{a}^T X$ has independent columns, hence the rank is d. □

Hence, if $G(Ax \times By)$ is a computation of a system of bilinear forms, $\boldsymbol{\theta} = X\boldsymbol{y}$ then G generates a $(n, k, \bar{d})$ linear code over F with $\bar{d} \geq d$, n the multiplicative complexity of the computation, which by Theorem 3.8 is at least $k + d - 1$.

Chapter 4

Algorithms for Polynomial Multiplication

4.1 Introduction

In this chapter we will introduce and expand on the idea of general algorithms for the multiplication of two polynomials $Z(u)$ and $Y(u)$. Usually the degrees of the two polynomials will be the same (taken as $\eta - 1$), but occasionally (when stated) they will be different, in which case deg $[Z(u)] = k - 1$ and deg $[Y(u)] = d - 1$.

Of course algorithms that are developed for deg$[Z(u)] =$ deg$[Y(u)]$ can easily be amended to algorithms for which deg$[Z(u)] \neq$ deg$[Y(u)]$, by setting the relevant coefficients to zero.

Example 4.1 If we have an algorithm for $\Phi(u) = Z(u)Y(u)$ where deg$[Z(u)] =$ deg$[Y(u)]$, then to obtain the algorithm for deg$[Z(u)] = k - 1$ and deg$[Y(u)] = d - 1$ (where $k > d$) we simply set

$$y_d = y_{d+1} = \cdots = y_{k-1} = 0$$

and as such some of the multiplications from the original algorithm may get set to zero and are therefore not needed. This method is similar when $d > k$, in this case we set the relevant z_i's to zero.

Note: When we say that an algorithm X is *better* than an algorithm Y, we shall mean that

using algorithm X requires less multiplications to achieve the required reconstruction than it would using algorithm Y.

4.2 General Background

The most general case that we can consider is when we are trying to find the multiplications to form a valid algorithm for the product

$$\Phi(u) = Z(u)Y(u) \tag{4.1}$$

where deg $[Z(u)] = k-1$ and deg $[Y(u)] = d-1$. We have seen in the previous chapter that since we are trying to find the coefficients of $\Phi(u)$ the problem can equivalently written as

$$\begin{aligned} \mathbf{\Phi} &= Z\boldsymbol{y} \\ &= A^T\boldsymbol{m} \\ &= A^T(G^T\boldsymbol{z} \times B\boldsymbol{y}) \end{aligned} \tag{4.2}$$

where the coefficients of $\Phi(u)$ are the elements of $\mathbf{\Phi}$. It is the matrix G^T we are interested in, as by Theorem 3.11, G is the generator matrix of a KM (n, k, d) code, where n is the multiplicative complexity, k is 1+deg $[Z(u)]$, and d is 1+deg $[Y(u)]$. While for most purposes we would like n to be as small as possible, it is sometimes necessary to give n a particular value. For example, the length may need to be a multiple of the dimension. For this reason optimal algorithms are not always required. We therefore seek a method of obtaining $\boldsymbol{m}$ and then G, since $\boldsymbol{m}$ uniquely defines G^T and B (and vice versa) in (4.2), or alternatively obtaining G^T directly. In the following we are concerned with the problem of finding $\boldsymbol{m}$, as this subject is of interest to many authors.

First of all we introduce some lower bounds on n, the minimum number of multiplications necessary for certain polynomial multiplication.

4.3 Lower Bounds on n

There are essentially two types of lower bounds:

1. finite lower bounds – these are true for all finite values of the variable, e.g. $M \geq 3\eta$ means M is greater than or equal to 3η for all finite η.
2. asymptotic lower bounds – these are true only as the variable tends to infinity, e.g. $M \geq 3\eta$, $\eta \to \infty$ means M is greater than or equal to 3η only for sufficiently large η, it does not say anything about M for small values of η.

Our interest in this thesis lies with finite lower bounds only as we would like to be able to construct algorithms of a practical use.

We now give some known lower bounds over $\mathbb{F}_q$, and some over $\mathbb{F}_2$ only. First we must introduce some definitions.

Definition 4.2 (i) Let $M_q(\eta-1)$ denote the number of multiplications needed to multiply two degree $\eta-1$ polynomials, over $\mathbb{F}_q$.
(ii) Let $M_q(\eta_1-1,\eta_2-1)$ denote the number of multiplications needed to multiply a degree η_1-1 polynomial by a degree η_2-1 polynomial, over $\mathbb{F}_q$.
(iii) Let $M_q(\eta-1,P(u))$ denote the number of multiplications needed to multiply two degree $\eta-1$ polynomials and reduce the the product modulo $P(u)$, over $\mathbb{F}_q$.
(iv) The function $f(x)=o(g(x))$ as x tends to infinity, means $\frac{f(x)}{g(x)} \to 0$ as $x \to \infty$.

Of course $M_q(\eta-1) \geq M_q(\eta-1,P(u))$, so the lower bounds for $M_q(\eta-1,P(u))$ are extremely useful. Winograd (1977) and then more explicitly Averbuch, Galil and Winograd (1986) proved that

$$M_q(\eta-1) > 2\eta-1 \quad \text{if} \quad q < 2\eta-2 \tag{4.3}$$

so over $\mathbb{F}_2$ this provides a relatively weak, but still useful bound. Lempel, Seroussi and Winograd (1983) prove that

$$M_q(\eta-1,P(u)) \geq \left(2+\frac{1}{q-1}\right)\eta - \left(\left\lceil \log_q\left(\frac{q\eta}{(q-1)\log_q e}\right)\right\rceil + \log_q e\right) \tag{4.4}$$

and so over $\mathbb{F}_2$ we have

$$M_2(\eta - 1, P(u)) \geq 3\eta - \left\lceil \frac{\ln 2\eta \ln 2}{\ln 2} \right\rceil - \frac{1}{\ln 2} \tag{4.5}$$

for multiplication of two degree $\eta - 1$ polynomials modulo an irreducible polynomial $P(u)$ of degree η. Bshouty (1992) proves that a similar result

$$M_q(\eta - 1, P(u)) \geq \left(2 + \frac{1}{q-1}\right)\eta - o(\eta) \quad \text{as } x \to \infty \tag{4.6}$$

applies for any polynomial $P(u)$ of degree η. Chudnovsky and Chudnovsky (1988) prove that

$$M_q(\eta - 1) \geq \frac{(\eta - 1)(q^{\eta - i - 1} - 1)q^{-\eta + i + 2}}{q - 1}, \tag{4.7}$$

for $i = 0, \ldots, \eta$, which provides for the field $\mathbb{F}_2$

$$M_2(\eta - 1) \geq (\eta - 1)(2^{\eta - i - 1} - 1)2^{-\eta + i + 2}, \tag{4.8}$$

for $i = 0, \ldots, \eta$.

4.4 A New Lower Bound for $M_2(\eta - 1, u^\eta)$

We provide here a new lower bound for $M_2(\eta-1, u^\eta)$, that proves useful when assessing how good some of the algorithms are in the next sections. The reason that we are concerned with modulo u^η stems from study into the wraparound part of the Improved CRT (see Chapter 2, Section 5). The proof is based on Winograd (1977), but amended for the problem.

Let

$$\boldsymbol{\theta} = \Upsilon(\boldsymbol{x})\boldsymbol{y} \tag{4.9}$$

be the system of bilinear forms of coefficients of the polynomial product

$$\theta(u) = X(u)Y(u) \pmod{u^\eta}, \tag{4.10}$$

where $X(u) = x_0 + \cdots + x_{\eta-1}u^{\eta-1}$, and $Y(u) = y_0 + \cdots + y_{\eta-1}u^{\eta-1}$.

Definition 4.3 Define the *companion matrix* of a polynomial $P(u) = p_0 + p_1 u + \cdots + p_{\eta-1}u^{\eta-1} + u^\eta$ to be the matrix

$$C_P = \begin{bmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -p_{\eta-2} \\ 0 & 0 & \cdots & 1 & -p_{\eta-1} \end{bmatrix} \tag{4.11}$$

Lemma 4.4 *If the elements of the vector $\Upsilon(\boldsymbol{x})\boldsymbol{y}$ of bilinear forms are the coefficients of*

$$\left(\sum_{i=0}^{\eta-1} x_i u^i\right)\left(\sum_{j=0}^{\eta-1} y_j u^j\right) \pmod{P(u)} \tag{4.12}$$

where the degree of $P(u)$ is η, then

$$\Upsilon(\boldsymbol{x}) = \left[\ \boldsymbol{x}\ \middle|\ C_P\boldsymbol{x}\ \middle|\ \ldots\ \middle|\ C_P^k\boldsymbol{x}\ \middle|\ \ldots\ \middle|\ C_P^{\eta-1}\boldsymbol{x}\ \right]. \tag{4.13}$$

Proof. We prove by induction, first the intial case, let $k = 0$, $C^0\boldsymbol{x} = \boldsymbol{x}$. Now the case $k = 1$,

$$\begin{aligned} C\boldsymbol{x} &= -p_0x^{\eta-1} + (1 - p_1x^{\eta-1}) + (x - p_2x^{\eta-1}) + \cdots + (x^{\eta-2} - p_{\eta-1}x^{\eta-1}) \\ &= 1 + x + \cdots + x^{\eta-2} - (p_0 + p_1 + \cdots + p_{\eta-1})x^{\eta-1}). \end{aligned} \tag{4.14}$$

Now let $X(u)u^k = t_0 + t_1u + \cdots + t_{\eta-1}u^{\eta-1} \pmod{P(u)}$ with $[t_0\ t_1\ \ldots t_{\eta-1}]^T = C_P^k\boldsymbol{x}$, then

$$\begin{aligned} C_P^{k+1} &= C_PC_P^k\boldsymbol{x} \\ &= C_P[t_0\ t_1\ \ldots t_{\eta-1}]^T \\ &= \begin{bmatrix} -p_0t_{\eta-1} \\ t_0 - p_1t_{\eta-1} \\ t_1 - p_2t_{\eta-1} \\ \vdots \\ t_{\eta-2} - p_{\eta-1}t_{\eta-1} \end{bmatrix} \end{aligned} \tag{4.15}$$

and

$$\begin{aligned} X(u)u^{k+1} &= (X(u)u^k)u \\ &= (t_0 + t_1u + \cdots + t_{\eta-1}u^{\eta-1})u \\ &= t_0u + t_1u^2 + \cdots + t_{\eta-1}u^{\eta} \\ &= -p_0t_{\eta-1} + (t_0 - p_1t_{\eta-1})u + \cdots + (t_{\eta-2} - p_{\eta-1}t_{\eta-1})u^{\eta-1} \pmod{P(u)} \end{aligned} \tag{4.16}$$

as expected. □

Lemma 4.5 (Column Rank Lemma) *Let $\Upsilon(\boldsymbol{x})$ be as in (4.9) and $\boldsymbol{v}$ be any non-zero vector then $\boldsymbol{v}\Upsilon(\boldsymbol{x})$ has column rank $max\{i : v_i \neq 0\}$ over $\mathbb{F}_q$.*

Proof. By Lemma 4.4, $\boldsymbol{v}\Upsilon(\boldsymbol{x})$ has columns $\boldsymbol{v}C^i\boldsymbol{x}$, where C is the companion matrix of u^η, i.e.

$$\Upsilon(\boldsymbol{x}) = \left[\ \boldsymbol{x}\ \middle|\ C\boldsymbol{x}\ \middle|\ C^2\boldsymbol{x}\ \middle|\ \cdots\ \middle|\ C^{\eta-1}\boldsymbol{x}\ \right].$$

Now let $\boldsymbol{\lambda}$ be a non-zero η - vector. Then

$$\begin{aligned} \sum_{i=0}^{\eta-1} \lambda_i\boldsymbol{v}C^i\boldsymbol{x} &= \left(\sum_{i=0}^{\eta-1} \lambda_i\boldsymbol{v}C^i\right)\boldsymbol{x} \\ &= 0 \qquad \iff \sum_{i=0}^{\eta-1} \lambda_i\boldsymbol{v}C^i = 0. \end{aligned} \tag{4.17}$$

If we look at the matrix with rows $\boldsymbol{v}C^i$ then we get

$$\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{v}C \\ \vdots \\ \boldsymbol{v}C^{\eta-1} \end{bmatrix} = \begin{bmatrix} v_0 & v_1 & \dots & v_{\eta-2} & v_{\eta-1} \\ v_1 & v_2 & \dots & v_{\eta-1} & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ v_{\eta-1} & 0 & \dots & 0 & 0 \end{bmatrix} \tag{4.18}$$

and the rank of this matrix is what we require. □

Lemma 4.6 (Weight Lemma) *Let $\boldsymbol{u}$, $\boldsymbol{v}$ be k - vectors over $\mathbb{F}_2$. If $wt\,(\boldsymbol{u}) \geq k - r$ then $wt\,(\boldsymbol{u}+\boldsymbol{v}) \leq k + r - wt\,(\boldsymbol{v})$.*

Proof. Without loss of generality let $\boldsymbol{u} = [1\ \cdots\ 1\ 0\ \cdots\ 0]$, with the number of 1's being $k - r + \epsilon$, and $\boldsymbol{v} = [0\ \cdots\ 0\ 1\ \cdots\ 1]$ be the worst case (i.e. the overlap of 1's is the biggest, and no zeros overlap), then overlap is $\mathrm{wt}(\boldsymbol{v}) - (r - \epsilon)$, so

$$\begin{aligned} \mathrm{wt}(\boldsymbol{u}+\boldsymbol{v}) &\leq k - \mathrm{wt}(\boldsymbol{v}) + (r - \epsilon) \\ &\leq k + r - \mathrm{wt}(\boldsymbol{v}) \end{aligned} \tag{4.19}$$

as $\epsilon \geq 0$. □

Let $\boldsymbol{\theta} = \Upsilon(\boldsymbol{x})\boldsymbol{y}$ be the system of bilinear forms as in (4.9). In the same way as explained in Section 3.2 we may put this in the form of a computation and further in the form

$$\boldsymbol{\theta} = A^T\boldsymbol{m}. \tag{4.20}$$

Let $\boldsymbol{m}$ be of length n, and be the least n such that an A^T exists. Now A^T is $(\eta \times n)$. As the η bilinear forms $\boldsymbol{\theta}$ are linearly independent then the matrix A^T is of rank η, therefore by rearrangement, if necessary, of the columns of A^T and the elements of $\boldsymbol{m}$, we may have the first η columns of A^T being linearly independent. Thus there exists a matrix W, say, such that

$$W\Upsilon(\boldsymbol{x})\boldsymbol{y} = WA^T\boldsymbol{m} = \left[I \,\middle|\, (A')^T\right]\boldsymbol{m}. \tag{4.21}$$

Definition 4.7 Let (i) $\boldsymbol{a}'_i$ =row i of $(A')^T$,
(ii) $V = V_s$ =span $\{\boldsymbol{e}_0, \dots, \boldsymbol{e}_{s-1}\}$, $\boldsymbol{e}_i = (0, \dots, 0, 1, 0, \dots, 0) \in \mathbb{F}_{2^n}$, with the 1 in position i,

(iii) $CR =$ column rank,

(iv) $n = \eta + k$,

(v) $s = 2k - 2\eta + 4$.

Theorem 4.8 *Let $\boldsymbol{v}$ =row i of W. (a) if $v_\eta = 1$ then wt $(\boldsymbol{a}_i') \geq \eta - 1$, (b) if $v_\eta = 0$ then wt $(\boldsymbol{a}_i') \leq s - 1$ and $\boldsymbol{v} \in V_s$.*

Proof. (a) We have

$$\boldsymbol{v}\Upsilon(\boldsymbol{x})\boldsymbol{y} = \left[\begin{array}{c|c} \boldsymbol{e}_i & \boldsymbol{a}_i' \end{array} \right] \boldsymbol{m}. \tag{4.22}$$

Therefore

$$\begin{aligned} 1 + \text{wt } (\boldsymbol{a}_i') &= \text{MC } (\boldsymbol{v}\Upsilon(\boldsymbol{x})\boldsymbol{y}) \\ &\geq \text{CR } (\boldsymbol{v}\Upsilon(\boldsymbol{x})) \\ &= \eta, \end{aligned} \tag{4.23}$$

by Column Rank Lemma (Lemma 4.5).

(b) W is non-singular so some row j (say $\boldsymbol{w}$) has $w_\eta = 1$. Now as $v_\eta = 0$, we have

$$(\boldsymbol{v} + \boldsymbol{w})_\eta = 1 \tag{4.24}$$

and

$$(\boldsymbol{v} + \boldsymbol{w})\Upsilon(\boldsymbol{x})\boldsymbol{y} = \left[\begin{array}{ccccccc|c} 0 & \ldots & 1 & \ldots & 1 & \ldots & 0 & \boldsymbol{a}_i' + \boldsymbol{a}_j' \end{array} \right] \boldsymbol{m}.$$

But, by part (a) and (4.24)

$$\text{wt } (\boldsymbol{a}_i' + \boldsymbol{a}_j') \geq \eta - 2 \tag{4.25}$$

By (a) we have $\text{wt}(\boldsymbol{a}_j') \geq \eta - 1$ as $w_\eta = 1$. Now putting this in the form of Lemma 4.6, i.e. $\text{wt}(\boldsymbol{a}_j') = \text{length} - r$, we have

$$\text{wt}(\boldsymbol{a}_j') \geq k - (k - \eta + 1)$$

Now using Lemma 4.6 we have

$$\text{wt}(\boldsymbol{a}_i' + \boldsymbol{a}_j') \leq k + (k - \eta + 1) - \text{wt}(\boldsymbol{a}_i'). \tag{4.26}$$

Therefore substituting (4.25) into (4.26) and rearranging we get

$$\mathrm{wt}(\boldsymbol{a}_i') \leq 2k - 2n + 3 = s - 1.$$

Now

$$\boldsymbol{v}\Upsilon(\boldsymbol{x})\boldsymbol{y} = \left[\begin{array}{c|c} \boldsymbol{e}_i & \boldsymbol{u}_i' \end{array} \right] \boldsymbol{m},$$

so, by Theorem 3.7

$$s \geq \mathrm{CR}\,(\boldsymbol{v}\Upsilon(\boldsymbol{x})).$$

Therefore, the biggest i such that $v_i = 1$ is s. So, by Lemma 4.5 $\boldsymbol{v} \in V_s$. □

Theorem 4.9 *If rows* $\boldsymbol{v}$, $\boldsymbol{w} \in W$ *end in* 1, *then*

$$\boldsymbol{v} + \boldsymbol{w} \in V_s\,.$$

Proof. First of all we write

$$(\boldsymbol{v} + \boldsymbol{w})\Upsilon(\boldsymbol{x})\boldsymbol{y} = \left[\begin{array}{c|c} \boldsymbol{e}_i' + \boldsymbol{e}_j' & \boldsymbol{a}_i' + \boldsymbol{a}_j' \end{array} \right] \boldsymbol{m}. \tag{4.27}$$

Now by Theorem 4.8, $\mathrm{wt}(\boldsymbol{a}_i')$,$\mathrm{wt}(\boldsymbol{a}_j') \geq \eta - 1$. Hence as

$$\mathrm{wt}(\boldsymbol{a}_i') \geq \eta - 1 = k - (k - \eta + 1)$$

then

$$\begin{aligned} \mathrm{wt}\,(\boldsymbol{a}_i' + \boldsymbol{a}_j') &\leq k + (k - \eta + 1) - \mathrm{wt}\,(\boldsymbol{a}_j') \\ &\leq k + (k - \eta + 1) - (\eta - 1) \\ &= 2k - 2\eta + 2 \\ &= s - 2. \end{aligned} \tag{4.28}$$

Therefore $s \geq \mathrm{CR}\,((\boldsymbol{v} + \boldsymbol{w})\Upsilon(\boldsymbol{x}))$, and as in Theorem 4.8 we have

$$\boldsymbol{v} + \boldsymbol{w} \in V_s.$$

□

Theorem 4.10 *The value n, the minimum number of multiplications for the polynomial product $\theta(u) = X(u)Y(u) \pmod{u^\eta}$, where $X(u) = x_0 + \cdots + x_{\eta-1}u^{\eta-1}$, and $Y(u) = y_0 + \cdots + y_{\eta-1}u^{\eta-1}$, and $\boldsymbol{\theta} = \Upsilon(\boldsymbol{x})\boldsymbol{y} = A^T\boldsymbol{m}$, $\boldsymbol{m}$ being a n-vector is bounded as follows*

$$n \geq \frac{5}{2}(\eta - 1).$$

Proof. Let the rows of W ending in 0 be $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p$, and let the rows of W ending in 1 be $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_q$. So $p + q + 1 = \eta$.

Now $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p \in V_s$, by Theorem 4.8. Similarly $\boldsymbol{v}_0 + \boldsymbol{v}_1, \ldots, \boldsymbol{v}_0 + \boldsymbol{v}_q \in V_s$ by Theorem 4.9. Now because the rows of W are linearly independent, these $p + q$ vectors are distinct and linearly independent, thus using Definition 4.7 (v)

$$p + q \leq \dim V_s = s = 2k - 2\eta + 4.$$

Combining this with $p + q = \eta - 1$ gives $n - 1 \leq 2k - 2\eta + 4$. Substituting $n = \eta + k$ from Definition 4.7 (iv) we get $\eta - 1 \leq 2n - 4\eta + 4$ and the result follows by rearrangement. □

4.5 A New Diagrammatic Representation

We have seen in the previous chapter that one way of describing an algorithm is of the form:

$$\Phi = A^T m, \tag{4.29}$$

in other words we have a set of multiplications $m_0, \ldots, m_{n-1}$ and each row i of A^T will pick out the necessary multiplications to form the ith coefficient of $\Phi(u)$.

We now introduce a slightly different way of displaying the vector m in a diagrammatic form. We are interested less in the matrix A^T (as long as one exists of course) as it is the m directly that enables us to obtain the generator matrix G as explained in Section 4.2, so for the rest of this section we will refer to an algorithm as the vector m, whether it is in vector form or the diagrammatic form to be explained.

If we plot a (z, y) grid where the z values range from 0 to the degree of $Z(u)$ and similarly for the y values, and mark each integer point we get Figure 4.1.

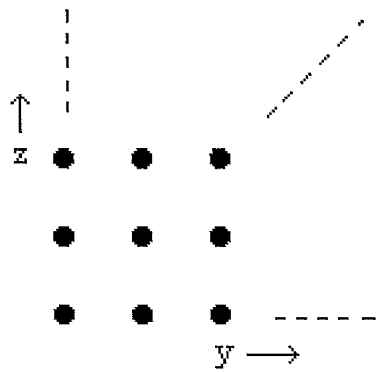

Figure 4.1: The (z, y) grid for algorithm representation

Now if we label the bottom left point z_0y_0, the next point right z_0y_1 and generally label point (i, j) as z_iy_j, it is easy to see that the coefficients of $\Phi(u) = Z(u)Y(u)$ are simply the left-right-down diagonals, as in Figure 4.2.

Multiplications can then be represented by circling the appropriate points. For example to represent $(z_0+z_1)y_0$ we would circle z_0y_0 and z_1y_0 and join them together, as in Figure 4.3.

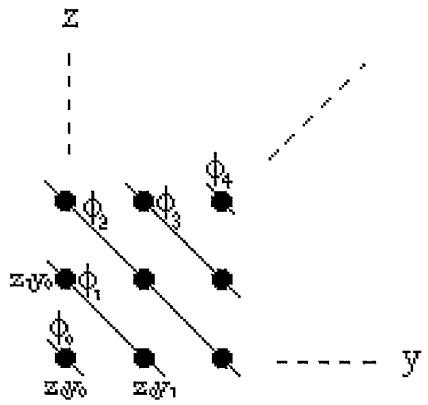

Figure 4.2: The ϕ_i as represented on the (z, y) grid

From now on we will omit the z and y labels on the axes, as they are not needed. For more complex multiplications (and algorithms) we can simply join two multiplications by a line as in Figure 4.4.

This method of displaying can be easily extended to any multiplication and indeed any algorithm, however complex, for example consider the multiplication $m = (z_0 + z_2 + z_3)(y_0 + y_2)$, then this can be represented as in Figure 4.5.

We now give an example to show the diagrammatic representation of an algorithm.

Example 4.11 Let $Z(u) = z_0 + z_1u + z_2u^2$ and $Y(u) = y_0 + y_1u$, with $\Phi(u) = Z(u)Y(u)$. Then we can have the following multiplications

$$\begin{array}{lcl|lcl} m_0 & = & z_0y_0 & m_3 & = & (z_1 + z_2)y_1 \\ m_1 & = & z_1y_1 & m_4 & = & (z_0 + z_2)y_0 \\ m_2 & = & (z_0 + z_1)(y_0 + y_1) & & & \end{array} \tag{4.30}$$

to reconstruct the $\Phi(u)$ as follows:

$$\Phi(u) = m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_1 + m_4)u^2 + (m_1 + m_3)u^3. \tag{4.31}$$

So the algorithm can be represented in diagrammatic form as in Figure 4.6. Note here

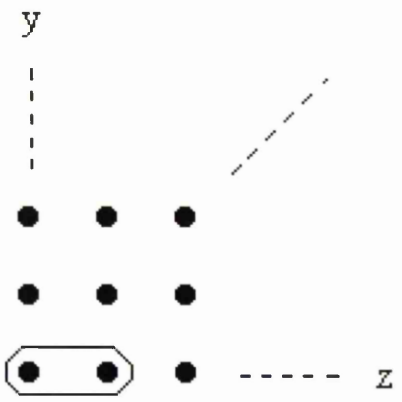

Figure 4.3: The multiplication $(z_0 + z_1)y_0$ represented in diagrammatic form

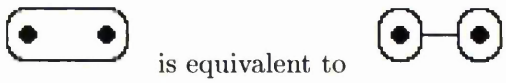

Figure 4.4: The multiplication $z_0(y_0 + y_1)$ as the join of z_0y_0 and z_0y_1 represented in diagrammatic form

for example that the multiplication z_1y_1 appears in three multiplications above and three circles below, as expected.

One of the main reasons for introducing this new way of displaying algorithms is that it gives us a handle on finding new algorithms from old ones. Basically if, for example, we had the two multiplications $m_0 = z_0y_0$ and $m_1 = (z_0 + z_1)y_0$, then we can form all the combinations of these two using $m_2 = z_1y_0$ and $m_3 = (z_0 + z_1)y_0$, or indeed $m_4 = z_0y_0$ and $m_5 = z_1y_0$, i.e.

$$\begin{array}{llllll} z_0y_0 & = & m_0 & = & m_2 + m_3 & = & m_4 \\ z_1y_0 & = & m_0 + m_1 & = & m_2 & = & m_5 \\ (z_0 + z_1)y_0 & = & m_1 & = & m_3 & = & m_4 + m_5. \end{array} \tag{4.32}$$

Definition 4.12 Consider two sets of multiplications

$$\{m_{a_1}, m_{a_2}, \ldots, m_{a_n}\} \text{ and } \{m_{b_1}, m_{b_2}, \ldots, m_{b_n}\}.$$

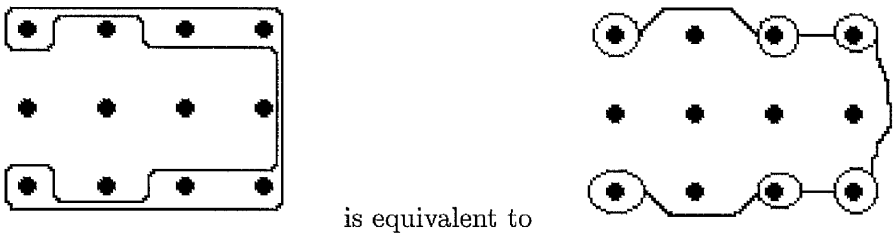

Figure 4.5: The multiplication $(z_0 + z_2 + z_3)(y_0 + y_2)$ represented in diagrammatic form

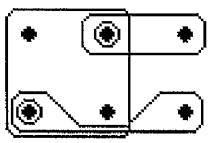

Figure 4.6: The algorithm of Example 4.11 represented in diagrammatic form

We will call these sets *equivalent* if all combinations of the elements of the first set can be made from particular combinations of the elements of the second set. In the description above we are concerned only with pairs of multiplications. We will call the operation of moving from one set of multiplications to an equivalent set *swapping.*

When we say that a set $\{m_{a_1}, m_{a_2}, \ldots, m_{a_n}\}$ is equivalent to a set $\{m_{b_1}, m_{b_2}, \ldots, m_{b_n}\}$ we mean there exists an invertible matrix $W = [w_{ij}]$ such that

$$\begin{bmatrix} m_{a_1} \\ m_{a_2} \\ \vdots \\ m_{a_n} \end{bmatrix} = W \begin{bmatrix} m_{b_1} \\ m_{b_2} \\ \vdots \\ m_{b_n} \end{bmatrix}. \tag{4.33}$$

Obviously when the number of multiplications in each set becomes large, finding whether another set is equivalent becomes quite hard. For this reason we are concerning ourselves only with sets of pairs of multiplications. Being concerned with sets of pairs of multiplications, if we label the sets $\{m_{a_1}, m_{a_2}\}$ and $\{m_{b_1}, m_{b_2}\}$ then if they are equivalent the following must hold.

$$\begin{bmatrix} m_{a_1} \\ m_{a_2} \end{bmatrix} = \begin{bmatrix} w_{00} & w_{10} \\ w_{01} & w_{11} \end{bmatrix} \begin{bmatrix} m_{b_1} \\ m_{b_2} \end{bmatrix}. \tag{4.34}$$

Now if we are working over $\mathbb{F}_2$ then there are only two possibilities for the matrix W such that the sets are distinct. So from one set of multiplications $\{m_{a_1}, m_{a_2}\}$ we get two equivalent sets $\{m_{a_1}, m_{a_1} + m_{a_2}\}$ and $\{m_{a_1} + m_{a_2}, m_{a_2}\}$.

This technique of finding equivalent multiplications is not very easy when they are not in the diagrammatic form and indeed as the multiplications get more complex so does this idea. From the diagrams however it is much more easy to see when this multiplication swapping can take place. In Figure 4.7 we have the multiplications $m_0 = z_0 y_0$ and $m_1 = z_2 y_0$ and the equivalent multiplications that will enable the algorithm to reconstruct the original polynomial.

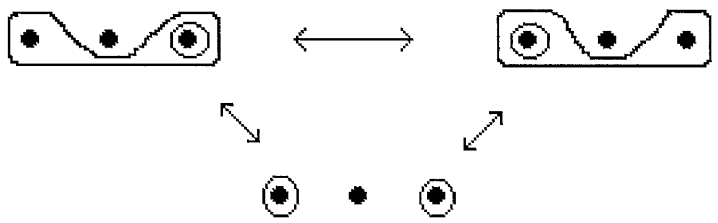

Figure 4.7: Diagrammatic representation of operation one on the multiplications

Of course the idea of multiplication swapping can be generalised and we give two generalisations in Figures 4.8 and 4.9.

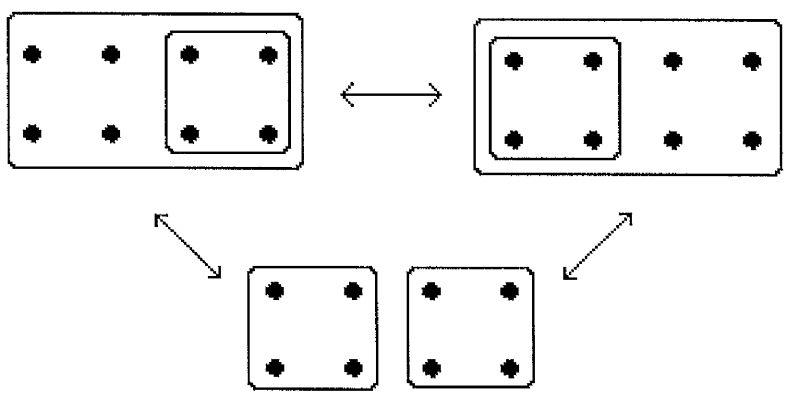

Figure 4.8: Diagrammatic representation of operation two on the multiplications

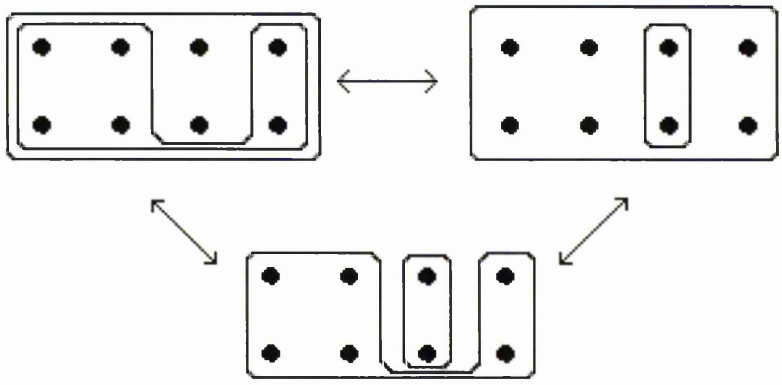

Figure 4.9: Diagrammatic representation of operation three on the multiplications

Further to the above operations we introduce the reciprocal operation. This is so because if ZY is an algorithm for Φ then $\overline{ZY}$ is an algorithm for $\overline{\Phi}$. This is represented in Figure 4.10.

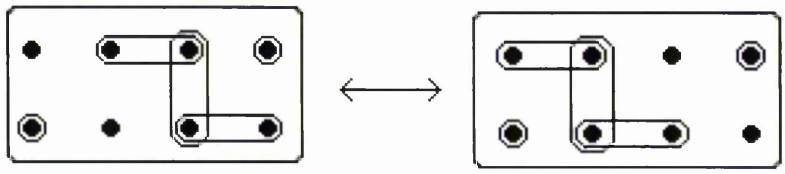

Figure 4.10: Diagrammatic representation of the reciprocal operation

With these swapping operations we can make many other algorithms from a given algorithm and this idea will be used in the next chapter to find all the KM codes (up to our technique of multiplication swapping) for a given value of k and d. There are other ways of swapping multiplications but they get very hard to generalise and so for now we will consider only the above four.

4.6 A New General Algorithm

In this section we develop a new algorithm, known as the *square, $P(u)$ algorithm*, the construction based on the diagrammatic approach introduced in the last section, and the resulting upper bound, known as the square, $P(u)$ (upper) bound.

Here we select $\eta = 1 + \deg[Z(u)] = 1 + \deg[Y(u)]$, so that if we were reducing some polynomials to $Z(u)$ and $Y(u)$ modulo $P(u)$, then $P(u)$ would have degree η. So for the simplest case, $\eta = 1$, we get $Z(u) = z_0$ and $Y(u) = y_0$, both of degree 0. The algorithm is obvious (we simply multiply z_0 by y_0) and given in Figure 4.11.

Figure 4.11: Our algorithm for $\eta = 1$

Now for the case $\eta = 2$, we get the 2×2 grid. If we place the $\eta = 1$ case in the bottom left and top right corner we can get ϕ_0 and ϕ_2, and ϕ_1 can be obtainable by simply including the multiplication $(z_0 + z_1)(y_0 + y_1)$ and we get Figure 4.12.

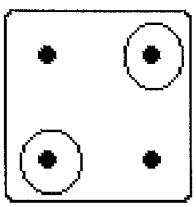

Figure 4.12: Our algorithm for $\eta = 2$

Now looking at the case $\eta = 3$, the 3×3 grid is therefore used. Placing the $\eta = 2$ case in the bottom left and top right corners we get an overlap occurring at $z_1 y_1$. Now we can get ϕ_0, ϕ_1, ϕ_3 and ϕ_4 and to obtain ϕ_2 we can simply include the multiplication $(z_0 + z_1 + z_2)(y_0 + y_1 + y_2)$ and we get the algorithm as in Figure 4.13.

