belong to the null space of the code. Given these 10-flats, the 9-flats have to be determined. However, notice that $\delta = \gcd(9, 12) = 3$. Then, $c' = 9/3 = 3$ and $m' = 12/3 = 4$. I.e. these 9-flats can be regarded as 3-flats in $EG(4, 2^3)$. At the next step, given the 3-flats in $EG(4, 2^3)$ the 2-flats in $EG(4, 2^3)$ (or the 6-flats in $EG(12, 2)$) have to be determined. However, $\delta' = \gcd(2, 4) = 2$ and therefore $c'' = 2/2 = 1$, $m'' = 4/2 = 2$ and these 2-flats in $EG(4, 2^3)$ can be regarded as 1-flats in $EG(2, 2^2)$. Finally, given the 1-flats in $EG(2, 2^2)$ (or the 6-flats in $EG(12, 2)$), the 0-flats have to be determined.

Note that using the above improved algorithm, we first determined the 6-flats from the 9-flats and then the 0-flats from the 6-flats; i.e. the original 10-step algorithm has been reduced down to 2-steps!

The above decoding path together with other paths representing decoding procedures for other Reed-Muller codes and Euclidean Geometry codes is given in figure (6).

2.3.4 Two-step decoding of all EG and PG codes.

A modification of the Reed algorithm capable of decoding all Euclidean Geometry and Projective Geometry codes in two steps is now stated.

Consider an r^{th} order Projective Geometry code. Then all r -flats of $PG(m, p^e)$ belong to its null space. The number of r -flats intersecting on a particular $(r-1)$ -flat is given by

$$\zeta = \frac{p^{e(m-r+1)} - 1}{p^e - 1} \quad (73)$$

(as in the previous section).

At the first step of the modified algorithm all r -flats are determined provided that at most $\lfloor \zeta/2 \rfloor$ errors occur, these r -flats are determined correctly in the usual way.

If $r = 1$, the process terminates at this point, while for $r = 2$ an additional step is required if the usual majority logic algorithm is used for decoding. However, for $r \geq 3$ at least two more steps are required.

The following method is able to decode all these codes in two steps as it is

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Services des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

**THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED**

THÈSES CANADIENNES

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

**LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE**

REED -MULLER CODES : STRUCTURE AND DECODING

by

RALLIS ANTONIADIS

(B.Sc. Athens, M.Sc. (I.S.S.) Carleton Univ.)

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Science

Department of Mathematics and Statistics

Carleton University

Ottawa, Ontario

Date : September 1986.

copyright

1986, Rallis Antoniadis

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-35849-1

B 507. Pu-Hemoa

B. Martinen
Thesis Supervisor

E.S. Bain bridge

(ii)

ABSTRACT

The structure and the properties of Reed-Muller codes are surveyed.

Several methods for the decoding of Reed-Muller codes are described and compared. Majority-logic decoding is first explained along with two improvements of the original Reed Algorithm. Next, a new decoding algorithm due to K. Tokiwa, T. Sugimura is stated. Finally, a special treatment to the first-order Reed-Muller codes is given and a fast decoding algorithm for them is presented.

ACKNOWLEDGEMENTS.

I would like to thank my supervisor Dr. Brian Mortimer for his helpful guidance throughout the present thesis.

I would also like to thank the Department of Mathematics and Statistics for supporting me during my studies.

Table of Contents

Acceptance sheet	ii
Abstract	iii
Acknowledgements	iv
Table of contents	v
Introduction	1
1. Introduction to Reed-Muller codes	3
1.1 Definitions and basic properties	3
1.2 Geometric interpretation	14
2. Decoding of Reed-Muller codes	22
2.1 Majority logic decoding of Reed-Muller codes	22
2.2 Geometric interpretation of the decoding scheme	30
2.3 Majority logic decoding (Part II)	35
2.3.1 Preliminaries	35
2.3.2 Majority decoding	38
2.3.3 Improvements of the Reed Algorithm	45
2.3.4 Two-step decoding of all EG and PG codes	50
2.4 New decoding algorithm for Reed-Muller codes	54
2.4.1 Simple iterated codes and the decoding algorithm	56
2.4.2 Algorithm for obtaining subcodes of an $R(r,m)$ code	59
2.4.3 A decoding algorithm for Reed-Muller codes	64
2.5 Fast decoding of first order Reed-Muller codes	74
2.5.1 Encoding of $R(1,m)$ and preliminary results	75
2.5.2 Decoding Algorithm for the $R(1,m)$ code based on FWT	83
2.5.3 A fast decoding algorithm for first order Reed-Muller codes	86
References	90

Introduction .

One of the major problems in digital data communication systems is the occurrence of errors in the data transmitted over a noisy channel.

Since the demand for efficient and reliable digital data transmission systems has been dramatically increased in the last few years because of the increasing use of automatic data processors and the rising need for long range communications, major efforts have been made by the communication engineers to control these errors so that reliable transmission of data can be obtained.

There are three essential phases of the coding problem :

- (i): To find codes that have the required error-correcting ability. (Usually, the most effective codes are long).
- (ii): To find a practical method of encoding.
- (iii): To find a practical error-correction method.

Since several classes of long, powerful codes as well as decoding procedures which can be implemented with a modest amount of hardware for several of these codes have been devised, the use of error-correcting codes has been made quite practical in recent data communication systems.

Reed-Muller codes are one of the oldest families of codes. They were first proposed by D. E. Muller [6] and a decoding algorithm for these codes was devised by I. S. Reed [9]. The basic properties of Reed-Muller codes as well as their geometric interpretation are stated in Chapter 1.

The major problem in the practical application of error-correcting codes is the complexity of decoding. One of the basic decoding complexity tradeoffs is that between decoding time and hardware cost. Decoding algorithms range from expensive combinatorial schemes that operate in minimum time to slow sequential schemes that require the minimal amount of hardware, while there seem to be only a few decoding schemes in the middle range.

The decoding of Reed-Muller codes is studied in Chapter 2 and various decoding schemes are stated.

In recent years, decoding schemes based on majority logic have received consid-

erable attention because the decoders are very easy to implement.

Reed-Muller codes are majority logic decodable codes and, hence, they have inherent all of the advantages of this class of codes. The majority logic decoding of Reed-Muller codes is described in section 2.1 followed by its geometric interpretation.

However, the complexity of majority logic decoding increases exponentially with the number of levels of majority logic which must be employed. After a few preliminary and majority logic results (given in sections 2.3.1 and 2.3.2) two improvements of the Reed algorithm are stated in section 2.3.3. The first improvement concerns the reduction of the number of majority gates, adders, etc. to be employed. The second improvement concerns the reduction of the number of steps of the decoding algorithm.

The Reed-Muller codes are members of a broader class of codes, namely, the Finite Geometry codes which include the Euclidean Geometry and Projective Geometry codes which can also be decoded with a majority logic decoding algorithm. A two-step decoding of all Euclidean Geometry and Projective Geometry codes is given in section 2.3.4.

The interpretation of Reed-Muller codes in terms of superimposition and a new decoding algorithm of much shorter delay than that of the majority logic algorithm (especially for large order Reed-Muller codes) proposed by K. Tokiwa, T. Sugimura et al. is the study of section 2.4.

A special treatment is given to the first order Reed-Muller codes which appear to be one of the most popular classes of codes¹ which results from their high correcting capacity as well as from the existence of fast decoding algorithms for them, two of which are given in section 2.5.

In particular, the number of operations required to decode a block using the second decoding algorithm for the first order Reed-Muller codes increases linearly in relation to the code length.

¹ They were used in the transmission of pictures from Mars.

Chapter 1

Introduction to Reed -Muller codes

Reed-Muller codes are one of the oldest families of codes. They are a class of binary group codes and can be regarded as cyclic codes with an over-all parity check added and they cover a wide range of rate and minimum distance.

Moreover, the binary Reed-Muller codes are the simplest examples of the class of geometrical codes all of which are majority-logic decodable codes. As a consequence, they can easily be implemented and this is one of the most attractive features of RM-codes.

1.1 Definitions and basic properties

Let $\mathbf{v} = (v_1, v_2, \dots, v_m)$ range over V^m (the set of all binary m -tuples), and $f(\mathbf{v}) = f(v_1, v_2, \dots, v_m)$ be a function the range of which is the set $\{0, 1\}$. Such a function is called *Boolean function* and it can be specified by its *truth table* which gives the values of f at all of its 2^m arguments. The 2^m -length vector which contains the values of f as $\mathbf{v}$ ranges over V^m is denoted by $\mathbf{f}$.

Definition : The r^{th} -order binary Reed-Muller code $R(r, m)$ of length $n = 2^m$, for $0 \leq r \leq m$, is the set of all vectors $\mathbf{f}$, where $f(v_1, v_2, \dots, v_m)$ is a Boolean function which is a polynomial of degree at most r .

Example : For simplicity reasons, consider the first order Reed-Muller code of length $8 = 2^3$ ($m = 3$).

Since the order $r = 1$, $f(v_1, v_2, v_3)$ is a Boolean function which is a polynomial of degree at most 1 ; i.e.

$$f(v_1, v_2, v_3) = a_0 1 + a_1 v_1 + a_2 v_2 + a_3 v_3, \quad a_i = 0 \text{ or } 1 \quad (i = 0, 1, 2, 3).$$

The 16 codewords of the first order RM-code of length 8 are shown below :

0	:	0 0 0 0 0 0 0 0
v_3	:	0 0 0 0 1 1 1 1
v_2	:	0 0 1 1 0 0 1 1
v_1	:	0 1 0 1 0 1 0 1
$v_2 + v_3$	:	0 0 1 1 1 1 0 0
$v_1 + v_3$	:	0 1 0 1 1 0 1 0
$v_1 + v_2$	:	0 1 1 0 0 1 1 0
$v_1 + v_2 + v_3$	:	0 1 1 0 1 0 0 1
1	:	1 1 1 1 1 1 1 1
$1 + v_1$	:	1 0 1 0 1 0 1 0
$1 + v_2$	:	1 1 0 0 1 1 0 0
$1 + v_3$	:	1 1 1 1 0 0 0 0
$1 + v_2 + v_3$	:	1 1 0 0 0 0 1 1
$1 + v_1 + v_3$	:	1 0 1 0 0 1 0 1
$1 + v_1 + v_2$	:	1 0 0 1 1 0 0 1
$1 + v_1 + v_2 + v_3$	:	1 0 0 1 0 1 1 0

Similarly, for the 2^{nd} order Reed-Muller code of length $8 = 2^3$, $f(v_1, v_2, v_3)$ is a Boolean function which is a polynomial of degree at most 2; i.e.

$$f(v_1, v_2, v_3) = a_0 1 + a_1 v_1 + a_2 v_2 + a_3 v_3 + a_{21} v_2 v_1 + a_{31} v_3 v_1 + a_{32} v_3 v_2,$$

where a_i, a_{kl} are 0 or 1 ($i = 0, \dots, 3$; $k = 2, 3$; $l = 1, 2$; ($k > l$)).

The 2^{nd} order RM-code of length 8 consists of 128 codewords.

Now let $n = 2^m$ and consider the binary vectors v_i , $i = 1, 2, \dots, m$ of length n consisting of 2^{m-1} blocks of equal length 2^1 , each of which contains 2^{1-1} consecutive entries equal to 0 followed by 2^{1-1} consecutive entries equal to 1, as shown in the following table :

Table 1.
$$\begin{array}{rcl} v_m & : & 00 \dots 00 \dots 00 \dots 00 \dots 00 11 \dots 11 \dots 11 \dots 11 \dots 11 \\ v_{m-1} & : & 00 \dots 00 \dots 01 \dots 11 \dots 11 00 \dots 00 \dots 01 \dots 11 \dots 11 \\ v_{m-2} & : & 00 \dots 01 \dots 10 \dots 01 \dots 11 00 \dots 01 \dots 10 \dots 01 \dots 11 \\ \vdots & & \vdots \\ v_1 & : & 01 \dots \dots \dots \dots \dots \dots \dots \dots 01 \end{array}$$

Next, consider the following arrangement of the vectors in (1) :

$$\begin{array}{c}
 1 \\
 v_1 \\
 v_2 \\
 v_2 v_1 \\
 v_3 \\
 v_3 v_1 \\
 v_3 v_2 \\
 v_3 v_2 v_1 \\
 v_4 \\
 v_4 v_1 \\
 v_4 v_2 \\
 v_4 v_2 v_1 \\
 v_4 v_3 \\
 \vdots \\
 v_4 v_3 v_2 v_1 \\
 \vdots \\
 v_m \\
 v_m v_1 \\
 \vdots \\
 v_m v_{m-1} \\
 \vdots \\
 v_m v_{m-1} \dots v_3 v_1
 \end{array} \tag{2}$$

According to the definition of the v_i 's preceding table 1, the vector corresponding to the product

$$v_k v_l \dots v_r \dots v_s$$

has its first

$$2^{k-1} + 2^{l-1} + \dots + 2^{r-1} + \dots + 2^{s-1}$$

entries all equal to 0, followed by a nonzero entry (i.e. 1). As a consequence, it can be readily checked that the arrangement (2) of the vectors (1) gives an $n \times n$ upper

triangular matrix with a diagonal consisting of 1's of the form

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & & & & \dots & & & & 1 \\ 0 & 0 & 1 & 1 & 0 & & & & \dots & & & & 1 \\ 0 & 0 & 0 & 1 & 0 & & & & \dots & & & & 1 \\ 0 & 0 & 0 & 0 & 1 & & & & \dots & & & & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & & & \dots & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & & & & \dots & & & & 1 \end{pmatrix}$$

The above matrix is of full rank and this implies that the vectors (1) are linearly independent. Since all the codewords of $R(m, m)$ are linear combinations of the above n vectors, we finally conclude that these vectors form a basis for $R(m, m)$.¹

Corollary 1 : For any r , $0 \leq r \leq m$, the r^{th} order Reed-Muller code $R(r, m)$ has a basis among the vectors (1).

Proof : Consider the vectors

$$1, v_1, v_2, \dots, v_m, v_1 v_2, v_2 v_1, \dots, v_m v_1, \dots, v_m v_{m-1}, \dots \quad (\text{up to degree } r)$$

This set of vectors is a subset of the vectors (1) which by Theorem 1 are linearly independent. Hence, the above set of vectors is linearly independent as well, and since by the definition of the r^{th} order Reed-Muller code $R(r, m)$ every codeword of $R(r, m)$ is a linear combination of them, we finally conclude that this set of vectors is a basis for $R(r, m)$. Since r was chosen arbitrarily from the set $\{1, 2, \dots, m\}$, the assertion is true for any r , $0 \leq r \leq m$.

What is the cardinality of the basis for $R(r, m)$? I.e. what is the dimension of the code?

Note that there are

$$k = 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r} = \sum_{i=0}^r \binom{m}{i}$$

basis vectors and therefore the dimension of the r^{th} order Reed-Muller codes of length $n = 2^m$ is

$$k = \sum_{i=0}^r \binom{m}{i} \quad (3)$$

¹ See also [3] for a proof based on Boolean functions.

(in the special case $r = m$, we have from (3) that $k = n$).

Example : Consider the case $m = 4$; i.e. the Reed-Muller codes of length $n = 2^4 = 16$. The 16 possible basis vectors are shown below :

Table 2.

1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	(i)
v_4	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	(ii)
v_3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1	(iii)
v_2	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1	(iv)
v_1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	(v)
$v_4 v_3$	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1	(vi)
$v_4 v_2$	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1	(vii)
$v_4 v_1$	0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0	(viii)
$v_3 v_2$	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1	(ix)
$v_3 v_1$	0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0	(x)
$v_2 v_1$	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1	(xi)
$v_4 v_3 v_2$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	(xii)
$v_4 v_3 v_1$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	(xiii)
$v_4 v_2 v_1$	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0	(xiv)
$v_3 v_2 v_1$	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1	(xv)
$v_4 v_3 v_2 v_1$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	(xvi)

Then the basis vectors for $R(r, 4)$ are given below :

Table 2a.

r	Rows
0	(i)
1	(i)-(v)
2	(i)-(xi)
3	(i)-(xv)
4	(i)-(xvi)

Let us now rearrange the above 16 vectors as in (2) :

1	:	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
v_1	:	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
v_2	:	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
$v_3 v_1$	:	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
v_3	:	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
$v_3 v_1$	:	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$v_3 v_2$	:	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$v_3 v_2 v_1$	:	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
v_4	:	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
$v_4 v_1$	:	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$v_4 v_2$	:	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
$v_4 v_2 v_1$	:	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
$v_4 v_3$	:	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
$v_4 v_3 v_1$	:	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
$v_4 v_3 v_2$	:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
$v_4 v_3 v_2 v_1$	:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

We obtain a 16×16 upper triangular binary matrix with 1's on the main diagonal of full rank, which indicates the linear independence of the 16 considered vectors.

One may construct the Reed-Muller codes of length $n = 2^m$ from the Reed-Muller codes of length 2^{m-1} as follows :

Consider all possible $m-1$ basis vectors for Reed-Muller codes of length 2^{m-1} . Then the possible basis vectors for Reed-Muller codes of length 2^m are :

$$\begin{aligned}
 &|1|1|, |v_1|v_1|, \dots, |v_{m-1}|v_{m-1}|, \\
 &|v_1 v_2|v_1 v_2|, \dots, |v_{m-1} v_{m-2}|v_{m-1} v_{m-2}|, \\
 &\dots, |v_{m-1} v_{m-2} \dots v_1|v_{m-1} v_{m-2} \dots v_1|,
 \end{aligned}$$

together with the vector $v_m = |0|1|$ and the products of v_m with the above vectors (where $|v_i|v_j|$ denotes concatenation of v_j after v_i). In fact, these products are of

the form :

$$v_m, v_m v_1 = |0|v_1|, v_m v_2 = |0|v_2|, \dots, v_m v_{m-1} \dots v_1 = |0|v_{m-1} \dots v_1|.$$

In this way, we obtain a total of $2^{m-1} + 2^{m-1} = 2^m$ distinct vectors which are also linearly independent and, therefore, these are all possible 2^m basis vectors for Reed-Muller codes of length 2^m .

Theorem 2 ([3]) : $R(r+1, m+1) = \{ |u|u+v| : u \in R(r+1, m), v \in R(r, m) \}$

Proof : By definition, a typical codeword f in $R(r+1, m+1)$ comes from a polynomial $f(v_1, \dots, v_{m+1})$ of degree at most $r+1$.

Consider the terms of $f(v_1, \dots, v_{m+1})$ which do not contain v_{m+1} as a factor and denote them as $g(v_1, \dots, v_m)$. If we now extract v_{m+1} as a common factor from $f(v_1, \dots, v_{m+1}) - g(v_1, \dots, v_m)$, we finally obtain

$$f(v_1, \dots, v_{m+1}) = g(v_1, \dots, v_m) + v_{m+1}h(v_1, v_2, \dots, v_m), \quad (4)$$

where $\deg(g(v_1, v_2, \dots, v_m)) \leq r+1$ and $\deg(h(v_1, v_2, \dots, v_m)) \leq r$.

If g and h denote the 2^m -length vectors corresponding to $g(v_1, \dots, v_m)$ and $h(v_1, \dots, v_m)$ respectively, then (by the definition of RM-codes) $g \in R(r+1, m)$ and $h \in R(r, m)$.

Consider now v_i , ($i = 1, 2, \dots, m$) as vectors of length 2^{m+1} ($v_i = |v_i|v_i|$), $v_{m+1} \neq |0|1|$ (where 0 and 1 have length 2^m) and $g(v_1, \dots, v_{m+1}) = g(v_1, \dots, v_m)$, $h(v_1, \dots, v_{m+1}) = v_{m+1}h(v_1, \dots, v_m)$.

