Reed-Muller Codes

Prateek Sharma

April 12, 2010

Abstract

This Seminar Report gives an introduction to the Reed-Muller family of Error Correcting Codes. The construction and various interpretations of the codewords is discussed. Reed Muller codes are primarily used because of their large error correcting ability and easy decoding - hence a survey of the decoding techniques and algorithms is presented. Some applications of the code are also presented. Some applications of the code to error correction and other areas are listed.

1 Introduction

Reed-Muller ($\mathcal{R}$) codes are amongst the oldest and most well-known codes. They were discovered by D. E. Muller and I. S. Reed in 1954. [**?**] [**?**] Reed-Muller codes have many interesting properties - they can be defined recursively, and are an infinite family of codes. The recursive definitions are studied in section **??**. Although they become weaker as their length increases, they are often used as building blocks in other codes. One of the major advantages of Reed-Muller codes is their relative simplicity to encode and decode messages. Several decoding algorithms are presented in 8. Reed-Muller codes, like many other codes, have tight links to design theory; we briefly investigate the link between Reed-Muller codes and affine geometries in **??**. Most famously used to transmit live images of mars in the Mariner-9 mission in 1972, [**?**], [7] , they have applications in a wide variety of areas- some of which are presented in 10.

1.1 Notation

F_q denotes a finite field of order q. Throughout, unless otherwise explicity stated, $q = 2$ and therefore $F = \{0, 1\}$, with the addition operation being equivalent to the binary exclusive-or (xor), and the multiplication operation being the binary *and*.

A vector field over F,(with the length of the vector being n) will be denoted by F^n.

We use a string of length n with elements in F_2 to write a vector in the vector space F_2^n . For example, if we have the vector $\mathbf{v} = (1, 0, 1, 1, 0, 1, 0, 1) \in F_2^8$, we simply write $\mathbf{v}$ as 10110101.

We shall also deal with boolean functions, and shall denote the boolean xor by $+$, instead of the customary $\oplus$ to maintain consistency with the addition operation in F_2.

Three parameters of linear error-correcting codes are it's length, dimension, and minimum distance, denoted by the triplet $[n, k, d]$. The elements of a code C are called the *codewords*. Codewords are represented as vectors of length n.

The distance between two vectors $\mathbf{u}$ and $\mathbf{v}$ which we are concerned about is the *Hamming-Distance* which is the number of positions in which 2 vectors differ. Analogously, the Hamming weight of a vector is defined as:

$$wt(x) = d(x, 0) = \begin{array}{ll} 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{array} \tag{1}$$

We will use $d(x, y) = wt(x-y)$ and $wt(x+y) = wt(x)+wt(y)-2wt(x*y)$, where $x*y = (x_1y_1, x_2y_2, \ldots, x_ny_n)$.

2 Construction of the Code

The simplest construction of Reed-Muller codes is by using boolean functions. Let $(x_1, x_2, \ldots, x_m) \in F^m$ be the set of binary m-tuples, and let $\mathbf{x} \equiv (x_1, x_2, \ldots, x_m)$. Consider a boolean function f, $f : F^{2^m} \rightarrow \{0, 1\}$. Thus $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_m)$ takes a binary m-tuple and returns either 0 or 1 .

$\mathbf{x}$ can take 2^m values. For each value of $\mathbf{x}$ we compute $f(\mathbf{x})$. This is the familiar and ubiquitous *truth-table* wherein a boolean function is evaluated for all possible inputs. In the example below, x_1, x_2, x_3 are rows and $f(x_1, x_2, x_3)$ is computed and represented as another row.

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
f	0	0	0	1	1	0	0	0

Since $(x_1, x_2, \ldots, x_m)$ can take 2^m values, $\mathbf{f}$ is a $n = 2^m$ length vector over F_2. Since there are 2^{2m} such boolean functions possible, this gives us a collection of 2^{2m} vectors, each of length 2^m.

The Reed-Muller codes are particular subsets of this collection as described below.

f can be written in the disjunctive normal form. Using logical operations, f can be represented as a function of the $x_1, x_2, \ldots, x_m$. In the above example, for instance, $f = x_1 \vee x_2 \vee x_3$.... We can represent any boolean function in the disjunctive Normal form (with or replaced by xor) [**?**], for instance in the above example $f =$

If there f is an $m - ary$ function, it can take the 2^m input values. Hence $\mathbf{f}$ and $x_1, x_2, \ldots_m$ have length 2^m. Since $\mathbf{f}$ is also a linear combination, it follows that the length of $x_1, x_2, ldots, x_m$ is 2^m.