For the cases $\eta = 4$ onwards we can generalise the construction of these algorithms. If we place the $\eta - 2$ algorithm in the bottom left and top right corners of the $\eta \times \eta$ grid, then we need three extra multiplications $(z_0 + z_{\eta-2})(y_0 + y_{\eta-2})$, $(z_1 + z_{\eta-1})(y_1 + y_{\eta-1})$ and

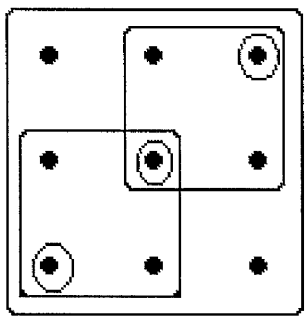

Figure 4.13: Our algorithm for $\eta = 3$

$(z_0 + z_1 + \cdots + z_{\eta-1})(y_0 + y_1 + \cdots + y_{\eta-1})$ to make up the algorithm. We therefore have the case $\eta = 4$ in Figure 4.14.

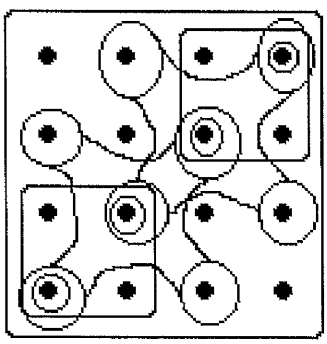

Figure 4.14: Our algorithm for $\eta = 4$

Now for the case $\eta = 5$ we get an overlap between the two $\eta = 3$ algorithms. This overlap is at the centre and is equivalent to the $\eta = 1$ algorithm. The construction is shown in Figure 4.15, giving the algorithm as in Figure 4.16.

Using this method we can easily construct the $\eta = 6$ case as shown in Figure 4.17. The enclosed part in each doted box is the algorithm for the $\eta = 4$ case. Note the overlap between the two $\eta = 4$ algorithms is the $\eta = 2$ algorithm. The full algorithm for the $\eta = 6$ case can be seen in Figure 4.18.

This construction is easily generalised for any $\eta \geq 5$ by taking the three multiplications

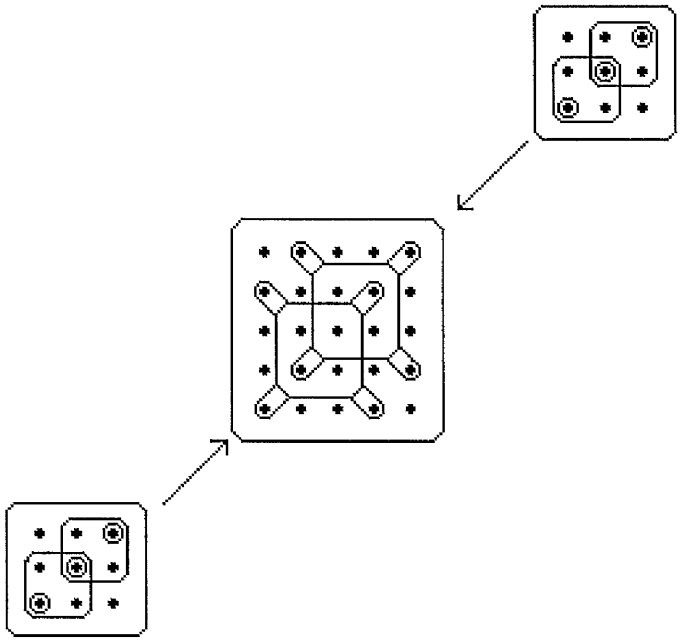

Figure 4.15: Constructing the $\eta = 5$ algorithm

$(z_0 + z_{\eta-2})(y_0 + y_{\eta-2})$, $(z_1 + z_{\eta-1})(y_1 + y_{\eta-1})$ and $(z_0 + z_1 + \cdots + z_{\eta-1})(y_0 + y_1 + \cdots + y_{\eta-1})$ and then placing on the diagram the $\eta - 2$ cases in the same way as we did with Figure 4.17 to get Figure 4.18. We can therefore get the number of multiplications needed to multiply two degree η polynomials using our algorithm, which we shall denote by $SQ_2(\eta)$, as follows:

$$SQ_2(\eta) = 2SQ_2(\eta - 2) - SQ_2(\eta - 4) + 3, \quad \eta \geq 5 \tag{4.35}$$

and if we set $SQ_2(0) = 0$ then this holds for $\eta \geq 4$.

We can solve these recurrences to get a closed form for the number of multiplications as follows:

$$SQ_2(\eta) = \begin{cases} \frac{3}{8}\eta^2 + \eta - \frac{3}{8}, & \eta \text{ odd} \\ \frac{3}{8}\eta^2 + \frac{3}{4}\eta, & \eta \text{ even.} \end{cases} \tag{4.36}$$

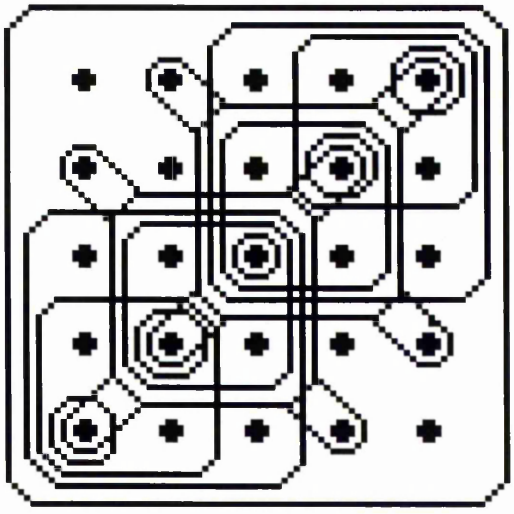

Figure 4.16: The $\eta = 5$ algorithm

We are going to prove this by induction. We have that

$$SQ_2(\eta) = 2SQ_2(\eta - 2) - SQ_2(\eta - 4) + 3 \quad , \eta \geq 4 \tag{4.37}$$

with the initial values as:

$\deg[P(u)] - 1$	1	2	3
no. mults	1	3	6

(4.38)

Now (4.36) is true for $0 \leq \eta \leq 4$, so assume true for $\eta < k$, $k > 4$, and we will look at $\eta = k$.

So for k even we have

$$\begin{aligned} SQ_2(k) &= 2SQ_2(k-2) - SQ_2(k-4) + 3 \\ &= 2.\tfrac{3}{8}(k-2)^2 + 2.\tfrac{3}{4}(k-2) \\ &\quad -\tfrac{3}{8}(k-4)^2 - \tfrac{3}{4}(k-4) + 3 \\ &= \tfrac{3}{8}k^2 + \tfrac{3}{4}k \end{aligned} \tag{4.39}$$

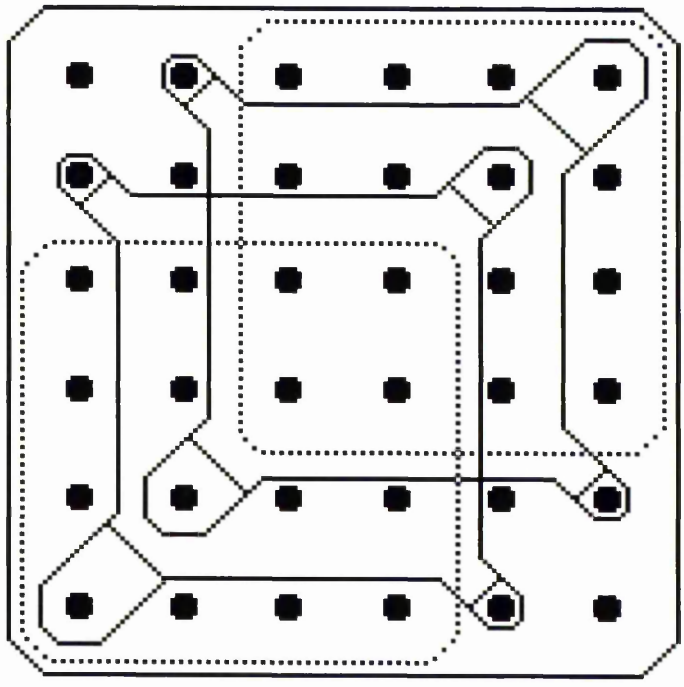

Figure 4.17: The making of the $\eta = 6$ algorithm

and for k odd we have

$$\begin{aligned} SQ_2(k) &= 2SQ_2(k-2) - 2SQ_2(k-4) + 3 \\ &= 2.\tfrac{3}{8}(k-2)^2 + 2.\tfrac{3}{4}(k-2) - 2.\tfrac{3}{8} \\ &\quad -\tfrac{3}{8}(k-4)^2 - \tfrac{3}{4}(k-4) + \tfrac{3}{8} + 3 \\ &= \tfrac{3}{8}k^2 + k - \tfrac{3}{8} \end{aligned} \tag{4.40}$$

as expected. So by induction the result (4.36) is true for all $k \geq 4$, and hence as it is true for $0 \leq \eta \leq 4$ it is true for all $k \geq 0$.

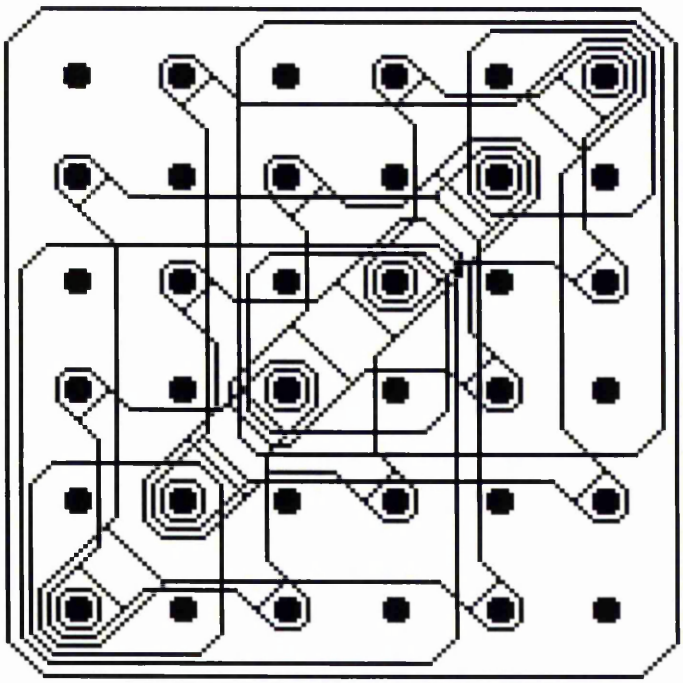

Figure 4.18: The $\eta = 6$ algorithm

4.7 Some Other Upper Bounds via Algorithms

In this section we state some existing upper bounds (algorithms) and where they originated. We first introduce what we shall call the *LSW algorithm and bound* (named after the paper Lempel, Seroussi and Winograd (1983)), and then provide two improvements to the basic procedure, namely the *LSW** and the *LSW* with wraparound.* We will also introduce the algorithm and bound that is based on the numerical approach used in Knuth (1981). In the next section we will display some graphs comparing all the upper bounds we have discussed.

4.7.1 The LSW Bounds

Lempel, Seroussi and Winograd (1983) develop an algorithm using the Chinese Remainder Theorem using ideas similar to those we were use to develop KM codes in Chapter 3. The closed bound they arrive at is not very close to the actual obtainable values so we will only consider the algorithm for small values of η. The theory can be found in the paper cited but here we give an example to demonstrate the idea fully.

Example 4.13 Suppose we want to find the number of multiplications needed to multiply two degree 3 polynomials. Using the construction of Lempel, Seroussi and Winograd (1983) we have

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 + z_3 u^3 \\ Y(u) &= y_0 + y_1 u + y_2 u^2 + y_3 u^3 \end{aligned} \tag{4.41}$$

and select $P(u) = u(u+1)(u^2+u+1)(u^3+u+1)$. Let $\Phi(u) = Z(u)Y(u)$. Then using the CRT (in the same way as Example 2.23) we get the following

i) modulo u

$$\begin{aligned} Z_1(u) &= Z(u) \pmod{u} \\ &= z_0 \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_1(u) &= Y(u) \pmod{u} \\ &= y_0. \end{aligned} \tag{4.42}$$

Let $m_0 = z_0 y_0$, then $\Phi_1(u) \equiv m_0 \pmod{u}$.

ii) modulo $u+1$

$$\begin{aligned} Z_2(u) &= Z(u) \pmod{u+1} \\ &= z_0 + z_1 + z_2 + z_3 \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_2(u) &= Y(u) \pmod{u+1} \\ &= y_0 + y_1 + y_2 + y_3. \end{aligned} \tag{4.43}$$

Let $m_1 = (z_0 + z_1 + z_2 + z_3)(y_0 + y_1 + y_2 + y_3)$, then $\Phi_2(u) \equiv m_1 \pmod{u+1}$.

iii) modulo $u^2 + u + 1$

$$\begin{array}{rcl|rcl} Z_3(u) & = & Z(u) \pmod{u^2+u+1} & Y_3(u) & = & Y(u) \pmod{u^2+u+1} \\ & = & (z_0 + z_2 + z_3) + (z_1 + z_2)u & & = & (y_0 + y_2 + y_3) + (y_1 + y_2)u. \end{array} \tag{4.44}$$

Let $m_2 = (z_0+z_2+z_3)(y_0+y_2+y_3)$, $m_3 = (z_1+z_2)(y_1+y_2)$ and $m_4 = (z_0+z_1+z_3)(y_0+y_1+y_3)$ then

$$\begin{aligned} \Phi_3(u) &\equiv m_2 + (m_2 + m_3 + m_4)u + m_3u^2 \pmod{u^2+u+1} \\ &\equiv (m_2 + m_3) + (m_2 + m_4)u \pmod{u^2+u+1}. \end{aligned} \tag{4.45}$$

iv) modulo $u^3 + u + 1$

$$\begin{array}{rcl|rcl} Z_4(u) & = & Z(u) \pmod{u^3+u+1} & Y_4(u) & = & Y(u) \pmod{u^3+u+1} \\ & = & (z_0 + z_3) + (z_1 + z_3)u + z_2u^2 & & = & (y_0 + y_3) + (y_1 + y_3)u + y_2u^2. \end{array} \tag{4.46}$$

Let

$$\begin{array}{rcl|rcl} m_5 & = & (z_0 + z_3)(y_0 + y_3) & m_8 & = & (z_0 + z_1)(y_0 + y_1) \\ m_6 & = & (z_1 + z_3)(y_1 + y_3) & m_9 & = & (z_1 + z_2 + z_3)(y_1 + y_2 + y_3) \\ m_7 & = & z_2y_2 & m_{10} & = & (z_0 + z_2 + z_3)(y_0 + y_2 + y_3) \end{array} \tag{4.47}$$

then

$$\begin{aligned} \Phi_4(u) &\equiv m_5 + (m_5 + m_6 + m_8)u + (m_5 + m_6 + m_7 + m_{10})u^2 \\ &\quad +(m_6 + m_7 + m_9)u^3 + m_7u^4 \pmod{u^3+u+1} \\ &\equiv (m_5 + m_6 + m_7 + m_9) + (m_5 + m_8 + m_9)u + (m_5 + m_6 + m_{10})u^2 \\ &\quad \pmod{u^3+u+1}. \end{aligned} \tag{4.48}$$

Now we must find $R_i(u), i = 1, \ldots, 4$.

i) modulo u

$$\begin{aligned} R_1(u)P_2(u)P_3(u)P_4(u) &\equiv 1 \pmod{P_1(u)} \\ R_1(u)(u^6 + u^4 + u + 1) &\equiv 1 \pmod{u}. \end{aligned} \tag{4.49}$$

Therefore $R_1(u) = 1$.

ii) modulo $u + 1$

$$\begin{aligned} R_2(u)P_1(u)P_3(u)P_4(u) &\equiv 1 \pmod{P_2(u)} \\ R_2(u)(u^6 + u^5 + u) &\equiv 1 \pmod{u+1}. \end{aligned} \tag{4.50}$$

Therefore $R_2(u) = 1$.

iii) modulo $u^2 + u + 1$

$$\begin{aligned} R_3(u)P_1(u)P_2(u)P_4(u) &\equiv 1 \pmod{P_3(u)} \\ R_3(u)(u^5 + u^4 + u^3 + u) &\equiv 1 \pmod{u^2 + u + 1}. \end{aligned} \tag{4.51}$$

Therefore $R_3(u) = u^2$.

iv) modulo $u^3 + u + 1$

$$\begin{aligned} R_4(u)P_1(u)P_2(u)P_3(u) &\equiv 1 \pmod{P_4(u)} \\ R_4(u)(u^4 + u) &\equiv 1 \pmod{u^3 + u + 1}. \end{aligned} \tag{4.52}$$

Therefore $R_3(u) = u^5$.

Now we can form the sum $\Phi(u)$, as follows

$$\begin{aligned} \Phi(u) &\equiv \Phi_1(u)R_1(u)Q_1(u) + \Phi_2(u)R_2(u)Q_2(u) + \Phi_3(u)R_3(u)Q_3(u) + \Phi_4(u)R_4(u)Q_4(u) \\ &\quad \pmod{P(u)} \\ &= \Phi_1(u)R_1(u)P_2(u)P_3(u)P_4(u) + \Phi_2(u)R_2(u)P_1(u)P_3(u)P_4(u) \\ &\quad + \Phi_3(u)R_3(u)P_1(u)P_2(u)P_4(u) + \Phi_4(u)R_4(u)P_1(u)P_2(u)P_3(u) \\ &\quad \pmod{P(u)} \\ &= m_0 + (m_0 + m_1 + m_3 + m_4 + m_5 + m_7 + m_8 + m_{10})u \\ &\quad + (m_2 + m_3 + m_5 + m_6 + m_7 + m_9)u^2 \\ &\quad + (m_3 + m_4 + m_6 + m_7 + m_8)u^3 \\ &\quad + (m_0 + m_2 + m_4 + m_5 + m_7 + m_8 + m_{10})u^4 \\ &\quad + (m_1 + m_2 + m_4 + m_5 + m_6 + m_7 + m_9)u^5 \\ &\quad + (m_0 + m_1 + m_2 + m_3 + m_6 + m_7 + m_8)u^6 \\ &\quad \pmod{P(u)} \end{aligned} \tag{4.53}$$

and as expected this gives

$$\begin{aligned} \Phi(u) &= z_0y_0 + (z_0y_1 + z_1y_0)u + (z_0y_2 + z_1y_1 + z_2y_0)u^2 \\ &\quad + (z_0y_3 + z_1y_2 + z_2y_1 + z_3y_0)u^3 + (z_1y_3 + z_2y_2 + z_3y_1)u^4 \\ &\quad + (z_2y_3 + z_3y_2)u^5 + z_3y_3u^6. \end{aligned} \tag{4.54}$$

Thus to multiply two degree three polynomials using the LSW algorithm takes eleven multiplications.

Note this method is recursive in that we use the algorithms for smaller degree polynomial multiplication to obtain algorithms for larger degree polynomial multiplication. We can get the following table of number of multiplications needed to multiply two degree $\eta - 1$ polynomials.

$\eta - 1$	0	1	2	3	4	5	6	7	8	9	10	11
no. mults (LSW)	1	3	6	11	16	22	27	33	38	44	49	55

With the above paragraph in mind if we can find better algorithms for small degree polynomial multiplication then this will result in the large degree polynomial multiplication taking less multiplications. We can improve on this algorithm by noting firstly (as proved in Kaminski (1985)) that $M_2(4, P(u)) = 9$, and secondly (via our square, $P(u)$ bound) that $M_2(5, P(u)) \leq 14$. So we can get better algorithms for smaller degree polynomial multiplication and this will then enable us to construct better algorithms for larger degree polynomial multiplication. Further, if we choose the $P(u)$ and its factorisation (obviously ensuring that the factors are pairwise coprime) we can get a better algorithm, which we denote by LSW^*. Let us demonstrate this with an example.

Example 4.14 If we wish to multiply two degree 13 polynomials using the LSW algorithm then we need $P(u) = u(u+1)(u^2+u+1)(u^3+u+1)(u^3+u^2+1)(u^4+u+1)(u^4+u^3+1)(u^4+u^3+u^2+u+1)(u^5+u^3+1)$ and thus this takes $1+1+3+6+6+11+11+11+16 = 66$ multiplications. Now if we use the LSW* with the same $P(u)$ the algorithm will take $1+1+3+6+6+9+9+9+14 = 58$. For this particular example the $P(u)$ is the best possible to reduce the number of multiplications needed.

We therefore get the following table for the number of multiplications needed when using the LSW* algorithm.

$\eta - 1$	0	1	2	3	4	5	6	7	8	9	10	11
no. mults (LSW*)	1	3	6	9	14	18	25	29	34	38	43	49

Now, further to the standard version of the CRT we may use the Improved version as found in Section 2.5 and apply the wraparound method to try to reduce the number of multiplications needed to multiply two polynomials. We denote the algorithm designed this way as the *LSW* with wraparound.* We give here an example and point the reader for reference to Example 2.23 for the full construction method.

Example 4.15 We want to multiply two degree 6 polynomials using the least possible number of multiplications. Using the LSW algorithm we could choose $P(u) = u(u^2+u+1)(u^3+u+1)(u^3+u^2+1)(u^4+u+1)$. Thus the reconstruction would take $1+3+6+6+11 = 27$ multiplications. Now using the LSW*, firstly using the same $P(u)$ would result in the number of multiplications being $1+3+6+6+9 = 25$. Secondly, we could choose a new $P(u)$ as $u^2(u^2+u+1)(u^3+u+1)(u^3+u^2+1)(u^3+u^2+u+1)$ and this would result in an algorithm taking only $3+3+6+6+6 = 24$ multiplications.

Now we wish to apply the wraparound technique. If we take $P(u) = (u+1)u^2(u^2+u+1)(u^3+u+1)(u^3+u^2+1)$ and $s = 2$ then this would result in an algorithm taking $1+3+3+6+6+3 = 22$ multiplications.

So we can get a table of the best possible number of multiplications for polynomial multiplication using the LSW* with wraparound method as follows.

$\eta - 1$	0	1	2	3	4	5	6	7	8	9	10	11
no. mults (LSW* with wrap.)	1	3	6	9	14	18	25	29	34	38	43	49

It must be noted here that the LSW* with wraparound algorithm is not always better than the LSW* algorithm but always at least as good (obviously as we can take the wraparound to be zero).

4.7.2 The Knuth bound

In the book Knuth (1981) Page 178, there is a numerical approach to multiplying two numbers. We can get a polynomial equivalent of this as follows. For $\eta = 2s$, if

$$\begin{aligned} z(u) &= z_0 + z_1 u + \cdots + z_{2s-1} u^{2s-1} \\ y(u) &= y_0 + y_1 u + \cdots + y_{2s-1} u^{2s-1}, \end{aligned} \tag{4.55}$$

write

$$\begin{aligned} z &= u^s z^{(1)} + z^{(0)} \; ; & z^{(1)} &= z_s + z_{s+1} u + \cdots + z_{2s-1} u^{s-1} \\ & & z^{(0)} &= z_0 + z_1 u + \cdots + z_{s-1} u^{s-1} \\ y &= u^s y^{(1)} + y^{(0)} \; ; & y^{(1)} &= y_s + y_{s+1} u + \cdots + y_{2s-1} u^{s-1} \\ & & y^{(0)} &= y_0 + y_1 u + \cdots + y_{s-1} u^{s-1}. \end{aligned} \tag{4.56}$$

Then

$$z(u)y(u) = (u^{2s} + u^s) z^{(1)} y^{(1)} + u^s (z^{(1)} - z^{(0)})(y^{(0)} - y^{(1)}) + (u^s + 1) z^{(0)} y^{(0)}.$$

This reduces the problem of multiplying a pair of degree $\eta - 1$ (η even) polynomials to that of multiplying three pairs of degree $\frac{\eta}{2} - 1$ polynomials.

This method can also be used (with slight modification) for $\eta = 2s + 1$, and we can get the following.

$$\begin{aligned} z(u) &= z_0 + z_1 u + \cdots + z_{2s} u^{2s} \\ y(u) &= y_0 + y_1 u + \cdots + y_{2s} u^{2s}, \end{aligned} \tag{4.57}$$

write

$$\begin{aligned} z &= u^s z^{(1)} + z^{(0)} \; ; & z^{(1)} &= z_s + z_{s+1} u + \cdots + z_{2s} u^s \\ & & z^{(0)} &= z_0 + z_1 u + \cdots + z_{s-1} u^{s-1} \\ y &= u^s y^{(1)} + y^{(0)} \; ; & y^{(1)} &= y_s + y_{s+1} u + \cdots + y_{2s} u^s \\ & & y^{(0)} &= y_0 + y_1 u + \cdots + y_{s-1} u^{s-1}. \end{aligned} \tag{4.58}$$

Then

$$z(u)y(u) = (u^{2s} + u^s) z^{(1)} y^{(1)} + u^s (z^{(1)} - z^{(0)})(y^{(0)} - y^{(1)}) + (u^s + 1) z^{(0)} y^{(0)}.$$

This reduces the problem of multiplying two degree $\eta - 1$ (η odd) polynomials to that of multiplying two pairs of degree $\frac{\eta-1}{2}$ polynomials and one pair of degree $\frac{\eta-1}{2} - 1$ polynomials.

If we let $K(\eta)$ be the number of multiplications needed to multiply two degree $\eta - 1$ polynomials using above method then we get that the Knuth bound can be written as the recursive formula

$$K(\eta) = \begin{cases} 3K\left(\frac{\eta}{2}\right), & \eta \text{ even} \\ 2K\left(\frac{\eta+1}{2}\right) + K\left(\frac{\eta-1}{2}\right), & \eta \text{ odd} \end{cases}. \tag{4.59}$$

4.8 A Comparison of Some of the Upper Bounds

In this section we give graphical comparisons for some of the upper bounds for $M_2(\eta)$, that we found in the previous sections. As we are only really interested in the values for smallish η, we plot the graphs up to $\eta = 14$ (η even), and $\eta = 15$ (η odd).

First of all we give the graphs of the bounds for η odd in Figure 4.19, and the graphs of the bounds for η even in Figure 4.20.

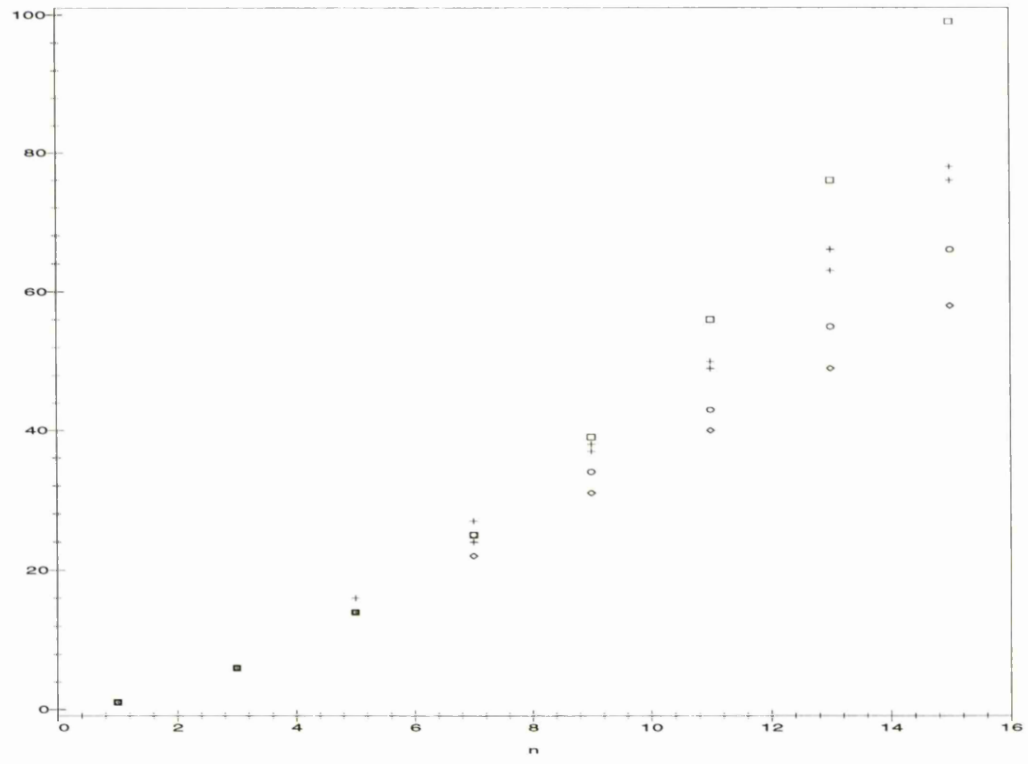

Figure 4.19: The graph of the five upper bounds for η odd.

The graphs are labelled as follows:

□	Square, $P(u)$
Upper +	Knuth
Lower +	LSW
○	LSW*
◇	LSW* with wraparound

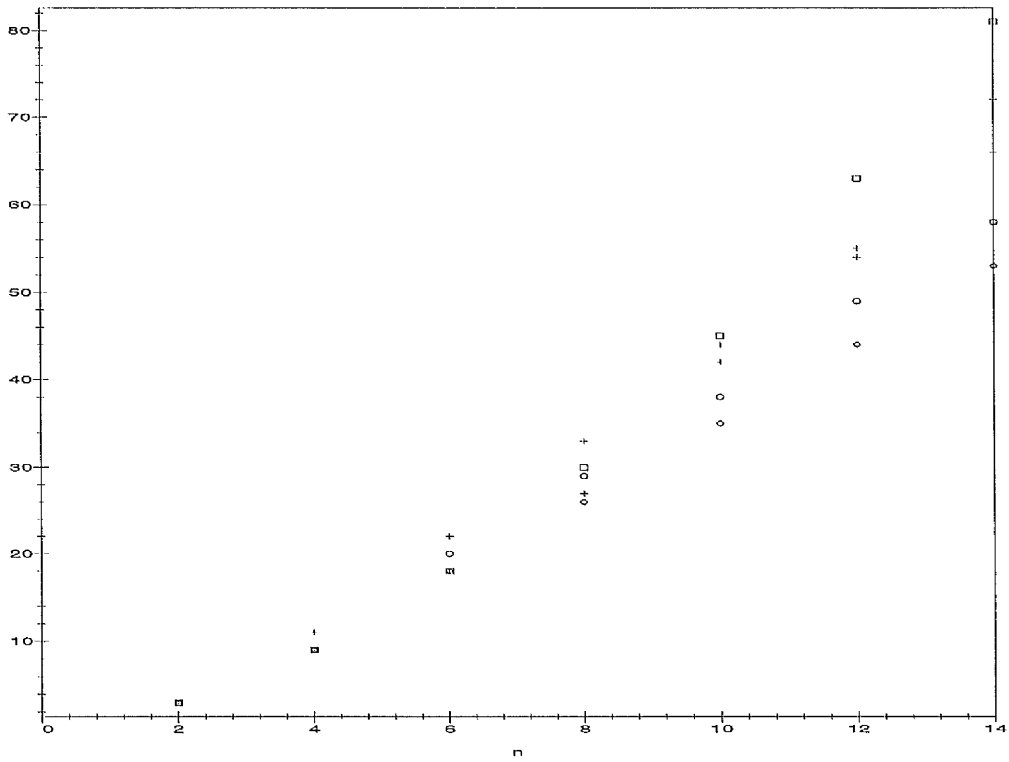

Figure 4.20: The graph of the five upper bounds for η even.

These graphs show that LSW* with wraparound algorithm is the best over the range indicated. However, as stated in Section 4.7, the LSW* algorithm is sometimes as good as the LSW* with wraparound.

4.9 A Generalised Upper Bound for $M_2(\eta - 1, u^\eta)$

The algorithm we develop here will be known as the *square, u^η algorithm.*

Starting from a base code and extending the length of the codewords we may be able to increase the minimum distance while keeping k the same. Using this idea we may consider a family of KM codes to be those having constant k and increasing d, by taking a constant $P(u)$ and letting s, the number of wraparound points, increase from 0. We therefore need a general algorithm for obtaining $\Phi(u) \equiv Z(u)Y(u)$ modulo u^s for varying s. Below we will develop such an algorithm.

Of course for $s = 0$ there is nothing to do so the first case we need to look at is $s = 1$. We get for this value

$$\begin{aligned} Z(u) &\equiv z_0 \pmod{u} \\ Y(u) &\equiv y_0 \pmod{u} \end{aligned} \tag{4.60}$$

and by letting $m_0 = z_0 y_0$ we get $\Phi(u) \equiv Z(u)Y(u) \pmod{u} = m_0$.