Then, the corresponding vector to $g(v_1, \dots, v_{m+1})$ is $|g|g|$, while the corresponding vector to $v_{m+1}h(v_1, \dots, v_{m+1})$ is $|0|h|$, since the first 2^m components of v_{m+1} are 0 and the following 2^m components are 1.

From the above discussion, we obtain that

$$f = |g|g| + |0|h|. \quad (5)$$

A basis vector of an r^{th} order Reed-Muller code $R(r, m)$ is the corresponding vector of a vector product of at most r different vectors chosen from

$$1, v_1, v_2, \dots, v_m$$

and, therefore, it has degree at most r .

Similarly, any basis vector of an $(m - r - 1)$ -order Reed-Muller code has degree at most $m - r - 1$. Therefore, the vector product of any basis vector of $R(r, m)$ with any basis vector of $R(m - r - 1, m)$ has degree at most $r + (m - r - 1) = m - 1$ and, hence, it is a basis vector of an $(m - 1)$ -order Reed-Muller code $R(m - 1, m)$.

As it will be shown in the following section, if we consider a vector space of dimension m over $GF(2)$ containing 2^m points P_i whose coordinates are all the binary vectors of length m , then the vectors v_i ($i = 1, 2, \dots, m$) can be thought of as incidence vectors of $(m - 1)$ -dimensional subspaces of the original vector space.

Since a vector product $v_i v_j$ has its k^{th} coordinate equal to 1 if and only if the k^{th} coordinate in both v_i and v_j is equal to 1 (i.e. if P_k belongs to both $(m - 1)$ -dimensional subspaces defined by v_i and v_j), we conclude that $v_i v_j$ is the incidence vector of the $(m - 2)$ -dimensional subspace which is the intersection of the two $(m - 1)$ -dimensional subspaces defined by v_i and v_j .

In general, a vector product consisting of l (distinct) factors chosen from

$$1, v_1, v_2, \dots, v_m$$

is the incidence vector of an $(m - l)$ -dimensional subspace of the original vector space which (subspace) is the intersection of all the $(m - 1)$ -dimensional subspaces having as incidence vectors the factors of the vector product.

The above observations together with the definition of the v_i 's imply that all these vector products have even weight, except in the case of $v_m v_{m-1} \dots v_2 v_1$ which is the incidence vector of a 0-dimensional subspace, i.e. of a single point, and therefore it has an odd weight.

As a consequence, the inner product of any two such vectors, which is equal to the sum (modulo 2) of the entries of their vector product, is zero. This, in turn, implies that any vector in the r^{th} order Reed-Muller code $R(r, m)$ is orthogonal to any basis vector for the $(m - r - 1)$ -order Reed-Muller code $R(m - r - 1, m)$ (i.e. to any vector which is the product of at most $m - r - 1$ of the vectors $v_1, \dots, v_m$).

The dimension of $R(r, m)$ is

$$k_1 = 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r} \quad (6)$$

and the dimension of $R(m - r - 1, m)$ is

$$k_2 = 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{m - r - 1} \quad (7)$$

and note that in general,

$$\binom{n}{k} = \binom{n}{n - k} \quad (8)$$

From (6), (7) and (8) we obtain :

$$\begin{aligned} k_1 + k_2 &= 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r} + \binom{m}{m - r - 1} + \dots + 1 \\ &= 1 + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r} + \binom{m}{r + 1} + \dots + \binom{m}{m} \\ &= \sum_{i=0}^m \binom{m}{i} = 2^m = n. \end{aligned} \quad (9)$$

Relation (9) implies that the sum of the dimensions k_1 and k_2 of $R(r, m)$ and $R(m - r - 1, m)$ respectively, is n .

From the above discussion, it follows that each of the codes $R(r, m)$ and $R(m - r - 1, m)$ is the dual of the other. Hence,

Theorem 3 : The dual to the Reed-Muller code $R(r, m)$ is also a Reed-Muller code and, in particular, $R(r, m)^\perp = R(m - r - 1, m)$, for $0 \leq r \leq m - 1$.

Theorem 4 : The minimum distance of the r th order Reed-Muller code $R(r, m)$ is 2^{m-r} .

Proof : It is a simple induction on m :

For $m = 0$ the result is trivially true (length=1, codewords : 0, 1; hence, $d = 1 = 2^0$). Now assume that for $m = k$, $R(r, k)$ has minimum distance

$$d = 2^{k-r} \quad (0 \leq r \leq k)$$

(and, therefore, there exists in $R(r, k)$ a codeword u such that $wt(u) = 2^{k-r}$).

For $m = k + 1$, consider $R(r, k + 1)$. Using the results of Theorem 2,

$$R(r, k + 1) = \{|u|u + v| : u \in R(r, k), v \in R(r - 1, k)\}.$$

Now let a and b be distinct codewords of $R(r, k + 1)$, with

$$a = |u_1|u_1 + v_1|, b = |u_2|u_2 + v_2| \text{ and } u_i \in R(r, k), v_i \in R(r - 1, k), i = 1, 2.$$

If $v_1 = v_2$, then $\text{dist}(a, b) = 2\text{dist}(u_1, u_2) \geq 2d_1$, where d_1 denotes the minimum distance of $R(r, k)$ and by the induction hypothesis

$$d_1 = 2^{k-r}.$$

Hence, if $v_1 = v_2$

$$d = \text{dist}(a, b) \geq 2^{k+1-r}. \quad (10)$$

Suppose now that $v_1 \neq v_2$. Then,

$$\begin{aligned} d = \text{dist}(a, b) &= \text{wt}(u_1 - u_2) + \text{wt}(u_1 - u_2 + v_1 - v_2) \\ &\geq \text{wt}(u_1 - u_2) + \text{wt}(v_1 - v_2) - \text{wt}(u_1 - u_2) \\ &= \text{wt}(v_1 - v_2) \geq d_2, \end{aligned}$$

where d_2 denotes the minimum distance of $R(r - 1, k)$ and by the induction hypothesis, $d_2 = 2^{k-r+1}$; i.e. for $v_1 \neq v_2$,

$$d = \text{dist}(a, b) \geq 2^{k+1-r}. \quad (11)$$

By the induction hypothesis, the existence of a codeword $u \in R(r, k)$ such that $\text{wt}(u) = 2^{k-r}$ is also implied.

Since $0 \in R(r - 1, k)$, we have that the codeword

$$u^* = |u|u + 0|$$

is a codeword of $R(r, k + 1)$ and furthermore,

$$\text{wt}(u^*) = 2\text{wt}(u) = 2 \cdot 2^{k-r} = 2^{(k+1)-r}. \quad (12)$$

From (10), (11) and (12), we finally have that :

$$d = 2^{(k+1)-r},$$

and the assertion is true for $m = k + 1$. Our induction is complete.

From the above discussion we conclude that :

For any integers m and r with $m > 0$ and $0 \leq r \leq m$, there exists a binary r^{th} order Reed-Muller code $R(r, m)$ (which is an extended cyclic code) having the following properties :

$$\text{length } n = 2^m$$

$$\text{dimension } k = 1 + \binom{m}{1} + \dots + \binom{m}{r}$$

$$\text{minimum distance } d = 2^{m-r}$$

$$R(r, m)^\perp = R(m - r - 1, m).$$

$R(r, m)$ consists of the vectors of values taken by all polynomials in the binary variables $v_1, \dots, v_m$ of degree at most r .

We close this section with the following

Definition : For $0 \leq r \leq m - 1$, the *punctured* RM code $R(r, m)^*$ is obtained by puncturing (deleting) the coordinate corresponding to $v_1 = v_2 = \dots = v_m = 0$ from all the codewords of $R(r, m)$.

$R(r, m)^*$ has length $2^m - 1$, minimum distance $2^{m-r} - 1$ and dimension $k = 1 + \binom{m}{1} + \dots + \binom{m}{r}$.

1.2 Geometric interpretation

Another way of thinking about $R(r, m)$ is in terms of finite geometries [3].

Consider a space of m dimension over $GF(2)$ (the field of two elements) containing 2^m points P , whose coordinates are all the binary vectors $(x_1, x_2, \dots, x_m)$ of length m . This space together with its affine subspaces is called the *Euclidean Geometry* $EG(m, 2)$ of dimension m over $GF(2)$ if we delete the origin we obtain the *Projective Geometry* $PG(m - 1, 2)$, and is also a subspace of the m -dimensional vector space over $GF(2)$. An l -dimensional subspace S of $EG(m, 2)$ is called an *l-flat*. An $(m - 1)$ -flat of $EG(m, 2)$ is also called a *hyperplane* of $EG(m, 2)$.

It should be also noted that in general $EG(m, p^t)$ and $EG(ms, p^t)$, p prime, are tightly and usefully related. An r -flat in $EG(m, p^t)$ satisfies a set of linear equations over $GF(p^t)$. This set of equations remains linear over $GF(p^t)$. Since these equations are satisfied in $EG(m, p^t)$ (of dimension m), they are also satisfied in $EG(ms, p^t)$ (of higher dimension ms). Hence, these equations determine a flat in $EG(ms, p^t)$ and more precisely, an rs -flat in $EG(ms, p^t)$. However, the converse is not necessarily true [7].

If we now form a matrix having as columns the coordinates of these 2^m points, then the rows of this matrix define the codewords v_i (see table 1) of $R(1, m)$. Each vector v_i has an 1 for all points that have their i th coordinate x_i equal to 0. I.e. each vector v_i is the incidence vector of an $(m-1)$ -dimensional x_i -hyperplane. The vector $v_i v_j$ has an 1 for every point in the intersection of the x_i and x_j -hyperplanes; i.e. it is the incidence matrix for an $(m-2)$ -dimensional flat through the origin, and so on.

Example : Consider the case $m = 4$. The Euclidean Geometry $EG(4, 2)$ consists of the following $2^4 = 16$ points $P_0, P_1, \dots, P_{15}$ shown below as column vectors:

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
v_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
v_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
v_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
v_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

The 16 points $P_0, P_1, \dots, P_{15}$ in the 4-dimensional vector space over $GF(2)$ defined by the columns of the above 4×16 binary matrix as well as the 3-dimensional hyperplanes the incidence vectors of which are the v_i 's as shown in figure (1). The x_1 -hyperplane has incidence vector v_1 and contains all points which have their first coordinate equal to 0; it is shown in figure (1) by the dotted lines. Similarly, v_2 is the incidence vector of the x_2 -hyperplane, which contains all points which have their second coordinate equal to 0 and is shown in figure (1) by a blue line, and so forth. The x_1, x_2, x_3, x_4 -hyperplanes are all 3-dimensional hyperplanes.

figure (1)

figure (2)

The products $v_i v_j$ are the incidence vectors of the 2-dimensional $x_{i,j}$ -flats which pass through the origin, are the intersections of the x_i and x_j -hyperplanes, and contain the points which have both their i th and j th coordinates equal to 0. These 2-dimensional flats are shown in figure (2).

The products $v_i v_j v_k$ are the incidence vectors of the $x_{i,j,k}$ -flats through the origin which are the intersections of the flats corresponding to $v_i v_j$ and $v_j v_k$ (i.e. of the $x_{i,j}$ and $x_{j,k}$ -flats) and they are 1-dimensional flats passing through the origin containing the points which have their i th, j th and k th coordinates equal to 0.

The intersection of all these 1-dimensional flats, gives the 0-dimensional flat passing through the origin, namely the origin itself which has all its coordinates equal to 0.

One may now generalize the above remarks and observe that the codewords of minimum weight in the r th order Reed-Muller code of length 2^m ($R(r, m)$) are the $(m-r)$ -dimensional flats in $EG(m, 2)$ as it is shown below.

Let S be a subset of $EG(m, 2)$ such that $|S| = 2^{m-r}$ and let H be any hyperplane $EG(m-1, 2)$ in $EG(m, 2)$ and H^* be the parallel to H hyperplane of $EG(m, 2)$ so that $EG(m, 2) = H \cup H^*$.

Suppose that h is the incidence vector of H in $EG(m, 2)$ which is a linear Boolean function of $v_1, v_2, \dots, v_m$ and therefore it is a codeword in $R(r, m)$ of weight 2^{m-1} . If g is the incidence vector of S , then (as we have already mentioned in similar cases above) gh is the incidence vector of $S \cap H$ and a codeword in $R(r+1, m)$, as well.

Since the minimum distance of a linear code is equal to the minimum weight of any nonzero codeword, and the minimum distance of $R(r+1, m)$ is equal to $2^{m-(r+1)} = 2^{m-r-1}$, we conclude that gh is either the 0 codeword, or it contains at least 2^{m-r-1} nonzero entries. This, in turn, implies that $|S \cap H| = 0$ or $|S \cap H| \geq 2^{m-r-1}$. But since $|S| = 2^{m-r} = |S \cap H| + |S \cap H^*|$, we finally have that $|S \cap H| = 0, 2^{m-r-1}$ or 2^{m-r} .

Hence, we have proved the following

Lemma 1 : If S is an $(m-r)$ -flat in $EG(m, 2)$ (and therefore $|S| = 2^{m-r}$) and

H is any hyperplane in $EG(m, 2)$, then

$$|S \cap H| = 0, 2^{m-r-1}, \text{ or } 2^{m-r}.$$

The converse is also true as it is shown in the following [3]

Lemma 2 (Rothchild and VanLint) : Let S be a subset of $EG(m, 2)$ such that $|S| = 2^{m-r}$, and $|S \cap H| = 0, 2^{m-r-1}$ or 2^{m-r} for all hyperplanes H in $EG(m, 2)$. Then S is an $(m-r)$ -dimensional flat in $EG(m, 2)$.

Proof : It is by induction on m .

For $m = 2$, let S be a subset of $EG(2, 2)$ such that $|S| = 2^{2-r}$, $0 \leq r \leq 2$. Note that $EG(2, 2)$ contains all 4 points of length 2 with coordinates in $GF(2)$, namely, $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 1)$.

For $r = 0$, $|S| = 2^2 = 4$. If $|S \cap H| = 0, 2$, or 4 for all hyperplanes H in $EG(2, 2)$, then S is an 2-dimensional flat.

For $r = 1$, $|S| = 2^1 = 2$. If $|S \cap H| = 0, 1$ or 2 for all hyperplanes H in $EG(2, 2)$, then S is an 1-dimensional flat.

For $r = 2$, $|S| = 2^0 = 1$. If $|S \cap H| = 0$ or 1 , for all hyperplanes H in $EG(2, 2)$, then S is a 0-dimensional flat. In all cases, S is an $(m-r)$ -dimensional flat (if it satisfies the hypothesis) and therefore, the result is true for $m = 2$.

Suppose now that the result is true for an integer k , with $2 \leq k < m$ and consider S to be a subset of $EG(m, 2)$ with the additional property that $|S| = 2^{m-r}$.

Suppose also that for some hyperplane H in $EG(m, 2)$, we have that

$$|S \cap H| = 2^{m-r}.$$

Then, $S \subseteq H$ (since $|S| = 2^{m-r}$) and therefore, S is contained in $EG(m-1, 2)$.

Consider now some hyperplane H^* in H . Then, there exists another hyperplane H'' of $EG(m, 2)$ such that $H^* = H \cap H''$ and $S \cap H^* = S \cap H''$; i.e. $|S \cap H^*| = 0, 2^{m-r-1}$ or 2^{m-r} . Equivalently, $|S \cap H^*| = 0, 2^{(m-1)-(r-1)-1}$ or $2^{(m-1)-(r-1)}$.

By the induction hypothesis, S is an $((m-1) - (r-1))$ -flat in $EG(m-1, 2)$; i.e. S is an $(m-r)$ -flat in $EG(m-1, 2)$, and therefore in $EG(m, 2)$.

If for some H we have $|S \cap H| = 0$, then we may replace H by its parallel hyperplane and apply entirely the same arguments.

We shall now prove that $|S \cap H| \neq 2^{m-r-1}$ for some hyperplane H in $EG(m, 2)$.

Indeed : Suppose on the contrary that $|S \cap H| = 2^{m-r-1}$ for any hyperplane H in $EG(m, 2)$. Then,

$$\begin{aligned} \sum_{H \in EG(m, 2)} |S \cap H|^2 &= \sum_{H \in EG(m, 2)} \left(\sum_{a \in S} \chi_H(a) \right)^2 \\ &= \sum_{H \in EG(m, 2)} \sum_{a \in S} \sum_{b \in S} \chi_H(a) \chi_H(b) \\ &= \sum_{a \in S} \sum_{b \in S} \sum_{H \in EG(m, 2)} \chi_H(a) \chi_H(b). \end{aligned} \quad (13)$$

However, note that there are $2^m - 1$ hyperplanes in $EG(m, 2)$ passing through a point and $2^{m-1} - 1$ through a line, and therefore (13) reduces to :

$$\sum_{H \in EG(m, 2)} |S \cap H|^2 = |S|(2^m - 1) + |S|(|S| - 1)(2^{m-1} - 1) \quad (14).$$

But $|S \cap H| = 2^{m-r-1}$ by assumption for any $H \in EG(m, 2)$. i.e. $|S \cap H|^2 = 2^{2m-2r-2}$ and therefore,

$$\sum_{H \in EG(m, 2)} |S \cap H|^2 = 2^m \cdot 2^{2m-2r-2} = 2^{3m-2r-2}.$$

Also, $|S| = 2^{m-r}$ and substituting in the R.H.S. of (14), we find

$$2^{2m-2r}(2^{m-1} + 2^{r-1} - 1) \neq 2^{3m-2r-2},$$

which is a contradiction.

From the above Lemma 1 and the discussion preceding it, we have the following

Theorem 5 ([3]) : Let g be a minimum weight codeword of $R(r, m)$, say $g = \chi(S)$. Then S is an $(m-r)$ -dimensional flat in $EG(m, 2)$. (This flat need not pass through the origin).

Theorem 6 : The incidence vector of any $(m-r)$ -flat in $EG(m, 2)$ is in $R(r, m)$.

Proof : Any $(m-r)$ -flat in $EG(m, 2)$ consists of all points v which satisfy r linear equations over $GF(2)$. Let these equations be the following :

$$\sum_{j=1}^m a_{ij} v_j = b_i, \quad i = 1, 2, \dots, r, \quad (15)$$

or,

$$\sum_{j=1}^m a_{ij} v_j + b_i + 1 = 1, \quad i = 1, 2, \dots, r \quad (16)$$

or, equivalently,

$$\prod_{i=1}^r \left(\sum_{j=1}^m a_{ij} v_j + b_i + 1 \right) = 1 \quad (17)$$

which is a polynomial of degree $\leq r$ in $v_1, \dots, v_m$. This implies that the flat is in $R(r, m)$.

From the above Theorems (5) and (6) we obtain

Theorem 7 : The codewords of minimum weight in $R(r, m)$ are exactly the incidence vectors of the $(m-r)$ -dimensional flats in $EG(m, 2)$.

Chapter 2

Decoding of Reed -Muller codes

2.1 Majority logic decoding of Reed -Muller codes .

An important feature of the Reed-Muller codes, as we have already mentioned, is that they can be decoded in a simple manner. The following example illustrates the decoding procedure.

Example : Let $m = 4$ and consider the 2^{nd} order Reed-Muller code $R(2, 4)$.
Then,

$$k = 1 + \binom{4}{1} + \binom{4}{2} = 11,$$

$$n = 2^4 = 16,$$

and therefore, $R(2, 4)$ is a $(16, 11)$ code with minimum distance

$$d = 2^{4-2} = 4.$$

Since d is even, the above code $R(2, 4)$ can simultaneously correct $\left\lfloor \frac{1}{2}(4 - 1) \right\rfloor = 1$ error and detect $\frac{4}{2} = 2$ errors. (See [3] for appropriate theory).

Consider now the 11 information symbols

$$a_0, a_4, a_3, a_2, a_1, a_{43}, a_{42}, a_{41}, a_{32}, a_{31}, a_{21}.$$

These are coded into the vector

$$\begin{aligned} x = & a_0 v_0 + a_4 v_4 + a_3 v_3 + a_2 v_2 + a_1 v_1 + a_{43} v_4 v_3 + a_{42} v_4 v_2 + a_{41} v_4 v_1 \\ & + a_{32} v_3 v_2 + a_{31} v_3 v_1 + a_{21} v_2 v_1 = (x_0, x_1, \dots, x_{15}), \end{aligned} \quad (18)$$

where the v 's are given in (Table 2).

Our objective is to recover these 11 information symbols from a received vector, even though errors may have occurred.

One may observe (see Table 2) that if the vector v_2, v_1 is omitted, then the first four components of any other basis vector add up to 0 (mod 2). This, in turn, implies that in the absence of errors, the sum of the first four components of the vector defined in (18) is equal to a_{21} (always in $GF(2)$); i.e.

$$x_0 + x_1 + x_2 + x_3 = a_{21} \quad (19)$$

But the same is true for all next three groups consisting of four subsequent components each; i.e.

$$x_4 + x_5 + x_6 + x_7 = a_{21} \quad (20)$$

$$x_8 + x_9 + x_{10} + x_{11} = a_{21} \quad (21)$$

$$x_{12} + x_{13} + x_{14} + x_{15} = a_{21} \quad (22)$$

Note that equations (19)-(22) give four independent determinations for a_{21} .

Of course, in the presence of errors not all of the above equations are valid. However, a single error can affect only one of them and therefore we can safely enough assign to a_{21} the majority vote and obtain it correctly.

In general, there are $\frac{2^m}{2^r} = 2^{m-r}$ such independent determinations for a_{21} which can be decoded correctly using the above procedure, as long as the errors do not cause any ambiguity and do not affect the majority of these determinations; i.e. a_{21} can be correctly decoded using the above method in the presence of at most $\frac{2^{m-r}}{2} - 1 = 2^{m-r-1} - 1$ errors.

The information symbols $a_{43}, a_{42}, a_{41}, a_{32}, a_{31}$ can be decoded in a quite similar manner. Once all of them have been determined, we subtract

$$a_{43}v_4v_3 + a_{42}v_4v_2 + a_{41}v_4v_1 + a_{32}v_3v_2 + a_{31}v_3v_1 + a_{21}v_2v_1 \quad (23)$$

from the received vector, which in the absence of errors is $(x_0, x_1, \dots, x_{15})$, to obtain the vector

$$x' = a_0v_0 + a_4v_4 + a_3v_3 + a_2v_2 + a_1v_1 = (x'_0, x'_1, \dots, x'_{15}). \quad (24)$$

If we now subdivide the vectors v_0, v_1, v_2, v_3, v_4 into eight subsequent pairs of (subsequent) components, then the sum of the components of each pair is 0 (mod 2).

This observation leads to the following independent determinations for a_1

$$\begin{array}{ll}
 x'_0 + x'_1 = a_1 & x'_8 + x'_9 = a_1 \\
 x'_2 + x'_3 = a_1 & x'_{10} + x'_{11} = a_1 \\
 x'_4 + x'_5 = a_1 & x'_{12} + x'_{13} = a_1 \\
 x'_6 + x'_7 = a_1 & x'_{14} + x'_{15} = a_1
 \end{array} \quad (25)$$

Similarly, we find that

$$\begin{array}{l}
 x'_0 + x'_2 = a_2 \\
 x'_1 + x'_3 = a_2
 \end{array}$$

and so on.

Note that determining the values of $a_1, \dots, a_4$ is an easier task since we now have eight votes for each of them and the majority vote in the presence of a single error, certainly gives a_i correctly.

Once all of a_i 's $i = 1, \dots, 4$ have been determined, we subtract the vector

$$a_4 v_4 + a_3 v_3 + a_2 v_2 + a_1 v_1 \quad (26)$$

from x' to obtain (again in the absence of errors)

$$x' - (a_4 v_4 + a_3 v_3 + a_2 v_2 + a_1 v_1) = a_0 v_0 \quad (27)$$

Since $v_0 = (1, 1, 1, \dots, 1)$, the vector $a_0 v_0$ is the 0 vector if $a_0 = 0$, or v_0 if $a_0 = 1$. So, choose a_0 to be 0 or 1 whichever occurs more frequently in $x' - (a_4 v_4 + a_3 v_3 + a_2 v_2 + a_1 v_1)$.

The above scheme first introduced by Irving S. Reed in 1954 [9] is called the *Reed decoding algorithm* and is the first recorded use of *majority logic decoding*.

Now let $r_{21}^{(1)}, r_{21}^{(2)}, r_{21}^{(3)}$ and $r_{21}^{(4)}$ denote the values of the four relations (19)-(22) for a_{21} and, in general, let $r_{ij}^{(1)}, r_{ij}^{(2)}, r_{ij}^{(3)}, r_{ij}^{(4)}$ denote the values of the corresponding relations for determining a_i , ($i = 2, 3, 4$; $j = 1, 2, 3, j < i$). Let also S_{ij} denote

the (arithmetic¹) sum of $r_{i,j}^{(1)}, r_{i,j}^{(2)}, r_{i,j}^{(3)}$ and $r_{i,j}^{(4)}$; i.e.

$$S_{i,j} = \sum_{k=1}^4 r_{i,j}^{(k)} \quad i = 2, 3, 4; j = 1, 2, 3; j < i. \quad (28)$$

Then, the majority decision test for $a_{i,j}$ is :

$$a_{i,j} = \begin{cases} 0 & \text{if } 0 \leq S_{i,j} < 2; \\ 1 & \text{if } 2 < S_{i,j} \leq 4; \\ \text{indeterminate} & \text{if } S_{i,j} = 2. \end{cases} \quad i = 2, 3, 4, j = 1, 2, 3, j < i \quad (29)$$

Similarly, if $r_i^{(k)}$ ($k = 1, 2, \dots, 8$) denote the values of the relations for a_i and S_i the corresponding sum of these values, then the majority decision test for a_i is :

$$a_i = \begin{cases} 0 & \text{if } 0 \leq S_i < 4; \\ 1 & \text{if } 4 < S_i \leq 8 \\ \text{indeterminate} & \text{if } S_i = 4. \end{cases} \quad i = 1, \dots, 4 \quad (30)$$

Finally, if $x'' = (x''_0, \dots, x''_{15}) = a_0 v_0$, then the majority decision test for a_0 is :

$$a_0 = \begin{cases} 0 & \text{if } \sum_{i=0}^7 x''_i < 8; \\ 1 & \text{if } \sum_{i=0}^7 x''_i > 8. \end{cases} \quad (31)$$

From the above discussion, arises the following question : How can one describe the scheme for determining which sums of symbols in the received vector should equal a given information symbol?

In order to answer this question, we first arrange the symbols in each of the original vectors $v_1, \dots, v_m$ as in Table(2) (for the case $m = 4$). I.e. for $m = 4$,

$$\begin{aligned} v_4 &= (0000000011111111) \\ v_3 &= (0000111100001111) \\ v_2 &= (0011001100110011) \\ v_1 &= (0101010101010101) \end{aligned} \quad (32)$$

Next, call the component corresponding to the j th 0 in v_i and the component corresponding to the j th 1 in v_i *matching components* for v_i . The 2^{m-1} sums of matching components are used to determine a_i . As a next step, we take a matching pair of

¹ $r_{i,j}^{(k)} = 0$ or 1, and we change the characteristic from 2 to 0.

components for v_i and with each of them the component that matches for v_j , to form a four-component sum. These 2^{m-2} sums (of four components each) are used to determine a_{ij} . Similarly, at the next step we take the sum of a pair of matching components v_i , together with their matching components for v_j and the matching components for v_k for each of these, to obtain a total of eight components. These 2^{m-3} sums are used to determine a_{ijk} , and so on.

The determination of the a_i 's ($i = 1, 2, 3, 4$) or, equivalently, the first-order redundancy relations applied to (32) can be represented in a tabular manner as follows : For a_1 :

$$v_1 = \begin{matrix} f_1^{(1)} & f_1^{(2)} & f_1^{(3)} & f_1^{(4)} & f_1^{(5)} & f_1^{(6)} & f_1^{(7)} & f_1^{(8)} \\ (0 & 1 & 0 & 1 & 0 & 1 & 0 & 1) \end{matrix} \quad (a)$$

For a_2 :

$$v_2 = \begin{matrix} f_2^{(1)} & f_2^{(2)} & f_2^{(3)} & f_2^{(4)} & f_2^{(5)} & f_2^{(6)} & f_2^{(7)} & f_2^{(8)} \\ (0 & 0 & 1 & 1 & 0 & 0 & 1 & 1) \end{matrix} \quad (b)$$

For a_3 :

$$v_3 = \begin{matrix} f_3^{(1)} & f_3^{(2)} & f_3^{(3)} & f_3^{(4)} & f_3^{(5)} & \dots & f_3^{(8)} \\ (0 & 0 & 0 & 0 & 1 & 1 & 1 & 1) \end{matrix} \quad (c)$$

For a_4 :

$$v_4 = \begin{matrix} f_4^{(1)} & \dots & f_4^{(8)} \\ (0 & 0 & 0 & 0 & 0 & 0 & 0 & 1) \end{matrix} \quad (d) \quad (33)$$

For instance, the eight independent and disjoint relations for a_1 are

$$a_1 = x_{2k} + x_{2k+1}, \quad k = 0, 1, 2, \dots, 7$$

and similar relations hold for the other a_i 's ($i = 2, 3, 4$).

The determination of the a_{ij} 's ($i = 2, 3, 4$; $j = 1, 2, 3$; $j < i$) or, equivalently, the second order redundancy relations can be represented in a tabular manner as follows : For a_{21} :

$$v_2 v_1 = \begin{matrix} f_{21}^{(1)} & f_{21}^{(2)} & f_{21}^{(3)} & f_{21}^{(4)} \\ (0 & 0 & 0 & 1) & (0 & 0 & 0 & 1) & (0 & 0 & 0 & 1) & (0 & 0 & 0 & 1) \end{matrix} \quad (a)$$

For a_{31} :

$$v_3 v_1 = \begin{matrix} f_{31}^{(1)} & f_{31}^{(2)} & f_{31}^{(3)} & f_{31}^{(4)} \\ (0 & 0 & 0 & 0 & 0 & 1 & 0 & 1) & (0 & 0 & 0 & 0 & 0 & 1 & 0 & 1) \end{matrix} \quad (b)$$

For a_{41} :

$$v_4 v_1 = (0000000001010101) \quad (c)$$

For a_{32} :

$$v_3 v_2 = (000000011000000011) \quad (d)$$

For a_{42} :

$$v_4 v_2 = (000000000000110011) \quad (e)$$

For a_{43} :

$$v_4 v_3 = (000000000000011111) \quad (f) \quad (34)$$

For instance, the four relations for the determination of a_{43} as given by the above scheme, are :

$$\begin{aligned}
 x_0 + x_4 + x_8 + x_{12} &= a_{43} \\
 x_1 + x_5 + x_9 + x_{13} &= a_{43} \\
 x_2 + x_6 + x_{10} + x_{14} &= a_{43} \\
 x_3 + x_7 + x_{11} + x_{15} &= a_{43}
 \end{aligned} \quad (35)$$

and similarly for the other second-order redundancy relations.

In general these (four) second-order redundancy relations will allow only one error; two errors will lead to indeterminacy.

For a better understanding of the coding and decoding scheme we give another

Example : Consider again the 2^{nd} order Reed-Muller code for $m = 4$ (i.e. $R(2,4)$) and suppose that the message

$$(a_0, a_1, a_2, a_3, a_4, a_{43}, a_{42}, a_{41}, a_{32}, a_{31}, a_{21}) = (10011100100) \quad (36)$$

was sent.

Then, the corresponding code is given by :

$$\begin{aligned}
 x &= 1 \cdot v_0 + 0 \cdot v_1 + 0 \cdot v_2 + 1 \cdot v_3 + 1 \cdot v_4 + 1 \cdot v_4 v_3 + 0 \cdot v_4 v_2 + 0 \cdot v_4 v_1 \\
 &\quad + 1 \cdot v_3 v_2 + 0 \cdot v_3 v_1 + 0 \cdot v_2 v_1 = (1111001100000011) \quad (37)
 \end{aligned}$$

and suppose that $\bar{x} = (1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1)$ was received instead.

Our objective is to recover the 11 information symbols given in (36) from the received vector $\bar{x}$ given above.

We first determine the a_{ij} 's ($i = 2, 3, 4$; $j = 1, 2, 3$; $j < i$). From the relations (34) and using our previous notation, we have that :

For a_{21} :

$$\begin{aligned} r_{21}^{(1)} &= x_0 + x_1 + x_2 + x_3 = 1 \\ r_{21}^{(2)} &= x_4 + x_5 + x_6 + x_7 = 0 \\ r_{21}^{(3)} &= x_8 + x_9 + x_{10} + x_{11} = 0 \longrightarrow S_{21} = 1 < 2 \longrightarrow a_{21} = 0 \quad (a) \\ r_{21}^{(4)} &= x_{12} + x_{13} + x_{14} + x_{15} = 0 \end{aligned}$$

For a_{31} :

$$\begin{aligned} r_{31}^{(1)} &= x_0 + x_1 + x_4 + x_8 = 1 \\ r_{31}^{(2)} &= x_2 + x_3 + x_6 + x_7 = 0 \\ r_{31}^{(3)} &= x_8 + x_9 + x_{12} + x_{13} = 0 \longrightarrow S_{31} = 1 < 2 \longrightarrow a_{31} = 0 \quad (b) \\ r_{31}^{(4)} &= x_{10} + x_{11} + x_{14} + x_{15} = 0 \end{aligned}$$

For a_{41} :

$$\begin{aligned} r_{41}^{(1)} &= x_0 + x_1 + x_8 + x_9 = 1 \\ r_{41}^{(2)} &= x_2 + x_3 + x_{10} + x_{11} = 0 \\ r_{41}^{(3)} &= x_4 + x_5 + x_{12} + x_{13} = 0 \longrightarrow S_{41} = 1 < 2 \longrightarrow a_{41} = 0 \quad (c) \\ r_{41}^{(4)} &= x_6 + x_7 + x_{14} + x_{15} = 0 \end{aligned}$$

For a_{32} :

$$\begin{aligned} r_{32}^{(1)} &= x_0 + x_2 + x_4 + x_6 = 1 \\ r_{32}^{(2)} &= x_1 + x_3 + x_5 + x_7 = 0 \\ r_{32}^{(3)} &= x_8 + x_{10} + x_{12} + x_{14} = 1 \longrightarrow S_{32} = 3 > 2 \longrightarrow a_{32} = 1 \quad (d) \quad (38) \\ r_{32}^{(4)} &= x_9 + x_{11} + x_{13} + x_{15} = 1 \end{aligned}$$

For a_{42} :

$$r_{42}^{(1)} = x_0' + x_2' + x_8' + x_{10}' = 0$$

$$r_{42}^{(2)} = x_1' + x_3' + x_9' + x_{11}' = 1$$

$$r_{42}^{(3)} = x_4' + x_6' + x_{12}' + x_{14}' = 0 \rightarrow S_{42} = 1 < 2 \rightarrow a_{42} = 0 \quad (e)$$

$$r_{42}^{(4)} = x_5' + x_7' + x_{13}' + x_{15}' = 0$$

For a_{43} :

$$r_{43}^{(1)} = x_0' + x_4' + x_8' + x_{12}' = 1$$

$$r_{43}^{(2)} = x_1' + x_5' + x_9' + x_{13}' = 0$$

$$r_{43}^{(3)} = x_2' + x_6' + x_{10}' + x_{14}' = 1 \rightarrow S_{43} = 3 > 2 \rightarrow a_{43} = 1 \quad (f)$$

$$r_{43}^{(4)} = x_3' + x_7' + x_{11}' + x_{15}' = 1$$

I.e. $a_{31} = a_{31} = a_{41} = a_{42} = 0$ and $a_{43} = a_{32} = 1$.

We now compute

$$\begin{aligned} & a_{43} \cdot v_4 v_3 + a_{42} \cdot v_4 v_2 + a_{41} \cdot v_4 v_1 + a_{32} \cdot v_3 v_2 + a_{31} \cdot v_3 v_1 + a_{21} \cdot v_2 v_1 = \\ & = 1 \cdot v_4 v_3 + 0 \cdot v_4 v_2 + 0 \cdot v_4 v_1 + 1 \cdot v_3 v_2 + 0 \cdot v_3 v_1 + 0 \cdot v_2 v_1 = \\ & = (0000001100001100) \end{aligned}$$

and we subtract it from the received vector (1011001100000011) to obtain $x' = (1011000000001111)$.

Next, we apply relations (33) to x' to determine a_1, a_2, a_3, a_4 .

For a_1 :

$$r_1^{(1)} = x_0' + x_1' = 1 \quad r_1^{(5)} = x_8' + x_9' = 0$$

$$r_1^{(2)} = x_2' + x_3' = 0 \quad r_1^{(6)} = x_{10}' + x_{11}' = 0$$

$$r_1^{(3)} = x_4' + x_5' = 0 \quad r_1^{(7)} = x_{14}' + x_{15}' = 0 \rightarrow S_1 = 1 < 4 \rightarrow a_1 = 0 \quad (a)$$

$$r_1^{(4)} = x_6' + x_7' = 0 \quad r_1^{(8)} = x_{12}' + x_{13}' = 0$$

For a_2 :

$$r_2^{(1)} = x_0' + x_2' = 0 \quad r_2^{(5)} = x_8' + x_{10}' = 0$$

$$r_2^{(2)} = x_1' + x_3' = 1 \quad r_2^{(6)} = x_9' + x_{11}' = 0$$

$$r_2^{(3)} = x_4' + x_6' = 0 \quad r_2^{(7)} = x_{12}' + x_{14}' = 0 \rightarrow S_2 = 1 < 4 \rightarrow a_2 = 0 \quad (b)$$

$$r_2^{(4)} = x_5' + x_7' = 0 \quad r_2^{(8)} = x_{13}' + x_{15}' = 0$$

For a_3 :

$$\begin{aligned}
 r_3^{(1)} &= x'_0 + x'_4 = 1 & r_3^{(5)} &= x'_8 + x'_{12} = 1 \\
 r_3^{(2)} &= x'_1 + x'_5 = 0 & r_3^{(6)} &= x'_9 + x'_{13} = 1 \\
 r_3^{(3)} &= x'_2 + x'_6 = 1 & r_3^{(7)} &= x'_{10} + x'_{14} = 1 \\
 r_3^{(4)} &= x'_3 + x'_7 = 1 & r_3^{(8)} &= x'_{11} + x'_{15} = 1
 \end{aligned}
 \rightarrow S_3 = 7 > 4 \rightarrow a_3 = 1 \quad (c) \quad (39)$$

For a_4 :

$$\begin{aligned}
 r_4^{(1)} &= x'_0 + x'_8 = 1 & r_4^{(5)} &= x'_4 + x'_{12} = 1 \\
 r_4^{(2)} &= x'_1 + x'_9 = 0 & r_4^{(6)} &= x'_5 + x'_{13} = 1 \\
 r_4^{(3)} &= x'_2 + x'_{10} = 1 & r_4^{(7)} &= x'_6 + x'_{14} = 1 \\
 r_4^{(4)} &= x'_3 + x'_{11} = 1 & r_4^{(8)} &= x'_7 + x'_{15} = 1
 \end{aligned}
 \rightarrow S_4 = 7 > 4 \rightarrow a_4 = 1 \quad (d)$$

I.e. $a_1 = a_2 = 0$ and $a_3 = a_4 = 1$.