For the case $s = 2$ we get that

$$\begin{aligned} Z(u) &\equiv z_0 + z_1 u \pmod{u^2} \\ Y(u) &\equiv y_0 + y_1 u \pmod{u^2} \end{aligned} \tag{4.61}$$

and by letting $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ we get

$$\Phi(u) \equiv Z(u)Y(u) \pmod{u^2} = m_0 + (m_0 + m_1 + m_2)u. \tag{4.62}$$

The case $s = 3$ is the first case to need less multiplications than the square, $P(u)$ algorithm for $n = 3$. We get that

$$\begin{aligned} Z(u) &\equiv z_0 + z_1 u + z_2 u^2 \pmod{u^3} \\ Y(u) &\equiv y_0 + y_1 u + y_2 u^2 \pmod{u^3} \end{aligned} \tag{4.63}$$

and by letting $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = z_2 y_2$, $m_3 = (z_0 + z_1)(y_0 + y_1)$ and $m_4 = (z_0 + z_2)(y_0 + y_2)$ then

$$\Phi(u) \equiv Z(u)Y(u) \pmod{u^3} = m_0 + (m_0 + m_1 + m_3)u + (m_0 + m_1 + m_2 + m_4)u^2. \tag{4.64}$$

This can be easily generalised for a given even value of s, to use the following multiplications

$$
\begin{array}{rcl|rcl}
m_0 & = & z_0 y_0 & m_{4s-6} & = & (z_2 + z_{s-3})(y_2 + y_{s-3}) \\
m_1 & = & z_1 y_1 & m_{4s-5} & = & (z_3 + z_4)(y_3 + y_4) \\
 & \vdots & & & \vdots & \\
m_{s-1} & = & z_{s-1} y_{s-1} & m_{5s-11} & = & (z_3 + z_{s-4})(y_3 + y_{s-4}) \\
m_s & = & (z_0 + z_1)(y_0 + y_1) & m_{5s-10} & = & (z_4 + z_{s-5})(y_4 + y_{s-5}) \\
m_{s+1} & = & (z_0 + z_2)(y_0 + y_2) & & \vdots & \\
 & \vdots & & m_{\frac{s^2}{4}+s-3} & = & (z_{\frac{s}{2}-2} + z_{\frac{s}{2}-1})(z_{\frac{s}{2}-2} + z_{\frac{s}{2}-1}) \\
m_{2s-2} & = & (z_0 + z_{s-1})(y_0 + y_{s-1}) & m_{\frac{s^2}{4}+s-2} & = & (z_{\frac{s}{2}-2} + z_{\frac{s}{2}})(z_{\frac{s}{2}-2} + z_{\frac{s}{2}}) \\
m_{2s-1} & = & (z_1 + z_2)(y_1 + y_2) & m_{\frac{s^2}{4}+s-1} & = & (z_{\frac{s}{2}-1} + z_{\frac{s}{2}})(y_{\frac{s}{2}-1} + y_{\frac{s}{2}}) \\
 & \vdots & & & & \\
m_{3s-3} & = & (z_1 + z_{s-2})(y_1 + y_{s-2}) & & & \\
m_{3s-2} & = & (z_2 + z_3)(y_2 + y_3) & & & \\
 & \vdots & & & &
\end{array}
\tag{4.65}
$$

With the coefficients ϕ_i, $i = 0, \ldots, s-1$ given as follows.

$$
\begin{aligned}
\phi_0 &= m_0 \\
\phi_1 &= m_s - m_0 - m_1 \\
\phi_2 &= m_1 + m_{s+1} - m_0 - m_2 \\
\phi_3 &= m_{2s-1} - m_1 - m_2 + m_{s+2} - m_0 - m_3 \\
\phi_4 &= m_2 + m_{2s} - m_1 - m_3 + m_{s+3} - m_0 - m_4 \\
\phi_5 &= m_{3s-2} - m_2 - m_3 + m_{2s+1} - m_1 - m_4 + m_{s+4} - m_0 - m_5 \\
\phi_6 &= m_3 + m_{2s+2} - m_1 - m_5 + m_{3s-1} - m_2 - m_4 + m_{s+5} - m_0 - m_6 \\
&\vdots \\
\phi_{s-2} &= m_{\frac{s-2}{2}} + m_{2s-2} - m_0 - m_{s-2} + m_{3s-4} - m_1 - m_{s-3} \\
&\quad + m_{4s-7} - m_2 - m_{s-4} + \cdots + m_{\frac{s^2}{4}+s-2} - m_{\frac{s}{2}-2} - m_{\frac{s}{2}} \\
\phi_{s-1} &= m_{2s-2} - m_0 - m_{s-1} + m_{3s-3} - m_1 - m_{s-2} + m_{4s-6} \\
&\quad - m_2 - m_{s-3} + \cdots + m_{\frac{s^2}{4}+s-1} - m_{\frac{s}{2}-1} - m_{\frac{s}{2}}
\end{aligned}
\tag{4.66}
$$

and similarly for s, odd, we use the following multiplications

$$
\begin{array}{rcl|rcl}
m_0 &=& z_0y_0 & m_{4s-6} &=& (z_2+z_{s-3})(y_2+y_{s-3}) \\
m_1 &=& z_1y_1 & m_{4s-5} &=& (z_3+z_4)(y_3+y_4) \\
 & \vdots & & & \vdots & \\
m_{s-1} &=& z_{s-1}y_{s-1} & m_{5s-11} &=& (z_3+z_{s-4})(y_3+y_{s-4}) \\
m_s &=& (z_0+z_1)(y_0+y_1) & m_{5s-10} &=& (z_4+z_{s-5})(y_4+y_{s-5} \\
m_{s+1} &=& (z_0+z_2)(y_0+y_2) & & \vdots & \\
 & \vdots & & m_{\frac{s^2}{4}+s-\frac{13}{4}} &=& (z_{\frac{s-1}{2}-2}+z_{\frac{s-1}{2}-1})(y_{\frac{s-1}{2}-2}+y_{\frac{s-1}{2}-1}) \\
m_{2s-2} &=& (z_0+z_{s-1})(y_0+y_{s-1}) & m_{\frac{s^2}{4}+s-\frac{9}{4}} &=& (z_{\frac{s-1}{2}-1}+z_{\frac{s-1}{2}})(y_{\frac{s-1}{2}-1}+y_{\frac{s-1}{2}}) \\
m_{2s-1} &=& (z_1+z_2)(y_1+y_2) & m_{\frac{s^2}{4}+s-\frac{5}{4}} &=& (z_{\frac{s-1}{2}-1}+z_{\frac{s-1}{2}+1})(y_{\frac{s-1}{2}-1}+y_{\frac{s-1}{2}+1}) \\
 & \vdots & & & & \\
m_{3s-3} &=& (z_1+z_{s-2})(y_1+y_{s-2}) & & & \\
m_{3s-2} &=& (z_2+z_3)(y_2+y_3) & & & \\
 & \vdots & & & &
\end{array}
\tag{4.67}
$$

With the coefficients ϕ_i, $i=0,\ldots,s-1$ given as follows.

$$
\begin{aligned}
\phi_0 &= m_0 \\
\phi_1 &= m_s - m_0 - m_1 \\
\phi_2 &= m_1 + m_{s+1} - m_0 - m_2 \\
\phi_3 &= m_{2s-1} - m_1 - m_2 + m_{s+2} - m_0 - m_3 \\
\phi_4 &= m_2 + m_{2s} - m_1 - m_3 + m_{s+3} - m_0 - m_4 \\
\phi_5 &= m_{3s-2} - m_2 - m_3 + m_{2s+1} - m_1 - m_4 + m_{s+4} - m_0 - m_5 \\
\phi_6 &= m_3 + m_{3s-1} - m_2 - m_4 + m_{2s+2} - m_1 - m_5 + m_{s+5} - m_0 - m_6 \\
&\vdots \\
\phi_{s-2} &= m_{2s-3} - m_0 - m_{s-2} + m_{3s-4} - m_1 - m_{s-3} + m_{4s-7} \\
&\quad -m_2 - m_{s-4} + \cdots + m_{\frac{s^2}{4}+s-\frac{9}{4}} - m_{\frac{s-1}{2}-1} - m_{\frac{s-1}{2}} \\
\phi_{s-1} &= m_{\frac{s-1}{2}} + m_{2s-2} - m_0 - m_{s-1} + m_{3s-3} - m_1 - m_{s-2} \\
&\quad +m_{4s-6} - m_2 - m_{s-3} + \cdots + m_{\frac{s^2}{4}+s-\frac{5}{4}} - m_{\frac{s-1}{2}-1} - m_{\frac{s-1}{2}+1}
\end{aligned}
\tag{4.68}
$$

Using the square,u^η algorithm we may construct an infinite family of KM codes where the wraparound part of the matrix is increased to increase the minimum distance of the code. Obviously if the $P(u)$ part of the matrix remains the same then the decoding procedure stays essentially the same (see Krishna (1987)), in that a circuit that was used to decode a codeword from the code with say $s = 0$ can be easily extended to the cases $s = 1, 2, \ldots$.

Example 4.16 Take $P(u) = u^2(u^2 + 1)$ and $s = 0$. For this example take $k = 3$ and $d = 2$ then

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 \\ Y(u) &= y_0 + y_1 u. \end{aligned} \tag{4.69}$$

Now using the process of the CRT we have $P(u) = P_1(u)P_2(u)$ so

(i) modulo $P_1(u) = u^2$

$$\begin{aligned} Z_1(u) &= Z(u) \pmod{u^2} \\ &= z_0 + z_1 u \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_1(u) &= Y(u) \pmod{u^2} \\ &= y_0 + y_1 u \end{aligned} \tag{4.70}$$

Let $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$. Then

$$\begin{aligned} \Phi_1(u) &\equiv Z_1(u) Y_1(u) \pmod{u^2} \\ &= m_0 + (m_0 + m_1 + m_2)u + m_2 u^2 \pmod{u^2} \\ &\equiv m_0 + (m_0 + m_1 + m_2)u \pmod{u^2} \end{aligned} \tag{4.71}$$

(ii) modulo $P_1(u) = u^2 + 1$

$$\begin{aligned} Z_2(u) &= Z(u) \pmod{u^2 + 1} \\ &= (z_0 + z_2) + z_1 u \end{aligned} \quad \Bigg| \quad \begin{aligned} Y_2(u) &= Y(u) \pmod{u^2 + 1} \\ &= y_0 + y_1 u \end{aligned} \tag{4.72}$$

Let $m_3 = (z_0 + z_2)y_0$, $m_4 = z_1 y_1$ and $m_5 = (z_0 + z_1 + z_2)(y_0 + y_1)$. Then

$$\begin{aligned} \Phi_2(u) &\equiv Z_2(u) Y_2(u) \pmod{u^2 + 1} \\ &= m_3 + (m_3 + m_4 + m_5)u + m_4 u^2 \pmod{u^2 + 1} \\ &\equiv (m_3 + m_4) + (m_3 + m_4 + m_5)u \pmod{u^2 + 1} \end{aligned} \tag{4.73}$$

Now we must find the $R_i(u), i = 1, 2$ as follows.

(i) modulo u^2

$$\begin{aligned} R_1(u)P_2(u) &\equiv 1 \pmod{P_1(u)} \\ R_1(u)(u^2 + 1) &\equiv 1 \pmod{u^2} \end{aligned} \tag{4.74}$$

Therefore $R_1(u) = 1$.

(ii) modulo $u^2 + 1$

$$\begin{aligned} R_2(u)P_1(u) &\equiv 1 \pmod{P_2(u)} \\ R_2(u)(u^2) &\equiv 1 \pmod{u^2+1} \end{aligned} \tag{4.75}$$

Therefore $R_2(u) = 1$.

Now we can form the sum, $\Phi(u)$ as follows.

$$\begin{aligned} \Phi(u) &\equiv \Phi_1(u)R_1(u)P_2(u) + \Phi_2(u)R_2(u)P_1(u) \pmod{P(u)} \\ &= (m_0 + (m_0 + m_1 + m_2)u)(u^2+1) + ((m_3 + m_4) + (m_3 + m_4 + m_5)u)u^2 \\ &\quad \pmod{(P(u))} \\ &= m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_3 + m_4)u^2 \\ &\quad +(m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^3 \end{aligned} \tag{4.76}$$

Now considering the ideas of Sections 3.2 and 3.4 we would like to put the above reconstruction of $\Phi(u)$ into a computation of the form

$$\mathbf{\Phi} = A^T(G^T\boldsymbol{z} \times B\boldsymbol{y}) \tag{4.77}$$

with $\boldsymbol{z} = (z_0, \ldots, z_{k-1})^T$, $\boldsymbol{y} = (y_0, \ldots, y_{d-1})^T$ and $\mathbf{\Phi} = (z_0y_0, z_0y_1 + z_1y_0, \ldots, z_{k-1}y_{d-1})^T$ as (3.23).

Now as explained at the end of Section 3.2 $(G^T\boldsymbol{z} \times B\boldsymbol{y})$ is simply a way of writing the multiplications $m_0, \ldots, m_{\eta-1}$ as a column vector and the A^T is simply picking out the relevent multiplications to form the coefficients of $\Phi(u)$.

Take m_0 as a start. This is equal to z_0y_0, so in vector form this would be

$$m_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \boldsymbol{z} \times \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{y}. \tag{4.78}$$

So the zero'th row of G^T is [1 0 0] and the zero'th row of B is [1 0]. Considering all the

multiplications we get

$$G^T\boldsymbol{z} \times B\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \tag{4.79}$$

where, for example m_5 is expressed by row 5 of G^T and B, i.e.

$$m_5 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \boldsymbol{z} \times \begin{bmatrix} 1 & 1 \end{bmatrix} \boldsymbol{y}. \tag{4.80}$$

Now to form $\boldsymbol{\Phi}$ we must pick out the relevent multiplications. This is done by refering to (4.76). We see that

$$\phi_0 = m_0,$$

so in matrix form this is

$$\phi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \end{bmatrix} \tag{4.81}$$

and so row 0 of A^T is $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. We refer similarly to (4.76) to form the rest of A^T and get

$$A^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.82}$$

Now the whole expression can be written as

$$\boldsymbol{\Phi} = A^T(G^T\boldsymbol{z} \times B\boldsymbol{y})$$

as we wanted.

Now by Theorem 3.11 and Section 3.4, G is the generator matrix of KM$(6,3,2)$ code.

$$G = \left[\begin{array}{cccccc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]. \tag{4.83}$$

We extend the ideas of blocks as in Definition 2.15 and further in Section 2.5 to partition the matrix G into blocks. A particular block of the matrix is now associated with a particular block in the CRT. For Example 4.16 this will result in us writing G as

$$G = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]. \tag{4.84}$$

A powerful feature of these codes is the ability by increasing N then keeping k the same automatically guarantees d to increase (as $N = k + d - 1 - s$). One way to implement this is to start with $s = 0$ and increase s.

Example 4.17 (i) So let us redo Example 4.16 with $P(u)$ the same and let $s = 1$. Keeping k the same will result in us increasing d by 1. We thus have $k = 3$ and $d = 3$ and so

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 \\ Y(u) &= y_0 + y_1 u + y_2 u^2. \end{aligned} \tag{4.85}$$

Now from block one (i.e. $P_1(u) = u^2$) we get the same multiplications i.e. $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ and from block two (i.e. $P_2(u) = u^2 + 1$) we get as $m_3 = (z_0 + z_2)(y_0 + y_2)$, $m_4 = z_1 y_1$ and $m_5 = (z_0 + z_1 + z_2)(y_0 + y_1 + y_2)$.

Now $\Phi_1(u)$, $\Phi_2(u)$, $R_1(u)$ and $R_2(u)$ are the same as Example 4.16 so we get (using Section 2.5 and noting that as we are using $s > 0$ then we get $\hat{\Phi}(u)$)

$$\hat{\Phi}(u) = m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_3 + m_4)u^2 + (m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^3. \tag{4.86}$$

Now consider the wraparound block of the CRT. We have

$$\begin{aligned} \overline{Z}(u) &= z_2 + z_1 u + z_0 u^2 \\ \overline{Y}(u) &= y_2 + y_1 u + y_0 u^2 \end{aligned} \tag{4.87}$$

and as $s = 1$ we get

$$\begin{aligned} \overline{Z}(u) &\equiv z_2 \pmod{u} \\ \overline{Y}(u) &\equiv y_2 \pmod{u}. \end{aligned} \tag{4.88}$$

Let $m_6 = z_2 y_2$, then according the Theorem 2.22 we have

$$\begin{aligned} \Phi(u) &= \hat{\Phi}(u) + m_6 P(u) \\ &= \hat{\Phi}(u) + m_6(u^4 + u^2) \\ &= m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_3 + m_4 + m_6)u^2 \\ &\quad + (m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^3 + m_6 u^4, \end{aligned} \tag{4.89}$$

which can be checked to be correct.

Now writing this in the form of a computation as (4.77) we see that (now partitioned into three blocks corresponding to $P_1(u)$, $P_2(u)$ and $s = 1$)

$$G_1 = \left[\begin{array}{ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] \tag{4.90}$$

and the first two blocks are the same as Example 4.16. Here G is the generator matrix of a KM$(7, 3, 3)$ code as expected.

(ii) Now let us redo Example 4.16 with $P(u)$ the same and let $s = 2$. Keeping k the same will result in us increasing d by two. We thus have $k = 3$ and $d = 4$ and so

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 \\ Y(u) &= y_0 + y_1 u + y_2 u^2 + y_3 u^3. \end{aligned} \tag{4.91}$$

Now from block one (i.e. $P_1(u) = u^2$) we get the same multiplications i.e. $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ and from block two (i.e. $P_2(u) = u^2 + 1$) we get as $m_3 = (z_0 + z_2)(y_0 + y_2)$, $m_4 = z_1(y_1 + y_3)$ and $m_5 = (z_0 + z_1 + z_2)(y_0 + y_1 + y_2 + y_3)$.

Now $\Phi_1(u)$, $\Phi_2(u)$, $R_1(u)$ and $R_2(u)$ are the same as Example 4.16 so we get (using Section 2.5 and noting that as we are using $s > 0$ then we get $\hat{\Phi}(u)$)

$$\hat{\Phi}(u) = m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_3 + m_4)u^2 + (m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^3. \tag{4.92}$$

Now consider the wraparound block of the CRT. We have

$$\begin{aligned} \overline{Z}(u) &= z_2 + z_1 u + z_0 u^2 \\ \overline{Y}(u) &= y_3 + y_2 u + y_1 u^2 + y_0 \end{aligned} \tag{4.93}$$

and as $s = 2$ we get

$$\begin{aligned} \overline{Z}(u) &\equiv z_2 + z_1 u \pmod{u^2} \\ \overline{Y}(u) &\equiv y_3 + y_2 u \pmod{u^2}. \end{aligned} \tag{4.94}$$

Let $m_6 = z_2 y_3$, $m_7 = z_1 y_2$ and $m_8 = (z_1 + z_2)(y_2 + y_3)$. We could now use Section 2.5 to reconstruct the polynomial in the same way as part (i) of this example, but it is sufficient to form G_2 directly from $\Phi = A^T(G_2^T \boldsymbol{z} \times B\boldsymbol{y})$ giving

$$G_2 = \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{4.95}$$

Now as can be seen here the matrix G_2 is just the matrix G_1 with two extra columns corresponding to the increase in s. Note here that if we are using the square, u^η algorithm then G_j is contained in G_i for $0 \leq i < j$.

So as k remains the same, this procedure can be used for any s, and we can thus get an infinite family of KM codes.

We will see in the following three chapters some general methods introduced to obtain the weight enumerators of this type of code (i.e. with increasing s).

Chapter 5

The Weight Enumerators of KM Codes

5.1 Introduction

We have seen in the previous chapters how KM codes were developed and constructed. In this chapter we take this a step further and move away from the direct construction of the polynomial approach. We would like, given simply the $P(u)$ and the number of wraparound coefficients s, to be able to construct the weight enumerator of the KM codes for any k and d. Obviously along the way it will be necessary to have knowledge of the way the algorithms for polynomial multiplication are used but in the end the families (defined by $k + d =$ constant, for a given $P(u)$ and s) will be dependent only on k (or alternatively d).

5.2 Shunting of the KM codes

Looking at the construction of KM codes, we may intuitively hope that as a decrease in k results in an increase in d we may simply reduce the number of rows of the generator matrix of a KM(n,k,d) code to get a KM$(n,k-1,d+1)$ code.

Thinking more carefully we would hope to remove the bottom row of the generator matrix of a KM(n,k,d) code corresponding to the non-wraproud blocks and the top row corresponding to the wraparound block, as reducing k by one implies reducing the powers of u in $Z(u)$ by one. Then using the CRT we firstly will not have use for the coefficient of u^{k-1} in $Z(u)$, and secondly $\overline{Z}(u)$ will now equal $z_{k-2}+z_{k-3}u+\cdots+z_0u^{k-2}$ implying a shift up one for the wraparound part of the matrix. We define this action.

Definition 5.1 Consider the generator matrix, G, of a KM(n,k,d) code formed by the CRT. Let G be partitioned into blocks corresponding to the factors $P_1(u),\ldots,P_L(u)$ and s, the wraparound. If we remove the bottom row of the blocks corresponding to the factors $P_1(u),\ldots,P_L(u)$ and remove the top row of the block corresponding to the wraparound and move this block up one we get a $((k-1)\times n)$ matrix. We say that we have *shunted* the generator matrix of the KM(n,k,d) code. If we have formed the generator matrix without using any wraparound then to *shunt* it we simply remove the bottom row.

We now show an example of shunting taking place and prove that (in this case at least) it does indeed form a KM code.

Example 5.2 Let $N=7$ and take $P(u)=u^2(u^2+1)(u^2+u+1)$, $s=1$, $k=6$ and $d=2$. We thus have

$$\begin{aligned} Z(u) &= z_0+z_1u+z_2u^2+z_3u^3+z_4u^4+z_5u^5 \\ Y(u) &= y_0+y_1u. \end{aligned} \tag{5.1}$$

Now using the Improved version of the CRT (in the same way as Example 4.17) we will form the generator matrix of a KM$(n,6,2)$ code. Let us now form the code.
(i) modulo u^2

$$\begin{aligned} Z_1(u) &= z_0+z_1u \pmod{u^2} \\ Y_1(u) &= y_0+y_1u \pmod{u^2}. \end{aligned} \tag{5.2}$$

Let $m_0 = z_0y_0$, $m_1 = z_1y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ then

$$\Phi_1(u) \equiv m_0 + (m_0 + m_1 + m_2)u \pmod{u^2} \tag{5.3}$$

(ii) modulo $u^2 + 1$

$$\begin{aligned} Z_2(u) &= (z_0 + z_2 + z_4) + (z_1 + z_3 + z_5)u \pmod{u^2+1} \\ Y_2(u) &= y_0 + y_1u \pmod{u^2+1}. \end{aligned} \tag{5.4}$$

Let $m_3 = (z_0+z_2+z_4)y_0$, $m_4 = (z_1+z_3+z_5)y_1$ and $m_5 = (z_0+z_1+z_2+z_3+z_4+z_5)(y_0+y_1)$ then

$$\Phi_2(u) \equiv (m_3 + m_4) + (m_3 + m_4 + m_5)u \pmod{u^2+1} \tag{5.5}$$

(ii) modulo $u^2 + u + 1$

$$\begin{aligned} Z_3(u) &= (z_0 + z_2 + z_3 + z_5) + (z_1 + z_2 + z_4 + z_5)u \pmod{u^2+u+1} \\ Y_3(u) &= y_0 + y_1u \pmod{u^2+u+1}. \end{aligned} \tag{5.6}$$

Let $m_6 = (z_0+z_2+z_3+z_5)y_0$, $m_7 = (z_1+z_2+z_4+z_5)y_1$ and $m_8 = (z_0+z_1+z_3+z_4)(y_0+y_1)$ then

$$\Phi_3(u) \equiv (m_6 + m_7) + (m_6 + m_8)u \pmod{u^2+u+1} \tag{5.7}$$

We must now find $R_i(u), i = 1, 2, 3$.

(i) modulo u^2

$$\begin{aligned} R_1(u)P_2(u)P_3(u) &= 1 \pmod{P_1(u)} \\ R_1(u)(u^4 + u^3 + u + 1) &= 1 \pmod{u^2} \end{aligned} \tag{5.8}$$

Therefore $R_1(u) = (u + 1)$.

(ii) modulo $u^2 + 1$

$$\begin{aligned} R_2(u)P_1(u)P_3(u) &= 1 \pmod{P_2(u)} \\ R_2(u)(u^4 + u^3 + u^2) &= 1 \pmod{u^2+1} \end{aligned} \tag{5.9}$$

Therefore $R_2(u) = u$.

(iii) modulo $u^2 + u + 1$

$$\begin{aligned} R_3(u)P_1(u)P_2(u) &= 1 \pmod{P_3(u)} \\ R_3(u)(u^4 + u^2) &= 1 \pmod{u^2+u+1} \end{aligned} \tag{5.10}$$

Therefore $R_2(u) = 1$.

Now we form $\hat{\Phi}(u)$ as

$$\begin{aligned}
\hat{\Phi}(u) &= \Phi_1(u)R_1(u)Q_1(u) + \Phi_2(u)R_2(u)Q_2(u) + \Phi_3(u)R_3(u)Q_3(u) \\
&= (m_0 + (m_0 + m_1 + m_2)u)(u+1)(u^4 + u^3 + u + 1) \\
&\quad +((m_3 + m_4) + (m_3 + m_4 + m_5)u)u(u^4 + u^3 + u^2) \\
&\quad +((m_6 + m_7) + (m_6 + m_8)u)(u^4 + u^2) \\
&= m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_6 + m_7)u^2 \\
&\quad +(m_1 + m_2 + m_3 + m_4 + m_6 + m_7)u^3 \\
&\quad +(m_0 + m_1 + m_2 + m_5 + m_6 + m_7)u^4 \\
&\quad +(m_0 + m_6 + m_6 + m_8)u^5 \\
&\quad +(m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^6 \\
&\equiv m_0 + (m_0 + m_1 + m_2)u + (m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7)u^2 \\
&\quad +(m_0 + m_5 + m_6 + m_8)u^3 \\
&\quad +(m_0 + m_1 + m_2 + m_5 + m_6 + m_7)u^4 \\
&\quad +(m_1 + m_2 + m_3 + m_4 + m_6 + m_8)u^5 \pmod{P(u)}.
\end{aligned} \tag{5.11}$$

Now looking at the wraparound we have

$$\begin{aligned}
\overline{Z}(u) &= z_5 + z_4u + z_3u^3 + z_2u^4 + z_1u^5 + z_0u^6 \\
\overline{Y}(u) &= y_1 + y_0u.
\end{aligned} \tag{5.12}$$

Then

$$\begin{aligned}
\overline{Z}(u) &\equiv z_5 \pmod{u} \\
\overline{Y}(u) &\equiv y_1 \pmod{u}.
\end{aligned} \tag{5.13}$$

Let $m_9 = z_5y_1$, so we have

$$\begin{aligned}
\Phi(u) &= \hat{\Phi}(u) + m_9P(u) \\
&= \hat{\Phi}(u) + m_9(u^6 + u^5 + u^3 + u^2).
\end{aligned} \tag{5.14}$$

So we have

$$\begin{aligned}
\Phi(u) &= m_0 + (m_0 + m_1 + m_2)u + (m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7 + m_9)u^2 \\
&\quad +(m_0 + m_5 + m_6 + m_8 + m_9)u^3 \\
&\quad +(m_0 + m_1 + m_2 + m_5 + m_6 + m_7)u^4 \\
&\quad +(m_1 + m_2 + m_3 + m_4 + m_6 + m_8 + m_9)u^5 + m_9u^6.
\end{aligned} \tag{5.15}$$

From the multiplications $m_i, i = 0, \ldots, 9$ and using the computation form of $\mathbf{\Phi} = A^T(G_1^T \boldsymbol{z} \times B_1 \boldsymbol{y})$ we get

$$G_1 = \left[\begin{array}{ccc|ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{array}\right], \tag{5.16}$$

the generator matrix of a KM$(10, 6, 2)$ code.

Now we wish to shunt this, so according to Definition 5.1 we remove the bottom row from blocks one, two and three, and remove the top row from block four (the last column). This will result in the matrix

$$G_1' = \left[\begin{array}{ccc|ccc|ccc|c} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]. \tag{5.17}$$

We must now decide if this is a KM$(10, 5, 3)$ code, or indeed any linear code with the parameters $(10, 5, 3)$. To prove this is a KM$(10, 5, 3)$ code we will do the example again with all the parameters the same except we will now take $k = 5$ and $d = 3$.

Example 5.3 We have now

$$\begin{aligned} Z(u) &= z_0 + z_1 u + z_2 u^2 + z_3 u^3 + z_4 u^4 \\ Y(u) &= y_0 + y_1 u + y_2 u^2, \end{aligned} \tag{5.18}$$

so the problem becomes

(i) modulo u^2. We get $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$, with $\Phi(u)$ (in terms of the m_i) the same as the previous example.

(ii) modulo $u^2 + 1$. We get $m_3 = (z_0 + z_2 + z_4)(y_0 + y_2)$, $m_4 = (z_1 + z_3)y_1$ and $m_5 = (z_0 + z_1 + z_2 + z_3 + z_4)(y_0 + y_1 + y_2)$, with $\Phi(u)$ the same as the previous example.

(iii) modulo u^2+u+1. We get $m_6=(z_0+z_2+z_3)(y_0+y_2)$, $m_1=(z_1+z_2+z_4)(y_1+y_2)$ and $m_2=(z_0+z_1+z_3+z_4)(y_0+y_1)$, with $\Phi(u)$ the same as the previous example.

Further $R_i(u), i=1,2,3$ are the same as before and so we get $\hat{\Phi}(u)$ the same as before(in terms of the $m_i, i=0,\ldots,8$). Now the wraparound will give

$$\begin{aligned}\overline{Z}(u) &= z_4+z_3u+z_2u^2+z_1u^3+z_0u^4\\ \overline{Y}(u) &= y_2+y_1u+y_0u^2.\end{aligned} \tag{5.19}$$

Then

$$\begin{aligned}\overline{Z}(u) &\equiv z_4 \pmod{u}\\ \overline{Y}(u) &\equiv y_2 \pmod{u}.\end{aligned} \tag{5.20}$$

Let $m_9=z_4y_2$, so we have

$$\begin{aligned}\Phi(u) &= \hat{\Phi}(u)+m_9P(u)\\ &= \hat{\Phi}(u)+m_9(u^6+u^5+u^3+u^2),\end{aligned} \tag{5.21}$$

giving the same (in terms of the m_i) as the previous example. We therefore get from $\mathbf{\Phi}=A^T(G_2^T\boldsymbol{z}\times B_2\boldsymbol{y})$ (note here that A is the same as before),

$$G_2=\left[\begin{array}{ccc|ccc|ccc|c} 1&0&1&1&0&1&1&0&1&0\\ 0&1&1&0&1&1&0&1&1&0\\ 0&0&0&1&0&1&1&1&0&0\\ 0&0&0&0&1&1&1&0&1&0\\ 0&0&0&1&0&1&0&1&1&1 \end{array}\right], \tag{5.22}$$

the generator matrix of a KM$(10,5,3)$ code. Therefore as $G_2=G_1'$ we have shown that G_1 shunts.

Unfortunately though, shunting does not always guarantee the formation of a generator matrix of a KM$(n,k-1,d+1)$ code from the generator matrix of a KM(n,k,d) code, nor indeed does it guarantee the formation of a non KM$(n,k-1,d-1)$ code as the following example will show.

Example 5.4 Let $N=5$ and take $P(u)=u^2(u^3+1)$, $k=4$ and $d=2$. We thus have

$$\begin{aligned}Z(u) &= z_0+z_1u+z_2u^2+z_3u^3\\ Y(u) &= y_0+y_1u+y_2u^2.\end{aligned} \tag{5.23}$$

Now using the CRT we will form the generator matrix of a KM$(n, 4, 2)$ code.

(i) modulo u^2

$$\begin{aligned} Z_1(u) &= z_0 + z_1 u \pmod{u^2} \\ Y_1(u) &= y_0 + y_1 u \pmod{u^2}. \end{aligned} \tag{5.24}$$

Let $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ then

$$\Phi_1(u) \equiv m_0 + (m_0 + m_1 + m_2)u \pmod{u^2} \tag{5.25}$$

(ii) modulo $u^3 + 1$

$$\begin{aligned} Z_2(u) &= (z_0 + z_3) + z_1 u + z_2 u^2 \pmod{u^3 + 1} \\ Y_2(u) &= y_0 + y_1 u \pmod{u^3 + 1}. \end{aligned} \tag{5.26}$$

Let $m_3 = (z_0 + z_3)y_0$, $m_4 = z_1 y_1$, $m_5 = (z_0 + z_1 + z_3)(y_0 + y_1)$, $m_6 = (z_0 + z_2 + z_3)y_0$ and $m_7 = (z_1 + z_2)y_1$, then

$$\Phi_2(u) \equiv (m_3 + m_4 + m_7) + (m_3 + m_4 + m_5)u + (m_3 + m_4 + m_6)u^2 \pmod{u^3 + 1} \tag{5.27}$$

We must now find $R_i(u), i = 1, 2$.

(i) modulo u^2

$$\begin{aligned} R_1(u)P_2(u) &= 1 \pmod{P_1(u)} \\ R_1(u)(u^3 + 1) &= 1 \pmod{u^2} \end{aligned} \tag{5.28}$$

Therefore $R_1(u) = 1$.

(ii) modulo $u^3 + 1$

$$\begin{aligned} R_2(u)P_1(u) &= 1 \pmod{P_2(u)} \\ R_2(u)u^2 &= 1 \pmod{u^3 + 1} \end{aligned} \tag{5.29}$$

Therefore $R_2(u) = u$.

Now we have

$$\begin{aligned} \Phi(u) &= \Phi_1(u)R_1(u)Q_1(u) + \Phi_2(u)R_2(u)Q_2(u) \\ &= (m_0 + (m_0 + m_1 + m_2)u)(u^3 + 1) \\ &\quad +((m_3 + m_4 + m_7) + (m_3 + m_4 + m_5)u + (m_3 + m_4 + m_6)u^2)u(u^2) \\ &\equiv m_0 + (m_0 + m_1 + m_2)u + (m_3 + m_4 + m_6)u^2 \\ &\quad +(m_0 + m_3 + m_4 + m_7)u^3 + (m_0 + m_1 + m_2 + m_3 + m_4 + m_5)u^4 \pmod{P(u)}. \end{aligned} \tag{5.30}$$

Therefore we get from the computation $\Phi = A^T(G_1^T z \times B_1 y)$

$$G_1 = \left[\begin{array}{ccc|ccccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right], \tag{5.31}$$

the generator matrix of a KM$(8, 4, 2)$ code. Now if we shunt this matrix we get (since no wraparound was used we simply remove the bottom row) we get

$$G_1' = \left[\begin{array}{ccc|ccccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]. \tag{5.32}$$

It is obvious that this is not the generator matrix of a KM$(8, 3, 3)$ code (or indeed any linear $(8, 3, 3)$ code) as there is at least one codeword of weight two. Therefore G_1 does not shunt.

So we would like to know when shunting does indeed produce a KM$(n, k-1, d+1)$ code from a KM(n, k, d). Due to the complexity of the problem of when shunting does give a KM code we have to restrict ourselves to certain cases.

In Example 5.2, note that we have shunted in a blockwise fashion, in that the blocks are all remaining the same size, and the algorithm used (i.e. the $\Phi(u)$ in terms of the m_i) remains the same. We further define the ith block to be *independent* if for block i, $\Phi_i(u)$ requires only those multiplications that are associated with block i. Note that in Example 5.2 all the blocks are independent. However, shunting is not restricted to generator matrices only having independent blocks, and we will consider shunting for other cases later in this chapter. For now we are considering only when shunting occurs where the blocks are independent and the algorithm remains the same for the shunted and the original code.

We are now in a position to give the standard version of the shunting theorem, first of all proving a necessary lemma.

Lemma 5.5 *Let $k \geq 2$ and $d \geq D$ (=deg[P(u)]). Suppose that for certain constants k_{ij}, k-vectors $\boldsymbol{\lambda}_j$ and d-vectors $\boldsymbol{\rho}_j$, the multiplications $m_j = (\boldsymbol{z}\boldsymbol{\lambda}_j^T)(\boldsymbol{y}\boldsymbol{\rho}_j^T)$ yield the identity:*

$$Z(u)Y(u) \equiv \sum_{i=0}^{D-1} \left(\sum_{j=0}^{n-1} k_{ij} m_j \right) u^i \pmod{P(u)}. \tag{5.33}$$

Then there are $d+1$-vectors $\boldsymbol{\sigma}_j$ such that the following identity holds:

$$Z(u)(Y(u) + y_d u^d) \equiv \sum_{i=0}^{D-1} \left(\sum_{j=0}^{n-1} k_{ij} m'_j \right) u^i \pmod{P(u)}, \tag{5.34}$$

where $m'_j = (\boldsymbol{z}\boldsymbol{\lambda}_j^T)(\boldsymbol{y}|y_d\boldsymbol{\sigma}_j^T)$.

Proof. First assume (5.33) holds. Let $\boldsymbol{a} = (a_0 \dots a_{D-1})$ be a vector of indeterminants. Set $\boldsymbol{y}$ equal to $\boldsymbol{a}$ extended by $0's$ up to length d, and let $\boldsymbol{\mu}_j$ be the truncation of $\boldsymbol{\rho}_j$ down to length D. Crucially, the latter two operations are well defined because $d \geq D$. We now have $\boldsymbol{y}\boldsymbol{\rho}_j^T = \boldsymbol{a}\boldsymbol{\mu}_j^T$, and so

$$Z(u)(a_0 + a_1 u + \cdots + a_{D-1}u^{D-1}) = \sum_{i,j} k_{ij} (\boldsymbol{z}\boldsymbol{\lambda}_j^T)(\boldsymbol{a}\boldsymbol{\mu}_j^T) u^i. \tag{5.35}$$

Having dropped the y's, start again with $y_0, \dots, y_d$ and let $\sum_{i=0}^{D-1} a_i u^i$ be the reduction modulo $P(u)$ of $y_0 + y_1 u + \cdots + y_d u^d$. Then $\boldsymbol{a} = (y_0 \dots y_d)R$ for the $d+1$ by D matrix R given in Definition 2.11, and substituting for $\boldsymbol{a}$ in (5.35) gives (5.34). □

We will now give an example to show this lemma in action.