Now we compute

$$a_1 v_1 + a_2 v_2 + a_3 v_3 + a_4 v_4 = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0).$$

and subtract it from x' to obtain

$$x'' = (1\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1).$$

Note that $\sum_{k=0}^{15} x''_k = 15 > 8$. Therefore, $a_0 = 1$ and the message which was sent is (according to the above scheme)

$$(1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0)$$

in agreement with (36).

The decoding procedure described above and illustrated in the preceding examples, has the advantage that it may be generalized to include any $R(r, m)$ codes while majority-logic decoding is a method of error-correction which is especially suited for machine implementation.

3.2 Geometric interpretation of the Decoding scheme

The decoding method described earlier, has an interesting geometric interpretation.

Consider again our reference example of the 2nd order Reed-Muller code with $m = 4$ $R(2,4)$ and the Euclidean geometry $EG(4,2)$ of dimension 4 over $GF(2)$ containing 2^4 points with coordinates all the binary vectors of length 4.

It has already been mentioned that as generator matrix G for $R(2,4)$, we take the 11 first rows of Table (2) (see also Table (2a)), and the message symbols

$$a = a_0 a_4 a_3 a_2 a_1 a_{43} a_{42} a_{41} a_{32} a_{31} a_{21} \quad (40)$$

are encoded into the codeword

$$\begin{aligned} x = aG &= a_0 v_0 + a_4 v_4 + \dots + a_1 v_1 + \dots + a_{31} v_3 v_1 + a_{21} v_2 v_1 \\ &= (x_0, \dots, x_{15}). \end{aligned} \quad (41)$$

i.e. each information symbol appears as a coefficient of a vector or vector product in (41).

Let us now determine which sums of symbols in the received vector should equal a given information symbol, say a_{32} .

We first note that a_{32} appears in (41) as coefficient of the vector product $v_3 v_2$ which is the incidence vector of a 2-dimensional flat containing four points, namely P_6, P_7, P_{14} and P_{15} (refer to Table (2) and discussion in section 1.2). Now through each of these points there is a perpendicular 2-dimensional flat containing four points. These are precisely the sets of points that correspond to the sums used to determine the information symbol a_{32} during decoding. The 2-dimensional flat whose incidence vector is $v_3 v_2$ as well as the four perpendicular 2-dimensional flats through the points P_6, P_7, P_{14} and P_{15} are shown in figure (3) with striped and shaded areas respectively.

Note that from Table (2),

$$v_3 v_2 = (0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1)$$

which is the incidence vector of the 2-dimensional plane containing the four points P_6, P_7, P_{14} and P_{15} (since the 1's are at the 6th, 7th, 14th and 15th position respectively), while the four perpendicular planes passing through P_6, P_7, P_{14} and P_{15} are

figure (3)

$P_0P_2P_6P_4$, $P_1P_3P_7P_5$, $P_8P_{10}P_{14}P_{12}$ and $P_9P_{11}P_{15}P_{13}$, respectively. If we now add the four components of the received vector x^* corresponding to the four points of each flat separately, we obtain the following four independent and disjoint determinations for a_{32} :

$$\begin{aligned} P_0P_2P_6P_4 : \quad x_0^* + x_2^* + x_6^* + x_4^* &= r_{32}^{(1)} \\ P_1P_3P_7P_5 : \quad x_1^* + x_3^* + x_7^* + x_5^* &= r_{32}^{(2)} \\ P_8P_{10}P_{14}P_{12} : \quad x_8^* + x_{10}^* + x_{14}^* + x_{12}^* &= r_{32}^{(3)} \\ P_9P_{11}P_{15}P_{13} : \quad x_9^* + x_{11}^* + x_{15}^* + x_{13}^* &= r_{32}^{(4)} \end{aligned}$$

which are precisely the determinations found earlier in (38d).

Next, consider the incidence vectors of these perpendicular flats. These are:

$$\begin{aligned} (1010101000000000) &\text{ for } P_0P_2P_6P_4 \\ (0101010100000000) &\text{ for } P_1P_3P_7P_5 \\ (0000000010101010) &\text{ for } P_8P_{10}P_{14}P_{12} \\ (0000000001010101) &\text{ for } P_9P_{11}P_{15}P_{13} \end{aligned}$$

One may observe that each one of the above four vectors has an odd number of 1's in common with v_3v_2 , but an even number in common with all other vectors that are products of at most two of the vectors v_i . This happens because later in the sum the desired coefficient will not drop out but all others will cancel.

The generalization for an arbitrary $R(r, m)$ code is now immediate. We first find a_σ , where $\sigma = \sigma_1 \dots \sigma_r$ ($\sigma_i \in \{1, \dots, m\}$), which appears in

$$a_0v_0 + a_mv_m + a_{m-1}v_{m-1} + \dots + a_1v_1 + a_{m,m-1}v_mv_{m-1} + \dots + a_{21}v_2v_1 + \dots \quad (42)$$

as a coefficient of a vector product of r of the vectors $v_1, \dots, v_m$; i.e. a_σ is the coefficient in (42) of $v_{\sigma_1}v_{\sigma_2} \dots v_{\sigma_r}$.

Each basis vector in the code is a product of r or fewer of the vectors $v_1, \dots, v_m$ (as we have already mentioned), and since an r -dimensional flat contains 2^r points, a product of r vectors v_i has 2^{m-r} 1's and is the incidence vector of an $(m-r)$ -dimensional subspace of $EG(m, 2)$. Through each of these points there is a perpendicular r -dimensional flat containing 2^r points. These are precisely the sets of

points that correspond to the sums used to determine the information symbols during decoding, provided that the incidence vector of each such set of points has an odd number of 1's in common with the vector $v_0, v_1, \dots, v_{n-1}$, but an even number in common with all other vectors which are products of r or fewer of the vectors v_i , so that later in the sum the desired coefficient a_0 will not drop out, while all the others will cancel (as we mentioned above).

These sets of points have this property because, being perpendicular to the given flat, each set intersects it in one point but each intersects every other flat of dimension r or more either not at all, or in a line, or in a higher-dimensional flat, in which case it has in common with it an even number of points 2^l , where l is the dimension of the intersection.

The above results are summarized in the following (proven also in [3])

Theorem 8 : If there are no errors, a_0 is given by

$$a_0 = \sum_{\sigma \in U_i} x_\sigma, \quad i = 1, \dots, 2^{m-r}, \quad (43)$$

where U_i are the perpendicular flats mentioned above, and $\sigma = \sigma_1 \sigma_2 \dots \sigma_r$. The above equations (43) give 2^{m-r} independent votes for a_0 .

It should be noted that other codes of a similar variety exist. Euclidean Geometry (EG) codes are a natural generalization of the (cyclic) Reed-Muller codes and, in fact, three such generalizations are involved. Namely, one may first consider codes with symbols from $GF(p)$, p prime, having length $p^{m'}$. One can also associate the $p^{m'}$ code points with different geometries; i.e. if $m' = sm$, $s \neq 1$, $m \neq 1$, then $EG(m', p)$, $EG(m, p^s)$ and $EG(s, p^m)$ can be used to construct majority-logic decodable, extended cyclic codes of length $p^{m'}$. Finally, one may construct codes with symbols from $GF(q)$, q prime power.

It will be shown in later sections that it is possible to correct about $\lfloor (d-1)/2 \rfloor$ random errors with EG codes using a majority logic decoder that employs the Reed algorithm (where d is the minimum distance of the code).

This is also true for Projective Geometry (PG) codes; another class of finite geometry codes which can be thought of as another generalization of the Reed-Muller codes.

2.3 Majority logic decoding (Part II).

Although majority-logic decoding is attractive because of its ease of implementation, the complexity of this type of decoder increases very rapidly (exponentially in fact) with the number of levels of majority logic which must be employed (or, with the number of steps). It is therefore important to decode the majority-logic decodable codes (hence the Reed-Muller codes and, in general, the Euclidean Geometry codes) in as few steps as possible. It is shown that the majority logic decoding process can be performed in $r + 1$ steps instead of $n - r$.

Before the majority-logic results, a few definitions and preliminary results are stated and the central (to majority-logic) concept of orthogonal parity check sums is introduced.

2.3.1 Preliminaries.

Let C be any $[n, k]$ code over $GF(q)$ (q prime power). The set of codewords of such a code is a subset of a set of n -tuples of the form

$$\mathbf{x} = (x_0, \dots, x_{n-1}), \quad x_i \in GF(q) \quad i = 0, \dots, n-1.$$

If $x_0, \dots, x_{k-1}$ are chosen to be the information symbols, then the remaining $n - k$ symbols are called the *parity symbols* and are determined from the information symbols by a set of linear equations

$$H\mathbf{x}^T = 0, \quad (44)$$

which every codeword $\mathbf{x}$ must satisfy and where H is the $(n - k) \times n$ parity check matrix of the code. In other words, (44) can be written in the form

$$x_i = \sum_{j=0}^{k-1} h_{ij} x_j, \quad i = k, k+1, \dots, n-1 \quad (45)$$

and we say that each of these $n - k$ equations defines a *parity set* for the code; it expresses a weighted sum of code symbols which is zero for all codewords. Let $\mathbf{x}^* = (x_0^*, x_1^*, \dots, x_{n-1}^*)$ denote the received n -tuple which differs from the transmitted

n -tuple x by a noisy sequence $(e_0, \dots, e_{n-1})$, because of errors that might have occurred during transmission. That is,

$$x_i^* = x_i + e_i \quad ; \quad i = 0, 1, \dots, n-1, \quad (46)$$

where x_i and e_i are elements of $GF(q)$. We define a *parity check* to be the following sum formed at the receiver

$$s_i = \sum_{j=0}^{k-1} h_{ij} x_j^* - x_i^* \quad ; \quad i = k, k+1, \dots, n-1 \quad (47)$$

From (45), (46) and (47), we also have that

$$s_i = \sum_{j=0}^{k-1} h_{ij} e_j - e_i \quad ; \quad i = k, k+1, \dots, n-1. \quad (48)$$

Therefore, a parity check is a summation (in $GF(q)$) of noise symbols which can be calculated at the receiver. The art of majority-logic decoding is to choose the best subset of the total of q^{n-k} such parity checks (we have used the fact that any linear combinations of the rows of H is a parity check). In the Reed-Muller case, this was done by choosing as parity checks the ones that correspond to the perpendicular flats through the points the incidence vector of which is the vector product corresponding to the given information symbol (see section 2.2).

Definition : Let $s_1, s_2, \dots, s_r$ denote check-sums on various noise digits. If a particular noise digit e_i is involved in every sum in the set, and no other error digit is checked by more than one sum, then the sums are said to be *orthogonal* on e_i . More generally, if every sum checks $e_1, \dots, e_j$ and no other error digit appears in more than one sum, then the check sums are orthogonal on the sum $e_1 + \dots + e_j$.

Example : If s_1, s_2, s_3, s_4, s_5 are given by

$$s_1 = e_1 + e_2 + e_4$$

$$s_2 = e_2 + e_3 + e_4 + e_5$$

$$s_3 = e_1 + e_5$$

$$s_4 = e_1 + e_3 + e_6$$

$$s_5 = e_1 + e_4 + e_5$$

then the sums s_1, s_3, s_4 are orthogonal on e_1 , while s_2, s_4 are orthogonal on e_3 and s_3, s_5 are orthogonal on $e_1 + e_5$.

Definition : A linear combination of the parity checks s_i 's given in (48) is said to be a *composite parity check* C_j . I.e.

$$C_j = \sum_{i=k}^{n-1} b_{ji} s_i, \quad (49)$$

where $b_{ji} \in GF(q)$.

The reason for considering orthogonal check sums is the following [7]—

Theorem 9 : If in a linear code there are at least $d - 1$ check sums orthogonal on each digit, then the code has minimum distance at least d .

Proof : It suffices to show that every nonzero codeword has weight at least d . So, consider one of the nonzero digits in a (nonzero) codeword, say x_i , and let $s_{i1}, s_{i2}, \dots, s_{i(d-1)}$ be the first $d - 1$ check sums orthogonal on x_i . Being the parity checks of a codeword, all of these check sums are zero. Since x_i appears in all of these sums, each equation has a nonzero entry and therefore there must exist at least one other nonzero entry in each sum. But since the sums are orthogonal on x_i , no two of the other $d - 1$ x_j 's are equal. Hence, there must exist at least another $d - 1$ (distinct) nonzero digits except x_i , which in turn implies that the minimum weight is d .

As a final remark, note that from (48) and (49) we have that

$$C_j = \sum_{i=k}^{n-1} b_{ji} \left(\sum_{l=0}^{k-1} h_{il} e_l - e_i \right) \quad (50)$$

or, equivalently,

$$C_j = \sum_{i=k}^{n-1} c_{ji} e_i, \quad (51)$$

where

$$c_{ji} = \begin{cases} \sum_{l=k}^{n-1} b_{jl} h_{li}, & \text{for } i = 0, 1, \dots, k-1; \\ b_{ji}, & \text{for } i = k, k+1, \dots, n-1. \end{cases} \quad (52)$$

Using now the above notation, we have the following

Definition : A set of J composite parity checks $\{C_j\}$ is said to be orthogonal on e_m if in equation (51)

$$c_{j,m} = 1 \quad j = 0, 1, \dots, J-1 \quad (53)$$

and

$$c_{j,i} = 0, \quad (54)$$

for all, but at most one index j and for any fixed i different from m .

Although the results are valid over any $GF(q)$, q prime power, we usually restrict ourselves to $GF(2)$.

2.3.2 Majority decoding.

We are now ready to give the following theorem which can be used to determine e_m from a set of J parity checks $\{C_j\}$ orthogonal on e_m [5].

Theorem 10 : Provided that $\lfloor \frac{J}{2} \rfloor$ or fewer of the $\{e_i\}$ that are checked by a set of J parity checks $\{C_j\}$ orthogonal on e_m are nonzero (i.e. there are $\lfloor \frac{J}{2} \rfloor$ or fewer errors in the corresponding received symbols), then e_m is given correctly as that value of $GF(2)$ which is assumed by the greatest fraction of the $\{C_j\}$. (Assume $e_m = 0$ in the case for which no value is assumed by a strict plurality of the $\{C_j\}$, and 0 is one of the several values with most occurrences).

Proof : Suppose that $e_m = \nu$ and assume that all other e_i that are checked by the J parity checks have value 0. Then, from (51) and (53) we have that

$$C_j = \nu \quad j = 0, 1, \dots, J-1. \quad (55)$$

Suppose initially that $\nu = 0$ and that there are at most $\lfloor \frac{J}{2} \rfloor$ errors in the corresponding received symbols. The $\lfloor \frac{J}{2} \rfloor$ nonzero noise digits can affect at most $\lfloor \frac{J}{2} \rfloor$ of the equations in (55) which implies that at least $\lfloor \frac{J}{2} \rfloor$ of the parity checks $\{C_j\}$ are still zero. Hence, zero is either the value with most occurrences in the set $\{C_j\}$ or one of the two values with the same largest number of occurrences; in either case the decoding rule described in the Theorem gives the correct value of e_m .

Next, suppose that $\nu \neq 0$. Since $e_m = \nu \neq 0$, then fewer than $\lfloor \frac{J}{2} \rfloor$ of the other noise digits checked by the parity checks $\{C_j\}$ are nonzero, and therefore more than $\lceil \frac{J}{2} \rceil$ of the equations in (55) are still correct. Again, e_m is given correctly by the decoding rule of the Theorem, and this completes the proof.

Corollary 2 : If there are J parity checks orthogonal on every e_i , the code can correct $\lfloor \frac{1}{2}J \rfloor$ errors.

Proof : It is immediate from the above Theorem.

Note that if the code is cyclic, given any set of J parity checks orthogonal on some e_i , a set of J parity checks orthogonal on other e_i 's can be found by cyclically shifting the given set.

It is also of some interest to note that although the above Theorem suggests that error vectors of weight greater than $\lfloor \frac{1}{2}J \rfloor$ will cause incorrect decoding, there are some cases in which the error vector has weight greater than $\lfloor \frac{1}{2}J \rfloor$ and the message may still be extracted correctly.

The decoding performed according to the algorithm given in the above Theorem is usually called *one-step majority logic decoding*.

Let $\bar{d}$ denote the minimum distance of the dual code of an (n, k) code. The maximum number of errors which can be corrected by one-step majority logic decoding is provided by the following theorem [15].

Theorem 11 : The number of errors which can be corrected by one-step majority logic decoding is at most

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

Proof : Since the maximum distance of the dual code is $\bar{d}$, then the minimum weight of the null space is $\bar{d}$ and therefore, there must exist at least $\bar{d}$ digits in each sum. Since these sums are orthogonal on some error digit e_i , this particular digit appears in all of the sums but the other $\bar{d} - 1$ digits appear in only one of the sums each. But there are another $n - 1$ digits in addition to e_i and therefore we are able to construct at most $\frac{(n-1)}{(\bar{d}-1)}$ orthogonal equations. The number of errors which can

be corrected by one-step majority logic decoding is at most half of these orthogonal equations; i.e. at most $(n-1)/2(d-1)$.

Although the previous result restricts the number of errors which some codes are able to correct, there are codes -such as Reed Muller codes- which can be decoded using several steps of majority logic.

Example : Consider the following nonzero parity checks of the (15,11) Hamming code :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
s_1 :	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
s_2 :	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0
s_3 :	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
s_4 :	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
s_5 :	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0
s_6 :	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0
s_7 :	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1
s_8 :	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
s_9 :	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0
s_{10} :	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1
s_{11} :	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0
s_{12} :	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1
s_{13} :	1	0	1	0	1	1	0	0	1	0	0	0	1	1	1
s_{14} :	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
s_{15} :	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1

Next, consider the following check sums :

$$s_1 = c_0 + c_1 + c_2 + c_3 + c_5 + c_7 + c_8 + c_{11}$$

$$s_9 = c_0 + c_1 + c_4 + c_8 + c_9 + c_{10} + c_{11} + c_{13}$$

$$s_{14} = c_0 + c_1 + c_3 + c_5 + c_6 + c_9 + c_{13} + c_{14}$$

$$s_5 = c_0 + c_4 + c_5 + c_6 + c_7 + c_9 + c_{11} + c_{12}$$

$$s_{15} = c_0 + c_1 + c_2 + c_4 + c_6 + c_7 + c_{10} + c_{14}$$

$$s_{13} = e_0 + e_2 + e_4 + e_6 + e_8 + e_{12} + e_{13} + e_{14}$$

Note that s_1 and s_9 are orthogonal on

$$T_1 = e_0 + e_1 + e_8 + e_{11}$$

and similarly, s_1 and s_{14} are orthogonal on

$$T_2 = e_0 + e_1 + e_3 + e_6$$

Assuming the existence of a single error, we can obtain the correct values of T_1 and T_2 using the majority logic test on s_1 and s_9 for T_1 and on s_1 and s_{14} for T_2 . However, T_1 and T_2 are orthogonal on $e_0 + e_1$, and therefore the correct value of $e_0 + e_1$ can be obtained using the majority logic test on T_1 and T_2 .

In entirely the same manner, s_5 and s_{15} are orthogonal on

$$T_3 = e_0 + e_4 + e_6 + e_7$$

while s_5 and s_{13} are orthogonal on

$$T_4 = e_0 + e_4 + e_5 + e_{12}$$

Again, assuming the existence of a single error, the correct values of T_3 and T_4 can be obtained using the majority logic test on s_5 and s_{15} , and on s_5 and s_{13} respectively. Since T_3 and T_4 are orthogonal on $e_0 + e_4$, a majority logic test will provide us the correct value of $e_0 + e_4$.

Finally, $e_0 + e_1$ and $e_0 + e_4$ are orthogonal on e_0 and a final majority test on these two will provide us the correct value of e_0 .

The majority logic process followed to obtain the correct value of e_0 can be also described by the circuit shown in figure (4) (observe that a two-input majority gate behaves like an AND gate).

Next, note that since the code is cyclic it suffices to design a circuit able to decode the first coordinate; the others are then corrected automatically, since the orthogonal sums for the remaining coordinates can be obtained from the same circuit by an appropriate cyclic shift of the input.

figure (4)

The decoder of figure (4) has three levels of majority gates (or majority logic). In general, a decoder which has L levels of majority logic is called an L -step decoder. Not surprisingly, L -step decoding is more powerful than one-step decoding. In fact, one can hope to correct about twice as many errors with L -step decoding as with one-step decoding, because of the following [7]

Theorem 12 : Let d denote the minimum distance of the dual code of an (n, k) code. Then the number of errors that can be corrected with L -step majority logic decoding (the Reed algorithm), t_L , is bounded by

$$t_L \leq \frac{n}{d} - \frac{1}{2}. \quad (60)$$

Proof : Because of the corollary to Theorem 10, for the code to be able to correct t_L errors, it must be possible to construct at least $2t_L$ check sums orthogonal on a set of digits common to all sums.

Suppose that the number of digits in this set is l . Since the minimum weight of the dual code (which equals its minimum distance) is d and these checks correspond to codewords in the dual code, we conclude that there must exist at least d digits in each sum.

If a_i denotes the number of digits in the i th parity check besides the l digits, then we must have

$$a_i + l \geq d. \quad (61)$$

Note that if $i \neq j$, then

$$a_i + a_j \geq d \quad (62)$$

since otherwise, the sum of the vectors corresponding to the two parity checks i and j , which is also a codeword in the null space of the code, would have weight less than d .

Let

$$S = \sum_{i=1}^{2t_L} a_i. \quad (63)$$

Since $n \geq S + l$, we have that $S \leq n - l$ and from (61),

$$\sum_{i=1}^{2t_L} a_i + \sum_{i=1}^{2t_L} l \geq \sum_{i=1}^{2t_L} d,$$

i.e.

$$S + 2t_L l \geq 2t_L d. \quad (64)$$

From (62),

$$2S \geq 2t_L d. \quad (65)$$

If we now eliminate l from (64) we get

$$S + 2t_L(n - S) \geq 2t_L d,$$

i.e.

$$(2t_L - 1)S \leq 2t_L(n - d). \quad (66)$$

Eliminating S between (65) and (66), we get

$$(2t_L - 1)t_L d \leq 2t_L(n - d),$$

i.e.

$$t_L \leq \frac{n}{d} - \frac{1}{2}.$$

For the Reed-Muller codes in particular, the following result is true [3] :

Theorem 13 : For the r^{th} order Reed-Muller code $R(r, m)$, $(r+1)$ -step majority decoding can correct

$$\left\lfloor \frac{1}{2}(d-1) \right\rfloor = \left\lfloor \frac{1}{2}(2^{m-r} - 1) \right\rfloor$$

errors.

Proof : It has already been mentioned that the dual code of $R(r, m)$ is $R(m-r-1, m)$. Then, Theorem 7 implies that the codewords of minimum weight in $R(m-r-1, m)$ are the incidence vectors of the $(m-(m-r-1))$ -flats in $EG(m, 2)$; i.e. of the $(r+1)$ -flats in $EG(m, 2)$.

The key to the proof is that the check sums corresponding to the $(r+1)$ -flats intersecting on a given r -flat V are orthogonal on the check sum corresponding to that r -flat. However, the check sums corresponding to all $(r+1)$ -flats are known to the decoder which can therefore determine all check sums corresponding to all r -flats with one level of majority logic.

So let U be any $(r+1)$ -flat containing V . There are 2^m points in $EG(m, 2)$ and 2^r points in an r -flat. There are $2^{r+1} - 2^r$ points in U and not in V . Each of the $2^m - 2^r$ points of $EG(m, 2)$ not in V determines a different U , all of which are intersecting on V . Also, no point outside the r -flat V can lie in more than one of the U 's since otherwise two $(r+1)$ -flats would be identical. In addition, every point must lie in one $(r+1)$ -flat since this particular point together with V determine completely an $(r+1)$ -flat containing V . Therefore, we must have

$$\frac{(2^m - 2^r)}{(2^{r+1} - 2^r)} = 2^{m-r} - 1 \quad (67)$$

different U 's any two of which are intersecting only in V . From (67) we can therefore have an estimate for the sum

$$\sum_{P \in V} y_P.$$

Obviously, the above estimate will be correct as long as no more than $\lfloor \frac{1}{2}(2^{m-r} - 1) \rfloor$ errors occur.

This process is repeated for all r -dimensional flats.

Next note that we may view the number of $(r+1)$ -flats as a function of r , the dimension of V . Thus, if W is an $(r-1)$ -flat and V is any r -flat containing W , then there are

$$2^{m-(r-1)} = 2^{m-r+1}$$

such V 's and the values of the corresponding check sums are already known from the previous stage. Hence, we may now obtain an estimate for the value of the sum

$$\sum_{P \in W} y_P$$

and proceeding in entirely the same way, after $r+1$ steps we are able to have an estimate for y_P for any point P , which will be correct provided of course that no more than $\lfloor \frac{1}{2}(2^{m-r} - 1) \rfloor = \lfloor \frac{1}{2}(d - 1) \rfloor$ errors occur.

2.3.3 Improvements of the Reed Algorithm.

The first improvement to be described concerns the number of majority gates, adders, etc. In fact it can reduce their number from an exponential function of the number of steps to a linear function, but at the expense of a linear delay.

Consider again the (15, 11) Hamming code. It has already been mentioned in section 2.3.2 that this is a cyclic code and it therefore suffices to design a decoder able to decode its first coordinate since the others are then corrected automatically, using precisely the same circuit; an appropriate cyclic shift of the original input is only required. A three-step decoding scheme for this code has been proposed (see figure (4)) as well. However, this code is in fact $R(2, 4)^*$ (the punctured 2^{nd} order Reed-Muller code of length 2^4).

In general, instead of the decoding scheme proposed in the previous section (Theorem 13) for the Reed-Muller codes, the punctured Reed-Muller code $R(r, m)^*$ can be used and due to the fact that it is cyclic [3], a single circuit to decode the first coordinate has to be constructed. The dual code $R(r, m)^{\perp}$ contains the incidence vectors of all $(r + 1)$ -flats in $EG(m, 2)$ which do not pass through the origin.

The *sequential code reduction* technique is now described by applying it to the decoder of figure (4) for the (15, 11) code. Referring to figure (4) let

$$m_3, m'_3, m''_3, \dots$$

denote the output from majority gate M_3 at successive times. In particular, we have

$$\begin{aligned} m_3 &= \hat{e}_0 + \hat{e}_1 \\ m'_3 &= \hat{e}_1 + \hat{e}_2 \\ m''_3 &= \hat{e}_2 + \hat{e}_3 \\ m'''_3 &= \hat{e}_3 + \hat{e}_4. \end{aligned} \tag{68}$$

Note that

$$m_3 + m'_3 + m''_3 + m'''_3 = \hat{e}_0 + \hat{e}_4 = m_6 \tag{69}$$

the output of majority gate M_6 orthogonal to m_3 . I.e. the output m_6 can be obtained after three clock cycles from M_3 without using gate m_6 , which can therefore be eliminated. The resulting circuit is given in figure (5).

In this particular example, if we are willing to introduce a delay equivalent to three clock cycles, we can reduce the number of majority gates by three, and the number of adders by two.

figure (5)

The above scheme applies to all cyclic codes and perhaps to other codes as well. Finding the best decoder for each case may not be an easy task at all [10].

The second improvement of the Reed algorithm concerns the number of steps of the algorithm. It can be applied to Reed-Muller codes as well as to other Euclidean Geometry codes, and it is proposed by E.J. Weldon, JR. [15].

For simplicity reasons, in the following paragraphs we use the term *determination of the r -flats* to refer to the determination of the check sums corresponding to the r -flats.

In general, for a prime p , an extended cyclic p -ary Euclidean Geometry code of order r has associated geometry $EG(m, p^s)$, length $n = p^{sm}$ and the additional property that the incidence vector of every $(r+1)$ -flat in this geometry ($EG(m, p^s)$) belongs to the null space of the code.

Given all the $(r+1)$ -flats, the r -flats can be determined by means of the Reed algorithm. Let ζ denote the number of $(r+1)$ -flats which intersect only on a particular r -flat (refer also to the proof of Theorem 13 for the case $p = 2, s = 1$). There are $p^{s(r+1)}$ points in an $(r+1)$ -flat. There are p^{sm} points in $EG(m, p^s)$ and p^{sr} points in an r -flat V . Let U be an $(r+1)$ -flat containing V . Then, there are $p^{s(r+1)} - p^{sr}$ points in U and not in V . As in the proof of Theorem 13, each of the $p^{sm} - p^{sr}$ points of $EG(m, p^s)$ not in V determines a different U , all of which are intersecting on V . Again, no point outside the flat V can lie in more than one of the U 's (otherwise two $(r+1)$ -flats would be identical) and every point must lie in one $(r+1)$ -flat (the one it determines together with V). Therefore, we must have

$$\zeta = \frac{p^{sm} - p^{sr}}{p^{s(r+1)} - p^{sr}} = \frac{p^{s(m-r)} - 1}{p^s - 1} = p^{s(m-r-1)} + p^{s(m-r-2)} + \dots + p^s + 1 \quad (70)$$

$(r+1)$ -flats U intersecting on a particular r -flat V .

It has already been mentioned that it is possible to decode these codes using majority logic in $r+1$ steps. However, when the integer ms is a composite number, the number of steps can be reduced even further, resulting in a substantial savings since the complexity of this type of decoder increases very rapidly with the number L of levels of majority logic which must be employed.

The improved algorithm is now described :

- The first step is identical to the original algorithm; i.e. the r -flats are determined by a majority test of the intersecting $(r+1)$ -flats.

- If $\gcd(r, m) = 1$, then the second step of the improved algorithm is identical to that of the original algorithm, as well; i.e. the $(r-1)$ -flats are determined from the r -flats.

- The process is continued until the c -flats are determined, where $\gcd(c, m) = \delta \neq 1$.

- Let $c' := c/\delta$ and $m' := m/\delta$. Note that a c' -flat in $EG(m', p^{\delta})$ is also a $\delta c'$ -flat in $EG(\delta m', p^{\delta})$ (although the converse is not necessarily true). As a consequence of this observation, the geometry $EG(m', p^{\delta})$ of considerably lower dimension than the original can be used. Since every $\delta c'$ -flat in $EG(\delta m', p^{\delta})$ is known, every c' -flat in $EG(m', p^{\delta})$ is known as well. The number of c' -flats orthogonal on a $(c'-1)$ -flat, ζ' , is given by

$$\begin{aligned} \zeta' &= \frac{p^{\delta m'} - p^{\delta(c'-1)}}{p^{\delta c'} - p^{\delta(c'-1)}} = p^{\delta(m'-c')} + p^{\delta(m'-c'-1)} + \dots + p^{\delta} + 1 \\ &= p^{\delta m - \delta c} + p^{\delta m - \delta c - \delta} + \dots + p^{\delta} + 1. \end{aligned} \quad (71)$$

However, since $r+1 > c$ (hence $sm - s(r+1) < sm - sc$), it can be easily checked from (70) and (71) that

$$\zeta' \geq \zeta \quad (72)$$

for all choices of p, r, s, c and m .

- Next, determine the $(c'-1)$ -flats from the given c' -flats. Since $\zeta' \geq \zeta$ these $(c'-1)$ -flats can be determined provided that at most $\lfloor \zeta/2 \rfloor$ errors occur. In other words, the improved algorithm does not affect the error-correcting capability of the decoder.

- The above process is now repeated to determine the $(c'-2)$ -flats from the $(c'-1)$ -flats and so on, while the same trick can be applied for a further reduction of the number of steps.

Example : Let $p = 2$, $m = 12$, $s = 1$, and consider the 9^{th} order Reed-Muller code. Then, the associated geometry is $EG(12, 2)$ and all 10-flats of this geometry

belong to the null space of the code. Given these 10-flats, the 9-flats have to be determined. However, notice that $\delta = \gcd(9, 12) = 3$. Then, $c' = 9/3 = 3$ and $m' = 12/3 = 4$. I.e. these 9-flats can be regarded as 3-flats in $EG(4, 2^3)$. At the next step, given the 3-flats in $EG(4, 2^3)$ the 2-flats in $EG(4, 2^3)$ (or the 6-flats in $EG(12, 2)$) have to be determined. However, $\delta' = \gcd(2, 4) = 2$ and therefore $c'' = 2/2, m'' = 4/2 = 2$ and these 2-flats in $EG(4, 2^3)$ can be regarded as 1-flats in $EG(2, 2^2)$. Finally, given the 1-flats in $EG(2, 2^2)$ (or the 6-flats in $EG(12, 2)$), the 0-flats have to be determined.

Note that using the above improved algorithm, we first determined the 6-flats from the 9-flats and then the 0-flats from the 6-flats; i.e. the original 10-step algorithm has been reduced down to 2-steps!

The above decoding path together with other paths representing decoding procedures for other Reed-Muller codes and Euclidean Geometry codes is given in figure (6).

2.3.4 Two-step decoding of all EG and PG codes.

A modification of the Reed algorithm capable of decoding all Euclidean Geometry and Projective Geometry codes in two steps is now stated.

Consider an r^{th} order Projective Geometry code. Then all r -flats of $PG(m, p^e)$ belong to its null space. The number of r -flats intersecting on a particular $(r-1)$ -flat is given by

$$\zeta = \frac{p^{e(m-r+1)} - 1}{p^e - 1} \quad (73)$$

(as in the previous section).

At the first step of the modified algorithm all r -flats are determined provided that at most $\lfloor \zeta/2 \rfloor$ errors occur, these r -flats are determined correctly in the usual way.

If $r = 1$, the process terminates at this point, while for $r = 2$ an additional step is required if the usual majority logic algorithm is used for decoding. However, for $r \geq 3$ at least two more steps are required.

The following method is able to decode all these codes in two steps as it is

shown in the following paragraphs.

Step 1 : Determine all $(r-1)$ -flats which intersect the high-order digit of the code.

Step 2 : Determine the 0-flats from the $(r-1)$ -flats found at Step 1.

Theorem 14 : The number of $PG(r, q)$ contained in a $PG(m, q)$, $q = p^e$, p prime, is

$$\frac{(q^{m+1} - 1)(q^{m+1} - q)(q^{m+1} - q^2) \dots (q^{m+1} - q^r)}{(q^{r+1} - 1)(q^{r+1} - q)(q^{r+1} - q^2) \dots (q^{r+1} - q^r)} = \begin{bmatrix} m+1 \\ r+1 \end{bmatrix}, \quad (74)$$

where

$$\begin{bmatrix} m+1 \\ r+1 \end{bmatrix}$$

is a q -ary Gaussian binomial coefficient defined by

$$\begin{bmatrix} x \\ 0 \end{bmatrix} = 1$$

$$\begin{bmatrix} x \\ k \end{bmatrix} = \frac{(q^x - 1)(q^{x-1} - 1) \dots (q^{x-k+1} - 1)}{(q^k - 1)(q^{k-1} - 1) \dots (q - 1)} \quad (75)$$

Proof : There are $(q^{m+1} - 1)(q^{m+1} - q)(q^{m+1} - q^2) \dots (q^{m+1} - q^r)$ ways of choosing $(r+1)$ independent points in $PG(m, q)$ to define a $PG(r, q)$, and there are $(q^{r+1} - 1)(q^{r+1} - q) \dots (q^{r+1} - q^r)$ ways of choosing $(r+1)$ independent points in a $PG(r, q)$. Hence, the number of distinct $PG(r, q)$ in a $PG(m, q)$ is given by (74).

Now let M denote the number of $(r-1)$ -flats in a $PG(m, p^e)$. From the above Theorem, we have that

$$\begin{aligned} M &= \frac{(p^{e(m+1)} - 1)(p^{e(m+1)} - p^e) \dots (p^{e(m+1)} - p^{e(r-1)})}{(p^{er} - 1)(p^{er} - p^e)(p^{er} - p^{2e}) \dots (p^{er} - p^{e(r-1)})} \\ &= \frac{\left(\frac{p^{e(m+1)} - 1}{p^e - 1}\right) p^e \left(\frac{p^{em} - 1}{p^e - 1}\right) \dots p^{e(r-1)} \left(\frac{p^{e(m-r+2)} - 1}{p^e - 1}\right)}{\left(\frac{p^{er} - 1}{p^e - 1}\right) p^e \left(\frac{p^{e(r-1)} - 1}{p^e - 1}\right) \dots p^{e(r-1)} \left(\frac{p^{e(r-r+2)} - 1}{p^e - 1}\right)} \\ &= \frac{(p^{em} + p^{e(m-1)} + \dots + p^e + 1)p^e}{(p^{e(r-1)} + p^{e(r-2)} + \dots + 1)p^e} \\ &= \frac{(p^{e(m-1)} + p^{e(m-2)} + \dots + 1) \dots p^{e(r-1)} (p^{e(m-r+1)} + \dots + 1)}{(p^{e(r-2)} + p^{e(r-3)} + \dots + 1) \dots p^{e(r-1)} 1} \\ &= \frac{(p^{em} + p^{e(m-1)} + \dots + p^e + 1)}{(p^{e(r-1)} + p^{e(r-2)} + \dots + 1)} \\ &= \frac{(p^{em} + p^{e(m-1)} + \dots + p^e) \dots (p^{em} + p^{e(m-1)} + \dots + p^{e(r-1)})}{(p^{e(r-1)} + \dots + p^e) \dots (p^{e(r-1)} + p^{e(r-2)}) p^{e(r-1)}} \end{aligned} \quad (76)$$

But each $(r-1)$ -flat contains $p^{s(r-1)} + p^{s(r-2)} + \dots + p^s + 1$ points. If M^0 denotes the number of $(r-1)$ -flats passing through a specific point, then M^0 is given by :

$$M^0 = M / \frac{(p^{sm} + p^{s(m-1)} + \dots + p^s + 1)}{(p^{s(r-1)} + \dots + p^s + 1)}$$

i.e.

$$M^0 = \frac{(p^{sm} + p^{s(m-1)} + \dots + p^s) \dots (p^{sm} + \dots + p^{s(r-1)})}{(p^{s(r-1)} + p^{s(r-2)} + \dots + p^s) \dots (p^{s(r-1)} + p^{s(r-2)})p^{s(r-1)}} \quad (77)$$

Similarly, its $(r-1)$ -flat contains

$$\frac{(p^{s(r-1)} + \dots + p^s + 1)(p^{s(r-2)} + \dots + p^s + 1)}{(p^s + 1)}$$

1-flats, while the number of 1-flats in $PG(m, p^s)$ is

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

If M^1 denotes the number of $(r-1)$ -flats passing through a given 1-flat (line), then M^1 is given by

$$M^1 = \frac{M(p^{s(r-1)} + \dots + p^s + 1)(p^{s(r-2)} + \dots + p^s + 1)/(p^s + 1)}{(p^{sm} + \dots + p^s + 1)(p^{s(m-1)} + \dots + p^s + 1)/(p^s + 1)} \quad (78)$$

Note that

$$\frac{M^0}{M^1} = \frac{p^{s(m-1)} + \dots + p^s + 1}{p^{s(r-2)} + \dots + p^s + 1} = \frac{p^{sm} - 1}{p^{s(r-1)} - 1} \quad (79)$$

Now consider the M^0 check sums on a particular digit corresponding to the $(r-1)$ -flats passing through a point. Each other digit appears in M^1 of these sums (since there are M^1 1-flats and any two points define an 1-flat).