Example 5.6 Let $P(u) = u^2+1$, $Z(u) = z_0+z_1u+z_2u^2$ and $Y(u) = y_0+y_1u+y_2u^2+y_3u^3$. Now let

$$\begin{array}{lcl|lcl} m_0 &=& z_0 y_0 & m_5 &=& (z_0+z_2)(y_0+y_2) \\ m_1 &=& z_1 y_1 & m_6 &=& z_0 y_3 \\ m_2 &=& z_2 y_2 & m_7 &=& z_1 y_3 \\ m_3 &=& (z_0+z_1)(y_0+y_1) & m_8 &=& z_2 y_3 \\ m_4 &=& (z_1+z_2)(y_1+y_1) & & & \end{array} \tag{5.36}$$

then

$$\begin{array}{lcl|lcl} \phi_0 &=& m_0 & \phi_3 &=& m_1+m_2+m_4+m_6 \\ \phi_1 &=& m_0+m_1+m_3 & \phi_4 &=& m_2+m_7 \\ \phi_2 &=& m_0+m_1+m_2+m_5 & \phi_5 &=& m_8. \end{array} \tag{5.37}$$

Now we have

$$\begin{aligned} Z(u)Y(u) \pmod{P(u)} &= (\phi_0+\phi_2+\phi_4)+(\phi_1+\phi_3+\phi_5)u \\ &= (m_1+m_5+m_7)+(m_0+m_2+m_3+m_4+m_6+m_8)u \end{aligned} \tag{5.38}$$

so in the same way as Lemma 5.5 we let

$$k_{ij} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{5.39}$$

Let $\boldsymbol{a} = (a_0\ a_1)$, and set $\boldsymbol{y} = (a_0\ a_1\ 0\ 0)$ and let $\boldsymbol{\mu}_j$ be the truncation of $\boldsymbol{\rho}_j$ down to length 2, (where we obtain $\boldsymbol{\rho}_j$ in the same way as Lemma 5.5). So we have

$$\begin{array}{lcl|lcl} \boldsymbol{\rho}_0 &=& (1\,0\,0\,0) & \boldsymbol{\mu}_0 &=& (1\ 0) \\ \boldsymbol{\rho}_1 &=& (0\,1\,0\,0) & \boldsymbol{\mu}_1 &=& (0\ 1) \\ \boldsymbol{\rho}_2 &=& (0\,0\,1\,0) & \boldsymbol{\mu}_2 &=& (0\ 0) \\ \boldsymbol{\rho}_3 &=& (1\,1\,0\,0) & \boldsymbol{\mu}_3 &=& (1\ 1) \\ \boldsymbol{\rho}_4 &=& (0\,1\,1\,0) & \boldsymbol{\mu}_4 &=& (0\ 1) \\ \boldsymbol{\rho}_5 &=& (1\,0\,1\,0) & \boldsymbol{\mu}_5 &=& (1\ 0) \\ \boldsymbol{\rho}_6 &=& (0\,0\,0\,1) & \boldsymbol{\mu}_6 &=& (0\ 0) \\ \boldsymbol{\rho}_7 &=& (0\,0\,0\,1) & \boldsymbol{\mu}_7 &=& (0\ 0) \\ \boldsymbol{\rho}_8 &=& (0\,0\,0\,1) & \boldsymbol{\mu}_8 &=& (0\ 0). \end{array} \tag{5.40}$$

We now have $\boldsymbol{y}\boldsymbol{\rho}_j^T = \boldsymbol{a}\boldsymbol{\mu}_j^T$, as Lemma 5.5 requires . Further we have

$$Z(u)(a_0+a_1u) \equiv \sum_{i,j} k_{ij}(\boldsymbol{z}\boldsymbol{\lambda}_j^T)(\boldsymbol{a}\boldsymbol{\mu}_j^T)u^i \pmod{P(u)} \tag{5.41}$$

by letting $Y(u) = a_0 + a_1u$ and $m_j = (\boldsymbol{z}\boldsymbol{\lambda}_j^T)(\boldsymbol{a}\boldsymbol{\mu}_j^T)$, as done in Lemma 5.5.

So we have effectively set $y_0 = a_0, y_1 = a_1, y_2 = 0, y_3 = 0$ in both sides, which is possible as the y's are indeterminates.

Now extend the $Y(u)$ we originally stated to $Y(u) = y_0 + y_1u + y_2u^2 + y_3u^3 + y_4u^4$, and let $Y(u)$ modulo $P(u)$ equal $\sum_{i=0}^{1} a_iu^i$, then $\boldsymbol{a} = (y_0 \dots y_d)R$, where R is the reduction matrix of $P(u)$ (see Definition 2.11), i.e.

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \end{bmatrix} \tag{5.42}$$

for $P(u) = u^2 + 1$. Now substituting $\boldsymbol{a}$ into (5.41) gives

$$Z(u)((y_0 + y_2 + y_4) + (y_1 + y_3)u) = \sum_{i,j} (z\boldsymbol{\lambda}_j^T)([y_0 + y_2 + y_4 \;\; y_1 + y_3]\,\boldsymbol{\mu}_j^T)u^i \pmod{(P(u))} \tag{5.43}$$

and as we are working modulo $P(u) = u^2 + 1$ this gives the expected result, as shown by Lemma 5.5.

Theorem 5.7 (Shunting Theorem) *A KM (n, k, d) code with independent blocks may be shunted to a KM $(n, k-1, d+1)$ code if $k \geq 2$ and $d \geq max_t(deg[P_t(u)],\, s)$.*

our multiplications required

Proof. First consider the case $s = 0$. By the block structure of the KM(n, k, d) code we firstly have from $P(u) = P_1(u) \dots P_L(u)$, for each block t, $1 \leq t \leq L$,

$$\Phi_t(u) \equiv (z_0 + \cdots + z_{k-1}u^{k-1})(y_0 + \cdots + y_{d-1}u^{d-1}) \pmod{P_t(u)} \tag{5.44}$$

is reconstructible using a unique set (as the blocks are independent) of multiplications, $\{m_{t_0}, \dots, m_{t_{\nu_t}}\}$, say. Further by the CRT

$$\Phi(u) \equiv (z_0 + \cdots + z_{k-1}u^{k-1})(y_0 + \cdots + y_{d-1}u^{d-1}) \pmod{P(u)} \tag{5.45}$$

is reconstructible using all the multiplications, $m_i, i = 0, \dots, n-1$.

For the following part of the proof we will denote the application of Lemma 5.5 on $\Phi_t(u)$ $(= Z_t(u)Y_t(u))$ by $\Phi_t'(u)$ (and similarly for $\Phi(u)$ and $\overline{\Phi}(u)$).

Now since $d \geq \deg[P_t(u)], 1 \leq t \leq L$ we have by Lemma 5.5, for each block t, $1 \leq t \leq L$ that

$$\Phi'_t(u) \equiv (z_0 + \cdots + z_{k-1}u^{k-1})(y_0 + \cdots + y_d u^d) \pmod{P_t(u)} \tag{5.46}$$

is reconstructible using a new set of multiplications $\{m'_{t_0}, \ldots, m'_{t_{\nu_t}}\}$. Further by the CRT (Theorem 2.14)

$$\Phi'(u) \equiv (z_0 + \cdots + z_{k-1}u^{k-1})(y_0 + \cdots + y_d u^d) \pmod{P(u)} \tag{5.47}$$

is reconstructible using all the multiplications, $m'_i, i = 0, \ldots, n-1$.

Since $d \geq s$ is also given, a slight modification of Lemma 5.5 shows that if there is a set of multiplications $\{m_{(L+1)_0}, \ldots, m_{(L+1)_{\nu_{L+1}}}\}$ that reconstruct

$$\overline{\Phi}(u) \equiv (z_{k-1} + \cdots + z_0 u^{k-1})(y_{d-1} + \cdots + y_0 u^{d-1}) \pmod{u^s} \tag{5.48}$$

then there are multiplications $\{m'_{(L+1)_0}, \ldots, m'_{(L+1)_{\nu_{L+1}}}\}$ that reconstruct

$$\overline{\Phi}'(u) \equiv (z_{k-1} + \cdots + z_0 u^{k-1})(y_d + \cdots + y_0 u^d) \pmod{u^s} \tag{5.49}$$

and by using the Improved Version of the CRT (Theorem 2.22), (5.47) can be reconstructed using all the multiplications, $m'_i, i = 0, \ldots, n-1$, even if $s \neq 0$.

Therefore we can reconstruct $\Phi'(u)$ itself provided $\deg[\Phi'(u)] < D + s$ (see Theorem 2.22). But this inequality is achieved subject to the required $k + d = D + s + 1$, provided we set $z_{k-1} = 0$, which is precisely what we do in shunting. Applying Theorem 3.11 with k replaced with $k-1$ and d replaced by $d+1$ completes the proof. □

This proof gives sufficient parameters for shunting although in some other cases these parameters can be amended slightly. This proof is extremely general in its approach in that the only real condition is that the blocks of the generator are independent. If we look more carefully at some algorithms we can see that as long as the shunted generator matrix represents a genuine algorithm, then it is a valid KM code. Let's look at an example where we shunt a KM code and the independent block structure is lost, but the shunted matrix is still a KM code.

Example 5.8 If we let $P(u) = (u^2 + u + 1)u^4$, $s = 1$. Then we get for $k = 5$ (and hence $d = 3$) the generator matrix of the KM $(12, 5, 3)$ code as

$$G = \left[\begin{array}{ccc|cccccccc|c} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]. \tag{5.50}$$

If we shunt this once then we get

$$G = \left[\begin{array}{ccc|cccccccc|c} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right], \tag{5.51}$$

which is a KM $(12, 4, 4)$ code. This is no surprise given the next theorem.

Theorem 5.9 *If $P(u) = P_1(u)P_2(u)\dots P_L(u)$, with one of the $P_i(u) = u^m$, $m = k - 1$, and the rest having deg$[P_j(u)] \leq d$, $j \neq i$, and $s > 0$, then starting from $k = d + 2$ the code will shunt at least once.*

Proof. By Theoren 5.7 all the blocks $P_j(u)$, $j \neq i$ will shunt. The block $P_i(u) = u^m$ is the problem. We have

$$\begin{aligned} Z(u) &= z_0 + \cdots + z_{k-1}u^{k-1} \\ Y(u) &= y_0 + \cdots + y_{k-3}u^{k-3} \end{aligned}, \tag{5.52}$$

which reduce modulo u^m to give

$$\begin{aligned} Z(u) &\equiv z_0 + \cdots + z_{k-2}u^{k-2} \\ Y(u) &\equiv y_0 + \cdots + y_{k-3}u^{k-3} \end{aligned}. \tag{5.53}$$

So there is no multiplication $z_{k-2}y_{k-2}$, but this is needed when shunting takes place. If however we look at the wraparound block for $k = d$, i.e. $k - 1 = d - 1$ we see

$$\begin{aligned} \overline{Z}(u) &= z_{k-2} + \cdots + z_0u^{k-2} \\ \overline{Y}(u) &= y_{k-2} + \cdots + y_0u^{k-2} \end{aligned}. \tag{5.54}$$

Then reducing these modulo u^s, $s > 0$ will always result in a multiplication $z_{k-2}y_{k-2}$ as required in the reconstruction. □

We now give the polynomial explanation of why in Example 5.8 the generator matrix does indeed shunt.

Example 5.10 We have $P(u) = (u^2 + u + 1)u^4$, $s = 1$. Taking $k = 5$ and $d = 3$ we have

$$\begin{aligned} Z(u) &= z_0 + z_1u + z_2u^2 + z_3u^3 + z_4u^4 \\ Y(u) &= y_0 + y_1u + y_2u^2 \end{aligned} \tag{5.55}$$

we get the required multiplications can be

$$\begin{array}{lcl|lcl} m_0 &=& (z_0+z_2+z_3)(y_0+y_2) & m_6 &=& (z_0+z_1)(y_0+y_1) \\ m_1 &=& (z_1+z_2+z_4)(y_1+y_2) & m_7 &=& (z_0+z_2)(y_0+y_2) \\ m_2 &=& (z_0+z_1+z_3+z_4)(y_0+y_1) & m_8 &=& (z_0+z_3)y_0 \\ m_3 &=& z_0y_0 & m_9 &=& (z_1+z_2)(y_1+y_2) \\ m_4 &=& z_1y_1 & m_{10} &=& z_4y_2 \\ m_5 &=& z_2y_2 & & & \end{array} \tag{5.56}$$

and the CRT gives the reconstructed polynomial as

$$\begin{aligned} \Phi(u) &= m_3 + (m_3 + m_4 + m_6)u + (m_3 + m_4 + m_5 + m_8)u^2 + \\ &\quad (m_3 + m_4 + m_5 + m_8 + m_9)u^3 + \\ &\quad (m_1 + m_2 + m_3 + m_5 + m_6 + m_8 + m_9 + m_{10})u^4 + \\ &\quad (m_0 + m_1 + m_3 + m_7 + m_8 + m_9 + m_{10})u^5 + \\ &\quad m_{10}u^6. \end{aligned} \tag{5.57}$$

If we now wish to shunt this code then we will get:

$$\begin{aligned} Z(u) &= z_0 + z_1u + z_2u^2 + z_3u^3 \\ Y(u) &= y_0 + y_1u + y_2u^2 + y_3u^3 \end{aligned} \tag{5.58}$$

and the multiplications directly turn out to be fine apart from in the block where we reduce the $Z(u)$ and $Y(u)$ modulo u^4 where there is no multiplication z_3y_3, but this multiplication is actually the wraparound so the polynomial can be reconstructed, and hence by Theorem 3.11 results in a KM code.

5.3 Minimum Length Codes for $N \leq 4$

5.3.1 Preliminaries

One way of finding all the possible weight enumerators of KM codes for a particular N is to vary the following:

1. k and d subject to the obvious that k and d are positive and they satisfy $k+d-1 = N$.
2. $P(u)$, its factors and their number L.
3. The wraparound value s.
4. The choice of the multiplications for each algorithm.

Also if we are looking at minimum length codes then it will not always be possible to have independent blocks.

Definition 5.11 Let the number of blocks (and hence the number of coprime factors of $P(u)$ in the CRT) be L if no wraparound was used or $L+1$ if a wraparound was incorporated. We say that a generator matrix of a KM Code has *semi-independent blocks* if $\Phi_i(u), i = 1, \ldots L$, can be reconstructed using only those multiplications from the blocks less than or equal to i. Obviously if a wraparound was incorporated then $\Phi(u)$ and hence $\overline{\Phi}(u)$ and $\overline{\Phi}(u)$ modulo u^s can be constructed using all the blocks.

Before moving on we need to clarify our idea of obtaining minimum length KM codes.

5.3.2 Reducing the length of KM Codes

Due to the construction of KM codes being directly from the multiplications needed to multiply two polynomials, any reduction in the length of the generator matrix can only be achieved if multiplications can be removed from the reconstruction of the polynomial - for the new matrix to still be a generator matrix the reconstruction must still possible. Let us demonstrate this with a simple example.

Example 5.12 Let $N = 3$ and take $P(u) = u^3$, $k = 2$ and $d = 2$, then

$$\begin{aligned} Z(u) &= z_0 + z_1 u \\ Y(u) &= y_0 + y_1 u. \end{aligned} \tag{5.59}$$

Now

$$\begin{aligned} Z(u) &= z_0 + z_1 u \pmod{u^3} \\ Y(u) &= y_0 + y_1 u \pmod{u^3}. \end{aligned} \tag{5.60}$$

Let $m_0 = z_0 y_0$, $m_1 = z_1 y_1$ and $m_2 = (z_0 + z_1)(y_0 + y_1)$ then by the CRT we can get

$$\Phi(u) = m_0 + (m_0 + m_1 + m_2)u + m_1 u^2 \tag{5.61}$$

and we get the generator matrix of a KM$(3, 2, 2)$ code as

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{5.62}$$

Now looking at the matrix we may initially say that we can remove column two as it is the sum of columns zero and one. However removing column two means that m_2 has been removed from the reconstruction (5.61). Now using only m_0 and m_1 the polynomial will not reconstruct, hence the reduced G is not guaranteed to be the generator matrix of a KM$(2, 2, 2)$ code. It is easy to see that the minimum distance would in fact be one, violating our expectation that the minimum distance would be two.

5.3.3 Standard Form of the Generator Matrix

It is well known that the generator matrix of any (n, k, d) code may be converted to a standard form with the same weight enumerator as the original, so we can obtain all the possible weight enumerators if we look at all the standard forms. The valid operations for converting a generator matrix, G, to its standard form are

- reduce G to its unique reduced echelon form by elementary row operations, i.e. switching distinct rows i, j $(R_i \leftrightarrow R_j)$ or adding a row to another, $(R_i \rightarrow R_i + R_j)$.
- perform column interchanges $C_i \leftrightarrow C_j$.

We get

$$G \approx \left[\; I_k \; \middle| \; U \; \right]$$

where the $k \times (n-k)$ matrix U is determined up to the order of its columns, and $\approx$ denotes equivalent up to the two operations above.

5.3.4 Extremal Cases

For the extremal cases of $k = 1$, and $k = N$ we can easily find the weight enumerators as the following theorem shows:

Theorem 5.13 *For any value of N, in extremal cases we have $n = N$ and the weight enumerator of the codes is as follows:*
(a) $k = 1$, $G = [1^N]$ and $A(x) = 1 + x^N$,
(b) $k = N$, $G = [I_N]$ and $A(x) = (1 + x)^N$,
where 1^N represents a N-vector of $1's$, and I_N represents the $(N \times N)$ identity matrix.

Proof. For (a) we have $d = N$ and $k = 1$, so from a KM point of view we have $Z(u)Y(u) = z_0(y_0 + \cdots + y_{N-1}u^{N-1})$, so the number of multiplications is exactly N. Thus $G = [1^N]$. For (b) we have $d = 1$ and $k = N$, so $Z(u)Y(u) = (z_0 + \cdots + z_{N-1}u^{N-1})y_0$, again using N multiplications. This time however $k = N$ and we get $G = [I_N]$. The number of codewords of weight r then equals the coefficient of x^r in $(1 + x)^N$. □

5.3.5 The Cases $N = 1, 2, 3$

We first investigate the cases $N = 1, 2, 3$ as these are the easiest, in fact they can be simply stated and proved in the following theorem.

Theorem 5.14 *The following are the only possible weight enumerators for KM codes for $N = 1, 2, 3$:*
(a) $N = 1$, $k = 1$, $d = 1$, $G \approx [1]$, $A(x) = 1$,
(b) $N = 2$, $k = 1$, $d = 2$, $G \approx [11]$, $A(x) = 1 + x^2$, $N = 2$, $k = 2$, $d = 1$, $G \approx [I_2]$, $A(x) = (1 + x)^2$,

(c) $N = 3$, $k = 1$, $d = 3$, $G \approx [111]$, $A(x) = 1+x^3$, $N = 3$, $k = 2$, $d = 2$, $G \approx \left[\begin{array}{cc|c} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right]$, $A(x) = 1 + 3x^2$, $N = 3$, $k = 3$, $d = 1$, $G \approx [I_3]$, $A(x) = (1 + x)^3$.

Proof. All follow from Theorem 5.13 except (c) $N = 3$, $k = 2$, $d = 2$. We have in this case $Z(u)Y(u) = (z_0+z_1u)(y_0+y_1u)$, and we know that the minimum number of multiplications for this is 3, hence $G \approx [I_2|U]$, with $U = \left[\begin{array}{cc} u_0 & u_1 \end{array}\right]^T$, but $d = 2$ implies that $u_0 = u_1 = 1$. □

5.3.6 The Case $N = 4$

As the length of the codes for the extremal cases is 4, for the other cases, i.e. $k = 2$, $d = 3$ and $k = 3$, $d = 2$, we must have the length greater than or equal to 4. We will show now that for both of these cases the minimum number of multiplications is actually 5.

Lemma 5.15 *For the case* $N = 4$, *with* $k = 3$, $d = 2$ *and* $k = 2$, $d = 3$ *we have that* $n_{min} = 5$.

Proof. We will do the case $k = 3$, $d = 2$. The other case follows similarly from this. We can write the problem as the following:

$$\begin{aligned} \begin{bmatrix} z_0y_0 \\ z_0y_1 + z_1y_0 \\ z_1y_1 + z_2y_0 \\ z_2y_1 \end{bmatrix} &= \begin{bmatrix} z_0 & 0 \\ z_1 & z_0 \\ z_2 & z_1 \\ 0 & z_2 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \\ &= E(\boldsymbol{z})\boldsymbol{y}. \end{aligned} \tag{5.63}$$

We know the problem can be done in five multiplications, so it remains to show that four multiplications are not sufficient. Suppose $m_0, \ldots m_3$ suffice, then the right hand side of (5.63) equals

$$U\,[m_0 \ldots m_3]^T,$$

where U is a 4×4 matrix over $\mathbb{F}_2$. Since the 4 forms of (5.63) are linearly independent, so are the 4 rows of U, which therefore has an inverse W, say. Now denoting the ith row of

λ	μ	w_0	w_1	w_2	w_3
1	0	0	0	0	arbitrary
0	1	arbitrary	0	0	0
1	1	w_1	w_2	w_3	arbitrary

Table 5.1: Table of all admissible values for λ and μ.

W by $\boldsymbol{w} = [w_0 \dots w_3]$ we have

$$\boldsymbol{w}E(\boldsymbol{z})\boldsymbol{y} = m_i \tag{5.64}$$

of complexity 1, and hence the inequalities

$$1 = M(\boldsymbol{w}E(\boldsymbol{z})\boldsymbol{y}) \geq \text{col. rank of } \boldsymbol{w}E(\boldsymbol{z}). \tag{5.65}$$

So for some scalars λ, μ not both zero

$$\lambda(w_0z_0 + w_1z_1 + w_2z_2) = \mu(w_1z_0 + w_2z_1 + w_3z_2), \tag{5.66}$$

which implies by equating the coefficients of z_0, z_1, z_2 that

$$\lambda w_0 = \mu w_1, \quad \lambda w_1 = \mu w_2, \quad \lambda w_2 = \mu w_3. \tag{5.67}$$

Using all admissible values of λ and μ we get Table 5.1.

Therefore there does not exist a valid 4×4 matrix W, hence 4 multiplications are not sufficient. □

Now we are in a position to look at all the weight enumerators of $(5, 3, 2)$ and $(5, 2, 3)$ codes. The case $(5, 2, 3)$ is easily shown in the next theorem.

Theorem 5.16 *The only weight enumerator for a* $(5, 2, 3)$ *code is* $A(x) = 1 + 2x^3 + x^4$.

Proof. The standard form of the generator matrix will be of the form

$$G \approx [I_2|U] \tag{5.68}$$

where the rows of U are 3-vectors u_0, u_1, say. Since the minimum weight is three we must have that $u_0 \neq u_1$, and u_0 and u_1 have weight at least two. Thus $u_0, u_1 \in \{101, 110, 011, 111\}$.

It is now easy to see that the four vectors formed from the linear combinations of the rows have weights $3, 3, 4$ in some order, and so the weight enumerator is as stated. □

Obviously therefore any KM code that is formed with the parameters $(5, 2, 3)$ will have a weight enumerator as above.

The case for the $(5, 3, 2)$ code is more interesting as more weight enumerators are possible as we shall now see.

Theorem 5.17 *The following are the only possible weight enumerators for a* $(5, 3, 2)$ *code.*

$A_{w_{min}}$	$A(x)$
2	$1 + 2x^2 + 4x^3 + x^4$
3	$1 + 3x^2 + 3x^3 + x^5$
4	$1 + 4x^2 + 3x^4$

Proof. First of all we know that the standard form of the generator matrix is of the type

$$G \approx [I_3|U], \tag{5.69}$$

where the rows of U are 2-vectors u_0, u_1, u_2. Note here that these vectors may be permuted amongst themselves, or the two columns of U may be swapped without affecting the weight enumerator. Now we have that the minimum distance possible is 2 so the $u_i, i = 0, 1, 2$ are all non-zero. They need not be distinct but obviously a column of zeros is forbidden, so if identical they must be 11. By a case by case analysis we can see in Table 5.2 that five distinct cases are possible (up to the permutation and column swapping as explained above).

It is now easy to obtain the weight enumerators as stated above, (1) and (2) having $A_{w_{min}} = 2$, (3) and (4) having $A_{w_{min}} = 3$ and (5) having $A_{w_{min}} = 4$. □

When designing codes we often ask for the largest possible d for a given n and k. In this thesis we would further like the number of codewords of weight d to be as low as possible, thus to have the maximum number of codewords with weight larger than d, leading to error correction in practice better than that guaranteed by the value of d. We thus get the following definition.

Case	u_0, u_1, u_2
(1)	10, 01, 11
(2)	10, 11, 11
(3)	10, 10, 11
(4)	11, 11, 11
(5)	10, 10, 01

Table 5.2: Table of all possible vectors (up to the equivalence explained above) of u_i for $(5, 3, 2)$ codes in standard form.

Definition 5.18 We say that the weight enumerator of a (n, k, d) code is *optimal* if $A_{w_{min}}$ is the minimum for all codes with the parameters n, k, d.

Also to be complete we would like to know when for all the possible choices 2–4 in Section 5.3.1 what weight enumerators are formed and when is it possible to convert a non-optimal weight enumerator code to an optimal weight enumerator code. For the current value of N (i.e. $N = 4$) we can easily see that there are only a few choices of $P(u)$, s and the multiplications when using the CRT such that the number of multiplications is the minimum (i.e. 5).

5.3.7 Using the CRT to form KM $(5, 3, 2)$ codes

In Table 5.3 we give the weight enumerators of all the KM codes that are directly formed (without any amendment of the multiplications) from the CRT for the parameters $N = 4, k = 3, d = 2$. Note Table 5.3 is complete as no other factorisation of $P(u)$ together with $s = 0, 1$ or 2 will, using the CRT, result in a KM code of length 5, using the parameters $k = 3$ and $d = 2$.

We can see from Table 5.3 that all the possibilities of $P(u)$ and s result in KM code with the optimal weight enumerator for the parameters used. We can however choose $P(u)$ and s and try to find linearly dependent multiplications to remove. The next example will illustrate this.

$P(u)$	s	Weight Enumerator, $A(x)$
$u(u+1)(u^2+u+1)$	0	$1+2x^2+4x^3+x^4$
$u(u^2+u+1)$	1	$1+2x^2+4x^3+x^4$
$u(u^2+1)$	1	$1+2x^2+4x^3+x^4$
$(u+1)u^2$	1	$1+2x^2+4x^3+x^4$
$(u+1)(u^2+u+1)$	1	$1+2x^2+4x^3+x^4$
$u(u+1)$	2	$1+2x^2+4x^3+x^4$

Table 5.3: The Weight Enumerators for KM codes where $N=4$ formed by the CRT

Example 5.19 With $N=4, k=3, d=2$, take $P(u)=u^3$ and $s=1$. This initially appears to need the following 6 multiplications

$$\begin{array}{lcl|lcl} m_0 &=& z_0y_0 & m_3 &=& (z_0+z_2)y_0 \\ m_1 &=& z_1y_1 & m_4 &=& (z_1+z_2)y_1 \\ m_2 &=& (z_0+z_1)(y_0+y_1) & m_5 &=& z_2y_1, \end{array} \tag{5.70}$$

and in order to keep a separate block for each $P(u), i=1,2$ we could choose m_0, m_1, m_2, m_3 and m_5 as these are sufficient to reconstruct (see Definition 2.19) $\Phi(u)(=m_0+(m_0+m_1+m_2)u+(m_0+m_1+m_3)u^2+m_5u^3)$ giving (using the same construction as in Example 4.16)

$$G=\left[\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right], \tag{5.71}$$

a different generator matrix than can be formed by Table 5.3 but still giving the optimal weight enumerator. This technique has removed the guarantee of independent blocks but still results in a valid KM code as the polynomial $\Phi(u)=Z(u)Y(u)$ can be reconstructed.

Of course it is not necessary to start by selecting a $P(u)$ and s then use the CRT, we may in fact be able to see a minimal algorithm straight away. The method of using the CRT to find algorithms (and hence generator matrices) is a very powerful one in that the selection of the multiplications is a recursive one, but we may use any of the algorithms

found in Chapter 4 (with maybe some amendments as explained in Section 4.1). For now we will concentrate on using the CRT.

Example 5.20 Let $N = 4, k = 3, d = 2$, $P(u) = u^2 + 1$ and $s = 2$. We thus get by the CRT

$$\begin{array}{rcl|rcl} m_0 & = & (z_0 + z_2)y_0 & m_3 & = & z_2 y_1 \\ m_1 & = & z_1 y_1 & m_4 & = & z_1 y_0 \\ m_2 & = & (z_0 + z_1 + z_2)(y_0 + y_1) & m_5 & = & (z_1 + z_2)(y_0 + y_1), \end{array} \tag{5.72}$$

and

$$\Phi(u) = (m_0+m_1+m_3+m_4+m_5)+(m_0+m_1+m_2+m_3)u+(m_3+m_4+m_5)u^2+m_3u^3 \tag{5.73}$$

But this takes six multiplications and they are all linearly independent. We can however get the reconstruction to work using the first three multiplications and another two if we choose carefully. Take $m_3' = (z_0 + z_1)(y_0 + y_1)$ and $m_4' = (z_1 + z_2)y_1$, then with m_0, m_1, m_2 as (5.72) we get

$$\begin{array}{rcl} \Phi(u) & = & (m_0 + m_1 + m_2 + m_3' + m_4') + (m_0 + m_2 + m_4')u \\ & & +(m_2 + m_3' + m_4')u^2 + (m_1 + m_4')u^3. \end{array} \tag{5.74}$$

The blocks are semi-independent (see Definition 5.11) as we can always get $\overline{\Phi}(u) \pmod{u^s}$ if we have $\Phi(u)$. We thus get the following generator matrix

$$G = \left[\begin{array}{ccc|cc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{array}\right] \tag{5.75}$$

with weight enumerator $A(x) = 1 + 2x^2 + 4x^3 + x^4$.

We extend this idea further to have a suitable set of multiplications such that the blocks are at least semi-independent and all the codes have optimal weight enumerator.

5.3.8 Multiplications for Optimal $A(x)$

As proved before in Lemma 5.15 we can actually achieve the product $\Phi(u) = (z_0 + z_1u + z_2u^2)(y_0 + y_1u)$ in 5 multiplications directly, but it is not easy to tabulate. In this section

we consider a set of multiplications such that the algorithms can be built up. Also we are insisting that the blocks are semi-independent for the given $P(u)$. First of all let us define some notation relating the multiplications and columns of the generator matrices formed from them.

Definition 5.21 Define the following

(i) If we have a set of multiplications $m_{i_0}, m_{i_1}, \ldots, m_{i_n}$, then we abbreviate the sum $m_{i_0} + m_{i_1} + \cdots + m_{i_n}$ as $m_{i_0 i_1 i_2 \ldots i_n}$,

(ii) The columns of G are denoted $\boldsymbol{g}_i^T$ and can be defined by

$$m_i = (\boldsymbol{z}\boldsymbol{g}_i^T)(\boldsymbol{y}\boldsymbol{h}_i^T) \tag{5.76}$$

where

$$\boldsymbol{z} = (z_0 \ldots z_{k-1}) \quad \text{and} \quad \boldsymbol{y} = (y_0 \ldots y_{d-1}). \tag{5.77}$$

(iii) Similarly to item (i), the generator matrix $[G_{i_0} G_{i_1} \ldots G_{i_n}]$ is written as $[G_{i_0 i_1 \ldots i_n}]$.

Table 5.4 contains a set of multiplications which give the optimal weight enumerator, $A(x)$, together with a set of alternatives m_i^* and m_i^{**} such that if exactly one m_i is replaced by an m_i^* or m_i^{**} the optimality of the weight enumerator remains. For example if we use m_0, m_1, m_2, m_3 and m_4 then we get the optimal weight enumerator, similarly we may use m_0^*, m_1, m_2, m_3 and m_4 to obtain the optimal weight enumerator. However, if we use m_0, m_1, m_2, m_3 and m_3^* then we cannot even reconstruct the polynomial $\Phi(u)$. The * and ** carry over to the abbreviating notation given in Definition 5.21.

Note: i) as we stated before once $\Phi(u)$ is found we automatically obtain $\overline{\Phi}(u) \pmod{u^s}$ for any s. For example, if $\Phi(u) = m_0 + (m_0 + m_1 + m_2)u + (m_0 + m_1 + m_3)u^2 + (m_1 + m_4)u^3$ then $\overline{\Phi}(u) = (m_1 + m_4) + (m_0 + m_1 + m_3)u + (m_0 + m_1 + m_2)u^2 + m_0 u^3$ and this can be reduced to the required s.

ii) calculating $\Phi(u) = Z(u)Y(u) \pmod{u^2 + a_1 u + a_0}$ requires three multiplications except in the case when $a_1 = 1, a_0 = 0$ as $u^2 + u = u(u + 1)$ and by the CRT this requires only

Optimality Set	Alternative 1	Alternative 2
$m_0 = z_0 y_0$	$m_0^* = (z_0 + z_1 + z_2)(y_0 + y_1)$	
$m_1 = z_1 y_1$	$m_1^* = (z_0 + z_1 + z_2)(y_0 + y_1)$	
$m_2 = (z_0 + z_1)(y_0 + y_1)$		
$m_3 = (z_0 + z_2)y_0$	$m_3^* = z_2 y_0$	$m_3^{**} = z_2(y_0 + y_1)$
$m_4 = (z_1 + z_2)y_1$	$m_4^* = z_2 y_1$	$m_4^{**} = z_2(y_0 + y_1)$

Table 5.4: Set of multiplications that can be used to form KM $(5, 3, 2)$ codes that have optimal weight enumerator. Note that $m_1^* = m_0^*$ and $m_4^{**} = m_3^{**}$.

two, i.e. $m_0 = z_0 y_0$ and $m_0^* = (z_0 + z_1 + z_2)(y_0 + y_1)$. To generalise this idea slightly $Z(u)Y(u) \pmod{u^2 + a_1 u + a_0} = m_0' + (m_0' + m_1' + m_2')u + m_1'(a_1 u + a_0)$ where

$$\begin{array}{rcl|rcl} m_0' & = & (z_0 + a_0 z_2)y_0 & m_2' & = & (z_0 + z_1 + a_0 z_2)(y_0 + y_1). \\ m_1' & = & (z_1 + a_0 z_2)y_1 & & & \end{array} \tag{5.78}$$

We are now in a position to give all the generator matrices for KM $(5, 3, 2)$ codes for all admissible choices of $P(u) = P_1(u) \ldots P_L(u)$ and s. They can be found in Table 5.5. A little explanation of the table is necessary before we display and prove its contents.

The multiplications are chosen to satisfy that the blocks of the generator matrix are semi-independent, and that $A(x)$ is optimal. Q_i stands for an arbitrary monic polynomial of degree i. Note however that as ZY modulo Q_3, where $Q_3 \neq u^3$ takes more multiplications than ZY modulo u^3 we consider this case separately and hence $Q_3 \neq u^3$.