The value of the particular error digit can be correctly obtained by performing the majority-logic test on the $(r-1)$ -flats (0 is assumed in the case of a tie), provided that at most $\left\lceil \frac{M^0}{2M^1} \right\rceil$ errors occurred.

Note that if $m > r > 1$ from (70) and (79) we get that $\frac{M^0}{M^1} > \frac{1}{2}$ and therefore, we have correct decoding result provided that $\left\lceil \frac{1}{2} \right\rceil$ errors occurred.

Next, consider the Euclidean Geometry $EG(m, p^s)$ obtained from the Projective Geometry $PG(m, p^s)$ by deleting the hyperplane H . From Theorem 14 we have that the number of $PG(r, p^s)$ in $PG(m, p^s)$ is

$$\begin{bmatrix} m+1 \\ r+1 \end{bmatrix}$$

Next, consider the Euclidean Geometry $EG(m, p^s)$ obtained from the Projective Geometry $PG(m, p^s)$ by deleting the hyperplane H . From Theorem 14 we have that the number of $PG(r, p^s)$ in $PG(m, p^s)$ is

$$\begin{bmatrix} m+1 \\ r+1 \end{bmatrix}$$

while the number of $PG(r, p^s)$ in H is

$$\begin{bmatrix} m \\ r+1 \end{bmatrix}$$

Since a $PG(r, p^s)$ either meets H in a $PG(r-1, p^s)$ or is contained in H , we have that the number of $EG(r, p^s)$ in an $EG(m, p^s)$ is the difference of the above numbers. In other words, we have just proved the following

Theorem 15 : The number of $EG(r, p^s)$ (r -flats) in an $EG(m, p^s)$ is

$$p^{s(m-r)} \begin{bmatrix} m \\ r \end{bmatrix},$$

where $\begin{bmatrix} m \\ r \end{bmatrix}$ is a p^s -ary Gaussian binomial coefficient defined as in Theorem 14.

Hence, entirely the same results hold for Euclidean Geometry codes, as well.

From the above discussion it can be seen that a two-step decoding (majority logic decoding) can be achieved for all Euclidean and Projective Geometry codes, either using the improved algorithm (when applicable) or using the modified algorithm.

However, it should be mentioned that it may be more economical to use the original algorithm possibly with the improved form even if more than two steps are required, since M^o can be very large and this can happen even with reasonable values of p, m, s because an M^o -input majority gate is required for the latter (modified) algorithm.

2.4 New decoding algorithm for Reed -Muller codes

Recall from Theorem 2 that if f is a codeword in $R(r, m)$, then f can be written in the form

$$f = |g|g| + |0|h|,$$

where $g \in R(r, m-1)$ and $h \in R(r-1, m-1)$. Schematically,

It is this *superimposition* in terms of which the Reed-Muller codes are interpreted in the *New decoding algorithm* (for Reed-Muller codes) to be described which is due to K. Tokiwa, T. Sugimura, et al [13].

The superimposition for (n, k, d) codes (i.e. codes of length n , information symbols k and minimum distance d) is shown in figure (7), where $Blk(i)$ denotes the i^{th} block.

A: a codeword in (n, k, d) code

B: a codeword in $(n, k', 2d)$ code

C: a codeword in $(2n, k + k', 2d)$ code

figure (7).

From figure (7) one may also observe that the superimposed code can be decomposed into *subcodes* (or *constituent codes*). We shall refer to this inverse operation

of superimposition as *decomposition*.

For the particular case of Reed-Muller codes, if A is a codeword in $R(r, m-1)$ and B is a codeword in $R(r-1, m-1)$, then C is a codeword in $R(r, m)$, where $1 \leq r \leq m-1$. It should also be pointed out that for the special case $r=0$ the $R(0, m)$ Reed-Muller code can be constructed by simply repeating twice a codeword in $R(0, m-1)$ while for $r=m$ the $R(m, m)$ Reed-Muller code can be regarded as the superimposed code which has the $R(m-1, m-1)$ codes as constituent codes or subcodes.

The interpretation of Reed-Muller codes in terms of superimposition enables us to describe the *New algorithm* without using majority logic.

However a decoding algorithm for a class of simple iterated codes defined below which will play an important role in the *New decoding algorithm* for the Reed-Muller codes as well as an algorithm for obtaining subcodes are first stated [13].

2.4.1 Simple iterated codes and the decoding algorithm.

Let C be a codeword in an (n, k, d) code, and consider the codeword C_I obtained by repeating I times the codeword C ; i.e.

$$C_I = \overbrace{|C|C|C| \dots |C|}^{I \text{ times}} \quad (80)$$

The (nI, k, dI) code having codewords of the form (80), where C is a codeword of a given (n, k, d) code, is called the *I-simple iterated code on the basis of the (n, k, d) code*.

Next, let

$$C_I = \overbrace{|C|C|C| \dots |C|}^{I \text{ times}} \quad (81)$$

be some transmitted codeword of an *I-simple iterated code* on the basis of the (n, k, d) code and suppose that

$$R = |C + e_1|C + e_2| \dots |C + e_I| \quad (82)$$

was received instead, where the weights of the error vectors e_i ($i = 1, 2, \dots, I$) are supposed to satisfy

$$\sum_{i=1}^I wt(e_i) \leq \frac{dI}{2}. \quad (83)$$

The decoding of simple iterated codes can be made as follows : [13]

- Decode each block $C + e_i$ of the received word R as $\hat{C}_i$ using any method.
- If error detection has been made above, then proceed to the next block unless $i = I$ in which case the error detection has been completed.
- If error correction has been made during the decoding of $C + e_i$ above, then calculate

$$N_i = \sum_{j=1}^I wt(C + e_j + \hat{C}_j)$$

which is the error weight if we assume that $\hat{C}_i$ is actually correct, and compare it with the threshold value $dI/2$ (where dI is the minimum distance of the code).

If N_i is less than $dI/2$, then we let $C \approx \hat{C}_i$ and the error correction is completed.

Otherwise, we proceed to the next block of the received word R and repeat the whole process, unless $i = I$ in which case the error detection has been completed.

Decoding algorithm for simple-iterated codes :

Step 1 : Let $i = 1$.

Step 2 : Decode $C + e_i$ as $\hat{C}_i$ of the received word R , using any appropriate method (eg. syndrome decoding since we can choose $n \ll nI$).

Step 3 : If error correction is made in Step 2, calculate the value of N_i from

$$N_i = \sum_{j=1}^I wt(C + e_j + \hat{C}_j). \quad (84)$$

If error detection is made in Step 2, then go to Step 5.

Step 4 : Compare N_i with the threshold value $dI/2$.

(i) : If $N_i < dI/2$, go to Step 6.

(ii) : If $N_i \geq dI/2$, go to Step 5.

Step 5 : If $i < I$ let $i = i + 1$ and go to Step 2.

If $i = I$, the error detection is completed.

Step 6 : Let $C = \hat{C}_1$, and the error correction is completed.

Note that in Step 2, either at least one of the blocks is successfully corrected, or all the blocks are error detected.

In the first case, suppose that the decoding of $C + e_k$ (i.e. of the k^{th} block of the received word R) yields the correct version of C ; i.e.

$$\hat{C}_k = C.$$

If the total number of errors occurred is less than or equal to $dI/2$ (in the latter case we assume that dI is even), then

$$\begin{aligned} N_k &= \sum_{j=1}^I wt(C + e_j + \hat{C}_k) \\ &= \sum_{j=1}^I wt(C + e_j + C) \\ &= \sum_{j=1}^I wt(e_j) \\ &\leq \frac{dI}{2}. \end{aligned}$$

On the other hand, if the decoding of $C + e_l$ is not equal to C , i.e.

$$\hat{C}_l \neq C,$$

then

$$\begin{aligned} N_l &= \sum_{j=1}^I wt(C + e_j + \hat{C}_l) \\ &\geq \sum_{j=1}^I [wt(C + \hat{C}_l) - wt(e_j)] \\ &\geq dI - \frac{dI}{2} \geq \frac{dI}{2}. \end{aligned}$$

In the second case (if and only if d is even), the error detection is completed in Step 5.

2.4.2 Algorithm for obtaining subcodes of an $R(r, m)$ code.

According to Theorem 2 (and as it has already been mentioned), the $R(r, m)$ Reed-Muller code can be decomposed into the $R(r, m-1)$ and $R(r-1, m-1)$ subcodes, as illustrated in figure (8).

This decomposition of $R(r, m)$ is called an *1-decomposition* of $R(r, m)$. However, in entirely the same manner, the $R(r, m-1)$ Reed-Muller code can be further decomposed into the $R(r, m-2)$ and the $R(r-1, m-2)$ subcodes to obtain a *2-decomposition* of $R(r, m)$, as illustrated in figure (9).

If we apply the above decomposition v times repeatedly, then we obtain a *v-decomposition* of the $R(r, m)$ Reed-Muller code.

The algorithm presented in this section, obtains each subcode of an $R(r, m)$ Reed-Muller code, while the principle of a *v-decomposition* of $R(r, m)$ is illustrated in figure (10).

In general, the following lemma is valid [13]:

Lemma 3 : Given an integer $v \in \{1, 2, \dots, m-r\}$, the $R(r, m)$ code can be decomposed into subcodes, the $R(r, m-v)$ and the $R(r-1, m-i)$ ($i = 1, 2, \dots, v$) Reed-Muller codes after *v-decomposition*.

Algorithm :

Let C^* be a codeword in the $R(r, m)$ Reed-Muller code of the form

$$C^* = |C_1|C_2| \dots |C_{2^v}|. \quad (85)$$

Step 1 : Let $C_k^{(0)} := C_k$, $k = 1, 2, \dots, 2^v$.

Let $i = 1$.

Step 2 : Obtain the j^{th} B_i from

$$B_i = |C_{2^{v-i}(2^j-2)+1}^{(i-1)} + C_{2^{v-i}(2^j-1)+1}^{(i-1)}| \dots |C_{2^{v-i}(2^j-1)}^{(i-1)} + C_{2^{v-i}2^j}^{(i-1)}| \quad (86)$$

for $j = 1, 2, \dots, 2^{i-1}$.

Step 3 : For $k = 1, 2, \dots, 2^v$ obtain $C_k^{(i)}$ by eliminating the B_i 's as follows :

$$|C_{2^{v-i}(2^j-2)+1}^{(i)}| \dots |C_{2^{v-i}(2^j-1)}^{(i)}| = |C_{2^{v-i}(2^j-2)+1}^{(i-1)}| \dots |C_{2^{v-i}(2^j-1)}^{(i-1)}| \quad (87)$$

$$|C_{2^{v-i}(2^j-1)+1}^{(i)}| \dots |C_{2^{v-i}2^j}^{(i)}| = |C_{2^{v-i}(2^j-1)+1}^{(i-1)}| \dots |C_{2^{v-i}2^j}^{(i-1)}| + B_i, \quad (88)$$

$$A^* \in R(r, m-1)$$

$$B \in R(r-1, m-1)$$

$$C \in R(r, m)$$

figure (8) : 1-decomposition of $R(r, m)$.

$$A \in R(r, m-2)$$

$$B_1 \in R(r-1, m-1)$$

$$B_2 \in R(r-1, m-2)$$

figure (9) : 2-decomposition of $R(r, m)$

$$A \in R(r, m - v)$$

$$B_i \in R(r - 1, m - i), \quad i = 1, 2, \dots, v$$

figure (10) : v -decomposition of $R(r, m)$

for $j = 1, 2, \dots, 2^{j-1}$

Step 4 . If $i < v$, let $i = i + 1$ and go to Step 2.

If $i = v$, go to Step 5.

Step 5 . All the A 's are obtained by :

$$A = |C_k^{(v)}| \quad , \quad k = 1, 2, \dots, 2^v \quad (89)$$

Example : Consider the $R(r, m)$ Reed-Muller code and let

$$C^* = |C_1|C_2|C_3|C_4|$$

From the first two steps of the algorithm, we get that

$$B_1 = |C_1 + C_3|C_2 + C_4|$$

and from Step 3 we have that

$$|C_1^{(1)}|C_2^{(1)}| = |C_1^{(0)}|C_2^{(0)}|$$

and

$$|C_3^{(1)}|C_4^{(1)}| = |C_3^{(0)}|C_4^{(0)}| + B_1$$

Next, the first B_2 is given by

$$B_2 = |C_1^{(1)} + C_3^{(1)}|$$

while the second B_2 is given by

$$B_2 = |C_3^{(1)} + C_4^{(1)}|$$

From Step 3 we get that

$$|C_1^{(2)}| = |C_1^{(1)}|$$

and

$$|C_3^{(2)}| = |C_3^{(1)}|$$

while

$$|C_2^{(2)}| = |C_2^{(1)}| + B_2$$

$$|C_4^{(2)}| = |C_4^{(1)}| + B_2.$$

Finally, the four A 's are given by :

$$A = |C_1^{(1)}|$$

$$A = |C_2^{(1)}| + B_2$$

$$A = |C_3^{(1)}|$$

$$A = |C_4^{(1)}| + B_2.$$

It should be noted that the $R(r, m)$ Reed-Muller code is the 2^v -simple iterated code and for $i = 1$ to v the $R(r-1, m-i)$ Reed-Muller code is the 2^{i-1} -simple iterated code. If $r-1 = 0$ then the 2^{i-1} -simple iterated code on the basis of the $R(0, m-i)$ Reed-Muller code can be constructed with the 2^v -simple iterated code on the basis of the $R(0, m-r-v)$ Reed-Muller code. If $r \geq 1$ then the $R(r-1, m-i)$ Reed-Muller code can be decomposed into the $R(r-1, m-v-1)$ and the $R(r-2, m-i-i')$ Reed-Muller codes ($i = 1, 2, \dots, v$; $i' = 1, 2, \dots, v+1-i$) after $(v+1-i)$ -decomposition. I.e. the original $R(r, m)$ Reed-Muller code has now been decomposed into the $R(r, m-v)$, the $R(r-1, m-v-1)$ and the $R(r-2, m-i-i')$ Reed-Muller codes ($i = 1, 2, \dots, v$; $i' = 1, 2, \dots, v+1-i$). The above operation is repeated until the subcodes of lowest order are of order 0.

Theorem 16 : [13] Given an integer $v \in \{1, 2, \dots, m-r\}$ the $R(r, m)$ Reed-Muller code can be decomposed into the $R(r-j, m-v-j)$ Reed-Muller codes ($j = 0, 1, \dots, r$) with the same minimum distance 2^{m-r-v} , each of which constitutes the 2^v -simple iterated code.

Example : Consider the $R(2, 6)$ Reed-Muller code, and assume that the minimum distance of each subcode is at least 4; i.e. (from Theorem 16)

$$2^{m-r-v} \geq 2^2 \Rightarrow v \leq m-r-2$$

and therefore,

$$v \in \{0, 1, 2, \dots, m-r-2\}.$$

For the particular case of $R(2, 6)$ we have that

$$v \leq 6 - 2 - 2 = 2$$

and therefore, a 2-decomposition of $R(2, 6)$ is attainable; $R(2, 6)$ is decomposed into $R(2, 5)$ and $R(1, 5)$, as shown in figure (11).

If we now consider $R(2, 5)$, we have that

$$v \leq 5 - 2 - 2 = 1$$

and therefore an 1-decomposition of $R(2, 5)$ is attainable as shown in figure (12).

Next, note that since $4 - 2 - 2 = 0$, a 0-decomposition is attainable for $R(2, 4)$, in other words, A cannot be decomposed any further.

However, $5 - 1 - 2 = 2$ and $4 - 1 - 2 = 1$ and therefore a 2-decomposition is attainable for $R(1, 5)$ while an 1-decomposition is attainable for $R(1, 4)$, both of which are shown in figure (13).

The $R(0, 3)$ subcodes can be further decomposed into $R(0, 2)$ subcodes. The complete decomposition of $R(2, 6)$ is given in figure (14).

2.4.3 A decoding algorithm for Reed-Muller codes.

A new algorithm for Reed-Muller codes applied for the 2^v -simple iterated code on the basis of the code with minimum distance 2^{m-r-v} is now explained by means of an example.

Again, it has been assumed that the minimum distance of each subcode is at least 4; i.e.

$$2^{m-r-v} \geq 2^2,$$

and therefore, v is a positive integer less than or equal to $m - r - 2$; i.e.

$$v \in \{1, 2, \dots, m - r - 2\}$$

(for a nontrivial decomposition to be obtained).

Example : Consider the $R(2, 6)$ Reed-Muller code the decomposition of which we have already obtained in figure (14) where $A \in R(2, 4)$, $B_{22}, B_{12} \in R(1, 3)$ and B_{21}, B_{12}, B_{11} are vectors of length 2^2 of the form $(0, 0, \dots, 0)$ or $(1, 1, \dots, 1)$.

$$C \in R(2,6)$$

$$A' \in R(2,5)$$

$$B_1 \in R(1,5)$$

figure (11) : decomposition of $R(2,6)$

$$A \in R(2,4)$$

$$B_2 \in R(1,4)$$

figure (12) : 1-decomposition of $R(2,5)$

figure.(13)

figure (14)

Suppose that C^* was the transmitted codeword, with

$$C^* = |C_1|C_2|C_3|C_4|, \quad (90)$$

where

$$C_4 = 1010010101011010$$

$$C_3 = 0011110011000011$$

$$C_2 = 0101010101010101$$

$$C_1 = 1100110011001100$$

and the word R was received instead, with

$$R = |R_1|R_2|R_3|R_4|, \quad (91)$$

where

$$R_1 = 1100010011001100$$

$$R_2 = 0101010101010101$$

$$R_3 = 0011110111000011$$

$$R_4 = 1010010101010010$$

The algorithm to be described first applies the decomposition algorithm given in section 2.4.2 on the received word in order to obtain a decomposition similar to the one given in figure (14), and then it decodes level by level starting from the lower levels and proceeding upwards using the algorithm described in section 2.4.1.

Note that all of B_{21}, B_{12} and B_{11} are vectors of length 2^3 consisting either entirely of 1's or entirely of 0's.

B_1 is first decoded :

(i). Obtain B_{11} as follows : Consider r_1 , with

$$r_1 = |R_1 + R_3|R_2 + R_4|$$

the regenerated vector corresponding to B_1 .

Then,

$$\begin{aligned} r_1 &= |1111100100001111|1111000000010111| \\ &= |11111001|00001111|11110000|00010111| \\ &:= |\beta_1|\beta_2|\beta_3|\beta_4| \end{aligned}$$

where

$$\beta_1 = 11111001$$

$$\beta_2 = 00001111$$

$$\beta_3 = 11110000$$

$$\beta_4 = 00010111$$

Next, let r_{11} be the regenerated vector corresponding to $|B_{11}|B_{11}|B_{11}|B_{11}|$, which can be obtained from r_1 as follows :

$$\begin{aligned} r_{11} &= |\beta_1 + \beta_3| \beta_2 + \beta_4| \\ &= |00001001|00011000| \\ &= |0000|1001|0001|1000| \\ &:= |B_{11} + e'_1|B_{11} + e'_2|B_{11} + e'_3|B_{11} + e'_4|, \end{aligned}$$

where

$$B_{11} + e'_1 = 0000$$

$$B_{11} + e'_2 = 1001$$

$$B_{11} + e'_3 = 0001$$

$$B_{11} + e'_4 = 1000$$

The errors e'_1, e'_2, e'_3, e'_4 which are caused by the (original or channel) errors e_i ($i = 1, 2, 3, 4$) during the regeneration process of r_1 and r_{11} are also called *pseudo-error vectors*.