Option	L	s	$P_1 \dots P_L$	Generator Matrix	Ind. Blocks	k_{min}
1	1	0	Q_4	$[G_{0123}G_0^*]$	Yes	1
$2a$	2	0	uQ_3	$[G_0\|G_{1234}]$	No	2
$2b$			$(u+1)Q_3$	$[G_0^*\|G_{1234}]$	No	2
$2c$			$(u+1)u^3$	$[G_0^*\|G_{0123}]$	Yes	1
$2d$			$u^2(u^2+1)$	$[G_{012}\|G_0^*G_3]$	No	1
$2e$			$(u^2+1)(u^2+u+1)$	$[G_{13}G_0^*\|G_{24}]$	No	2
$2f$			$u^2(u^2+u+1)$	$[G_{012}\|G_{34}]$	No	2
$2g$			$(u^2+u)(u^2+u+1)$	$[G_0G_0^*\|G_{234}]$	Yes	1
3	3	0	$u(u+1)(u^2+u+1)$	$[G_0\|G_0^*\|G_{234}]$	Yes	1
$4a$	1	1	Q_3	$[G_4^*\|G_{0123}]$ (wrap first)	No	1
$4b$			u^3	$[G_{0123}\|G_4^*]$	Yes	1
$5a$	2	1	uQ_2	$[G_0\|G'_{012}\|G_4^*]$	Yes	1
$5b$			$(u+1)Q_2$	$[G_0^*\|G'_{012}\|G_4^*]$	Yes	1
$6a$	1	2	u^2	$[G_{012}\|G_{34}]$	No	2
$6b$			u^2+1	$[G_{13}G_0^*\|G_{24}]$	No	2
$6c$			u^2+u+1	$[G_{234}\|G_0G_0^*]$	No	3
$6d$			u^2+u	$[G_0G_0^*\|G_{14}G_3^*]$	Yes	1
7	2	2	$u(u+1)$	$[G_0\|G_0^*\|G_{14}G_3^*]$	Yes	1
$8a$	1	3	u	$[G_0\|G_{134}G_4^*]$	Yes	1
$8b$			$u+1$	$[G_0^*\|G_{134}G_4^*]$	Yes	1

Table 5.5: Generator Matrices G for KM $(5,3,2)$ codes for all admissible choices of $P(u) = P_1(u)\dots P_L(u)$ and s

5.3.9 Multiplications for Non-Optimal $A(x)$

A natural question to further the study of the weight enumerator is to ask whether all the other weight enumerators for the given n, k, d can be found as KM codes. For the case $k = 3, d = 2$ here it is easy to see that the other weight enumerators are possible. See the next example for proof of this.

Example 5.22 (i) $P(u) = Q_4$, then the algorithm can be

$$\begin{array}{lll} m_0 = z_0 y_0 & & \\ m_1 = z_1 y_1 & & \\ m_2 = (z_0 + z_1)(y_0 + y_1) & G = [G_{012} G_3^* G_4^*] & (5.79) \\ m_3 = z_2 y_0 & & \\ m_4 = z_2 y_1 & A(x) = 1 + 4x^2 + 3x^4, & \end{array}$$

(ii) or the algorithm could be

$$\begin{array}{lll} m_0 = z_1 y_1 & & \\ m_1 = (z_0 + z_1)(y_0 + y_1) & & \\ m_2 = (z_0 + z_2) y_0 & G = [G_{123} G_3^* G_4^*] & (5.80) \\ m_3 = z_2 y_0 & & \\ m_4 = z_2 y_1 & A(x) = 1 + 3x^2 + 3x^3 + x^5. & \end{array}$$

5.4 Shunting of the KM$(5, 3, 2)$ codes

Theorem 5.7 says that each KM$(5, 3, 2)$ code will shunt to a KM$(5, 2, 3)$ and a KM$(5, 1, 4)$ code if the blocks are independent and $d \geq \max(\deg[P_t(u)], s)$. This happens for Options 2c, 2g, 3, 4b, 5a, 5b, 6d, 7, 8a and 8b of Table 5.5. Theorem 5.9 doesn't hold in the case $k = 3, d = 2$. Notice that in Option 1 of Table 5.5 even though we have an independent block Theorems 5.7 and 5.9 do not hold so it is fortuitous(over our theorems) that this generator matrix shunts to $k = 1$. Another algorithm that can be used for $P(u) = Q_4$ is $m_0 = z_0y_0, m_1 = z_1y_1, m_2 = (z_0 + z_1)(y_0 + y_1), m_3 = (z_0 + z_2)y_0, m_4 = (z_1 + z_2)y_1$. These multiplications will form a valid algorithm, the block is independent (obviously!) but it only shunts to $k = 2$.

5.5 Minimum Length Codes for $N = 5$

We now move onto the harder case and try to construct all the minimum length codes for $N = 5$. As will be found in this chapter this problem is much harder than the $N < 5$ cases. We continue with the knowledge of Section 5.3.1 – 4.

5.5.1 Standard Form of the Generator Matrix

In order to find all the weight enumerators that may be possible via KM construction we must find all the weight enumerators for any kind of linear binary code. We do this in the same way Section 5.3.3 by looking at the standard form of the generator matrix.

For $N = 5$ we can have the following possibilities for k and d.

$$\begin{aligned} k &= 4 \ , \ d = 2 \\ k &= 3 \ , \ d = 3 \\ k &= 2 \ , \ d = 4. \end{aligned} \tag{5.81}$$

We must first find out the minimum length that these codes can have and for this we return to the polynomial approach of KM.

Theorem 5.23 *The minimum number of multiplications needed to multiply two polynomials* $Z(u) = z_0 + \cdots + z_{k-1}u^{k-1}$ *and* $Y(u) = y_0 + \cdots + y_{d-1}u^{d-1}$ *for the values given in (5.81) is 6 for each of the three cases.*

Proof. This is done in a similar way to Lemma 5.15. □

Now the cases for $k = 3, d = 3$ and $k = 2, d = 4$ are the simplest and so shall be dealt with first.

Theorem 5.24 *The only possible weight enumerators for* $(6, 3, 3)$ *linear codes and* $(6, 2, 4)$ *linear codes are*

$$\begin{aligned} A(x) &= 1 + 4x^3 + 3x^4 \\ A(x) &= 1 + 3x^4, \end{aligned} \tag{5.82}$$

respectively.

Proof. First consider the case for the $(6,3,3)$ linear code. Looking at the standard form of the generator matrix we get

$$G = [I_3|U], \tag{5.83}$$

with $U = \begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \\ w_0 & w_1 & w_2 \end{bmatrix}$. Now by similar arguments to those given in Theorem 5.17, we can see that the only possibilities for $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$ are

Case	$\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}$
(1)	$111, 110, 011$
(2)	$110, 011, 101$

all giving the weight enumerator of $A(x) = 1 + 4x^3 + 3x^4$. For the $(6,2,4)$ linear code we have the standard form of the generator matrix as

$$G = [I_2|U] \tag{5.84}$$

with $U = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 \\ v_0 & v_1 & v_2 & u_3 \end{bmatrix}$, easily giving that the only possible vectors $\boldsymbol{u}, \boldsymbol{v}$ are $\boldsymbol{u}, \boldsymbol{v} = 1110, 0111$, and so the only possible weight enumerator is $A(x) = 1 + 3x^4$. □

From Theorem 5.23 we know that KM codes exist for $k = 3, d = 3$ and $k = 2, d = 4$ of length six and so by Theorem 5.24 must have the weight enumerators as stated.

Now the case for the weight enumerators of a $(6,4,2)$ linear code is far more interesting as the following theorem will show.

Theorem 5.25 *The only possible weight enumerators of a* $(6,4,2)$ *linear code are*

$A_{w_{min}}$	$A(x)$
3	$1 + 3x^2 + 8x^3 + 3x^4 + x^6$
4	$1 + 4x^2 + 6x^3 + 3x^4 + 2x^5$
6	$1 + 6x^2 + 4x^3 + x^4 + 4x^5$
	$1 + 6x^2 + 9x^4$
7	$1 + 7x^2 + 7x^4 + x^6$

Proof. For the values of n, k, d we have the standard form of the generator matrix is

$$G = [I_4|U] \tag{5.85}$$

with $U = \left[u^{(0)}u^{(1)}u^{(2)}u^{(3)}\right]^T$, and $u^{(i)} = (u_0^{(i)}, u_1^{(i)}), i = 0, 1, 2, 3$. Now, as in Theorem 5.17, we get the following table for the $u^{(i)}$ and the corresponding weight enumerator:

Case	$u^{(0)}$	$u^{(1)}$	$u^{(2)}$	$u^{(3)}$	$A(x)$
(1)	11	11	11	11	$1+6x^2+4x^3+x^4+4x^5$
(2)	11	11	11	01	$1+4x^2+6x^3+3x^4+2x^5$
(3)	10	10	10	11	$1+6x^2+4x^3+x^4+4x^5$
(4)	10	10	10	01	$1+7x^2+7x^4+x^5$
(5)	11	11	01	10	$1+3x^2+8x^3+3x^4+x^5$
(6)	10	10	01	11	$1+4x^2+6x^3+3x^4+2x^5$
(7)	11	11	01	01	$1+4x^2+6x^3+3x^4+2x^5$
(8)	10	10	01	01	$1+6x^2+9x^4$

giving all the possible weight enumerators as stated. □

As in Section 5.3-4, we would like to be able to to show that firstly the optimal weight enumerator can be found for a KM code and secondly what other weight enumerators can be found from KM codes.

5.5.2 Direct use of the CRT to form KM $(6, 4, 2)$ Codes

Below we display as examples, the only two KM codes that are formed directly from the CRT (i.e. without any amendments to the multiplications) such that the code length, n, is six. For each we have $N = 5, k = 4, d = 2$.

Example 5.26 Let $P(u) = (u+1)u^3$ and $s = 1$. Then using Appendix 1 we get the multiplications as follows:

$$\begin{array}{lcl|lcl} m_0 & = & (z_0+z_1+z_2+z_3)(y_0+y_1) & m_3 & = & (z_0+z_1)(y_0+y_1) \\ m_1 & = & z_0y_0 & m_4 & = & (z_0+z_2)y_0 \\ m_2 & = & z_1y_1 & m_5 & = & z_3y_1, \end{array} \tag{5.86}$$

where m_0 is from $u+1$, $m_1, \ldots, m_4$ are from u^3 and m_5 is from the wraparound. This gives the generator matrix as

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, A(x) = 1 + 4x^2 + 6x^3 + 3x^4 + 2x^5. \tag{5.87}$$

Example 5.27 Let $P(u) = u(u+1)$ and $s = 3$. Then using Appendix 1 we get the multiplications as follows:

$$\begin{array}{lcl|lcl} m_0 & = & z_0 y_0 & m_3 & = & z_2 y_0 \\ m_1 & = & (z_0 + z_1 + z_2 + z_3)(y_0 + y_1) & m_4 & = & (z_2 + z_3)(y_0 + y_1) \\ m_2 & = & z_3 y_1 & m_5 & = & (z_1 + z_3) y_1, \end{array} \tag{5.88}$$

where m_0 is from u, m_1 is from $u+1$ and $m_2, \ldots, m_5$ are from the wraparound. This gives the generator matrix as

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}, \quad A(x) = 1 + 4x^2 + 6x^3 + 3x^4 + 2x^5. \tag{5.89}$$

This, in the first instance does not look too good as the optimal weight enumerator does not occur using this method. However, as we shall see soon the optimal weight enumerator can be found by using the diagrammatic techniques introduced in Chapter 4. Before we use this method let us show what other weight enumerators are possible by certain amendments of the CRT in order to reduce the number of multiplications necessary.

Rules for multiplication amending

1. If the multiplication occurs in any other part of the algorithm then one of them can be omitted.
2. If three multiplications are linearly dependent then one of them can be omitted.

3. If two multiplications m_a, m_b, say, always occur together in the reconstruction of $\Phi(u)$, then we might be able to invent a new multiplication, m_{n+1}, say, such that m_{n+1} together with c, say, of the other multiplications, not including m_a and m_b gives the sum of the original two, i.e.

$$m_a + m_b = \sum_{1 \le j \le c, i_j \neq a,b} m_{i_j} + m_{n+1}. \tag{5.90}$$

4. If when we look at the reconstruction of $\Phi(u)$ a multiplication is not needed, it may be left out.

Using these rules we are able to use the CRT to construct algorithms where the number of multiplications needed is greater than six and then possibly reduce the number needed to six (while still being able to reconstruct $\Phi(u)$). Obviously this may not always be possible. We give a comprehensive example to explain the intricacies of some of these rules, then using the algorithms of Appendix 1, we shall show all the options for $P(u)$ and s such that the number of multiplications can be reduced to six.

Example 5.28 We have $N = 5, k = 4, d = 2$. Let $P(u) = (u^2+1)(u^3+u^2+u)$, so using Appendix 2 we have, firstly from reducing modulo u^2+1 the multiplications

$$\begin{array}{rcl|rcl} m_0 &=& (z_0+z_2)y_0 & m_2 &=& (z_0+z_1+z_2+z_3)(y_0+y_1) \\ m_1 &=& (z_1+z_3)y_1 & & & \end{array} \tag{5.91}$$

and secondly from reducing modulo u^3+u^2+u the multiplications

$$\begin{array}{rcl|rcl} m_3 &=& z_0y_0 & m_6 &=& (z_1+z_2)y_1 \\ m_4 &=& (z_1+z_3)y_1 & m_7 &=& (z_0+z_2+z_3)y_0. \\ m_5 &=& (z_0+z_1+z_3)(y_0+y_1) & & & \end{array} \tag{5.92}$$

But finding $\Phi_2(u)$ does not need m_4 so it can be omitted at this stage. Now the rest of the multiplications are linearly independent so Rules (1) and (2) do not apply. We need to look at the reconstruction if we want to try and apply Rules (3) and (4). We have

$$\begin{array}{rcl|rcl} \phi_0 &=& m_3 & \phi_3 &=& m_2+m_3+m_5+m_7 \\ \phi_1 &=& m_0+m_1+m_3+m_5+m_7 & \phi_4 &=& m_0+m_1+m_2+m_3+m_5+m_6. \\ \phi_2 &=& m_2+m_5+m_6 & & & \end{array} \tag{5.93}$$

All the multiplications are used at least once and so Rule (4) cannot be used, but we see that m_0 and m_1 always occur together, so we might be able to use Rule (3). Indeed, if we define $m_8 = z_3 y_1$, then $m_0 + m_1 = m_8 + m_2 + m_3 + m_5 + m_6$ and the reconstruction can be achieved using $m_2, m_3, m_5, m_6, m_7, m_8$, giving the generator matrix as

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \quad A(x) = 1 + 4x^2 + 6x^3 + 3x^4 + 2x^5. \tag{5.94}$$

Further to the above example we are able to use the coprime factorisation of all polynomials of the required degree from Appendix 2, and the algorithms of Appendix 1 to form Table 5.6.

We have thus shown by our standard algorithms of Appendix 1 that only two of the possible weight enumerators for any $(6, 4, 2)$ linear codes are possible as KM codes. We now move on to using the diagrammatic representation of Chapter 4.

5.5.3 Using the Diagrammatic Representation for $N = 5$

We will first show the diagrams for the examples in the previous sub-section. Firstly the algorithm of Example 5.26 is displayed in Figure 5.1, then in Figure 5.2 can be found the diagrammatic representation of the algorithm of Example 5.27. Figure 5.3 contains the diagrammatic representation of Example 5.28.

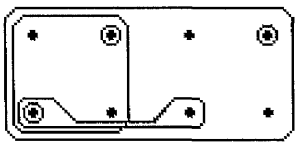

Figure 5.1: Diagram of the algorithm for Example 5.26

Also we can reduce our square, $P(u)$ algorithm of Section 4.6 to the present problem. Indeed if we take the $k = 4, d = 4$ case of Figure 4.14 and reduce it to $k = 4, d = 2$ we get Figure 5.4.

$P(u)$	s	$A(x)$
$(u^2+1)(u^3+u^2+u)$	0	$1+4x^2+6x^3+3x^4+2x^5$
$u^2(u^3+u^2+u+1)$	0	$1+6x^2+9x^4$
$u(u+1)(u^2+u+1)$	1	$1+4x^2+6x^3+3x^4+2x^5$
$(u^2+u)(u^2+u+1)$	1	$1+4x^2+6x^3+3x^4+2x^5$
$u(u^3+1)$	1	$1+4x^2+6x^3+3x^4+2x^5$
$(u+1)(u^3+u^2+u)$	1	$1+4x^2+6x^3+3x^4+2x^5$
$u^2(u^2+1)$	1	$1+4x^2+6x^3+3x^4+2x^5$
$(u+1)u^3$	1	$1+4x^2+6x^3+3x^4+2x^5$
$u(u^3+u^2+u+1)$	1	$1+4x^2+6x^3+3x^4+2x^5$
u^3	2	$1+6x^2+9x^4$
$u(u^2+1)u^3$	2	$1+4x^2+6x^3+3x^4+2x^5$
u^3+u	2	$1+4x^2+6x^3+3x^4+2x^5$
$u^2(u+1)$	2	$1+4x^2+6x^3+3x^4+2x^5$
(u^3+u^2)	2	$1+4x^2+6x^3+3x^4+2x^5$
u^3+u^2+u+1	2	$1+4x^2+6x^3+3x^4+2x^5$
u^2	3	$1+6x^2+9x^4$
u^2+1	3	$1+6x^2+9x^4$
$u(u+1)$	3	$1+4x^2+6x^3+3x^4+2x^5$
(u^2+u)	3	$1+4x^2+6x^3+3x^4+2x^5$

Table 5.6: Weight Enumerators of all the KM codes formed by the CRT with amending rules as above

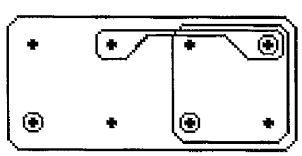

Figure 5.2: Diagram of the algorithm for Example 5.27

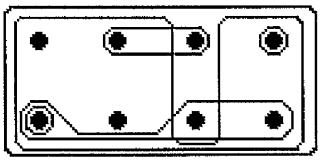

Figure 5.3: Diagram of the algorithm for Example 5.28

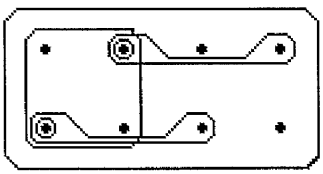

Figure 5.4: Reduced version of our square, $P(u)$ algorithm

The algorithm of Figure 5.4 has a weight enumerator associated with it as $A(x) = 1 + 6x^2 + 9x^4$.

Now we use the operations described in Figures 4.7-10 to obtain other algorithms. This may enable us to find some other weight enumerators for KM codes. It is noted here that operation four (as described in Figure 4.10) will not change the weight enumerator as it is the same as rotating the generator matrix by 180 degrees. It turns out that up to the four allowed operations there are two equivalence classes of algorithms with typical representatives as in Figure 5.5.

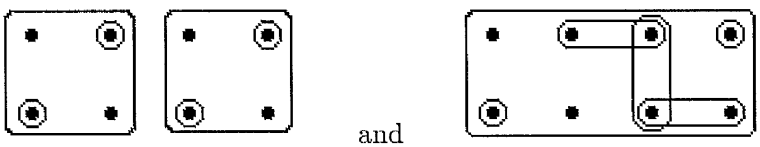

Figure 5.5: The two typical representatives of the equivalence classes

Looking at the weight enumerators associated with each algorithm of the two classes we see for example, that the algorithm in Figure 5.6 gives the optimal weight enumerator, $A(x) = 1 + 3x^2 + 8x^3 + 3x^4 + x^6$, for the given values $k = 4, d = 2$.

Note here that this is not the only algorithm (obviously since one of the operations is operation four) with the optimal weight enumerator associated with it. The results obtained

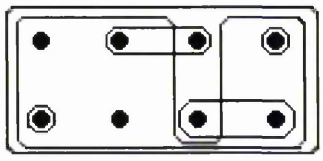

Figure 5.6: Algorithm with the optimal weight enumerator

here lead us to state the following.

Theorem 5.29 *Using the algorithms of Appendix 1, the CRT construction and the operations described in Figures 4.7-10, the only weight enumerators of KM codes for* $k = 4, d = 2$ *are*

$$\begin{aligned} A(x) &= 1 + 3x^2 + 8x^3 + 3x^4 + x^6 \\ A(x) &= 1 + 4x^2 + 6x^3 + 3x^4 + 2x^5 \\ A(x) &= 1 + 6x^2 + 9x^4. \end{aligned}$$

We also pose the following.

Conjecture 5.30 *For* $k = 4, d = 2$, *the only weight enumerators that KM codes can have are those given in Theorem 5.29.*

So we have shown that for $N \leq 5$ there exist KM codes of minimum length possible for the given k and d having optimal weight enumerator.

5.6 KM codes for $N \geq 6$

The techniques used in the previous sections results in data that requires more time than available, and will be attempted at a later stage.

Chapter 6

The Weight Enumerators of Shunted Families of KM Codes, I

6.1 Introduction and Dual Codes

If we wish to find the weight enumerator of a KM(n, k, d) code and if k is large but $n - k$ is small, it is computationally more efficient to calculate the weight enumerator of the dual KM code (an $(n, n-k, d')$ linear code) and then use the MacWilliams Identities (equations (2.2)) to dualise back to the weight enumerator of the primary (original) KM code.

As stated in Definition 2.1 we need to find $n - k$ independent relations on the codeword entries $c_0 \ldots c_{n-1}$, say, as this will then enable us to find the dual KM code and hence the weight enumerator of the dual KM code.

We will develop a method of finding the weight enumerators of families of KM codes. In this chapter we will consider families to mean codes with the same $P(u)$ but with a range of values of s, starting with $s = 0$. We will study the effect on the weight enumerator as s increases.

6.2 Further Ideas on Reduction

We extend here the ideas of Section 2.3.2 in order to apply them directly to the present material. In the same style as Definition 2.11, if we have $X(u)$ of degree $h-1$ and we reduce it modulo $P(u)$ of degree D, $(D < h-1)$, to $a(u)$ then obviously $a(u)$ will have degree $D-1$ and there are constants r_{ij} such that

$$u^j \pmod{P(u)} = \sum_{i=0}^{D-1} r_{ij} u^i. \tag{6.1}$$

Hence we have that

$$a(u) \pmod{P(u)} = \sum_{j=0}^{h-1} x_j \sum_{i=0}^{D-1} r_{ij} u^i, \tag{6.2}$$

giving

$$a_i = \sum_{j=0}^{h-1} x_j r_{ij}. \tag{6.3}$$

Again we may write this in the matrix form $\boldsymbol{a} = \boldsymbol{x}R$, where $R = [r_{ij}]$ is $h \times D$. We will explain a little further after noting the shift in the rows of R, i.e.

$$\text{row } i+1 \text{ of } R = \begin{bmatrix} 0 & r_{i,0} & \dots & r_{i,D-2} \end{bmatrix} + r_{i,D-1} \begin{bmatrix} p_0 & p_1 & \dots & p_{D-1} \end{bmatrix}$$

where the p_i are the coefficients of $P(u)$. Now if we have an algorithm for the product modulo $P(u)$ of two polynomials $a(u)$ and $b(u)$ of degree $D-1$ we have the multiplications $m_i = (\boldsymbol{a}\boldsymbol{\lambda}_i^T)(\boldsymbol{b}\boldsymbol{\rho}_i^T)$, where $\boldsymbol{\lambda}_i^T$ and $\boldsymbol{\rho}_i^T$ are d - vectors. We can form a vector $\boldsymbol{a}S$, with $S = [\boldsymbol{\lambda}_0^T \boldsymbol{\lambda}_1^T \dots]$ for the $\boldsymbol{a}$ part of the multiplication.

Note here that the first D columns of S commonly form the identity matrix I_D. Now if $Z(u)$ reduces modulo $P_t(u)$ to $a(u)$ then from Definition 2.11 we have

$$\boldsymbol{a} = \boldsymbol{z}R, \tag{6.4}$$

where R is defined as above, and so $\boldsymbol{a}S = \boldsymbol{z}RS$, and finally we see that

$$G_t = RS \tag{6.5}$$

where $\boldsymbol{z}R$ represents reduction modulo $P_t(u)$.

We are now in a position to use these techniques to assist our goal of finding the weight enumerators.

The Periodicity of R and G_t

We look a little further at the structure of R and G_t and note some periodicity properties that are exhibited by them. For the following we shall consider $L = 1$ only, although for the case $L > 1$ we simply apply the $L = 1$ case to each block. First a definition concerning period is needed.

Definition 6.1 The following are equivalent definitions of e, *the order of* u *modulo* $P(u)$

- e is the least positive integer, such that $u^e \equiv 1 \pmod{P(u)}$,
- e is the least positive integer such that $P(u)|(u^e - 1)$.

Now, row i of R is the vector $\begin{bmatrix} r_{i0} & \dots & r_{i,D-1} \end{bmatrix}$, so we have from 6.1

$$\text{row } i = \text{row } j \iff u^i \equiv u^j \pmod{P(u)} \iff e|(j-i). \tag{6.6}$$

In particular the rows (if they are present) of R repeat with period e, and hence so do the rows of $G_t = RS$.

6.3 Obtaining the $n-k$ Independent Relations

If we encode the information word $\boldsymbol{z} = z_0 \dots z_{k-1}$ to the codeword $\boldsymbol{c} = c_0 \dots c_{n-1}$ then the equation relating these two can of course be written

$$\boldsymbol{z}G = \boldsymbol{c}. \tag{6.7}$$

We would like to find the z_i in terms of the c_i as this will then give at least some of the required relations. Returning to the polynomial approach we have

Theorem 6.2 *(a) The coefficients in* $\hat{Z}(u)$ $(= Z(u) \pmod{P(u)})$ *are linear combinations of* $\boldsymbol{z}G$,

(b) The rank of G_t *equals the degree* D_t *of* $P_t(u)$

Proof. (a) First all by the definition of a computation we know that the multiplications, m_i, giving block G_t, say $a \le i \le b$, satisfy

$$Z_t(u)Y_t(u) = \sum_{j=0}^{D_t-1} \left(\sum_{i=a}^{b} k_{ij} m_i \right) u^j \pmod{P_t(u)} \tag{6.8}$$

for certain constants k_{ij}, where as we know from (5.76) $m_i = (\boldsymbol{z}\boldsymbol{g}_i^T)(\boldsymbol{y}\boldsymbol{h}_i^T)$, $\boldsymbol{g}_i^T$ being the ith column of G and $\boldsymbol{h}_i^T$ being a d-vector. In particular, we may set $y_0 = 1$ and $y_i = 0, i = 1, \dots, d$, so that $Y_t(u) = 1$ and

$$Z_t(u) = \sum_{j=0}^{D_t-1} \left(\sum_{i=a}^{b} l_{ij} (\boldsymbol{z}\boldsymbol{g}_i^T) \right) u^j \tag{6.9}$$

where l_{ij} equals k_{ij} multiplied by the 0th entry of H_j. Since equality holds for each t, the CRT shows that the coefficients in $\hat{Z}(u)$ are linear combinations of the $\boldsymbol{z}\boldsymbol{g}_i^T$, $(0 \le i \le n-1)$.
(b) There are D_t coefficients of powers of u in (6.9), hence $D_t \le$column rank of G_t. But the latter cannot exceed D_t, since each $\boldsymbol{z}\boldsymbol{g}_i^T$ is a linear combination of the D_t columns of the matrix $\boldsymbol{z}R$, with R as in (6.4). This proves (b). □

We will separate the relations we can find into three distinct categories for ease of understanding.

Definition 6.3 (i) A linear relation between entries c_i is of *inner* type if it corresponds to a relation between columns of the same block.

(ii) If $s = 0$ then the expressions for $z_k, \ldots, z_{D-1}$ must all equal zero, giving $D - k = k + d - 1 - k = d - 1$ relations known as *outer relations.* If $s > 0$ then the outer relations refer to $\hat{z}_i$ and Theorem 6.4 is used to convert these back to the z_i's.

(iii) If $s > 0$ then we can get inner relations from the wraparound block. These are known as *wraparound relations* and are converted to relations among the c_i's by equating columns in (6.7)

Theorem 6.4 *Let $P(u)$ have reciprocal $Q(u) = q_0 + q_1 u + \cdots + q_E u^E$, of degree E. Define $Q_j(x) = q_0 + q_1 x + \cdots + q_j x^j$, $0 \le j \le E$. Then*

(a) with coefficients d_i independent of k,

$$\hat{z}_i = z_i, \quad \text{if} \quad 0 \le i < D - E \quad \text{case } D > E \tag{6.10}$$

$$\hat{z}_{D-j} = z_{D-j} + \sum_{i \ge 0} d_i z_{D+i} \quad (D + i < k), \quad \text{if} \quad 1 \le j \le E, \tag{6.11}$$

where $d_n = q_1 d_{n-1} + q_2 d_{n-2} + \cdots + q_E d_{n-E}$ with initial values $d_{-s} = \delta_{sj}$, $(1 \le s \le E)$.

(b) Let $G_j(x) = \sum_{r \ge 0} d_{r-j} x^r = d_{-j} + d_{-j+1} x + \cdots$. Then $G_j(x) = \frac{Q_{j-1}(x)}{Q(x)}$.

Proof. (a) By definition $\hat{Z}(u)$ satisfies $Z(u) = \hat{Z}(u) + a(u)P(u)$ for some polynomial $a(u) = \sum_{i=0}^{k-1-D} a_i u^i$. Writing this in matrix form we get

$$\begin{bmatrix} z_0 - \hat{z}_0 \\ z_1 - \hat{z}_1 \\ \vdots \\ z_{D-1} - \hat{z}_{D-1} \\ z_D \\ \vdots \\ z_{m+D} \end{bmatrix} = \begin{bmatrix} q_D & 0 & 0 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ q_{D-1} & q_D & 0 & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ q_{D-2} & q_{D-1} & q_D & 0 & \dots & \dots & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & & & & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & & & & \vdots \\ q_1 & q_2 & q_3 & \dots & q_D & 0 & & & & 0 \\ q_0 & q_1 & q_2 & q_3 & \dots & q_D & 0 & & & 0 \\ \vdots & \ddots & & & & & \ddots & & & \vdots \\ 0 & & \ddots & & & & & q_{D-2} & q_{D-1} & q_D \\ 0 & & & \ddots & & & & & q_{D-2} & q_{D-1} \\ \vdots & & & & & & & & \vdots & \vdots \\ 0 & & & & & & & \ddots & q_0 & q_1 \\ 0 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 & q_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix},$$

$$= Ma \tag{6.12}$$

but the definition of $Q(x)$ implies that the first $D-E$ rows of M are zero so $z_i - \hat{z}_i = 0$ for $i = 0, \dots D-E$.

(b) Let M_i denote the polynomial equivalent of row i of M. We thus have

$$M_{D-j} = x^{-j}(Q(x) - q_0 - \cdots - q_{j-1}x^{j-1}), \quad 1 \le j \le D \tag{6.13}$$

and as each of the first $D-1$ rows is a linear combination of the last m rows we get

$$M_{D-j} = \sum_{i \ge 0} d_i x^i Q(x), \quad 1 \le j \le D. \tag{6.14}$$

Equating equations (6.13) and (6.14)

$$\begin{aligned} x^{-j}(Q(x) - q_0 - \cdots - q_{j-1}x^{j-1}) &= \sum_{i\geq 0} d_i x^i Q(x) \\ -x^{-j}(q_0 + \cdots + q_{j-1}x^{j-1}) &= -x^{-j}Q(x) + \sum_{i\geq 0} d_i x^i Q(x) \\ x^{-j}Q_{j-1}(x) &= \left(\sum_{i\geq 0} d_i x^i + x^{-j}\right) Q(x) \\ &= \left(\sum_{i\geq -D} d_i x^i\right) Q(x), \end{aligned} \tag{6.15}$$

where $d_{-D}\ldots d_{-1} = 0\ldots 010\ldots 0$, the 1 being in position $-j$.

Now equating coefficients of x^{n+j} $(n \geq 0)$ in the last equation of (6.15) we have

$$x^{-j}Q_{j-1}(x) = \left(\sum_{i\geq -D} d_i x^i\right) Q(x). \tag{6.16}$$

Expanding this we get

$$\begin{aligned} q_0x^{-j} + q_1x^{-j+1} + \cdots + q_jx^{-1} &= (d_{-D}x^{-D} + d_{-D+1}x^{-D+1} + \cdots \\ &\quad \cdots + d_{-j}x^{-j} + \cdots + d_{-1}x^{-1} + \cdots) \\ &\quad (q_0 + q_1x + \cdots + q_Ex^E) \\ &= q_0d_{-D}x^{-D} + (q_0d_{-D+1} + q_1d_{-D})x^{-D+1} + \\ &\quad (q_0d_{-D+2} + q_1d_{-D+1} + q_2d_{-D})x^{-D+2} + \cdots \\ &\quad \cdots + (q_0d_n + q_1d_{n-1} + \cdots + q_Ed_{n-E})x^n + \cdots \end{aligned} \tag{6.17}$$

as $q_{E+1}, \ldots$ are zero. Now equating coefficients of x^{n+j} $(n \geq 0)$ we get

$$0 = q_0d_n + q_1d_{n-1} + \cdots + q_Ed_{n-E} \tag{6.18}$$

and as $q_0 = 1$ we get

$$d_n = q_1d_{n-1} + \cdots + q_Ed_{n-E} \tag{6.19}$$

as required.

For (b) look again at (6.15). We have

$$Q_{j-1}(x) = \left(\sum_{i\geq -D} d_i x^{i+j}\right) Q(x). \tag{6.20}$$

Now as $d_{-D} \dots d_{-j-1} = 0 \dots 0$ this can be written

$$Q_{j-1}(x) = \left(\sum_{i \geq 0} d_{i-j} x^i \right) Q(x), \tag{6.21}$$

i.e.

$$G_j(x) = \frac{Q_{j-1}(x)}{Q(x)}, \tag{6.22}$$

and that completes the proof. □

We now give some more results on the $G_i(x)$ and the $Q_i(x)$.

Theorem 6.5 *(a)* $G_E(x) = 1 - x^E G_1(x)$, *hence* $G_1(x)$ *and* $G_E(x)$ *have period ord[*$Q(x)$*] and satisfy* $d_0 = 1$,

(b) If $q_j = 0$ *then* $G_{j+1}(x) = G_j(x)$.

Proof. (a) $G_E(x) = \frac{Q_{E-1}(x)}{Q(x)} = \frac{Q(x) - x^E}{Q(x)} = 1 - x^E \frac{1}{Q(x)} = 1 - x^E G_1(x)$.
(b) Let $q_j = 0$, then $Q_j(x) = Q_{j-1}(x)$, so $\frac{Q_j(x)}{Q(x)} = \frac{Q_{j-1}(x)}{Q(x)}$, i.e. $G_{j+1}(x) = G_j(x)$. □

Theorem 6.6 *Let* $G_j(x)$ *have period* e, *fixed* j, *then*

(a) $e |$ *ord[*$Q(x)$*],*

(b) $e > max\left(\left[\frac{E}{2}\right], j-1, E-j\right)$.

Proof. (a) Let $e' = \text{ord}[Q(x)]$, so $\frac{1}{Q(x)}$ has a period e'. Therefore,

$$\begin{aligned} &\Leftrightarrow && \tfrac{1}{Q(x)}(1 + x^{e'}) && \text{is a polynomial} \\ &\Leftrightarrow && \tfrac{Q_{j-1}(x)}{Q(x)}(1 + x^{e'}) && \text{is a polynomial} \\ &\Leftrightarrow && \tfrac{Q_{j-1}(x)}{Q(x)} && \text{has period } e' \end{aligned} \tag{6.23}$$

and as period $G_j(x) = e$, it follows that $e |$ ord$[Q(x)]$.
(b) We know $G_j(x) = \frac{Q_{j-1}(x)}{Q(x)}$. Now the maximum degree of $G_j(x)$ is $E-j$ and ord$[G_j(x)] =$ period $[G_j(x)] \geq \deg[G_j(x)]$, so $e > E - j$.

There cannot be j zeros otherwise no recurrence would occur, so $e > j - 1$.

Amongst initial values $d_{-E}, \dots, d_{-1}$ there exists a run of at least $\left[\frac{E}{2}\right]$, but no run of e zeros occurs, else all $d_i = 0$, therefore $e > \left[\frac{E}{2}\right]$. □

6.3.1 The Matrix Method

If $s = 0$, then if we let $\boldsymbol{z} = (z_0 \dots z_{D-1})$ we are able to obtain the $\boldsymbol{z}$ by matrix inversion.

We know by the CRT, Theorem 2.14, the equation $\boldsymbol{z}G = \boldsymbol{c}$ has a unique solution for $\boldsymbol{z}$ in terms of the $\boldsymbol{c}$. We shall reduce G to an invertible matrix. Now, the columns of $G_t = RS$ are linear combinations of those of R. Since by Theorem 6.2 the rank of G_t is D_t we may, by rearranging if necessary, take the first D_t columns as independent. Indeed, for simplicity we shall assume that, as is usually the case, these first D_t columns are R itself, and write

$$\underline{G} = [R_1 R_2 \dots R_L], \tag{6.24}$$

where R_i corresponds to reduction $P_i(u)$. Now the equation $\boldsymbol{z}\underline{G} = \underline{\boldsymbol{c}}$ is obtained from $\boldsymbol{z}G = \boldsymbol{c}$ by deleting certain columns of G and their counterparts in $\boldsymbol{c}$, and so has a unique solution for $\boldsymbol{z}$ in terms of $\underline{\boldsymbol{c}}$. But since $\underline{G}$ is a $D \times D$ matrix, uniqueness implies that $\underline{G}$ is invertible, and $\boldsymbol{z} = \underline{\boldsymbol{c}}\underline{G}^{-1}$. Thus with the same entry deletions indicated by underlining,

$$\left[\underline{\zeta}_0^T \cdots \underline{\zeta}_{D-1}^T\right] = \underline{G}^{-1}, \tag{6.25}$$

where $z_i = \boldsymbol{\zeta}_i \boldsymbol{c}^T, (0 \leq i \leq D-1)$.

The method of matrix inversion is most useful and applicable when finding $\underline{G}$ is easier than the CRT calculation. A type of code fairly easily handled by this type of calculation is given in the following definition of which the idea was first introduced in Hoggar and Pickavance (1995), and formally defined in Hoggar (1997).

Definition 6.7 If the degree of $P_t(u)$ is a constant w for each $1 \leq t \leq L$ we say the resulting KM code has *constant degree* or it is a *constant degree KM code.* We apply this definition for all values of s.

Since the $P_t(u)$ must all be coprime, one way when considering constant degree KM codes, is to let $P_1(u) = u^w, P_2(u) = (u^w + 1)$ and take $P_t(u)$ as an irreducible of degree w for $t > 2$. Note here that the definition allows a wider choice than this, but for our purposes this shall be kept rigid.

We will pursue the rest of this chapter concerning ourselves with the constant degree KM codes with $w = 2$, the work can be relatively easily generalised.

Example 6.8 Looking at the constant degree KM code with $w = 2$ and $L = 3$ (the maximum for the given w) we must have $P(u) = u^2(u^2+1)(u^2+u+1)$. So

$$G = \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right], \tag{6.26}$$

and so the reduced matrix $\underline{G}$ is

$$\underline{G} = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right]. \tag{6.27}$$

If $\underline{G}$ has six rows then it is square and as explained before, by the CRT, it has an inverse $\underline{G}^{-1}$. This can easily be obtained as

$$\underline{G}^{-1} = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]. \tag{6.28}$$

Then $\boldsymbol{z} = \underline{\boldsymbol{c}}\underline{G}^{-1}$ or alternatively going back from the underlined matrix to the original,

$z = cG^{(-1)}$, where

$$G^{(-1)} = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] = \left[\zeta_0^T \dots \zeta_5^T\right]. \tag{6.29}$$

So the outer relations (see Definition 6.3 (ii)) can be obtained.

6.4 The KM$(9, k, 7-k)$ Codes

As in Example 6.8 we will use $P(u) = u^2(u^2+1)(u^2+u+1)$. We have three inner relations (see Definition 6.3 (i))

$$\begin{aligned} E_1 &= 111000000 \\ E_2 &= 000111000\ , \\ E_3 &= 000000111 \end{aligned} \tag{6.30}$$

and the outer relations (see Definition 6.3 (ii)) $\zeta_k, \ldots, \zeta_{D-1}$ obtained from (6.29).

Before continuing with the study we will give some general notation from vector space theory, that will help in the explanation of the work in this section and subsequent ones.

Definition 6.9 Let U be the space of n-vectors over $\mathbb{F}_2$, viewed as sequences of m triples (from the number of blocks) $\boldsymbol{u} = \boldsymbol{u}_1\boldsymbol{u}_2\ldots\boldsymbol{u}_m$ followed by $n-3m$ single digits. As will be shown in subsequent sections the number of single digits can conveniently be kept below three. Now if V is any subset of U, then define the following

E	The span of the set $\{E_1, E_2, \ldots, E_m\}$, where E_i is the generalised version of (6.30).
$\boldsymbol{u}+V$	The *translate* of V by $\boldsymbol{u}$, i.e. the set $\{\boldsymbol{u}+\boldsymbol{v} : \boldsymbol{v} \in V\}$.
S_k	The span of the set $\{\zeta_k, \zeta_{k+1}, \ldots, \zeta_{D-1}\}$. Note this is considered a subset of U and is extended by zeros up to length n. Here $0 \le k \le D-1$, and if $k > D-1$ or $k < 0$ then $S_k = \{0\}$.
$\zeta_{i_1 i_2 \ldots i_n}$	The sum of the vectors $\zeta_{i_1}, \zeta_{i_2}, \ldots, \zeta_{i_n}$.
$t(\boldsymbol{u})$	The number of nonzero triples of $\boldsymbol{u}$, i.e. $\#\{\boldsymbol{u}_i \neq 000 : i = 1, 2, \ldots, m\}$. Thus $t(\boldsymbol{0}) = 0$.
$t_i(V)$	The number of vectors $\boldsymbol{u} \in V$, with $t(\boldsymbol{u}) = i$, i.e. $\#\{\boldsymbol{u} \in V : t(\boldsymbol{u}) = i\}$.
R(V)	The weight enumerator of V.

Now back to the present problem, we have $m = 3$ and $D = 6$. As we need $9-k$ independent relations to form the dual code, and as we are only concerned with the weight enumerator, it does not matter which relations we use. We thus can say that the dual code

is generated by $E_1, E_2, E_3, \zeta_k, \ldots, \zeta_5$, or

$$\text{Dual KM}(9, k, 7-k) \text{ code} = \bigcup_{u \in S_k} (u + E) \tag{6.31}$$

Now we could simply obtain the dual code by obtaining the whole set from (6.31) and looking at each vector, but we would like a more systematic approach. The next few lemmas prove vital in taking the problem away from looking at every single vector in the dual code.

Lemma 6.10 *For vectors of length* $3m$, *we have* $R(u + E) = (x + x^2)^{t(u)}(1 + x^3)^{m-t(u)}$

Proof. Split each vector of $u + E$ into m triples, i.e. $u_1 + E_1, u_2 + E_2, \ldots, u_m + E_m$. Then letting $V_i = u_i + E_i$ we have

$$u + E = V_1 \times V_2 \times \cdots \times V_m \tag{6.32}$$

and

$$R(u + E) = R(V_1)R(V_2) \ldots R(V_m). \tag{6.33}$$

Now by the construction of the ζ_i the third digit of each triple is 0, so u_i is one of 000,100,010,110. Hence $R(V_i) = 1 + x^3$ if $u_i = 000$ or $x + x^2$ otherwise, and $R(u + E)$ is as stated. □

Note here that $R(E) = (1 + x^2)^3$ since the only vector u such that $u + E = E$ consists of three triple zeros.

We have in Lemma 6.10 the ability to find the weight enumerator, $B(x)$, of the dual code if we obtain the extra relevant relations. These extra relations can arise via Definition 6.3 (ii), so for $k = 5$ we have $z_5 = 0$ leading to ζ_5 being a relation and we have thus

$$\begin{aligned} \text{Dual KM}(9, 5, 2) \text{ code} &= \bigcup_{u \in S_5} (u + E) \\ &= E \cup (\zeta_5 + E). \end{aligned} \tag{6.34}$$

For $k = 4$ we have $z_4 = 0$ and $z_5 = 0$ leading to ζ_5, ζ_4 and their sum, ζ_{45}, being the required generators so

$$\begin{aligned} \text{Dual KM}(9, 4, 3) \text{ code} &= \bigcup_{u \in S_4} (u + E) \\ &= E \cup (\zeta_5 + E) \cup (\zeta_4 + E) \cup (\zeta_{45} + E). \end{aligned} \tag{6.35}$$

Now using Lemma 6.10 we can find the weight enumerators using this method, for example for the KM$(9,5,2)$ code

$$B(x) = R(E) + R(\zeta_5 + E),$$

and as $\zeta_5 = 110100010$, $t(\zeta_5) = 3$ and

$$\begin{aligned} B(x) &= (1+x^3)^3 + (x+x^2)^3 \\ &= 1 + 4x^3 + 3x^4 + 4x^6 + 3x^5 + x^9. \end{aligned} \tag{6.36}$$

Now if we let $\beta_t = (x+x^2)^t(1+x^3)^{3-t}$ then for the dual KM$(9,k,7-k)$ code

$$B(x) = \sum_{u \in S_k} \beta_{t(u)} \tag{6.37}$$

and using the MacWilliams identities (see Theorem 2.6 and the subsequent paragraph) we can obtain the weight enumerator of the KM$(9,k,7-k)$ code as

$$\begin{aligned} A(x) &= \tfrac{1}{2^{6-k}} \sum_{u \in S_k} \alpha_{t(u)} \\ &= \tfrac{1}{2^{6-k}} \left(a_0\alpha_0 + a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3\right), \text{say,} \end{aligned} \tag{6.38}$$

where $\alpha_t = (1-x^2)^t(1+3x^2)^{3-t}$, the MacWilliams transform of β_t.

The following lemma will enable us to produce a table of weight enumerators of the KM$(9,k,7-k)$ codes.

Lemma 6.11 *With $t(\boldsymbol{u})$ defined as in Definition 6.9 we have for $2 \le k \le 5$*

$$t(\boldsymbol{u}) = \begin{cases} 3 & \text{for } \boldsymbol{u} = \boldsymbol{z}_j, 2 \le j \le 5, \text{and } \boldsymbol{z}_{23}, \boldsymbol{z}_{45} \\ 2 & \text{otherwise} \end{cases} \tag{6.39}$$

Proof. By inspection of the vectors $\boldsymbol{z}_i, 2 \le i \le 5$ we see that all the sums except $\boldsymbol{z}_k, \boldsymbol{z}_{23}$ and $\boldsymbol{z}_{45}$ result in only one 000 appearing. □

In Table 6.1 we can see the weight enumerators of the primary and dual KM$(9,k,7-k)$ codes, together with the a_i, the coefficients of α_i in $A(x)$, as in (6.38). Note we also include the case $k=1, d=6$ as the results from this will be used for a more complex problem later on in this chapter. The results for this particular case are shown in Lemma 6.12, they are either obtained by direct methods or using the results of Chapter 7.

Lemma 6.12 *For $k = 1$ we must use ζ_1 which will introduce vectors with $t(\boldsymbol{u}) = 1$, and we get for this value of k*

$$a_0 = 1, a_1 = 3, a_2 = 15, a_3 = 13. \tag{6.40}$$

k	d	a_0	a_1	a_2	a_3	$A(x)$	$B(x)$
5	2	1	0	0	1	$1+3x^2+15x^4+13x^6$	$1+4x^3+3x^4+3x^5+4x^6+x^9$
4	3	1	0	0	3	$1+9x^4+6x^6$	$1+6x^3+9x^4$
3	4	1	0	3	4	$1+3x^4+4x^6$	$1+3x^2+13x^3+15x^4$
2	5	1	0	9	6	$1+3x^6$	$1+9x^2+27x^3+27x^4$
1	6	1	3	15	13	$1+x^6$	$1+3x+18x^2+46x^3+60x^4+60x^5$ $+46x^6+18x^7+3x^8+x^9$

Table 6.1: Weight enumerators of KM$(9, k, 7-k)$ codes

6.4.1 Explicit Formulae for the A_i

We need a systematic way of obtaining $a_i(d)$. As a first step we deduce a compact expression for the A_i from (6.38). We have

$$\alpha_t = \sum_{r=0}^{3} b_{tr} x^{2r} \quad \text{where} \quad [b_{tr}] = \begin{bmatrix} 1 & 9 & 27 & 27 \\ 1 & 5 & 3 & -9 \\ 1 & 1 & -5 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix} \tag{6.41}$$

and

$$A(x) = 2^{k-6} \sum_{u \in S_k} \alpha_{t(u)} = 2^{1-d} \left(a_0\alpha_0 + a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3\right). \tag{6.42}$$

Now we know $a_0 = 1$ and $a_0 + a_1 + a_2 + a_3 = 2^{d-1}$, i.e. $a_1 + a_2 + a_3 = 2^{d-1} - 1$. So for $d \geq 2$ we have $A_0 = 1, A_1 = 0$ and equating the constant coeffients of (6.42) gives

$$1 = 2^{k-6}(1 + a_1 + a_2 + a_3), \tag{6.43}$$

the same as we found above.

Lemma 6.13 *With $t(\boldsymbol{u})$ defined as in Definition 6.9, (a) $t(\boldsymbol{u}) \neq 1$ for $\boldsymbol{u} \in S_k$ and $k \geq 2$, (b) $t(\boldsymbol{u}) \neq 2$ for $\boldsymbol{u} \in S_k$ and $k \geq 4$.*

Proof. (a) Let $\boldsymbol{u} = \boldsymbol{u}_1\boldsymbol{u}_2\boldsymbol{u}_3$ be in S_2, with $t(\boldsymbol{u}) = 1$. Then $\boldsymbol{u}$ is a linear combination of $\zeta_2, \ldots, \zeta_5$, with not all coefficients zero. So with $A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ and $I' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, for some 2-vector $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ not both zero

$$\begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\lambda} & \boldsymbol{\mu} \end{bmatrix} \begin{bmatrix} A & A+I' & I' \\ A+I' & A & I' \end{bmatrix}. \tag{6.44}$$

Now, $t(\boldsymbol{u}) = 1$ means that exactly two of the $\boldsymbol{u}_i$ are zero. But $\boldsymbol{u}_1 = \boldsymbol{u}_2 = \mathbf{0}$ implies

$$\boldsymbol{\lambda}A + \boldsymbol{\mu}(A + I') = \mathbf{0} = \boldsymbol{\lambda}(A + I') + \boldsymbol{\mu}A,$$

which implies $\boldsymbol{\lambda} = \boldsymbol{\mu} = \mathbf{0}$. Similarly for $\boldsymbol{u}_2 = \boldsymbol{u}_3 = \mathbf{0}$ and $\boldsymbol{u}_1 = \boldsymbol{u}_3 = \mathbf{0}$. Hence $t(\boldsymbol{u}) \neq 1$ for $\boldsymbol{u} \in S_2$ and $k \geq 2$.

(b) Simply inspect ζ_4, ζ_5 and ζ_{45}. □

Now Lemma 6.13 says that $a_1 = 0$ for $d \leq 5$ so, replacing $k-6$ by $d-1$, (6.43) becomes

$$a_2 + a_3 = 2^{d-1} - 1. \tag{6.45}$$

Now if we let $d \geq 3$ then $A_2 = 0$ and $0 = 2^{1-d}(9a_0 + 5a_1 + a_2 - 3a_3)$, i.e.

$$a_2 = 3a_3 - 9. \tag{6.46}$$

Now solving (6.45) and (6.46) we get $a_2 = 3(2^{d-3} - 1)$ and $a_3 = 2^{d-3} + 2$. So for $d \geq 3$, we have $A_4 = 3(2^{5-d} - 1)$ and $A_6 = 2^{5-d} + 2$. Thus we have the more general version of Table 6.1 as Table 6.2.

k	d	$A(x)$
$k = 5$	$d = 2$	$1 + 3x^2 + 15x^4 + 13x^6$
$2 \leq k \leq 4$	$3 \leq k \leq 5$	$1 + 3(2^{5-d} - 1)x^4 + (2^{5-d} + 2)x^6$

Table 6.2: Weight enumerators of $\mathrm{KM}(9, k, 7-k)$ codes in terms of d

6.5 Extending the KM$(9, k, 7-k)$ Codes to KM$(10, k, 8-k)$ Codes

We now extend the method of Section 6.4 to incorporate wraparound, $s = 1$. The generator matrices of the KM$(10, k, 8-k)$ codes will be those of (6.26) with the extra column $[00\ldots01]^T$, corresponding to the multiplication $z_{k-1}y_{d-1}$ required for $\overline{Z}(u)\overline{Y}(u)$ modulo u^s. As we discussed in Section 5.2 the 1 in the last column is always in the last row of this column, whatever the value of k. This being so we are able to get the equation $z_{k-1} = c_9$ from

$$\left[\begin{array}{ccc} z_0 & \ldots & z_{k-1} \end{array}\right] \left[\begin{array}{c|c|c|c} & & & 0 \\ & & & 0 \\ & & & \vdots \\ & & & 1 \end{array}\right] = \left[\begin{array}{ccc} c_0 & \ldots & c_9 \end{array}\right]. \tag{6.47}$$

Implicitly we can extend the ζ_i by one zero, to have length 10. This will also leave their values of t unchanged. In the same way as Section 6.4 we can now give the necessary $10-k$ independent generators for the dual code, and further

$$\text{Dual KM}(10, k, 8-k) \text{ code} = \bigcup_{u \in S_k \cup (\zeta'_{k-1} + S_k)} (u + E). \tag{6.48}$$

Concerning the weight enumerator of this code we see that

$$R(u+E) = \begin{cases} \beta_{t(u)} & \text{if} \quad u \in S_k \\ x\beta_{t(u)} & \text{if} \quad u \in \zeta'_{k-1} + S_k \end{cases}. \tag{6.49}$$

Now as the code length has increased to 10 the MacWilliams transform of $\beta_{t(u)}$ is changed to $\alpha_{t(u)} = (1+x)(1-x^2)^{t(u)}(1+3x^2)^{3-t(u)}$, and the MacWilliams transform of $x\beta_{t(u)}$ which we shall denote by $\alpha'_{t(u)}$ is equal to $\frac{(1-x)}{(1+x)}(1-x^2)^{t(u)}(1+3x^2)^{3-t(u)}$. So we have the weight enumerator, $B(x)$, of the dual KM$(10, k, 8-k)$ code as

$$B(x) = \sum_{u \in S_k} \beta_{t(u)} + \sum_{v \in (\zeta'_{k-1} + S_k)} x\beta_{t(v)} \tag{6.50}$$

and for the primary KM$(10, k, 8-k)$ code as

$$\begin{aligned} A(x) &= \frac{1}{2^{7-k}} \left(\sum_{u \in S_k} \alpha_{t(u)} + \sum_{v \in (\zeta'_{k-1} + S_k)} \alpha'_{t(v)} \right) \\ &= \frac{1}{2^{7-k}} \left(a_0\alpha_0 + a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3 + a'_1\alpha'_1 + a'_2\alpha'_2 + a'_3\alpha'_3 \right). \end{aligned} \tag{6.51}$$

Now using Lemma 6.11 we can produce a table of the weight enumerators of the KM$(10, k, 8-k)$ codes. These are given in Table 6.3.

k	d	a_2	a'_2	a_3	a'_3	$A(x)$
6	2	0	0	0	1	$1 + 3x^2 + 6x^3 + 15x^4 + 12x^5 + 13x^6 + 14x^7$
5	3	0	0	1	2	$1 + 3x^3 + 9x^4 + 6x^5 + 6x^6 + 7x^7$
4	4	0	3	3	1	$1 + 3x^4 + 6x^5 4x^6 + 2x^7$
3	5	3	6	4	2	$1 + 3x^5 + 3x^6 + x^7$
2	6	9	12	6	4	$1 + x^6 + 2x^7$

Table 6.3: Weight enumerators of KM$(10, k, 8-k)$ codes

6.5.1 Explicit Formulae for the A_i

Once again it would be nice to find a systematic way to obtain a formula for the A_i. This can be obtained by finding formulae for the a_i and a'_i, expressing them all in terms of d. First we must look at the α_i and α'_i and note that

$$\begin{aligned} \alpha_t &= \textstyle\sum_{r=0}^{3} \left(b_{t,r} x^{2r} + b_{t,r} x^{2r+1} \right) \\ \alpha'_t &= \textstyle\sum_{r=0}^{3} \left(b_{t,r} x^{2r} - b_{t,r} x^{2r+1} \right), \end{aligned} \tag{6.52}$$

where

$$[b_{tr}] = \begin{bmatrix} 1 & 9 & 27 & 27 \\ 1 & 5 & 3 & -9 \\ 1 & 1 & -5 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix}. \tag{6.53}$$

Also we have that

$$A(x) = 2^{k-7} \left(\sum_{u \in S_k} \alpha_{t(u)} + \sum_{v \in \zeta'_{k-1} + S_k} \alpha'_{t(v)} \right). \tag{6.54}$$

Now from (6.52) and (6.54), replacing $k-7$ by $1-d$ we obtain for $d \geq 2$

$$\begin{aligned} 2^{d-1}A_{2r} &= b_{0,r} + b_{1,r}(a_1 + a_1') + b_{2,r}(a_2 + a_2') + b_{2,r}(a_2 + a_2') \\ 2^{d-1}A_{2r+1} &= b_{0,r} + b_{1,r}(a_1 - a_1') + b_{2,r}(a_2 - a_2') + b_{2,r}(a_2 - a_2') \end{aligned} . \tag{6.55}$$

Looking at the construction and size of S_k and $\zeta_{k-1} + S_k$ we can see that as the dashed values for d become undashed for $d+1$, for $2 \leq d \leq 6$,

$$\begin{aligned} a_1(d+1) &= a_1(d) + a_1'(d) \\ a_2(d+1) &= a_2(d) + a_2'(d) \\ a_3(d+1) &= a_3(d) + a_3'(d) \end{aligned} \tag{6.56}$$

and

$$\begin{aligned} a_1 + a_2 + a_3 &= 2^{d-2} - 1 \\ a_1' + a_2' + a_3' &= 2^{d-2}. \end{aligned} \tag{6.57}$$

Now the a_i and a_i' are trivial to determine for $d = 2$, so we can use them for initial values in (6.56). For $d = 2$ we have $a_0 = 1, a_1 = a_1' = a_2 = a_2' = a_3 = 0, a_3' = 1$. Note for $d = 2$ we also have $A_0 = 1$ and $A_1 = 0$ (obvious) giving from (6.55) for $2 \leq d \leq 6$,

$$\begin{aligned} 2^{d-1} &= 1 + (a_1 + a_1') + (a_2 + a_2') + (a_3 + a_3') \\ 0 &= 1 + (a_1 + a_1') + (a_2 - a_2') + (a_3 - a_3'), \end{aligned} \tag{6.58}$$

We will obtain the formulae for this in two ways. Firstly we will use the known fact from Lemma 6.13 that for $\zeta_5, \ldots, \zeta_2$ there are no vectors with two zero triples (i.e. $a_1 = a_1' = 0$) and consider the case where we need to use ζ_1 separately. Secondly we will keep it (initially) more general (at least in the sense of we will not assume that $a_1 = a_1' = 0$ and involve them in the calculation.)

We know from Lemma 6.11 that $a_1 = a_1' = 0$ for $2 \leq d \leq 5$ so we restrict ourselves to that for the moment. As both of the equations in (6.58) can actually be obtained from (6.57) we must find another. If we set $d \geq 3$ then we have that $A_2 = 0$ and can obtain from (6.55) for $3 \leq d \leq 5$

$$0 = 9 + (a_2 + a_2') - 3(a_3 + a_3'). \tag{6.59}$$

Now solving the first equation of (6.58) and (6.59) we get for $3 \leq d \leq 5$

$$a_2 + a_2' = 3(2^{d-3} - 1), a_3 + a_3' = 2^{d-3} + 2. \tag{6.60}$$

Using these we can obtain A_{2r} directly from (6.53) and (6.55), but to find A_{2r+1} we must find a_2, a_2', a_3 and a_3' individually. For this we use (6.60) and (6.56) and obtain for $4 \leq d \leq 5$

$$a_2 = 3(2^{d-4} - 1), a_2' = 3.2^{d-4}, a_3 = 2^{d-4} + 2, a_3' = 2^{d-4}. \tag{6.61}$$

Note that the formulae for A_{2r} apply for $d \geq 3$ as (6.56) is not used. We therefore get Table 6.4. Note in the table that the weight enumerators for $k = 1, 2$ are determined trivially in the usual way.

k	d	$A(x)$						
6	2	1	$+3x^2$	$+6x^3$	$+15x^4$	$+12x^5$	$+13x^6$	$+14x^7$
5	3	1		$+3x^3$	$+3(2^{5-d}-1)x^4$	$+6x^5$	$+(2^{5-d}+2)x^6$	$+7x^7$
4	4	1			$+3(2^{5-d}-1)x^4$	$+3.2^{5-d}x^5$	$+(2^{5-d}+2)x^6$	$+2^{5-d}x^7$
3	5	1			$+3(2^{5-d}-1)x^4$	$+3.2^{5-d}x^5$	$+(2^{5-d}+2)x^6$	$+2^{5-d}x^7$
2	6	1					$+x^6$	$+2x^7$
1	7	1						$+x^7$

Table 6.4: Weight enumerators of KM$(10, k, 8-k)$ codes

Now we will consider the more general form of starting with a_i and $a_i', i = 1, 2, 3$ unknown. We will however see that this leads to a less general result.

Here if we set $d \geq 3$ then we get the equation

$$0 = 9 + 5(a_1 + a_1') + (a_2 + a_2') - 3(a_3 + a_3'). \tag{6.62}$$

But unlike before there are too many unknowns to solve the first equality of (6.58) and (6.62) so we must find another equation. Setting $d \geq 4$ will not result in a new equation from (6.55) so we must try $d \geq 5$. Therefore we can set $A_4 = 0$ and obtain

$$0 = 27 + 3(a_1 + a_1') - 5(a_2 + a_2') + 3(a_3 + a_3'). \tag{6.63}$$

We can now solve part one of (6.58), (6.62) and (6.63) to give

$$\begin{aligned} a_1 + a_1' &= 3(2^{d-5} - 1) \\ a_2 + a_2' &= 3(2^{d-4} + 1) \\ a_3 + a_3' &= 7(2^{d-5} - 1) \end{aligned}, \tag{6.64}$$

Giving for $5 \leq d \leq 7$, that $A_6 = 2^{7-d} - 1$. Now to find A_7 we must use (6.56) and so obtain $A_7 = 2^{7-d}$ for $d \geq 6$ only. This provides only a limited amount of information, but is still useful in our study.

6.6 Extending to KM$(12, k, 9-k)$ Codes

Here we incorporate the wraparound $s = 2$ to the existing model of Section 6.4. The generator matrices for the KM$(12, k, 9-k)$ codes will be those of (6.26) with the extra column shown below in the scheme for general k.

$$\begin{bmatrix} z_0 & \cdots & z_{k-1} \end{bmatrix} \left[\begin{array}{c|c|c|ccc} & & & 0 & 0 & 0 \\ & & & \vdots & \vdots & \vdots \\ & & & 0 & 0 & 0 \\ & & & 0 & 1 & 1 \\ & & & 1 & 0 & 1 \end{array}\right] = \begin{bmatrix} c_0 & \cdots & c_{11} \end{bmatrix}. \tag{6.65}$$

The extra multiplications come from the calculation $\overline{Z}(u)\overline{Y}(u)$ modulo u^2, and are $m_9 = z_{k-1}y_{d-1}, m_{10} = z_{k-2}y_{d-2}, m_{11} = (z_{k-2}+z_{k-1})(y_{d-2}+y_{d-1})$. The last three columns of the generator matrices sum to zero, giving the relation $c_9 + c_{10} + c_{11} = 0$, which if we express as $E_4\mathbf{c}^T = 0$ then we can take $m = 4$ in Definition 6.9. This implies extending the vectors ζ_i by three zeros to be of length 12 also. In a similar way to the $s = 1$ case, we can consider columns 9 and 10 of both sides of (6.65) and get

$$z_{k-1} = c_9, z_{k-2} = c_{10}. \tag{6.66}$$

Now using Theorem 6.4 we have $z_i = \hat{z}_i = \zeta_i\mathbf{c}^T$ for $0 \leq i \leq 5$ if $k \leq 6$ with corresponding dual code generators of

$$\zeta'_{k-1} = \zeta_{k-1} + e_9, \zeta'_{k-2} = \zeta_{k-2} + e_{10}. \tag{6.67}$$

However, when $k = 7$ we have quite an individual case to consider. This is due to the fact that at first sight we need ζ_6 as $k-1 = 6$, and yet we do not have a ζ_6. If we revert back to the construction of the polynomial $Z(u)Y(u)$ modulo $P(u)$, then we can only reconstruct $Z(u)$ uniquely using the CRT, because $\deg[Z(u)] < \deg[P(u)]$. This is not the case here and in fact we have

$$Z(u) = \hat{Z}(u) + \lambda P(u), \tag{6.68}$$

and equating coefficients of u^i gives

$$\begin{aligned} z_6 &= \lambda \\ z_5 &= \hat{z}_5 + \lambda \end{aligned}, \tag{6.69}$$

so we have $z_{56} = \hat{z}_5$.

Note here that this is a special case of the generalised theory given by Theorem 6.4, which we will show in action now. We have $P(u) = u^6 + u^5 + u^3 + u^2$, hence the reciprocal polynomial $Q(u) = u^4 + u^3 + u + 1$, so $D = 6$ and $E = 4$. Now we have $k = 7$ and $d = 2$ and we want (from (6.11)) $D - j = 5$, so $j = 1$. This gives

$$\begin{aligned} \hat{z}_5 &= z_5 + \sum_{i \geq 0}^{k-D-1} d_i z_{6+i} \\ &= z_5 + d_0 z_6 \end{aligned}, \tag{6.70}$$

where $d_0 = q_1 d_{-1} + q_2 d_{-2} + q_3 d_{-3} + q_4 d_{-4}$ and $d_{-4} \ldots d_{-1} = 0001$. So with $Q(u)$ as stated above we get $d_0 = 1$, and hence $\hat{z}_5 = z_{56}$.

Now from the matrix (6.65) we see that $z_{56} = c_{11}$. Note this is not unique. So the generator we can use is $\zeta_5 + e_{11}$. The dual KM$(12, 7, 2)$ code is thus obtained as the span of E_{1234} and $\zeta_5 + e_{11}$.

We now have a formula for the dual KM$(12, k, 9-k)$ code as follows:

$$\begin{aligned} \text{Dual KM}(12, k, 9-k) \text{ code} &= \bigcup_{u \in \text{Span}\{\zeta'_{k-2}, \zeta'_{k-1}, \zeta_k, \ldots, \zeta_5\}} (u + E) && , 1 \leq k \leq 6 \\ &= \text{Span}\,\{\zeta'_5 + E\} && , k = 7 \end{aligned} \tag{6.71}$$

Now we have vectors of length 12, i.e. 4×3, so we use Lemma 6.10 with $m = 4$. We can easily find the weight enumerator of the dual code and hence the primary code in the same way as Section 6.4. The results are given in Table 6.5.

6.6.1 Explicit Formulae for the A_i

We have

$$\begin{aligned} A(x) &= 2^{k-8} \sum_{u \in \text{Span}\{\zeta'_{k-1}, \zeta'_{k-2}, S_k\}} \alpha_{t(u)} \\ &= 2^{1-d} \left(a_0 \alpha_0 + a_1 \alpha_1 + a_2 \alpha_2 + a_3 \alpha_3\right), \end{aligned} \tag{6.72}$$

k	d	$A(x)$
7	2	$1+4x^2+30x^4+52x^6+41x^8$
6	3	$1+17x^4+24x^6+21x^8$
5	4	$1+6x^4+16x^6+9x^8$
4	5	$1+12x^6+3x^8$
3	6	$1+4x^6+3x^8$
2	7	$1+3x^8$

Table 6.5: Weight enumerators of KM$(12, k, 9-k)$ codes

where

$$\alpha_{t(u)} = \sum_{r=0}^{4} b_{t(u),r} x^{2r} \tag{6.73}$$

and

$$[b_{tr}] = \begin{bmatrix} 1 & 12 & 54 & 108 & 81 \\ 1 & 8 & 18 & 0 & -27 \\ 1 & 4 & -2 & -12 & 9 \\ 1 & 0 & -6 & 8 & -3 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix}. \tag{6.74}$$

Now we know $a_0 = 1$ and so

$$a_1 + a_2 + a_3 + a_4 = 2^{d-1} - 1. \tag{6.75}$$

Also if $d \geq 2$ then $A_0 = 1, A_1 = 0$ and equating the constant coefficients in (6.72) gives $1 = 2^{1-d}(1 + a_1 + a_2 + a_3 + a_4)$, which is the same as (6.75).

The following lemma will help with the solving of the problem.

Lemma 6.14 *For $d \leq 7$ we have $a_1 = 0$.*

Proof. From Lemma 6.12 the only time for $n = 9$ that $a_1 \neq 0$ is when ζ_1 is used, but in the present problem these are always dashed if used, and hence the fourth triple is non-zero, adding one to the value of t. Hence $a_1 = 0$ for $d \leq 7$. □

Now from Lemma 6.14, $a_1 = 0$ for $d \leq 7$ so (6.75) becomes

$$2^{d-1} - 1 = a_2 + a_3 + a_4. \tag{6.76}$$

Now if we let $d \geq 3$ then $A_2 = 0$ and $0 = 2^{1-d}(12a_0 + 8a_1 + 4a_2 - 4a_4)$, i.e.

$$12 = 4a_4 - 4a_2. \tag{6.77}$$

We need the following lemma to proceed further.

Lemma 6.15 *For* $3 \leq d \leq 5$ *we have* $a_2 = 0$.

Proof. This can be seen by looking at the vectors $\zeta_5, \ldots, \zeta_2$ from Section 6.4. The only way to obtain $t(\boldsymbol{u}) = 2$ is $\zeta_{34}, \zeta_{24}, \zeta_{234}, \zeta_{235}$ and ζ_{245}, but for the given values of d these will always be dashed and so t will increase by one. Also we could obtain $t(\boldsymbol{u}) = 1$ and the dashed vectors will increase t to two, but $t(\boldsymbol{u}) = 1$ cannot occur. Therefore $t(\boldsymbol{u}) \neq 2$. □

Back to the problem. Solving (6.76) and (6.77) we get for $3 \leq d \leq 5$,

$$a_4 = 3, a_3 = 2^{d-1} - 4\ . \tag{6.78}$$

So we have $d = 2$ and $3 \leq d \leq 5$, but can we find a relation for $d \geq 5$. If we set $d \geq 4$ then we do not get a new relation from (6.72) so set $d \geq 5$, i.e. $A_4 = 0$ and get $0 = 2^{1-d}(54 - 2a_2 - 6a_3 + 6a_4)$, i.e.

$$54 = 2a_2 + 6a_3 - 6a_4. \tag{6.79}$$

Now solving (6.76),(6.77) and (6.79) we get

$$a_2 = 6(2^{d-5} - 1), a_3 = 8(2^{d-6} + 1), a_4 = 3(2^{d-4} - 1), \tag{6.80}$$

and Table 6.6 can be formed.