Using the decoding algorithm for the 4-simple iterated code on the basis of the $R(0, 2)$ Reed-Muller code (see section 2.4.1), we obtain

$$B_{11} = 0000. \quad (92)$$

(ii) Next, obtain B_{12} as follows :

B_{11} is first eliminated in r_1 :

$$\beta_1^{(1)} := \beta_1 = 11111001$$

$$\beta_2^{(1)} := \beta_2 = 00001111$$

$$\beta_3^{(1)} := \beta_3 + |B_{11}|B_{11}| = 11110000$$

$$\beta_4^{(1)} := \beta_4 + |B_{11}|B_{11}| = 00010111.$$

The regenerated vector r_{12} corresponding to $|B_{12}|B_{12}|B_{12}|B_{12}|$ is obtained by

$$\begin{aligned} r_{12} &= |\beta_1^{(1)} + \beta_2^{(1)}|\beta_3^{(1)} + \beta_4^{(1)}| \\ &= |1\ 1\ 1\ 1|0\ 1\ 1\ 0|1\ 1\ 1\ 0|0\ 1\ 1\ 1| \\ &:= |B_{12} + e'_5|B_{12} + e'_6|B_{12} + e'_7|B_{12} + e'_8|, \end{aligned}$$

where

$$B_{12} + e'_5 = 1\ 1\ 1\ 1$$

$$B_{12} + e'_6 = 0\ 1\ 1\ 0$$

$$B_{12} + e'_7 = 1\ 1\ 1\ 0$$

$$B_{12} + e'_8 = 0\ 1\ 1\ 1$$

and e'_i ($i = 5, 6, 7, 8$) are the pseudo-error vectors.

Again, using the decoding algorithm (section 2.4.1) for the 4-simple iterated code on the basis of the $R(0, 2)$ Reed-Muller code, we obtain

$$B_{12} = 1\ 1\ 1\ 1. \quad (93)$$

(iii). Now, eliminate B_{12} and obtain B_{13} as below :

$$\beta_1^{(2)} := \beta_1^{(1)} = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1$$

$$\beta_2^{(2)} := \beta_2^{(1)} + |B_{12}|B_{12}| = 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0$$

$$\beta_3^{(2)} := \beta_3^{(1)} = 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0$$

$$\beta_4^{(2)} := \beta_4^{(1)} + |B_{12}|B_{12}| = 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0.$$

Then, the regenerated vector r_{13} corresponding to $|B_{13}|B_{13}|B_{13}|B_{13}|$ is given by :

$$\begin{aligned} r_{13} &= |\beta_1^{(2)}|\beta_2^{(2)}|\beta_3^{(2)}|\beta_4^{(2)}| \\ &= |1\ 1\ 1\ 1\ 1\ 0\ 0\ 1|1\ 1\ 1\ 1\ 0\ 0\ 0\ 0|1\ 1\ 1\ 1\ 0\ 0\ 0\ 0|1\ 1\ 1\ 0\ 1\ 0\ 0\ 0| \\ &:= |B_{13} + e'_9|B_{13} + e'_{10}|B_{13} + e'_{11}|B_{13} + e'_{12}| \end{aligned}$$

where,

$$B_{13} + e'_9 = 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1$$

$$B_{13} + e'_{10} = 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0$$

$$B_{13} + e'_{11} = 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0$$

$$B_{13} + e'_{12} = 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0$$

and, as usual, e_i ($i = 9, 10, 11, 12$) are the pseudo-error vectors.

Using the decoding algorithm for the 4-simple iterated code but now on the basis of the $R(1, 2^3)$ Reed-Muller code, we obtain

$$B_{13} = 11110000. \quad (94)$$

Note that

$$B_1 = |B_{13}|B_{13} + (|B_{12}|B_{12}|)|B_{13} + (|B_{11}|B_{11}|)|B_{13} + (|B_{12}|B_{12}|) + (|B_{11}|B_{11}|). \quad (95)$$

From (92), (93), (94) and (95) it is readily obtained

$$B_1 = 1111000000001111111000000001111. \quad (96)$$

B_2 is next decoded :

(i). Eliminate B_1 in R and obtain B_{21} :

$$|R_1^{(1)}|R_2^{(1)}| := |R_1|R_2|$$

$$|R_3^{(1)}|R_4^{(1)}| := |R_3|R_4| + B_1$$

where,

$$R_1^{(1)} = 1100010011001100$$

$$R_2^{(1)} = 0101010101000101$$

$$R_3^{(1)} = 1100110111001100$$

$$R_4^{(1)} = 0101010101011101$$

The regenerated vector r_2 corresponding to $|B_2|B_2|$ is obtained by

$$r_2 = |R_1^{(1)} + R_2^{(1)}|R_3^{(1)} + R_4^{(1)}|.$$

i.e.

$$\begin{aligned} r_2 &= |1001000110001001|1001100010010001| \\ &:= |10010001|10001001|10011000|10010001| \\ &:= |\gamma_1|\gamma_2|\gamma_3|\gamma_4|, \end{aligned}$$

where,

$$\gamma_1 = 10010001$$

$$\gamma_2 = 10001001$$

$$\gamma_3 = 10011000$$

$$\gamma_4 = 10010001$$

Next, the regenerated vector r_{21} corresponding to $|B_{21}|B_{21}|B_{21}|B_{21}|$ is obtained from r_2 as follows

$$\begin{aligned} r_{21} &= |\gamma_1 + \gamma_2| \gamma_3 + \gamma_4| \\ &= |00011000|00001001| \\ &= |0001|1000|0000|1001| \\ &:= |B_{21} + e''_1|B_{21} + e''_2|B_{21} + e''_3|B_{21} + e''_4|, \end{aligned}$$

where,

$$B_{21} + e''_1 = 0001$$

$$B_{21} + e''_2 = 1000$$

$$B_{21} + e''_3 = 0000$$

$$B_{21} + e''_4 = 1001$$

The error vectors e''_i ($i = 1, 2, 3, 4$) are also called pseudo-error vectors.

Using the decoding algorithm for the 4-simple iterated code on the basis of the $R(0, 2)$ Reed-Muller code, we obtain

$$B_{21} = 0000. \quad (97)$$

The process continues in entirely the same way as in the decoding of B_1 ; i.e.

(ii). Eliminate B_{21} in r_2 and obtain B_{22} :

$$\gamma_1^{(1)} := \gamma_1 = 10010001$$

$$\gamma_2^{(1)} := \gamma_2 + |B_{21}|B_{21}| = 10001001$$

$$\gamma_3^{(1)} := \gamma_3 = 10011000$$

$$\gamma_4^{(1)} := \gamma_4 + |B_{21}|B_{21}| = 10010001$$

The regenerated vector r_{22} corresponding to $|B_{22}|B_{22}|B_{22}|B_{22}|$ is obtained as below :

$$\begin{aligned} r_{22} &= |\gamma_1^{(1)}|\gamma_2^{(1)}|\gamma_3^{(1)}|\gamma_4^{(1)}| \\ &= |10010001|10001001|10011000|10010001| \\ &:= |B_{22} + e_5''|B_{22} + e_6''|B_{22} + e_7''|B_{22} + e_8''| \end{aligned}$$

where,

$$B_{22} + e_5'' = 1001 \bigg| 0001$$

$$B_{22} + e_6'' = 10001001$$

$$B_{22} + e_7'' = 10011000$$

$$B_{22} + e_8'' = 10010001$$

and e_i'' ($i = 5, 6, 7, 8$) are the pseudo-error vectors.

The application of the decoding algorithm for the 4-simple iterated code on the basis of the $R(1, 3)$ Reed-Muller code, results in

$$B_{22} = 10011001. \quad (98)$$

But B_2 is given by

$$B_2 = |B_{22}|B_{22} + (|B_{21}|B_{21}|). \quad (99)$$

and therefore (99) together with (98) and (97) yield

$$B_2 = 1001100110011001. \quad (100)$$

And, finally, we have reached the decoding of A :

B_2 is eliminated as below :

$$R_1^{(2)} := R_1^{(1)} = 1100010011001100$$

$$R_2^{(2)} := R_2^{(1)} + B_2 = 1100110011011100$$

$$R_3^{(2)} := R_3^{(1)} = 1100110111001100$$

$$R_4^{(2)} := R_4^{(1)} + B_2 = 1100110011000100$$

The regenerated vector r corresponding to $|A|A|A|A|$ is obtained by

$$\begin{aligned} r &= |R_1^{(2)}|R_2^{(2)}|R_3^{(2)}|R_4^{(2)}| \\ &:= |A + e_1|A + e_2|A + e_3|A + e_4|, \end{aligned}$$

where,

$$A + e_1 = 1100010011001100$$

$$A + e_2 = 1100110011011100$$

$$A + e_3 = 1100110111001100$$

$$A + e_4 = 1100110011000100$$

and, e_i ($i = 1, 2, 3, 4$) are the original (channel) error vectors.

Using the decoding algorithm for the 4-simple iterated code on the basis of the $R(2, 4)$ Reed-Muller code, we obtain,

$$A = 1100110011001100. \quad (101)$$

The transmitted C^* is now given by

$$\begin{aligned} C^* &= |C_1|C_2|C_3|C_4| \\ &= |A|A + B_2|A + (|B_{13}|B_{13} + (|B_{12}|B_{12}|))|A + B_2 + \\ &\quad + (|B_{13} + (|B_{11}|B_{11}|)|B_{13} + (|B_{12}|B_{12}|) + (|B_{11}|B_{11}|))|. \end{aligned}$$

I.e.

$$\begin{aligned} C_1 &= A \\ &= 1100110011001100 \\ C_2 &= A + B_2 \\ &= 01010101010101 \\ C_3 &= A + (|B_{13}|B_{13} + (|B_{12}|B_{12}|)) \\ &= 0011110011000011 \\ C_4 &= A + B_2 + (|B_{13} + (|B_{11}|B_{11}|)|B_{13} + (|B_{12}|B_{12}|) + (|B_{11}|B_{11}|)) \\ &= 1010010101011010 \end{aligned}$$

and the decoding process has been completed.

An outline of the *New Algorithm* for the decoding of the $R(r, m)$ Reed-Muller code is now stated [13].

Algorithm for decoding of the $R(r, m)$ Reed-Muller code :

Step 1 : Obtain the regenerated vector corresponding to the simple iterated code in Theorem 16.

Step 2 : Decode this vector using the algorithm of section 2.4.1.

Step 3 : Eliminate the codeword in the simple iterated code obtained, and return to *Step 1*.

Note that in *Step 2*, we used the decoding algorithm for the 2^v -simple iterated code on the basis of the code with minimum distance 2^{m-r-v} , where v is given, and $v \in \{1, 2, \dots, m-r-2\}$.

It can be shown (see [13]) that the decoding delay of the above algorithm is much shorter compared to that of the majority-logic algorithm for the cyclic Reed-Muller codes especially when r is relatively large.

A second feature of the algorithm is that when applied to the $R(r, m)$ Reed-Muller code, it can successfully detect 2^{m-r-1} errors. However, its most interesting feature is that the new decoder has a practicable size.

The reason the above algorithm possesses all of these features, is that despite the conventional algorithms where the decoder produces only one symbol at a time, in this new algorithm the decoder produces all the information symbols in a subcode at a time.

2.5 Fast decoding of first-order Reed-Muller codes .

The class of first order Reed-Muller codes is doubtless one of the most popular classes of codes. The fact that $R(1, m)$ are $[2^m, m+1, 2^{m-1}]$ codes, which has as a consequence their low rate and their high correcting capacity, makes them particularly suited to very noisy channels, especially when the message is very precious and the sender is willing to use most of the channel for error control ($R(1, 5)$ was successfully used in the Mariner 9 spacecraft to transmit pictures from Mars). Because of these distinguished features, although the general techniques for encoding and decoding Reed-Muller codes can also be applied to $R(1, m)$, several other special techniques have been proposed for this particular class of codes.

The encoding scheme of $R(1, m)$ together with some preliminary results are first stated, followed by a decoding algorithm based on the Fast Walsh Transform (FWT). A detailed description of the decoding circuit (Green machine) can be found in [3]. Finally, a fast decoding algorithm for $R(1, m)$ due to S. N. Litsyn and O. I. Shekhovtsov [2] is stated in which the number of operations required to decode a block increases linearly in relation to the code length, compared to the $O(n \log n)$ complexity of the algorithm based on the Fast Walsh Transform.

2.5.1 Encoding of $R(1, m)$ and preliminary results.

There is nothing complicated in the encoding of an $R(1, m)$ Reed-Muller code.

Consider the case $m = 4$, and let $u_0 u_1 u_2 u_3 u_4$ be a message. The codeword corresponding to $(u_0 u_1 u_2 u_3 u_4)$ is given by :

$$(x_0 \dots x_{15}) = (u_0 \dots u_4) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (102)$$

and the corresponding circuit is given in figure (15).

Note the existence of a clock circuit in figure (15). It generates the states $t_1 t_2 t_3 t_4 = 0000, 0001, 0010, 0011, 0100, 0101, \dots, 1111$ or, equivalently, it counts from 0 to 15. The result of the circuit is

$$u_0 + t_1 u_1 + t_2 u_2 + t_3 u_3 + t_4 u_4,$$

which is actually the codeword $x_0 x_1 x_2 \dots x_{15}$ (see also (102)).

The generalization of the above scheme to any $R(1, m)$ Reed-Muller code is immediate. The upper registers will contain the information symbols $u_0 u_1 \dots u_m$ while the clock circuit will generate all successive states corresponding to the binary representation of the numbers $1, 2, \dots, 2^m - 1$. The resulting codeword is then given by

$$u_0 + t_1 u_1 + t_2 u_2 + \dots + t_m u_m. \quad (103)$$

figure (15)

Before proceeding to the decoding scheme, we state a few definitions and results for a better understanding of the decoding algorithm. For further information, the reader may refer to [3] and [2].

Definition : If $A = (a_{ij})$ is an $m \times m$ matrix and $B = (b_{ij})$ is an $n \times n$ matrix over any field, the *Kronecker product* of A and B is the $mn \times mn$ matrix obtained from B by replacing every entry b_{ij} by the $m \times m$ matrix $b_{ij}A$. Symbolically,

$$A \otimes B = (b_{ij}A).$$

Note that if

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

then

$$I_2 \otimes H_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}$$

while

$$H_2 \otimes I_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

i.e.

$$I_2 \otimes H_2 \neq H_2 \otimes I_2$$

which implies that the Kronecker product of two matrices A and B is not commutative.

The definition of Hadamard matrices which were introduced by Hadamard [1] and in an earlier use by Sylvester [12] as well as a few properties of them are now stated.

Definition : The Hadamard matrix H_n of order 2^m is defined inductively by

$$H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$H_{2^m} = H_{2^{m-1}} \otimes H_2, \quad m \geq 2. \quad (104)$$

Example : For $m = 2$, (104) yields :

$$H_4 = H_2 \otimes H_2.$$

i.e.

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad (105)$$

Our objective now is to express the matrix H_{2^m} in such a form that the matrix product $H_{2^m} x^T$ requires a computational effort as low as possible

Consider the case $m = 3$. Then,

$$H_8 = H_4 \otimes H_2 = \begin{pmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{pmatrix} = \begin{pmatrix} H_4 & 0 \\ 0 & H_4 \end{pmatrix} \cdot \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix} \quad (106)$$

But quite similarly,

$$H_4 = H_2 \otimes H_2 = \begin{pmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{pmatrix} = \begin{pmatrix} H_2 & 0 \\ 0 & H_2 \end{pmatrix} \cdot \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} \quad (107)$$

Combining (106) and (107) together, we obtain

$$H_8 = \begin{pmatrix} \begin{pmatrix} H_2 & 0 \\ 0 & H_2 \end{pmatrix} \cdot \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} H_2 & 0 \\ 0 & H_2 \end{pmatrix} \cdot \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} \end{pmatrix} \cdot \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix}$$

i.e.

$$\begin{aligned} H_8 &= \begin{pmatrix} H_2 & 0 & 0 & 0 \\ 0 & H_2 & 0 & 0 \\ 0 & 0 & H_2 & 0 \\ 0 & 0 & 0 & H_2 \end{pmatrix} \cdot \begin{pmatrix} I_2 & I_2 & 0 & 0 \\ I_2 & -I_2 & 0 & 0 \\ 0 & 0 & I_2 & I_2 \\ 0 & 0 & I_2 & -I_2 \end{pmatrix} \cdot \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix} \\ &= (H_2 \otimes I_4)(I_2 \otimes H_2 \otimes I_2)(I_4 \otimes H_2) \\ &= (H_2 \otimes I_2 \otimes I_2)(I_2 \otimes H_2 \otimes I_2)(I_2 \otimes I_2 \otimes H_2) \end{aligned} \quad (108)$$

where 0 denotes a null matrix of the appropriate size and I_n denotes the $n \times n$ identity matrix.

Using the fact that

$$I_{2^m} = I_{2^{m-1}} \otimes I_2 \quad (109)$$

the generalization of the above result (given in (108)) is now immediate :

Theorem 17 : (*The Fast Hadamard Transform Theorem*):

$$H_{2^m} = \prod_{i=1}^m (I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m-i}}) \quad (110)$$

Proof : It is a simple induction on m .

For $m = 1$ the result is trivially true ($H_2 = H_2$).

Assume that the assertion is true for m .

Let $C_m^{(i)}$ denote the i^{th} factor of the product given in (110); i.e.

$$C_m^{(i)} = I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m-i}}, \quad 1 \leq i \leq m.$$

Then

$$\begin{aligned} C_{m+1}^{(i)} &= I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m+1-i}} \\ &= I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m-i}} \otimes I_2 \\ &= C_m^{(i)} \otimes I_2 \end{aligned}$$

and

$$C_{m+1}^{(m+1)} = I_{2^m} \otimes H_2.$$

Hence,

$$\begin{aligned} \prod_{i=1}^{m+1} C_{m+1}^{(i)} &= \prod_{i=1}^{m+1} (I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m+1-i}}) \\ &= \prod_{i=1}^m (C_m^{(i)} \otimes I_2) \cdot (I_{2^m} \otimes H_2) \\ &= \left(\prod_{i=1}^m C_m^{(i)} \right) \cdot I_{2^m} \otimes \left(\prod_{i=1}^m I_2 \right) \cdot H_2 \\ &= H_{2^m} \otimes H_2 \\ &= H_{2^{m+1}} \end{aligned}$$

and our induction has been completed.

Next, consider a permutation matrix P the action of which on $(x_0, \dots, x_{2^m-1})$ is given by

$$P(x_0, x_1, \dots, x_{2^m-1})^T = (x_0, x_{2^m-1}, x_1, x_{2^m-2}, \dots, x_{2^m-1-1}, x_{2^m-1})^T \quad (111)$$

which splits the vector $(x_0, x_1, \dots, x_{2^m-1})$ into two halves and then produces an ideal shuffle. *

Theorem 18 : If P is a permutation matrix as in (111), then

$$P^m = I. \quad (112)$$

Proof : A permutation P acting on a vector

$$x = (x_0, x_1, \dots, x_{2^m-1})$$

moves an element x_i ($i = 0, 1, \dots, 2^m - 1$) of the vector x to a position j , such that

$$j \equiv 2i \pmod{(2^m - 1)}.$$

Let k be a nonnegative integer so that P^k when applied to a vector x fixes x_i , i.e.

$$2^k \cdot i \equiv i \pmod{(2^m - 1)}$$

or

$$i \cdot (2^k - 1) \equiv 0 \pmod{(2^m - 1)}. \quad (113)$$

Note that for $k = m$ (113) is valid for any $i \in \{0, 1, \dots, 2^m - 1\}$. This implies that the order of P divides m ; hence,

$$P^m = I.$$

Theorem 19 : If P is a permutation matrix as defined in (111) and A_k , $k = 0, 1, \dots, m$ are $n \times n$ matrices, then

$$P \cdot (A_0 \otimes A_1 \otimes A_2 \otimes \dots \otimes A_m) \cdot P^{-1} = A_m \otimes A_0 \otimes A_1 \otimes \dots \otimes A_{m-1}. \quad (114)$$

Proof : The action of P on a vector has already been described above. Its action on a matrix A is analogous. However, in

$$PAP^{-1}$$

P acts on the rows of A and then on the columns of A .