6.6.2 Another Look at the Dual Code Construction

We finish off this section by noting that

$$\text{Span}\ \{\zeta'_{k-1}, \zeta'_{k-2}, S_k\} = S_k \cup (\zeta'_{k-1} + S_k) \cup (\zeta'_{k-2} + S_k) \cup (\zeta'_{k-2,k-1} + S_k), \tag{6.81}$$

k	d	$A(x)$
7	2	$1+4x^2+30x^4+52x^6+41x^8$
$4 \le k \le 6$	$3 \le d \le 5$	$1+6(2^{5-d}-1)x^4+8(2^{4-d}+1)x^6+3(2^{6-d}-1)x^8$
$2 \le k \le 4$	$5 \le d \le 7$	$1+4(2^{7-d}-1)x^6+3x^8$

Table 6.6: Weight enumerators of KM$(12, k, 8-k)$ codes

but note that the 4th triple of $\zeta'_{k-1}, \zeta'_{k-2}$ and $\zeta'_{k-2,k-1}$ will never equal 000 and so the value of t for these vectors will be one plus the value of t for the undashed versions.

So if we write (6.72) as

$$A(x) = 2^{1-d}\left(\sum_{u\in S_k}\alpha_{t(u)} + \sum_{u\in S_{k-2}\setminus S_k}\alpha_{t(u)+1}\right) \tag{6.82}$$

and further as

$$A(x) = 2^{1-d}\sum_{i=0}^{4} a_i\alpha_i \tag{6.83}$$

then

$$\begin{aligned} a_i &= t_i(S_k) + t_{i-1}(S_{k-2}\setminus S_k) \\ &= t_i(S_k) + t_{i-1}(S_{k-2}) - t_{i-1}(S_k), \end{aligned} \tag{6.84}$$

and we can use the results of Section 6.4 (in particular Table 6.1) to solve the problem directly. (Note that a similar idea is also used in Hoggar (1997), and will be investigated further in the next chapter.) We present one of the cases as an example to illustrate this method.

Example 6.16 From Table 6.1 we see for the KM$(9,3,4)$ code $a_{0123} = 1.0.3.4$ and for the KM$(9,1,6)$ code $a_{0123} = 1.3.15.13$. So for the present problem if we require the weight enumerator for the KM$(12,3,6)$ code, then we use (6.84) and the above values.

For the KM$(12,3,6)$ code

$$\begin{aligned} a_0 &= t_0(S_3) + t_{-1}(S_1) - t_{-1}(S_3) \\ &= 1+0-0 \end{aligned}, \tag{6.85}$$

and similarly $a_1 = 0+1-1 = 0, a_2 = 3+3-0 = 6, a_3 = 4+15-3 = 16, a_4 = 0+13-4$, and $A(x)$ can be found from (6.83) as $1+4x^6+3x^8$, as expected.

Now using this method we can obtain the same results as Table 6.5 apart from the $k = 2, d = 7$ case as this needs the $k = 0, d = 7$ case from Table 6.1. However this is not much of a loss since the case $k = 2, d = 7$ is easy to work out independently. We will obtain a method to get round this in Section 7.4.

6.6.3 Future Work

Using the ideas developed here and the infinite scheme for the wraparound introduced in Section 4.9 we can develop theory and obtain results for any value of s. Time, however, did not permit this.

6.7 Constant Degree Codes with $w = 3$

We can with $P(u) = u^3(u^3+1)(u^3+u+1)(u^3+u^2+1)$ get a base code, G, of length 24, as follows

$$\left[\begin{array}{cccccc|cccccc|cccccc|cccccc}
1&0&1&0&0&1&1&0&1&0&0&1&1&0&1&0&0&1&1&0&1&0&0&1\\
0&1&1&0&1&1&0&1&1&0&1&1&0&1&1&0&1&1&0&1&1&0&1&1\\
0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0&1&1&1\\
0&0&0&0&0&0&1&0&1&0&0&1&1&1&0&0&1&0&1&0&1&1&1&0\\
0&0&0&0&0&0&0&1&1&0&1&1&0&1&1&1&0&0&1&1&0&1&0&1\\
0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&1&0&1&1&1&0&0&1&0\\
0&0&0&0&0&0&1&0&1&0&0&1&1&0&1&1&1&0&0&1&1&1&0&0\\
0&0&0&0&0&0&0&1&1&0&1&1&1&0&1&0&0&1&1&0&1&0&0&1\\
0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&0&1&1&0&1&1&0&1&1\\
0&0&0&0&0&0&1&0&1&0&0&1&0&0&0&1&1&1&0&0&0&1&1&1\\
0&0&0&0&0&0&0&1&1&0&1&1&1&1&0&0&1&0&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1&0&0&1&1&0&1&0&1
\end{array}\right]. \quad (6.86)$$

From this we can invert the reduced matrix $\underline{G}$ to give $\underline{G}^{-1}$ and therefore to get the

twelve available relations $\zeta_i, i = 0 \ldots 11$ as

$$\begin{bmatrix} \zeta_0 \\ \vdots \\ \zeta_{11} \end{bmatrix} = \left[\begin{array}{cccccc|cccccc|cccccc|cccccc} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&1&0&0&0&0&0&1&1&0&0&0&0&1&1&0&1&0&0 \\ 0&1&0&1&0&0&1&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&1&0&1&0&0&0&1&0&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&1&0&0&1&1&0&0&0&0 \\ 1&0&0&0&0&0&1&1&0&1&0&0&1&0&0&0&0&0&1&1&0&1&0&0 \\ 0&1&0&0&0&0&1&1&0&1&0&0&1&1&0&0&0&0&0&1&0&1&0&0 \\ 0&0&0&1&0&0&1&1&0&1&0&0&1&1&0&1&0&0&0&0&1&0&0&0 \\ 1&0&0&1&0&0&0&1&0&1&0&0&1&0&0&1&0&0&0&1&0&1&0&0 \\ 0&1&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1&1&0&0&0&0 \end{array}\right] \tag{6.87}$$

Here instead of having the vectors divided as triples, we divide them into six-tuples. This of course may increase the complexity quite considerably but still the same method is used. We get the inner relations for each block as 111000, 001110 and 010101 and so if we set $S = \text{Span}\{111000, 001110, 010101\}$ then the contribution to the weight enumerator for each of the eight possible possible six-tuples are given in Table 6.7. Note that we do not need to look at the sixty-four possible six-tuples as there can never be a one in the third, fifth or sixth position. This eases our problem quite considerably.

Weight Enumerator	Vector	Associated t
$1 + 4z^3 + 3z^4$	000000	t_0
$3z^2 + 4z^3 + z^6$	100100	t_1
$z + 2z^2 + 2z^3 + 2z^4 + z^5$	100000, 010000, 110000, 000100, 010100, 110100	$t_2 = (4 - t_1 - t_2)$

Table 6.7: Table of the contributions to the weight enumerator

Definition 6.17 Define the following

$E_{i,1}$	The vector 111000 in the ith six-tuple of a 24-vector.
$E_{i,2}$	The vector 001110 in the ith six-tuple of a 24-vector.
$E_{i,3}$	The vector 010101 in the ith six-tuple of a 24-vector.
E	The span of the set $\{E_{11}, E_{12}, E_{13}, E_{21}, \ldots, E_{43}\}$.
S_k	The span of the set $\{\zeta_k, \ldots \zeta_{11}\}$.
$t_0(\boldsymbol{u})$	The number of times the six-tuple 000000 appears in $\boldsymbol{u}$.
$t_1(\boldsymbol{u})$	The number of times the six-tuple 100100 appears in $\boldsymbol{u}$ (see Table 6.7).

Lemma 6.18 *The weight enumerator of* $\boldsymbol{u} + E$ *is*

$$(1 + 4z^3 + 3z^4)^{t_0(\boldsymbol{u})}(3z^2 + 4z^3 + z^6)^{t_1(\boldsymbol{u})}(z + 2z^2 + 2z^3 + 2z^4 + z^5)^{4-t_0(\boldsymbol{u})-t_1(\boldsymbol{u})}$$

Proof. This is proved in the same way as Lemma 6.10. □

Now knowing the $\zeta_i, i = 0, \ldots, 11$, we can easily form the family of KM$(24, k, 13-k)$ codes by setting the relevant ζ_i to zero, and using the general formula as

$$\text{Dual KM}(24, k, 13-k) \text{ code} = \bigcup_{\boldsymbol{u} \in S_k} (\boldsymbol{u} + E). \tag{6.88}$$

So if we take $\beta_{i,j} = (1 + 4z^3 + 3z^4)^i(z + 2z^2 + 2z^3 + 2z^4 + z^5)^j(3z^2 + 4z^3 + z^6)^{4-i-j}$ then we have

$$B(x) = \sum_{\boldsymbol{u} \in S_k} \beta_{t_0(\boldsymbol{u}), t_1(\boldsymbol{u})} \,. \tag{6.89}$$

We could now try and develop the theory in the same way as in the $w = 2$ case by forming the $\beta_{i,j}$ for all the possible values of i and j, but as can easily be seen there are fifteen. Forming a matrix of the MacWilliams transforms of these was seen as too out of the scope of this chapter, but the problem, although being harder could be approached in the same way as the $w = 2$ case. We therefore opt at the present to obtain the weight enumerators by looking at all the necessary vectors. For example if we want to know the weight enumerator of the KM$(24, 11, 2)$ codes then we need 13 generators. We have 12 given by S and the other one can be found as $\zeta_{11} = 0$. Now $\zeta_{11} = [010100000100100000110000]$ and so $t_0(\zeta_{11}) = 0$,

$t_1(\zeta_{11}) = 0$ and $t_2(\zeta_{11}) = 4$. So the weight enumerator of the dual KM$(24, 11, 2)$ code is

$$\begin{aligned} B(z) &= (1 + 4z^3 + 3z^4)^0(z + 2z^2 + 2z^3 + 2z^4 + z^5)^4(3z^2 + 4z^3 + z^6)^0 + (1 + 4z^3 + 3z^4)^4 \\ &= 1 + 16z^3 + 13z^4 + 8z^5 + 128z^6 + 232z^7 + 242z^8 + 584z^9 + 1056z^{10} + 1032z^{11} \\ &\quad 1010z^{12} + 1368z^{13} + 1344z^{14} + 760z^{15} + 269z^{16} + 88z^{17} + 32z^{18} + 8z^{19} + z^{20} \end{aligned} \tag{6.90}$$

Then we can use the MacWilliams identities to find the weight enumerator of the primary code. We thus get Table 6.8.

We could now develop, in association with the theory of Sections 6.4-6.6, the weight enumerators of the KM$(25, k, 14-k)$ codes ($s = 1$) and further. Instead we just give a brief view of the $s = 1$ case to show the generalising nature of this theory.

6.7.1 The KM$(25, k, 14 - k)$ Codes

With the inclusion of the $s = 1$ part of the matrix the dual code can be formed as (using the same ideas as Section 6.3)

$$\text{Dual KM}(25, k, 14-k) \text{ code} = \bigcup_{u \in S_k} (u + E) + \bigcup_{v \in \zeta'_{k-1} + S_k} (v + E) \tag{6.91}$$

and therefore the weight enumerator of the dual code is

$$B(x) = \sum_{u \in S_k} \beta_{t_0(u), t_1(u)} + \sum_{v \in \zeta_{k-1} + S_k} x\beta_{t_0(v), t_1(v)} \, . \tag{6.92}$$

Again the theory could be further developed here but we feel that it has been explained and proven that the theory developed for the $w = 2$ case can be naturally extended to larger cases. Finally, by looking at the necessary vectors we are able to form the dual code weight enumerator and then dualise back to the primary code, the results of which are shown in Table 6.9.

k	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
12	1			16	12	0	96	144	54	256	576
11	1			8	4	0	48	72	30	128	288
10	1			1	3	0	28	37	13	64	136
9	1						14	20	8	32	68
8	1						5	10	3	16	38
7	1						2	3	1	8	20
6	1									4	12
5	1									2	4
4	1										4
3	1										
2	1										

k	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}
12	432	364	768	864	432	81
11	216	180	384	432	216	41
10	115	89	192	220	103	22
9	56	44	96	110	52	11
8	28	22	48	53	26	6
7	18	10	24	26	11	4
6	12	4	12	12	4	3
5	6	4	6	4	2	3
4	0	4	0	4	0	3
3	0	4	0	0	0	3
2	0	2	0	0	0	1

Table 6.8: Weight Distribution of KM$(24, k, 13-k)$ codes

k	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
12	1			8	12	8	48	120	102	152	416
11	1			1	10	1	28	57	48	81	200
10	1				1	3	14	34	25	37	100
9	1						5	19	13	21	54
8	1						2	6	8	10	28
7	1							2	3	5	16
6	1									2	6
5	1										6
4	1										
3	1										
2	1										

k	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}
12	504	396	568	816	648	257	40
11	267	190	283	412	315	135	19
10	124	103	141	206	162	62	11
9	58	50	70	101	83	32	5
8	36	20	36	50	38	19	2
7	20	10	18	24	18	10	1
6	14	10	6	10	10	5	0
5	0	10	0	10	0	5	0
4	4	4	0	0	4	3	0
3	0	2	2	0	0	1	2
2	0	1	1	0	0	0	1

Table 6.9: Weight Distribution of KM$(25, k, 14-k)$ codes

Chapter 7

Weight Enumerators of Shunted Families of KM codes, II

7.1 Introduction

In this chapter we will introduce and develop some theory concerning the relationship between sets of vectors and KM Codes. The work has strong links with that of Hoggar (1997) but here we give a more thorough explanation of it. Also we try to consistently keep the idea of our general families of KM Codes at the forefront. The results obtained from Hoggar (1997) will be indicated as such.

7.2 Shifts and Related Topics

We start this section off with a definition.

Definition 7.1 Let $\boldsymbol{a} = \left[\begin{array}{cccc} a_0 & \ldots & a_{n-1} \end{array} \right]$, of length n and define $a_{-1} = a_n = 0$. Then
(i) The operator δ acts on $\boldsymbol{a}$ as

$$\boldsymbol{a}^{\delta} = \delta(\boldsymbol{a}) = \left[\begin{array}{cccc} a_0 - a_{-1} & a_1 - a_0 & \ldots & a_n - a_{n-1} \end{array} \right],$$

to form a vector of length $n+1$.
(ii)The operator Δ acts on $\boldsymbol{a}$ as

$$\boldsymbol{a}^{\Delta} = \Delta(\boldsymbol{a}) = \left[\begin{array}{cccc} a_0 - a_1 & a_1 - a_2 & \ldots & a_{n-1} - a_n \end{array} \right],$$

to form a vector of length n.
(iii) The operator σ, known as the shift of $\boldsymbol{a}$, acts on $\boldsymbol{a}$ as

$$\boldsymbol{a}^{\sigma} = \sigma(\boldsymbol{a}) = \left[\begin{array}{cccc} 0 & a_0 & \ldots & a_{n-1} \end{array} \right],$$

to form a vector of length $n+1$.
(iv) The operator τ acts on $\boldsymbol{a}$ as

$$\boldsymbol{a}^{\tau} = \tau(\boldsymbol{a}) = \left[\begin{array}{cccc} a_0 & a_0 + a_1 & \ldots & a_{n-1} + a_n \end{array} \right],$$

to form a vector of length $n+1$.

We now introduce a few new vectors to investigate and demonstrate the usefulness and power of these shifts
(v) $\boldsymbol{\alpha}(m) = \left[\begin{array}{cccc} \alpha_0 & \alpha_1 & \ldots & \alpha_m \end{array} \right], \alpha_t = (1-x^2)^t(1+3x^2)^{m-t}$.

This way of representing the α_i's (as defined in the previous chapter) is to investigate the using of the vectors and relating the weight enumerator to the weight distribution.
(vi) $\boldsymbol{b}(n) = \left[\begin{array}{cccc} 1 & -n & \binom{n}{2} & \ldots & (-1)^n.1 \end{array} \right]$. Thus $\boldsymbol{b}(0) = 1, \boldsymbol{b}(1) = \left[\begin{array}{cc} 1 & -1 \end{array} \right]$.

Remark 7.2 For a vector $\boldsymbol{a}$ of length n we interpret $\boldsymbol{a}.\boldsymbol{\alpha}(m)$ as $\left[\begin{array}{cc} \boldsymbol{a} & 0^{m-n} \end{array} \right].\boldsymbol{\alpha}(m)$ if $n < m$ and not defined if $n > m$.

Lemma 7.3 $\boldsymbol{b}(n)^\delta = \boldsymbol{b}(n+1)$.

Proof.

$$\begin{aligned}(\boldsymbol{b}(n)^\delta)_i &= (-1)^i \binom{n}{i} - (-1)^{i-1} \binom{n}{i-1} \\ &= (-1)^i \left(\binom{n}{i} + \binom{n}{i-1} \right) \\ &= (-1)^i \binom{n+1}{i} \\ &= (\boldsymbol{b}(n+1))_i \ .\end{aligned} \tag{7.1}$$

□

Lemma 7.4 $\boldsymbol{a} = [a_0, \ldots, a_n]$. *If* $\sum a_i = 0$ *then* $\boldsymbol{a} = \boldsymbol{b}^\delta$ *for some* $\boldsymbol{b} = (b_0, \ldots, b_{n-1})$.

Proof. The $(n+1)$ - vectors $\boldsymbol{e}_i - \boldsymbol{e}_{i+1}, (0 \leq i \leq n-1)$ are linearly independent and lie in the vector space $\{\boldsymbol{x} \in \mathbb{R}^{n+1} : \sum x_i = 0\}$, and so form a basis of that space. Hence $\boldsymbol{a}$ is a unique linear combination $\sum_{i=0}^{n-1} b_i(\boldsymbol{e}_i - \boldsymbol{e}_{i+1}) = \sum b_i \delta(\boldsymbol{e}_i) = \delta(\boldsymbol{b})$. □

We now give a few Lemmas that demonstrate the power of the shifts when applying to $\boldsymbol{\alpha}(m)$, especially the ability to reduce the length of $\boldsymbol{\alpha}$.

Theorem 7.5 *We can link* δ *to the length of* $\boldsymbol{\alpha}$ *by the following,*
(a) $\boldsymbol{a}^\delta.\boldsymbol{\alpha}(m) = 4x^2\boldsymbol{a}.\boldsymbol{\alpha}(m-1)$.
(b) $\boldsymbol{b}(n).\boldsymbol{\alpha}(m) = (4x^2)^n.(1+3x^2)^{m-n}$.

Proof. A proof of (a) can be found in Hoggar (1997), but is a consequence of the following two lemmas. To prove (b) we also use Lemma 7.3. □

Lemma 7.6 *Let* $\boldsymbol{a} = [a_0, \ldots, a_{n-1}], \boldsymbol{b} = [b_0, \ldots, b_n]$, *then* $\boldsymbol{a}^\delta.\boldsymbol{b} = \boldsymbol{a}.\boldsymbol{b}^\Delta$.

Proof. We have

$$\begin{aligned} \boldsymbol{a}^\delta.\boldsymbol{b} &= \sum_{i=0}^{n}(a_i - a_{i-1})b_i \\ &= \sum_{i=0}^{n-1} a_i b_i - \sum_{i=0}^{n} a_{i-1}b_i \\ &= \sum_{i=0}^{n-1}(a_i b_i - a_i b_{i+1}) \\ &= \sum_{i=0}^{n-1} a_i(b_i - b_{i+1}) \\ &= \boldsymbol{a}.\boldsymbol{b}^\Delta \, . \end{aligned} \tag{7.2}$$

□

Lemma 7.7 $\Delta(\boldsymbol{\alpha}(m)) = 4x^2\boldsymbol{\alpha}(m-1)$.

Proof. For $0 \leq i \leq m-1$, we have

$$\begin{aligned} (\Delta(\boldsymbol{\alpha}(m)))_i &= \alpha_i - \alpha_{i+1}, \\ &= (1-x^2)^i(1+3x^2)^{m-i} - (1-x^2)^{i+1}(1+3x^2)^{m-1-i} \\ &= (1-x^2)^i(1+3x^2)^{m-1-i}(1+3x^2-(1-x^2)) \\ &= 4x^2(\boldsymbol{\alpha}(m-1))_i \, . \end{aligned} \tag{7.3}$$

□

For interest we show that the operator σ has a similarly powerful effect on the vector $\boldsymbol{\alpha}(m)$.

Theorem 7.8 (Hoggar (1997))

$$\boldsymbol{a}^\sigma.\boldsymbol{\alpha}(m) = (1-x^2)\boldsymbol{a}.\boldsymbol{\alpha}(m-1). \tag{7.4}$$

Proof.

$$\begin{aligned} \boldsymbol{a}^\sigma.\boldsymbol{\alpha}(m) &= \sum_{i=0}^{m-1} a_i(1-x^2)^{i+1}(1+3x^2)^{m-i-1} \\ &= (1-x^2)\sum_{i=0}^{m-1} a_i(1-x^2)^i(1+3x^2)^{m-1-i} \\ &= (1-x^2)\boldsymbol{a}.\boldsymbol{\alpha}(m-1) \, . \end{aligned} \tag{7.5}$$

□

7.3 Back to KM Codes

We start this section with a definition.

Definition 7.9 (i) If we have the set of vectors S_k, then let $\underline{abc} = \underline{abc}_k = a\zeta_{k-1} + b\zeta_{k-2} + c\zeta_{k-3} + S_k$, with the shorter writing of $\underline{ab} = \underline{ab0} = a\zeta_{k-1} + b\zeta_{k-2} + S_k$.
(ii) Let $\boldsymbol{a}(S_k)$ denote vector $[a_0 \ \ldots \ a_m]$, such that m is the number of triples in each vector of S_k and $a_i = t_i(S_k)$.
(iii) When confusion may arise, we will explicitly write $B(m)$ for the matrix B used in the construction of the weight enumerators where the vectors have number of triples equal to m. This should save any confusion between the B's.

Example 7.10 There are various ways of using the short notation in Definition 7.9, part (i), and here we show a few useful examples, which can be proved easily.

1. $\underline{10} = (S_{k-1} \setminus S_k)$,
2. $\underline{01}_{k-1} = \{\underline{001}, \underline{101}\}$,
3. $\boldsymbol{a}(S_{k-1} \setminus S_k) = \boldsymbol{a}(S_{k-1}) - \boldsymbol{a}(S_k)$.

These will be used later on in this section.

We now state the link between all the previous results in this chapter (i.e. those of Lemma 7.3 - Theorem 7.8) and the possible use of the matrices B in the previous chapter, which will prove excellent when trying to standardise our methods for finding the weight enumerators. We extend this from the version found in Hoggar (1997) (where Lemma 7.3 - Theorem 7.8 are used) to include the $k = 7$ which can be proved directly, with $S_7 = \{0\}$ and $S_{-1} = 2S_0$.

Lemma 7.11 *The following relate δ and σ for our present problem,*
(a) $\boldsymbol{a}(S_k)^\sigma B(4) = 2^{6-k}\boldsymbol{a}(S_{6-k})^\delta$, $(0 \le k \le 7)$,
(b) $\boldsymbol{a}(S_k)^\delta B(4) = 2^{8-k}\boldsymbol{a}(S_{6-k})^\sigma$, $(0 \le k \le 7)$.

Now in a similar manner to Hoggar (1997), we will develop new ways using these shifting techniques to find the weight enumerators of the shunted families of KM codes.

7.3.1 The KM$(9, k, 7-k)$ Codes

For ease of reading let us repeat the matrix B from (6.41) that we shall need during this section.

$$B = \begin{bmatrix} 1 & 9 & 27 & 27 \\ 1 & 5 & 3 & -9 \\ 1 & 1 & -5 & 3 \\ 1 & -3 & 3 & -1 \end{bmatrix}. \tag{7.6}$$

Now we know that we can write the general formula for the weight enumerator as

$$\begin{bmatrix} 1 & A_2 & A_4 & A_6 \end{bmatrix} = 2^{k-6} \begin{bmatrix} 1 & a_1 & a_2 & a_3 \end{bmatrix} B, \tag{7.7}$$

but by Lemma 6.13, the fact that $A_i = 0$, for $1 \leq i < d$ and by noting that α_i only have even powers of x we can divide (7.7) as

$$\begin{aligned} [1\ 0\ 0\ A_6] &= 2^{k-6}[1\ a_1\ a_2\ a_3]B, & 0 \leq k \leq 2 \\ [1\ 0\ A_4\ A_6] &= 2^{k-6}[1\ 0\ a_2\ a_3]B, & 2 \leq k \leq 4 \\ [1\ A_2\ A_4\ A_6] &= 2^{k-6}[1\ 0\ 0\ a_3]B, & 4 \leq k \leq 6 \end{aligned} . \tag{7.8}$$

Now we state and prove in a slightly different way the following lemma which can be found in Hoggar (1997).

Lemma 7.12 (Hoggar (1997)) *With matrix B as in (7.6) and any scalars μ, ν and λ we have (a) if $[1\ 0\ 0\ r] = \mu[1\ a\ b\ c]B$ then $[abc] = [3(\frac{\mu}{16} - 1) \quad 3(\frac{\mu}{16} + 1) \quad 7\left(\frac{\mu}{16}\right) - 1]$ and $r = \frac{64}{\mu} - 1$,*
(b) if $[1\ 0\ q\ r] = \nu[1\ 0\ b\ c]B$ then $[b\ c] = [3(\frac{\nu}{4} - 1) \quad \nu + 2]$ and $[q\ r] = [3\left(\frac{16}{\nu} - 1\right) \quad \frac{16}{\nu} + 2]$,
(c) if $[1\ p\ q\ r] = \lambda[1\ 0\ 0\ c]B$ then $c = \lambda - 1$ and $[p\ q\ r] = [3\left(\frac{4}{\lambda} - 1\right) \quad 3\left(\frac{8}{\lambda} + 1\right) \quad \frac{28}{\lambda} - 1]$.

Proof. This is generally the same as the proof in Hoggar (1997), but here we are keeping the factors out of the λ and so we get three different scalars. □

We now use this fact and examine the first and third cases from (7.8). Comparing (7.8) and Lemma 7.12 (a) we have that $\mu = 2^{6-k}$ and then part one of (7.8) becomes

$$\begin{bmatrix} 1 & 3(2^{2-k} - 1) & 3(2^{3-k} - 1) & 7.2^{2-k} - 1 \end{bmatrix} B = 2^{6-k} \begin{bmatrix} 1 & 0 & 0 & 2^k - 1 \end{bmatrix}, \qquad 0 \leq k \leq 2. \tag{7.9}$$

Similarly comparing (7.8) and Lemma 7.12 (c) we have that $\lambda = 2^{6-k}$ and then part three of (7.8) becomes

$$\left[\begin{array}{cccc} 1 & 0 & 0 & 2^{6-k}-1 \end{array}\right] B = 2^{6-k} \left[\begin{array}{cccc} 1 & 3(2^{k-4}-1) & 3(2^{k-3}-1) & 7.2^{k-4}-1 \end{array}\right], \qquad 4 \leq k \leq 6. \tag{7.10}$$

With these results we can state the following theorem.

Theorem 7.13 *$A_{2i}(k) = a_i(S_{6-k})$ and $\boldsymbol{a}(S_k)B = 2^{6-k}\boldsymbol{a}(S_{6-k})$ for $0 \leq k \leq 6$.*

Proof. The case $2 \leq k \leq 4$ can be seen in Hoggar (1997). The other two cases $0 \leq k \leq 2$ and $4 \leq k \leq 6$ can be seen in the explanation above. □

So we can get a complete table of the weight enumerators of the KM$(9, k, 7-k)$ codes as Table 7.1.

k	$A(x)$			
$0 \leq k \leq 2$	1			$+(2^k-1)x^6$
$2 \leq k \leq 4$	1		$+3(2^{k-2}-1)x^4$	$+(2^{k-2}+2)x^6$
$4 \leq k \leq 6$	1	$+3(2^{k-4}-1)x^2$	$+3(2^{k-3}-1)x^4$	$+(7.2^{k-4}-1)x^6$

Table 7.1: Complete table of weight enumerators of KM$(9, k, 7-k)$ codes

We can also obtain a complete table of the a_i for each k which can then be used for the construction of the larger family in the way of Section 6.6.2. These are shown in Table 7.2, and can also be found in Hoggar (1997).

k	a_0	a_1	a_2	a_3
$0 \leq k \leq 2$	1	$3(2^{2-k}-1)$	$3(2^{3-k}+1)$	$7.2^{2-k}-1$
$2 \leq k \leq 4$	1	0	$3(2^{4-k}-1)$	$2^{4-k}+2$
$4 \leq k \leq 6$	1	0	0	$2^{6-k}-1$

Table 7.2: Complete table of the a_i for KM$(9, k, 7-k)$ codes

7.3.2 The KM$(10, k, 8-k)$ Codes

We have a new generator which gives (the dash automatically being taken care of by the x in the second sum)

$$B(x) = \sum_{u \in S_k} \beta_{t(u)} + \sum_{u \in \zeta_{k-1}+S_k} x\beta_{t(u)}. \tag{7.11}$$

Now we know the MacWilliams Transform of β_t is $(1+x)\alpha_t$ and similarly $x\beta_t$ gives $(1-x)\alpha_t$. Also as $S_{k-1} = S_k \cup (\zeta_{k-1} + S_k)$, i.e. $\zeta_{k-1} + S_k = S_{k-1} \setminus S_k$, then

$$\begin{aligned} t_i(S_{k-1}) &= t_i(S_k \cup (\zeta_{k-1} + S_k)) \\ &= t_i(S_k) - t_i(\zeta_{k-1} + S_k) \\ &= t_i(S_k) - (t_i(S_{k-1}) - t_i(S_k)) \\ &= 2t_i(S_k) - t_i(S_{k-1}) \, . \end{aligned} \tag{7.12}$$

So

$$\begin{aligned} A(x) &= 2^{k-7} \left(\sum_{u \in S_k} (1+x)\alpha_{t(u)} + \sum_{u \in \zeta_{k-1}+S_k} (1-x)\alpha_{t(u)} \right) \\ &= 2^{k-7} \left(\sum_{u \in S_{k-1}} \alpha_{t(u)} + x \left(\sum_{u \in S_k} \alpha_{t(u)} - \sum_{u \in \zeta_{k-1}+S_k} \alpha_{t(u)} \right) \right) \end{aligned} \tag{7.13}$$

and we can write as is seen in Hoggar (1997),

$$\left[\begin{array}{cccc} 1 & A_2 & A_4 & A_6 \end{array} \right] = 2^{k-7} \left[\begin{array}{cccc} t_0(S_{k-1}) & t_1(S_{k-1}) & t_2(S_{k-1}) & t_3(S_{k-1}) \end{array} \right] B \tag{7.14}$$

and

$$\begin{aligned} \left[\begin{array}{cccc} 0 & A_3 & A_5 & A_7 \end{array} \right] = 2^{k-7} \Big[& 2t_0(S_k) - t_0(S_{k-1}) \quad 2t_1(S_k) - t_1(S_{k-1}) \\ & 2t_2(S_k) - t_2(S_{k-1}) \quad 2t_3(S_k) - t_3(S_{k-1}) \Big] B \end{aligned} \, . \tag{7.15}$$

Using Theorem 7.13 we can simplify these to

$$\left[\begin{array}{cccc} 1 & A_2 & A_4 & A_6 \end{array} \right] = \boldsymbol{a}(S_{7-k}) \tag{7.16}$$

and

$$\left[\begin{array}{cccc} A_1 & A_3 & A_5 & A_7 \end{array} \right] = \boldsymbol{a}(S_{6-k}) - \boldsymbol{a}(S_{7-k}). \tag{7.17}$$

Now we can get Table 7.3, the even weights of the KM$(10, k, 7-k)$, Table 7.4, the vectors $\boldsymbol{a}(S_{6-k})$, and Table 7.5 which contains the largest possible ranges for k. Note here that we are again taking the unusual step of finding $\boldsymbol{a}(S_{-1})$, which we are taking as $2\boldsymbol{a}(S_0)$ as we are just adding the zero vector ζ_{-1}.

k	A_0	A_2	A_4	A_6
0	1	0	0	0
$1,2,3$	1	0	0	2^k-1
$3,4,5$	1	0	$3(2^{k-3}-1)$	$2^{k-3}+2$
$5,6,7$	1	$3(2^{k-5}-1)$	$3(2^{k-4}+1)$	$7.2^{k-5}-1$

Table 7.3: The even weights of the KM$(10,k,8-k)$ codes

k	a_0	a_1	a_2	a_3
$0,1,2$	1	0	0	2^k-1
$2,3,4$	1	0	$3(2^{k-2}-1)$	$2^{k-2}+2$
$4,5,6$	1	$3(2^{k-4}-1)$	$3(2^{k-3}+1)$	$7.2^{k-4}-1$
7	2	$6(2^{k-5}-1)$	$6(2^{k-4}+1)$	$2(7.2^{k-5}-1)$

Table 7.4: Table of the vectors $\boldsymbol{a}(S_{6-k})$

k	A_1	A_2	A_3	A_4	A_5	A_6	A_7
1	0	0	0	0	0	2^k-1	2^{k-1}
2	0	0	0	0	0	2^k-1	2^{k-1}
3	0	0	0	$3(2^{k-3}-1)$	3.2^{k-3}	$2^{k-3}+2$	2^{k-3}
4	0	0	0	$3(2^{k-3}-1)$	3.2^{k-3}	$2^{k-3}+2$	2^{k-3}
5	0	0	3.2^{k-5}	$3(2^{k-3}-1)$	3.2^{k-4}	$2^{k-3}+2$	7.2^{k-5}
6	0	$3(2^{k-5}-1)$	3.2^{k-5}	$3(2^{k-4}+1)$	3.2^{k-4}	$7.2^{k-5}-1$	7.2^{k-5}
7	1	$3(2^{k-5}-1)$	$3(2^{k-5}-1)$	$3(2^{k-4}+1)$	$3(2^{k-4}-1)$	$7.2^{k-5}-1$	$7.2^{k-5}-1$

Table 7.5: The weight distribution of the KM$(10,k,8-k)$ codes

7.3.3 The KM$(12, k, 9-k)$ Codes

Further to Section 6.6.2 we present here some more ideas for using the $\boldsymbol{a}$ to obtain the weight enumerators. This technique is the same as the one used in Hoggar (1997).

As we see from Section 6.6.1, the α's used (see (6.73)) contain only even powers of x and so the weight enumerators of the KM$(12, k, 9-k)$ contain only even powers of x. We can thus write (6.83), using (6.84), as

$$\left[\begin{array}{ccccc} A_0 & A_2 & A_4 & A_6 & A_8 \end{array}\right] = 2^{k-8}\left[\boldsymbol{a}(S_k)^{\delta} + \boldsymbol{a}(S_{k-2})^{\sigma}\right] B. \tag{7.18}$$

Theorem 7.14 (Hoggar (1997)) *For the KM$(12, k, 8-k)$ codes, $(0 \leq k \leq 7)$,*

$$\left[\begin{array}{ccccc} A_0 & A_2 & A_4 & A_6 & A_8 \end{array}\right] = \boldsymbol{a}(S_{6-k})^{\sigma} + \boldsymbol{a}(S_{8-k})^{\delta} \tag{7.19}$$

.