Let a_{ij} be an entry in

$$A_m \otimes A_0 \otimes A_1 \otimes \dots \otimes A_{m-1}.$$

Let also

$$i = i_0 + i_1 2 + i_2 2^2 + \dots + i_m 2^m$$

and

$$j = j_0 + j_1 2 + j_2 2^2 + \dots + j_m 2^m$$

be the binary representations of i and j respectively.

Recall that

$$A \otimes B = (h, A).$$

Therefore,

$$a_{ij} = a_{i_m j_m}^{(m-1)} a_{i_{m-1} j_{m-1}}^{(m-2)} \dots a_{i_1 j_1}^{(0)} a_{i_0 j_0}^{(m)},$$

where $a_{i,j}^{(r)}$ denotes the i,j entry of the matrix A_r .

Suppose that a_{ij} is the image of an element a_{ij} .

If i is even, then $\hat{i} = i/2$.

If i is odd, then $\hat{i} = \frac{i + (2^m - 1)}{2}$.

Then,

$$\hat{i} = i_1 + i_2 2 + \dots + i_m 2^{m-1} + i_0 2^m.$$

Similar observations hold for j as well.

Hence, a_{ij} is the image of the element $a_{\hat{i}\hat{j}}$ such that

$$a_{\hat{i}\hat{j}} = a_{i_0 j_0}^{(m)} a_{i_m j_m}^{(m-1)} \dots a_{i_1 j_1}^{(0)} \in A_0 \otimes A_1 \otimes \dots \otimes A_m.$$

This implies that

$$P \cdot (A_0 \otimes A_1 \otimes \dots \otimes A_m) \cdot P^{-1} = A_m \otimes A_0 \otimes A_1 \otimes \dots \otimes A_{m-1}.$$

Theorem 20 : The Hadamard matrix H_{2^m} can be written in the form

$$\begin{aligned} H_{2^m} &= \{(H_2 \otimes I_{2^{m-1}}) \cdot P\}^m \\ &:= B^m, \end{aligned} \tag{115}$$

where $B = (H_2 \otimes I_{2^{m-1}}) \cdot P$ is a matrix containing two nonzero elements in each row.

Proof : Consider the following Kronecker product of 2×2 matrices :

$$H_2 \otimes I_{2^{m-1}} = H_2 \otimes I_2 \otimes I_2 \otimes \dots \otimes I_2.$$

From Theorem 19 we have that

$$\begin{aligned} P(H_2 \otimes I_2 \otimes I_2 \otimes \dots \otimes I_2)P^{-1} &= I_2 \otimes H_2 \otimes I_2 \otimes \dots \otimes I_2 \\ &= I_2 \otimes H_2 \otimes I_{2^{m-2}} \\ P^2(H_2 \otimes I_2 \otimes I_2 \otimes \dots \otimes I_2)P^{-2} &= P(I_2 \otimes H_2 \otimes I_2 \otimes \dots \otimes I_2)P^{-1} \\ &= I_2 \otimes I_2 \otimes H_2 \otimes I_2 \otimes \dots \otimes I_2 \\ &= I_{2^2} \otimes H_2 \otimes I_{2^{m-3}} \end{aligned} \quad (116)$$

$$\begin{aligned} P^{m-1}(H_2 \otimes I_2 \otimes I_2 \otimes \dots \otimes I_2)P^{-m+1} &= I_2 \otimes I_2 \otimes I_2 \otimes \dots \otimes H_2 \\ &= I_{2^{m-1}} \otimes H_2. \end{aligned}$$

However, from Theorem 17 we have that

$$\begin{aligned} H_{2^m} &= \prod_{i=1}^m (I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m-i}}) \\ &= (H_2 \otimes I_{2^{m-1}})(I_2 \otimes H_2 \otimes I_{2^{m-2}}) \dots (I_{2^{m-1}} \otimes H_2). \end{aligned} \quad (117)$$

From (116) and (117) we have that

$$\begin{aligned} H_{2^m} &= (H_2 \otimes I_{2^{m-1}})[P(H_2 \otimes I_{2^{m-1}})P^{-1}][P^2(H_2 \otimes I_{2^{m-1}})P^{-2}] \dots \\ &\quad \dots [P^{m-1}(H_2 \otimes I_{2^{m-1}})P^{-m+1}] \\ &= [(H_2 \otimes I_{2^{m-1}})P][(H_2 \otimes I_{2^{m-1}})P] \dots [(H_2 \otimes I_{2^{m-1}})P^{-m+1}] \\ &= [(H_2 \otimes I_{2^{m-1}})P]^{m-1}[(H_2 \otimes I_{2^{m-1}})P^{-m+1}]. \end{aligned} \quad (118)$$

However, from Theorem 18 we have that

$$P^m = I$$

and therefore

$$P \cdot P^{m-1} = I$$

or

$$P = P^{-m+1}$$

(since P is an invertible permutation matrix).

Hence,

$$\begin{aligned} H_{2^m} &= [(H_2 \otimes I_{2^{m-1}})P]^m \\ &:= B^m, \end{aligned}$$

where $B = (H_2 \otimes I_{2^{m-1}})P$, and since B is a permutation of the columns of the matrix $H_2 \otimes I_{2^{m-1}}$ which has exactly two nonzero entries in each row we conclude that B has precisely two nonzero entries in each row as well.

Note that B involves first permuting the incoming data and then adding and subtracting these permuted values in pairs. The permutation can be accomplished with a prewired network and the additions can be performed in parallel for the different pairs of data. Since a single multiplication by $(H_2 \otimes I_{2^{m-1}})$ requires $n = 2^m$ additions and this has to be performed m times, we have a total of $m \cdot 2^m = n \cdot m = n \cdot \log n$ additions which is a substantial savings compared to the $n(n-1)$ required when $H_{2^m} \mathbf{x}^T$ is performed directly.

2.5.2 Decoding algorithm for the $R(1, m)$ code based on FWT.

Consider again the $R(1, 3)$ Reed-Muller code, and suppose that 0's have been replaced by -1 in the code vectors. Then the code can be completely specified by the Hadamard matrix H_8 , where

$$H_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

and as it has already been mentioned,

$$\begin{aligned} H_8 &= H_2 \otimes H_2 \otimes H_2 \\ &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \end{aligned} \quad (119)$$

The fact that H_8 can be written in the form (119), enables us to express any one of its rows in the form

$$v = v_1 \otimes v_2 \otimes v_3, \quad (120)$$

where v is a row of H_8 and $v_i = (v_{i1}, v_{i2})$ with $v_{i1} = 1$, $i = 1, 2$, $v_{31} = \pm 1$, $v_{32} = \pm 1$, $i = 1, 2, 3$.

If we now allow $v_{31} = \pm 1$, then (120) provides all the codewords of $R(1, 3)$.

In general, if v is a codeword in $R(1, m)$ with 0's been replaced by -1, then it can be shown ([14]) that v can be written as a Kronecker product of m cofactor vectors of length 2 consisting of ± 1 .

In particular, if we let $v^{(0)}$ be the first row of the 2×2 Hadamard matrix H_2 and $v^{(1)}$ be the second row of this Hadamard matrix H_2 , i.e.

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} v^{(0)} \\ v^{(1)} \end{pmatrix} \quad (121)$$

and $h_{b_{m-1} \dots b_1 b_0}$ is the $b_{m-1} \dots b_1 b_0$ row of H_{2^m} (where $b_{m-1} \dots b_1 b_0$ is the binary representation of the index of the row ($b_i = 0, 1$)), then

$$h_{b_{m-1} \dots b_1 b_0} = v_1 \otimes v_2 \otimes \dots \otimes v_m,$$

where

$$v_i = \begin{cases} (1, 1), & \text{if } b_{i-1} = 0; \\ (1, -1), & \text{if } b_{i-1} = 1. \end{cases} \quad (122)$$

In order to include all the codewords of $R(1, m)$, we have to allow v_m to be $(-1, \pm 1)$, as well.

Therefore, a codeword $v \in R(1, m)$ can be written in the form

$$v = v_1 \otimes v_2 \otimes \dots \otimes v_m,$$

where

$$v_i = (v_{i1}, v_{i2}),$$

$$v_{i1} = 1, i = 1, 2, \dots, m-1; v_{m1}, v_{i2} = \pm 1, i = 1, 2, \dots, m.$$

We are now ready to state the following [2]

FWT-based decoding algorithm for $R(1, m)$.

Suppose that the vector w is received.

Step 1: Let $i = m$.

Step 2: Replace w with $((I_{2^{i-1}} \otimes H_2 \otimes I_{2^{m-i}}) \cdot w^T)^T$.

Step 3: Let $i = i - 1$. If $i \geq 1$, go to *Step 2*.

Step 4: In the resultant vector, determine the maximum element in modulus. The binary representation of the index of this element, with the 0's replaced by -1, corresponds to the vector

$$v_{(2)} = (v_{12}, v_{22}, \dots, v_{m2}),$$

and the sign of this element is the sign of v_{m1} .

Theorem 21 : The above algorithm correctly decodes first order Reed-Muller codes and its complexity is $O(n \log n)$.

Proof : At the second step the algorithm multiplies the received vector w by the factors of the factorization of H_{2^m} which we have already obtained in Theorem 17. This operation is equivalent to multiplying the received vector w by the parity check matrix of $R(1, m)$.

After the multiplication has been completed, the syndrome of w has been determined. The maximum element in modulus in the syndrome determines the row of H_{2^m} with the most elements in common with w . The binary representation of the index of this element determines (as it has already been mentioned) the vector $v_{(2)}$, while the sign of the number determines the sign of v_{m1} (since the received vector w can also be $1 + h$, where h is a row of H_{2^m} in which case $v_{m1} = -1$).

The algorithm requires $n \log n$ addition operations, $(n - 1)$ comparison operations, and n operations of taking the modulus. It follows that its complexity is $O(n \log n)$.

2.5.3 A fast decoding algorithm for first-order Reed-Muller codes.

A remarkable observation in the algorithm given in the preceding section which has as a consequence the reduction of its complexity is that the calculation of all the components in the resultant vector is a superfluity.

So suppose again that the vector w is received and consider the following [2]

New decoding algorithm for the $R(1, m)$ Reed-Muller codes.

Step 1 : Let $i := n$, $j := 1$.

Step 2 : Represent the received vector w of dimension i in the form of a $2 \times (i/2)$ matrix S having as first row the odd-numbered components of w and as second row the even-numbered components of w .

Step 3 : Multiply matrix H_2 by the matrix S obtained at Step 2 and denote the resulting $2 \times (i/2)$ matrix by $\tilde{S}$.

Step 4 : Add the elements of the columns of $\tilde{S}$ taken in modulus and denote the 2-dimensional resulting column vector by s .

Step 5 : Find the greater element in s whose number defines $v_{j,2}$; i.e. $v_{j,2} = 1$ if the greater element is the first element, or $v_{j,2} = 0$ if it is the second element.

Step 6 : If $j < m$, then the row of $\tilde{S}$ with number equal to the number of the maximum element of s is rewritten in place of the first $(i/2)$ positions of w . Set $j := j + 1$, $i := i/2$ and go to Step 2.

Step 7 : If $j = m$, determine the number of the maximum element of $\tilde{S}$ in modulus which determines $v_{m,2}$; the sign of this element determines the sign of $v_{m,1}$.

The validity as well as the complexity of the above algorithm are given in the following [2]

Theorem 22 : The algorithm described above ensures decoding of the $R(1, m)$ Reed-Muller code within the limits of the code distance with linear complexity.

Proof : For the first part of the Theorem, note that in a noiseless channel (with no distortion), the vector s has the form $(0, n)$ or $(n, 0)$, $\forall j$; the components of s interchange depending on the value of $v_{i,2}$, $i = 1, 2, \dots, m$.

Note also that an error in w alters the element of the corresponding column of $\tilde{S}$ by ± 2 and correspondingly the component of s (since it corresponds to subtraction

instead of addition, or vice versa).

Suppose that $t < n/4$ errors are present in the received word. Then any component of s can vary by at most $2t < \frac{2n}{4} = \frac{n}{2}$ and the number of the maximum components of s is the same.

As far as the complexity concerns, at the main step of the algorithm corresponding to a vector w of length i , $(2i - 2)$ addition (and subtraction) operations, one comparison and i operations taking the modulus are required. At the final step one comparison-type operation for determining the sign of the number is also needed.

Hence, the number of additions is given by

$$\sum_{j=1}^m (2 \frac{n}{2^{j-1}} - 2) = 4n - 2 \log n - 4.$$

The number of comparisons is

$$m + 1 = \log n + 1$$

and the number of operations taking the modulus is

$$n \sum_{j=1}^m \frac{1}{2^{j-1}} = 2n - 2.$$

Hence, the complexity of the above algorithm is $O(n)$.

The advantages of the new decoding algorithm for the $R(1, m)$ Reed-Muller codes are now immediate. Its linear complexity yields a significant gain in operating speed and since the algorithm requires operations over the real numbers, software decoding can be easily implemented.

Compared to the FWT-based algorithm, the new decoding algorithm has the additional advantage that the delay in decoding an information symbol with a smaller value index is less than the delay in decoding subsequent information symbols and a simulation (of the algorithm) indicated that it is able to correct a substantial number of errors of multiplicity greater than that guaranteed by the code distance [2].

Example : Consider the $R(1, 4)$ Reed-Muller code ($d = 8$).

Let

$$v = 11 - 1 - 111 - 1 - 111 - 1 - 111 - 1 - 1$$

be the transmitted codeword with 0's replaced by -1 and suppose that

$$w = 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ -1 \ 1 \ 1 \ 1 \ -1 \ 1 \ 1 \ 1 \ -1 \ -1$$

was received instead.

$$i = 2^4 = 16$$

$$j = 1$$

$$S := \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 \end{pmatrix}$$

$$\tilde{S} := H_2 S = \begin{pmatrix} 2 & -2 & 2 & 0 & 2 & -2 & 2 & -2 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$s := \begin{pmatrix} 14 \\ 2 \end{pmatrix}$$

$$14 > 2 \Rightarrow v_{12} := 1$$

$$k = 1$$

$$w := (2 \ -2 \ 2 \ 0 \ 2 \ -2 \ 2 \ -2)$$

$$j := 2$$

$$i := 8$$

$$S := \begin{pmatrix} 2 & 2 & 2 & 2 \\ -2 & 0 & -2 & -2 \end{pmatrix}$$

$$\tilde{S} := H_2 S = \begin{pmatrix} 0 & 2 & 0 & 0 \\ 4 & 2 & 4 & 4 \end{pmatrix}$$

$$s := \begin{pmatrix} 2 \\ 14 \end{pmatrix}$$

$$2 < 14 \Rightarrow v_{22} = -1$$

$$k := 2$$

$$w := (4 \ 2 \ 4 \ 4)$$

$$j := 3$$

$$i := 4$$

$$S := \begin{pmatrix} 4 & 4 \\ 2 & 4 \end{pmatrix}$$

$$\tilde{S} = H_2 S = \begin{pmatrix} 6 & 8 \\ 2 & 0 \end{pmatrix}$$

$$s := \begin{pmatrix} 14 \\ 2 \end{pmatrix}$$

$$14 > 2 \Rightarrow v_{32} := 1$$

$$k := 1$$

$$w := (6 \ 8)$$

$$j = 4$$

$$j := 2$$

$$S := \begin{pmatrix} 6 \\ 8 \end{pmatrix}$$

$$\tilde{S} = H_2 S = \begin{pmatrix} 14 \\ -2 \end{pmatrix}$$

$$s := \begin{pmatrix} 14 \\ 2 \end{pmatrix}$$

$$14 > 4 \Rightarrow v_{42} := 1$$

$$\text{sgn}(v_{41}) = \text{sgn}(14) = +1 \Rightarrow v_{41} := 1$$

Hence, the transmitted codeword was :

$$\begin{aligned} v &= (v_{11} \ v_{12}) \otimes (v_{21} \ v_{22}) \otimes (v_{31} \ v_{32}) \otimes (v_{41} \ v_{42}) \\ &= (1 \ 1) \otimes (1 \ -1) \otimes (1 \ 1) \otimes (1 \ 1) \\ &= 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \ -1 \ -1 \ 1 \ 1 \end{aligned}$$

as expected.

References

- [1] Hadamard J. : *Resolution d' une question relative aux déterminants*, Bull.Sci. Math: 17(1893) pp. 240-248.
- [2] Litsyn S. N., Shekhovtsov O. I. : *Fast decoding algorithm for first-order Reed-Muller codes*, Probl. Inf. Transm. (USA), April-June 1983; 19(2), pp. 87-91.
- [3] MacWilliams F. J., Sloane N. J. A. : *The Theory of Error-Correcting Codes*, New York: North-Holland, 1977.
- [4] Manley H. J., Mattson H. F. Jr., Schatz J. R. : *Some Applications of Good's Theorem*, I.E.E.E. Trans. Info. Theory 26(1980), pp. 475-476.
- [5] Massey J. L. : *Threshold decoding*, M.I.T. press, Cambridge, MA., 1963.
- [6] Muller D. E. : *Application of Boolean Algebra to Switching Circuit design and to error detection*, I.E.E.E. Trans. Computers, 3 (1954), pp. 6-12.
- [7] Peterson W. W., Weldon E. J. Jr. : *Error-Correcting Codes*, second edition, M.I.T. press, Cambridge, MA., 1972.
- [8] Posner E. C. : *Combinatorial structures in Planetary Reconnaissance*, in *Error Correcting Codes*, H. B. Mann Ed., New York, John Wiley, pp. 149-162, 1969
- [9] Reed I. S. : *A class of multiple-error-correcting codes and the decoding scheme*, I.E.E.E. Trans. Info. Theory 4(1956), pp. 38-49.
- [10] Rudolph L. D., Hartmann C. R. P. : *Decoding by sequential code reduction*, I.E.E.E. Trans. Info. Theory 19(1973), pp. 549-555.
- [11] Rushforth C. K. : *Fast Fourier-Hadamard decoding of orthogonal codes*, Info and Control 15(1969), pp. 33-37.
- [12] Sylvester J. J. : *Thoughts of inverse orthogonal matrices, simultaneous sign successions and tessellated pavements in two or more colors, with applications to Newton's rule, ornament tile-work, and the theory of Numbers*, Phil Mag 34(1867), pp. 461-475.
- [13] Tokiwa K., Sugimura T., et al. : *New decoding algorithm for Reed-Muller codes*, I.E.E.E. Trans. Info. Theory (USA) Sept. 1982, 28(5,pt.1), pp. 779-787.
- [14] Welch J. E. Jr. and Guinn D. F. : *The Fast Fourier Hadamard Transform and*

2

of/de

2

MICRO

its use in signal representation and classification, EASCON '68 record *Electronics and Aerospace Systems convention*, 1968 pp. 561-573.

- [15] Weldon E. J., Jr. : *Some results on majority-logic decoding*, in *Error Correcting Codes*, H. B. Mann Ed., New York, John Wiley, pp. 149-162 (1969).

END

23/12.87

FIN