Proof. Apply the results of Lemma 7.11 to (7.18). □

We need to obtain the general results of $\boldsymbol{a}(S_k)^{\sigma}$ and $\boldsymbol{a}(S_k)^{\delta}$ so we use Table 7.2 and the definitions of σ and δ to obtain Tables 7.6 and 7.7. Note that an equivalent to Table 7.7 appears in Hoggar (1997).

k	$(\boldsymbol{a}(S_k)^{\sigma})_0$	$(\boldsymbol{a}(S_k)^{\sigma})_1$	$(\boldsymbol{a}(S_k)^{\sigma})_2$	$(\boldsymbol{a}(S_k)^{\sigma})_3$	$(\boldsymbol{a}(S_k)^{\sigma})_4$
-1	0	2	$2.3(2^{1-k}-1)$	$2.3(2^{2-k}+1)$	$7.2^{1-k}+2$
$0,1,2$	0	1	$3(2^{2-k}-1)$	$3(2^{3-k}+1)$	$7.2^{2-k}-1$
$2,3,4$	0	1	0	$3(2^{4-k}-1)$	$2^{4-k}+2$
$4,5,6$	0	1	0	0	$2^{6-k}-1$

Table 7.6: The elements of $\boldsymbol{a}(S_k)^{\sigma}$

We can now get a complete table of the weight enumerators of the KM$(12, k, 9-k)$ codes, as found in Table 7.8.

7.3.4 The KM$(14, k, 10-k)$ Codes

In this section we give a thorough explanation of the techniques developed in Hoggar (1997), but we keep our mind on our families of KM codes in the sense of increasing s as in Section

k	$(\boldsymbol{a}(S_k)^\delta)_0$	$(\boldsymbol{a}(S_k)^\delta)_1$	$(\boldsymbol{a}(S_k)^\delta)_2$	$(\boldsymbol{a}(S_k)^\delta)_3$	$(\boldsymbol{a}(S_k)^\delta)_4$
$0,1,2$	1	$3.2^{2-k}-4$	$3(2^{2-k}+2)$	$2^{2-k}-4$	$1-7.2^{2-k}$
$2,3,4$	1	-1	$3(2^{4-k}-1)$	$5-2^{5-k}$	$-2-2^{4-k}$
$4,5,6$	1	-1	0	$2^{6-k}-1$	$1-2^{6-k}$

Table 7.7: The elements of $\boldsymbol{a}(S_k)^\delta$

k	A_0	A_2	A_4	A_6	A_8
$0,1,2$	1	0	0	0	2^k-1
$2,3,4$	1	0	0	2^k-4	3
$4,5,6$	1	0	$6(2^{k-4}-1)$	$2^{k-2}+8$	$3(2^{k-3}-1)$
7	1	$3.2^{k-6}-2$	$3.5.2^{k-6}$	$25.2^{k-6}+2$	$3.7.2^{k-6}-1$

Table 7.8: The weight distribution of the KM$(12,k,9-k)$ codes

4.9 (see specifically Examples 4.14 and 4.15 for this idea). Now we are concerned with $s=3$ which results in the multiplications

$$\begin{array}{lcl|lcl} m_9 &=& z_{k-1}y_{d-1} & m_{12} &=& z_{k-3}y_{d-3} \\ m_{10} &=& z_{k-2}y_{d-2} & m_{13} &=& (z_{k-3}+z_{k-1})(y_{d-3}+y_{d-1})\ . \\ m_{11} &=& (z_{k-2}+z_{k-1})(y_{d-2}+y_{d-1}) & & & \end{array} \tag{7.20}$$

This is slightly different from what is found in Hoggar (1997), but as we shall see (obviously, as the weight enumerator is unchanged by column manipulation, see Section 5.3.3) the weight enumerator is the same, although the method of getting to it has to be amended.

So we have the general shunted families as represented by

$$[z_0 \ldots z_{k-1}] \left[\begin{array}{ccc|cc} & & & 0 & 0 \\ & & & \vdots & \vdots \\ & & & 0 & 0 \\ & & & 1 & 1 \\ & & & 0 & 0 \\ & & & 0 & 1 \\ & & & 1 & 0 \\ & & & 1 & 1 \end{array}\right] = [c_0 \ldots c_{13}]\,. \tag{7.21}$$

From (7.21) we see that we have the relation $c_9+c_{10}+c_{11}=0$, which we include as the generator, E_4, as in the previous case for $s=2$. We also have another similar relation

as $c_9 + c_{12} + c_{13} = 0$, and the usual ones by equating the coefficients each side of (7.21), i.e. $z_{k-1} = c_9, z_{k-2} = c_{10}$ and $z_{k-3} = c_{12}$. These last four relations can be converted to the generators

$$\begin{aligned} e' &= e_9 + e_{12} + e_{13} \\ \zeta'_{k-1} &= \zeta_{k-1} + e_9 \\ \zeta'_{k-2} &= \zeta_{k-2} + e_{10} \\ \zeta'_{k-3} &= \zeta_{k-3} + e_{12} \end{aligned}$$

For $k = 2$ the generator ζ'_{k-3} is just e_{12}, but in the cases $k = 7$ and $k = 8$ we must use Theorem 6.4. We therefore get for $(1 \leq k \leq 6)$

$$\text{Dual KM}(14, k, 10-k) \text{ code} = \bigcup_{u \in \text{Span}\ \{e', \zeta'_{k-1}, \zeta'_{k-2}, \zeta'_{k-3}, S_k\}} (u + E). \tag{7.22}$$

For $k = 7$, just the same as in Section 6.6 we get that $\hat{z}_5 = z_{56}$ which from the matrix (7.21) will give the relation $\zeta'_5 = \zeta_5 + e_9 + e_{10}$, giving the dual code as

$$\text{Dual KM}(14, 7, 3) \text{ code} = \bigcup_{u \in \text{Span}\ \{e', \zeta'_5, \zeta'_4\}} (u + E). \tag{7.23}$$

The $k = 8$ case is similar to the $k = 7$ case so from Theorem 6.4 we get $\hat{z}_5 = z_{567}$, which from the matrix (7.21) can form the generator $\zeta'_5 = \zeta_5 + e_9 + e_{10} + e_{12}$. The dual code can therefore be defined as

$$\text{Dual KM}(14, 8, 2) \text{ code} = \bigcup_{u \in \text{Span}\ \{e', \zeta'_5\}} (u + E). \tag{7.24}$$

Similarly for completeness we give the dual code of the KM$(14, 9, 1)$ as

$$\text{Dual KM}(14, 9, 1) \text{ code} = \bigcup_{u \in \text{Span}\ \{e'\}} (u + E). \tag{7.25}$$

We now attempt to give the most general form of the weight enumerator construction, but noting the difference between the work of Hoggar (1997) and the work here being that we are using a slightly different form of the wraparound part of the matrix. As we shall see from Theorem 7.16 we restrict ourselves to the range $(3 \leq k \leq 6)$ for the even weights but are able to obtain a result for the odd weights for the range $2 \leq k \leq 6$. In fact the range

can be made bigger but the extremes are obtained directly. The cases $k = 1, 2, 7, 8, 9$ can be worked out individually (of course the cases $k = 1, 2, 9$ are easy and the remaining are explained fully above).

Now adding $\boldsymbol{e}'$ and $\boldsymbol{\zeta}_{k-3}$ has a more complicated affect on $B(x)$ and therefore $A(x)$ so we partition the dual code into four sets V_{cd}, with $c, d \in \mathbb{F}_2$, where

$$V_{cd} = \bigcup_{a,b \in \mathbb{F}_2} (a\boldsymbol{\zeta}'_{k-1} + b\boldsymbol{\zeta}'_{k-2} + c\boldsymbol{\zeta}'_{k-3} + d\boldsymbol{e}' + S_k). \tag{7.26}$$

Now any element of V_{cd} can be written $\boldsymbol{v} = a\boldsymbol{\zeta}'_{k-1} + b\boldsymbol{\zeta}'_{k-2} + c\boldsymbol{\zeta}'_{k-3} + d\boldsymbol{e}' + \boldsymbol{u}$ for $\boldsymbol{u} \in S_k$. The vectors of S_{k-3} have fourth triple 000, but the extra digits coming from the dashed convert the fourth triple to $[a + d, b, 0]$ so we see that $t(a\boldsymbol{\zeta}_{k-1} + b\boldsymbol{\zeta}_{k-2} + c\boldsymbol{\zeta}_{k-3} + \boldsymbol{u})$ is incremented by one when $\boldsymbol{v}_4 \neq 000$, i.e. when

$$abd \neq 000 \text{ or } 101. \tag{7.27}$$

Further, if r is the number of ones among the last two digits of $\boldsymbol{v}$, i.e. $c + d$ and d, then $R(\boldsymbol{v} + E) = x^r \beta_{t(\boldsymbol{v})}$. We can now derive Table 7.9, as follows.

First of all the case $cd = 00$ is directly from (7.18). For the case $cd = 01$, by (7.26) we have that the fourth triple of a vector V_{cd} contributes 1 to $t(\boldsymbol{v}), \boldsymbol{v} \in V_{cd}$, except when $ab = 10$. Hence

$$\begin{aligned} (\boldsymbol{a}(V_{cd}))_i &= t_i(\underline{10}) + t_{i-1}(\underline{00}, \underline{01}, \underline{11}) \\ &= t_{i-1}(\underline{00}, \underline{01}, \underline{10}, \underline{11}) + t_i(\underline{10}) - t_{i-1}(\underline{10}) \\ &= t_{i-1}(S_{k-2}) + t_i(\underline{10}) - t_{i-1}(\underline{10}) \\ &= \boldsymbol{a}\left((S_{k-2})^\sigma + \underline{10}^\delta\right)_i . \end{aligned} \tag{7.28}$$

For the case $cd = 10$, by (7.26) we have that the fourth triple of a vector V_{cd} contributes 1 to $t(\boldsymbol{v}), \boldsymbol{v} \in V_{cd}$, except when $ab = 00$. Hence

$$\begin{aligned} (\boldsymbol{a}(V_{cd}))_i &= t_i(\underline{001}) + t_{i-1}(\underline{011}, \underline{101}, \underline{111}) \\ &= t_{i-1}(\underline{001}, \underline{011}, \underline{101}, \underline{111}) + t_i(\underline{001}) - t_{i-1}(\underline{001}) \\ &= \boldsymbol{a}\left((S_{k-3} \setminus S_{k-2})^\sigma + \underline{001}^\delta\right)_i . \end{aligned} \tag{7.29}$$

For the case $cd = 11$, by (7.26) we have that the fourth triple of a vector V_{cd} contributes 1 to $t(\boldsymbol{v}), \boldsymbol{v} \in V_{cd}$, except when $ab = 10$. Hence

$$\begin{aligned}(\boldsymbol{a}(V_{cd}))_i &= t_i(\underline{101}) + t_{i-1}(\underline{001}, \underline{011}, \underline{111}) \\ &= t_{i-1}(\underline{001}, \underline{011}, \underline{101}, \underline{111}) + t_i(\underline{101}) - t_{i-1}(\underline{101}) \\ &= \boldsymbol{a}\left((S_{k-3} \setminus S_{k-2})^\sigma + \underline{101}^\delta\right)_i .\end{aligned} \tag{7.30}$$

For the case $k = 7$ we have $V_{cd} = \bigcup_{b \in \mathbb{F}_2} (b\zeta'_{k-2} + c\zeta'_{k-3} + de')$ and we look at each case separately. For example $cd = 10$ gives $V_{cd} = \{\zeta'_4, \zeta'_4 + \zeta'_5\}$ and so $\boldsymbol{a}(V_{cd}) = 00011$.

For the case $k = 8$ the set $V_{cd} = (c\zeta'_{k-3} + de')$, and again we look at each case separately. For example $cd = 01$ gives $V_{cd} = \{e'\}$, and so $\boldsymbol{a}(V_{cd}) = 01000$.

cd	$\boldsymbol{a}(V_{cd})$	$\boldsymbol{a}(V_{cd}), k = 7$	$\boldsymbol{a}(V_{cd}), k = 8$	r	$2^{10-k}(A(x)$, part of)
00	$\boldsymbol{a}(S_{k-2})^\sigma + \boldsymbol{a}(S_k)^\delta$	10001	10000	0	$\boldsymbol{a}.(1 + x^2 + 2x)\boldsymbol{\alpha}$
01	$\boldsymbol{a}(S_{k-2})^\sigma + \boldsymbol{a}(\underline{10})^\delta$	01001	01000	2	$\boldsymbol{a}.(1 + x^2 - 2x)\boldsymbol{\alpha}$
10	$\boldsymbol{a}(S_{k-3} \setminus S_{k-2})^\sigma + \boldsymbol{a}(\underline{001})^\delta$	00011	00001	1	$\boldsymbol{a}.(1 - x^2)\boldsymbol{\alpha}$
11	$\boldsymbol{a}(S_{k-3} \setminus S_{k-2})^\sigma + \boldsymbol{a}(\underline{101})^\delta$	00002	00001	1	$\boldsymbol{a}.(1 - x^2)\boldsymbol{\alpha}$

Table 7.9: Contribution to $A(x)$ for all the values of cd

Theorem 7.15 *The odd weight distribution of the $KM(14, k, 10-k)$ codes are*

$$\left[\begin{array}{ccccc} A_1 & A_3 & A_5 & A_7 & A_9 \end{array}\right] = \begin{cases} \boldsymbol{a}(S_{6-k})^\sigma - \boldsymbol{a}(S_{7-k})^\sigma, & 1 \le k \le 7 \\ [\ 0\ \ 2\ \ 18\ \ 54\ \ 54\ ], & k = 8 \end{cases} \tag{7.31}$$

Proof. The $k = 1$ case is obvious. For $2 \le k \le 6$ in the same way as Hoggar (1997), we see from Table 7.9 that the only contribution to odd powers in $A(x)$ is

$$2x(\boldsymbol{a}(S_{k-2})^\sigma + \boldsymbol{a}(S_k)^\delta).\boldsymbol{\alpha} - 2x(\boldsymbol{a}(S_{k-2})^\sigma + \boldsymbol{a}(\underline{10})^\delta).\boldsymbol{\alpha}. \tag{7.32}$$

Converting this to matrix form we get

$$\begin{aligned}[A_{2i+1}] &= 2^{k-10}2(\boldsymbol{a}(S_{k-2})^\sigma + \boldsymbol{a}(S_k)^\delta - \boldsymbol{a}(S_{k-2})^\sigma - \boldsymbol{a}(\underline{10})^\delta).B(4) \\ &= 2^{k-9}(2\boldsymbol{a}(S_k)^\delta - \boldsymbol{a}(S_{k-1})^\delta).B(4) .\end{aligned} \tag{7.33}$$

Now using Lemma 7.11 we get

$$\begin{aligned}[A_{2i+1}] &= 2^{k-9}(2.2^{8-k}\boldsymbol{a}(S_{6-k})^{\sigma} + 2^{7-k}\boldsymbol{a}(S_{7-k})^{\sigma}) \\ &= \boldsymbol{a}(S_{6-k})^{\sigma} - \boldsymbol{a}(S_{7-k})^{\sigma} \ .\end{aligned} \tag{7.34}$$

The cases $k = 7$ and $k = 8$ are obtained by direct calculation, the case $k = 7$ can be included by noting that it is equal to $\boldsymbol{a}(S_{-1})^{\sigma} - \boldsymbol{a}(S_0)^{\sigma}$ for our definition of S_{-1}. □

Theorem 7.16 *The even weight distribution of the $KM(14, k, 10-k)$ codes are (where $T_k = 2^{k-7}(\boldsymbol{a}(\underline{01}_{k-1}).B(3))$)*

$$\left[\begin{array}{cccccc} A_0 & A_2 & A_4 & A_6 & A_8 & A_{10} \end{array}\right] = \begin{cases} [\ 1\ 0\ 0\ 0\ 0\ 0\ ], & k = 1 \\ [\ 1\ 0\ 0\ 1\ 0\ 0\ ], & k = 2 \\ \sigma\delta\boldsymbol{a}(S_{8-k}) + \delta^2\boldsymbol{a}(S_{9-k}) & \\ +2^{-1}\sigma(\tau\boldsymbol{a}(S_{7-k}) + \delta T_k), & 3 \le k \le 6 \\ [\ 1\ 0\ 15\ 21\ 20\ 7\ ], & k = 7 \\ [\ 1\ 3\ 28\ 40\ 43\ 13\ ], & k = 8. \end{cases} \tag{7.35}$$

Proof. The cases $k = 1$ and $k = 2$ are easy to work out individually. The case $3 \le k \le 6$ is obtained by the results of Table 7.9 as each of the values of cd contribute to the even weights, as in Hoggar (1997), and using the results in Example 7.10. The reason that this way does not provide results for the $k = 2$ case is due to the fact that the range in Lemma 7.11 would be violated. Also $k = 7$ is not included here because of the fact that $\boldsymbol{a}(S_{k-2})^{\sigma} + \boldsymbol{a}(\underline{10})^{\delta} \neq [\ 0\ 1\ 0\ 0\ 1\ ]$, or alternatively noting that we have not defined (7.32) for $k > 6$. The cases $k = 7$ and $k = 8$ are obtained by direct calculation using the results of Table 7.9. □

We can now form the table of the weight distribution of the KM$(14, k, 10-k)$ codes which is given in Table 7.10.

7.3.5 Future Work

The methods introduced in this chapter are theoretically easily extended to any value of s as we have an infinite scheme for the wraparound as introduced in Section 4.9. There

k	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
1	1										1
2	1								1	2	
3	1							3	2	1	1
4	1						3	6	3	2	1
5	1					3	9	6	5	7	1
6	1				4	6	16	12	7	14	4
7	1			1	15	9	21	27	20	27	7
8	1		3	2	28	18	40	54	43	54	13

Table 7.10: The Weight Distribution of the KM$(14, k, 10-k)$ codes

was not enough time, however, for the work to be completed. The groundwork has been thoroughly set and future work will be completed in due course.

Chapter 8

Future Work

As explained briefly in the two previous chapters there is still much that can be done on the work of KM codes. We believe that they still hold an important part in the mathematical world and the more known about all sides of them the better.

Indeed classifying them will enable them to be used in more specific situations. As the development of GH-ARQ and related schemes increases, so codes with similar parameters to those of KM are needed and this may hold the future of them. The linear algebra used in the previous two chapters will hopefully give insight into other areas of mathematics and it is hoped that other areas can be directly related to this area to further the study.

The results of the previous two chapters highlight the relation between the shunted codes and this may be used for many purposes when the code characteristics need to be altered.

Of course the work done in chapter 4 is still of the upmost importance not only to KM codes, but also to complexity theory, and it is hoped that the new diagrammatic method will help others to find useful and important algorithms which can, of course, then be used to construct better KM codes. Also, other new areas of complexity theory can be directly used to further the study of KM codes and it is hoped that the two areas can move ahead together.

Appendix 1

Below we give a set of standard algorithms for multiplication of the polynomials $Z(u) = z_0 + z_1u + z_2u^2 + z_3u^3$ and $Y(u) = y_0 + y_1u$, modulo a polynomial $P(u)$, before and after multiplication (for use in the CRT formation of KM codes among other uses).

Modulo, $P(u)$	Multiplications	Algorithm modulo $P(u)$
u	$m = z_0y_0$	m
$u+1$	$m = (z_0 + z_1 + z_2 + z_3)(y_0 + y_1)$	m
u^2	$m_0 = z_0y_0$ $m_1 = z_1y_1$ $m_2 = (z_0 + z_1)(y_0 + y_1)$	$m_0 + (m_0 + m_1 + m_2)u$
$u^2 + 1$	$m_0 = (z_0 + z_2)y_0$ $m_1 = (z_1 + z_3)y_1$ $m_2 = (z_0 + z_1 + z_2 + z_3)(y_0 + y_1)$	$(m_0 + m_2) + (m_0 + m_2)u$
$u^2 + u$	$m_0 = z_0y_0$ $m_1 = (z_1 + z_2 + z_3)y_1$ $m_2 = (z_0 + z_1 + z_2 + z_3)(y_0 + y_1)$	$(m_0 + m_2) + (m_0 + m_2)u$
$u^2 + u + 1$	$m_0 = (z_0 + z_2 + z_3)y_0$ $m_1 = (z_1 + z_2)y_1$ $m_2 = (z_0 + z_1 + z_3)(y_0 + y_1)$	$(m_0 + m_2) + (m_0 + m_1)u$

Algorithms, I

Modulo, $P(u)$	Multiplications	Algorithm modulo $P(u)$
u^3	$m_0 = z_0 y_0$ $m_1 = z_1 y_1$ $m_2 = (z_0 + z_1)(y_0 + y_1)$ $m_3 = (z_0 + z_2)y_0$	$m_0 + (m_0 + m_1 + m_2)u +$ $(m_0 + m_1 + m_3)u^2$
$u^3 + 1$	$m_0 = (z_0 + z_3)y_0$ $m_1 = z_1 y_1$ $m_2 = (z_0 + z_1 + z_3)(y_0 + y_1)$ $m_3 = (z_1 + z_2)y_1$ $m_4 = (z_0 + z_2 + z_3)y_0$	$m_0 + (m_0 + m_1 + m_2)u +$ $(m_0 + m_1 + m_4)u^2$
$u^3 + u$	$m_0 = z_0 y_0$ $m_1 = (z_1 + z_3)y_1$ $m_2 = (z_0 + z_1 + z_3)(y_0 + y_1)$ $m_3 = (z_1 + z_2 + z_3)y_1$ $m_4 = (z_0 + z_2)y_0$	$m_0 + (m_0 + m_2 + m_3)u +$ $(m_0 + m_1 + m_4)u^2$

Algorithms, II

Modulo, $P(u)$	Multiplications	Algorithm modulo $P(u)$
u^3+u+1	$m_0 = (z_0+z_3)y_0$ $m_1 = (z_1+z_3)y_1$ $m_2 = (z_0+z_1)(y_0+y_1)$ $m_3 = (z_1+z_2+z_3)y_1$ $m_4 = (z_0+z_2+z_3)y_0$	$m_0+(m_0+m_2+m_3)u+$ $(m_0+m_3+m_4)u^2$
u^3+u^2	$m_0 = z_0y_0$ $m_1 = z_1y_1$ $m_2 = (z_0+z_1)(y_0+y_1)$ $m_3 = (z_1+z_2+z_3)y_1$ $m_4 = (z_0+z_2+z_3)y_0$	$m_0+(m_0+m_1+m_2)u+$ $(m_0+m_3+m_4)u^2$
u^3+u^2+1	$m_0 = (z_0+z_3)y_0$ $m_1 = z_1y_1$ $m_2 = (z_0+z_1+z_3)(y_0+y_1)$ $m_3 = (z_1+z_2+z_3)y_1$ $m_4 = (z_0+z_2)y_0$	$(m_0+m_1+m_3)+$ $(m_0+m_1+m_2)u+$ $(m_0+m_3+m_4)u^2$

Algorithms, III

Modulo, $P(u)$	Multiplications	Algorithm modulo $P(u)$
u^3+u^2+u	$m_0 = z_0y_0$ $m_1 = (z_1+z_3)y_1$ $m_2 = (z_0+z_1+z_3)(y_0+y_1)$ $m_3 = (z_1+z_2)y_1$ $m_4 = (z_0+z_2+z_3)y_0$	$m_0+(m_0+m_2+m_3)u+$ $(m_0+m_3+m_4)u^2$
u^3+u^2+u+1	$m_0 = (z_0+z_3)y_0$ $m_1 = (z_1+z_3)y_1$ $m_2 = (z_0+z_1+z_3)(y_0+y_1)$ $m_3 = (z_1+z_2)y_1$ $m_4 = (z_0+z_2+z_3)y_0$	$(m_0+m_1+m_3)+$ $(m_0+m_2+m_3)u+$ $(m_0+m_3+m_4)u^2$

Algorithms, IV

Wraparound, s	Multiplications	Algorithm modulo u^s
1	$m = z_3y_1$	m
2	$m_0 = z_3y_1$ $m_1 = z_2y_0$ $m_2 = (z_2+z_3)(y_0+y_1),$	$m_0+(m_0+m_1+m_2)u$
3	$m_0 = z_3y_1$ $m_1 = z_2y_0$ $m_2 = (z_2+z_3)(y_0+y_1)$ $m_3 = (z_1+z_3)y_1$ $m_4 = (z_1+z_2)y_0,$	$m_0+(m_0+m_1+m_2)u+$ $(m_0+m_1+m_3)u^2$

Algorithms, V

Appendix 2

In this appendix we give tables of all polynomials of degree less than or equal to five over $\mathbb{F}_2$, together with their factorisations. The complete factorisation is in bold, the others are coprime factorisations, i.e. in the form $P_1(u)\dots P_t(u)$ such that $(P_i(u), P_j(u)) = 1, i \neq j$.

Polynomial	Factorization		Polynomial	Factorization
u	$\boldsymbol{u}$		u^3	$\boldsymbol{u^3}$
$u+1$	$\boldsymbol{u+1}$		u^3+1	$\boldsymbol{(u+1)(u^2+u+1)}$
u^2	$\boldsymbol{u^2}$		u^3+u	$\boldsymbol{u(u+1)^2}$
u^2+1	$\boldsymbol{(u+1)^2}$			$u(u^2+1)$
u^2+u	$\boldsymbol{u(u+1)}$		u^3+u+1	$\boldsymbol{u^3+u+1}$
u^2+u+1	$\boldsymbol{u^2+u+1}$		u^3+u^2	$\boldsymbol{u^2(u+1)}$
			u^3+u^2+1	$\boldsymbol{u^3+u^2+1}$
			u^3+u^2+u	$\boldsymbol{u(u^2+u+1)}$
			u^3+u^2+u+1	$\boldsymbol{(u+1)^3}$

Degree 1, 2 and 3 Factorisations

Polynomial	Factorization
u^4	$\boldsymbol{u^4}$
u^4+1	$\boldsymbol{(u+1)^4}$
u^4+u	$\boldsymbol{u(u+1)(u^2+u+1)}$
	$(u^2+u)(u^2+u+1)$
	$u(u^3+1)$
	$(u+1)(u^3+u^2+u)$
u^4+u+1	$\boldsymbol{u^4+u+1}$
u^4+u^2	$\boldsymbol{u^2(u+1)^2}$
	$u^2(u^2+1)$
u^4+u^2+1	$\boldsymbol{(u^2+u+1)^2}$
u^4+u^2+u	$\boldsymbol{u(u^3+u+1)}$
u^4+u^2+u+1	$\boldsymbol{(u+1)(u^3+u^2+1)}$
u^4+u^3	$\boldsymbol{u^3(u+1)}$
u^4+u^3+1	$\boldsymbol{u^4+u^3+1}$
u^4+u^3+u	$\boldsymbol{u(u^3+u^2+1)}$
u^4+u^3+u+1	$\boldsymbol{(u+1)^2(u^2+u+1)}$
	$(u^2+1)(u^2+u+1)$
$u^4+u^3+u^2$	$\boldsymbol{u^2(u^2+u+1)}$
$u^4+u^3+u^2+1$	$\boldsymbol{(u+1)(u^3+u+1)}$
$u^4+u^3+u^2+u$	$\boldsymbol{u(u+1)^3}$
	$u(u^3+u^2+u+1)$
$u^4+u^3+u^2+u+1$	$\boldsymbol{u^4+u^3+u^2+u+1}$

Degree 4 Factorisations

Polynomial	Factorization
u^5	$\boldsymbol{u^5}$
u^5+1	$\boldsymbol{(u+1)(u^4+u^3+u^2+u+1)}$
u^5+u	$\boldsymbol{u(u+1)^4}$
	$u(u^4+1)$
u^5+u+1	$\boldsymbol{(u^2+u+1)(u^3+u^2+1}$
u^5+u^2	$\boldsymbol{u^2(u+1)(u^2+u+1)}$
	$u^2(u^3+1)$
	$(u^3+u)(u^2+u+1)$
	$(u+1)(u^4+u^3+u^2)$
u^5+u^2+1	$\boldsymbol{u^5+u^2+1}$
u^5+u^2+u	$\boldsymbol{u(u^4+u+1)}$
u^5+u^2+u+1	$\boldsymbol{(u+1)^2(u^3+u+1)}$
	$(u^2+1)(u^3+u+1)$
u^5+u^3	$\boldsymbol{u^3(u+1)^2}$
	$u^3(u^2+1)$
u^5+u^3+1	$\boldsymbol{u^5+u^3+1}$
u^5+u^3+u	$\boldsymbol{u(u^2+u+1)^2}$
	$u(u^4+u^2+1)$
u^5+u^3+u+1	$\boldsymbol{(u+1)(u^4+u^3+1)}$
$u^5+u^3+u^2$	$\boldsymbol{u^2(u^3+u+1)}$
$u^5+u^3+u^2+1$	$\boldsymbol{(u+1)^3(u^2+u+1)}$
	$(u^3+u^2+u+1)(u^2+u+1)$
$u^5+u^3+u^2+u$	$\boldsymbol{u(u+1)(u^3+u^2+1)}$
	$u(u^4+u^2+u+1)$
	$(u+1)(u^4+u^3+u)$
	$(u^2+u)(u^3+u^2+1)$
$u^5+u^3+u^2+u+1$	$\boldsymbol{u^5+u^3+u^2+u+1}$

Degree 5 Factorisations Part 1

Polynomial	Factorization
u^5+u^4	$\boldsymbol{u^4(u+1)}$
u^5+u^4+1	$\boldsymbol{(u^3+u+1)(u^2+u+1)}$
u^5+u^4+u	$\boldsymbol{u(u^4+u^3+1)}$
u^5+u^4+u+1	$\boldsymbol{(u+1)^5}$
$u^5+u^4+u^2$	$\boldsymbol{u^2(u^3+u^2+1)}$
$u^5+u^4+u^2+1$	$\boldsymbol{(u+1)(u^4+u+1)}$
$u^5+u^4+u^2+u$	$\boldsymbol{u(u+1)^2(u^2+u+1)}$
	$u(u^2+1)(u^2+u+1)$
	$u(u^4+u^3+u+1)$
	$(u^2+1)(u^3+u^2+u)$
	$(u^3+u)(u^2+u+1)$
$u^5+u^4+u^2+u+1$	$\boldsymbol{u^5+u^4+u^2+u+1}$
$u^5+u^4+u^3$	$\boldsymbol{u^3(u^2+u+1)}$
$u^5+u^4+u^3+1$	$\boldsymbol{(u+1)^2(u^3+u^2+1)}$
	$(u^2+1)(u^3+u^2+1)$
$u^5+u^4+u^3+u$	$\boldsymbol{u(u+1)(u^3+u+1)}$
	$u(u^4+u^3+u^2+1)$
	$(u+1)(u^4+u^2+u)$
	$(u^2+u)(u^3+u+1)$
$u^5+u^4+u^3+u+1$	$\boldsymbol{u^5+u^4+u^3+u+1}$
$u^5+u^4+u^3+u^2$	$\boldsymbol{u^2(u+1)^3}$
	$u^2(u^3+u^2+u+1)$
$u^5+u^4+u^3+u^2+1$	$\boldsymbol{u^5+u^4+u^3+u^2+1}$
$u^5+u^4+u^3+u^2+u$	$\boldsymbol{u(u^4+u^3+u^2+u+1)}$
$u^5+u^4+u^3+u^2+u+1$	$\boldsymbol{(u+1)(u^2+u+1)^2}$
	$(u+1)(u^4+u^2+1)$

Degree 5 Factorisations Part 2

References

AVERBUCH, A., WINOGRAD, S. & GALIL, Z.,Classification of all the Minimal Bilinear Algorithms for Computing the Coefficients of the Products of Two Polynomials Modulo a Polynomial, *Lecture Notes In Computer Science* **226** 31-39 (1986)

BSHOUTY, N.H., A Lower Bound for the Multiplication of Polynomials Modulo a Polynomial, *Information Processing Letters* **41** 321-326 (1992)

CHUDNOVSKY, D.V. & CHUDNOVSKY, G.V., Algebraic Complexities and Algebraic Curves over Finite Fields, *Journal of Complexity* **4** 285-316 (1988)

FIDUCCIA, C.M., Fast Matrix Multiplication, *Proc. of the 3rd Annual ACM Symposium on the Theory of Computing, Shaker Heights, Ohio*, 45-49 (1971)

HILL, R., *A First Course in Coding Theory* Oxford University Press, Oxford (1986)

HOGGAR, S.G. & PICKAVANCE, K., The Weight Distribution of KM Codes, *IMA Conference on the Applications of Combinatorial Mathematics* Oxford University Press (1997)

HOGGAR, S.G., Weight Enumerators of KM Codes with Wraparound, *To Appear in Discrete Mathematics* (1997)

HOPCROFT, J. & MUSINSKI, J., Duality Applied to the Complexity of Matrix Multiplication and other Bilinear Forms, *SIAM Journal Computing* **2** 159-173 (1973)

KAMINSKI, M., A Lower Bound for Polynomial Multiplication *Theoretical Computer Science* **40** 319-322 (1985)

KNUTH, D.E., *The Art of Computer Programming, Vol. 2*, Addison-Wesley, Reading, MA (1981)

KRISHNA, H., On a signal Processing Algorithm Based Class of Linear Codes, *Applicable Algebra in Engineering, Communication and Computing* **4**, 41-57 (1993)

KRISHNA, H., Computational Complexity of Bilinear Forms, *Lecture Notes in Control and Information Sciences* **94**, Springer-Verlag (1987)

KRISHNA, H. & MORGERA, S.D., A New Error Control Scheme for Hybrid ARQ Systems, *IEEE Trans. Communications*, **35** 981-989 (1987)

LANG, S., *Algebra,* Addison-Wesley; Reading, Mass. (1993)

LEMPEL, A., SEROUSSI, G. & WINOGRAD, S., On the Complexity of Multiplication in Finite Fields, *Theoretical Computer Science* **22** 285-296 (1983)

LEMPEL, A. & WINOGRAD, S., A New Approach to Error-Correcting Codes, *IEEE Trans. Information Theory*, **23** 503-508 (1977)

LIDL, R. & NIEDERREITER, H., Finite Fields, *Encyclopedia of Math. and its Applications*, **20** Addison-Wesley; Reading, Mass. (1983)

LIN, S. & COSTELLO JR., D.J., *Error-Control Coding: Fundamentals and Applications*, Prentice Hall, Inc. Englewood Cliffs (1983)

LINT, J.H. VAN, *Introduction to Coding Theory*, Springer, New York (1982)

MCFARLANE, I., *Generalised Type-II Hybrid Automatic Repeat Request Schemes and KM Codes,* MSc Thesis, University of Glasgow Mathematics Department (1991)

MACWILLIAMS, F.J., *A Theorem on the Distribution of Weights in a Systematic Code*, Bell System Tech. J., **42** 79-94 (1963)

MACWILLIAMS, F.J. & SLOANE, N.J.A., *The Theory of Error-Correcting Codes*, North Holland, Amsterdam (1977)

WINOGRAD, S., On the Number of Multiplications Necessary to Compute Certain Functions, *Comms. on Pure and Applied Maths.* **XXIII** 165-179 (1970)

WINOGRAD, S., Some Bilinear Forms Whose Multiplicative Complexity Depends on the Field of Constants, *Mathematical Systems Theory* **10** 162-180 (1977)