Let the set of boolean monomials be:

$$M = \{1, x_1, x_2, \ldots, x_m, x_1x_2, \ldots, x_{m-1}x_m, x_1x_2x_3, \ldots, x_1x_2 \ldots x_m\}$$

Where $\mathbf{1}$ is the all ones vector.

There are

$$1 + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{m} = 2^m$$

distinct monomials.

A Boolean polynomial is a linear combination (with coefficients in F_2) of boolean monomials. The degree of the boolean polynomial is the degree of the its highest term (monomial).

$$f = 1 + a_1x_1 + a_2x_2 + \ldots + a_mx_m + a_{12}x_1x_2 + \ldots + a_{12\ldots r}x_1x_2 \ldots x_r + \ldots$$

The polynomials (we will drop the term boolean from here on) of degree 1 are simply the linear combination of m vectors.

$$\mathbf{1} + a_1x_1 + a_2x_2 + \ldots + a_mx_m \tag{2}$$

The above

With this, we can now define Reed-Muller codes of length m and order r as :

$$\mathcal{R}(r, m) = \{\mathbf{f} : degree(f(x_1, \ldots, x_m)) = r\} \tag{3}$$

$\mathcal{R}(r, m)$ is a linear code.

Proof. As defined, $\mathcal{R}(r, m)$ is a linear combination of the monomials of degree $\leq r$. The vectors $x_1, x_2, \ldots, x_m$ are clearly linearly independent. Monomials of higher degree are composed by component-wise multiplication of the above vectors, and hence are also linearly independent. Thus the boolean monomial set M consists of linearly independent vectors, and thus any subset of that is also independent. □

From the above proof: The monomials of degree $\leq r$ thus form a basis for $\mathcal{R}(r, m)$

Let $G(r, m)$ denote the *Generator Matrix* of $\mathcal{R}(r, m)$. It consists of all the monomials from set M of degree $\leq r$.

$$G(r,m) = \begin{pmatrix} \mathbf{1} \\ \mathbf{x_1} \\ \mathbf{x_2} \\ \vdots \\ \mathbf{x_m} \\ \mathbf{x_1 x_2} \\ \vdots \\ \mathbf{x_1 x_2 \dots x_r} \end{pmatrix} \tag{4}$$

$\mathcal{R}(0, m)$ is the repetition code. At the other extreme $\mathcal{R}(m, m)$ is a code consisting of all possible binary sequences of length m.

2.1 clarification

In what follows, we frequently switch back and forth between B(r, v1 , . . . , vm) and R(r, m). Doing so in the most explicit manner would make the reasoning a lot harder to read, and for this reason we decided to treat codewords and Boolean functions as interchangeable. In fact, boolean logic using xor is also called reed-muller logic! [**?**]

Properties: Length, $n = 2^m$ Minimum Distance, $d = 2^{m-r}$ Dimension, $k = 1 + ..$

The rate of an $[n, k]$ code is dened as k/n. (Thinking of the code as having nk check digits and k message digits.

2.2 Code properties

We have seen that the first order Reed-Muller code $\mathcal{R}(1, m)$ consists of all vectors $u_0\mathbf{1} + \sum_{i=1}^{m} u_i \mathbf{v_i}$ with $u_i = 0$ or 1. Define the *orthogonal code* $\prime_m$ to be the $[2m, m, 2m1]$ code consisting of the vectors $\sum_{i=1}^{m} u_i \mathbf{v_i}$ Then

$$\mathcal{R}(1, m) = \mathcal{O}_m(\mathbf{1} + \mathcal{O}_m)$$

Proof. The definition of the orthogonal codes is exactly similar to the corresponding Reed-Muller codes, with only the complements of the codewords missing. □

Uniqueness: Any linear code with parameters $[2^m, m + 1, 2^m]$ is equivalent to the first order Reed-Muller code.

*from [**?**]. DO THIS PROOF.* □

Run-length properties at [**?**]

2.3 Recursive Formulation

Theorem 1. $\mathcal{R}(r + 1, m + 1) = \{\mathbf{u}|\mathbf{u} + \mathbf{v} : \mathbf{u} \in \mathcal{R}(r + 1, m), \mathbf{v} \in \mathcal{R}(r, m)\}$

This is known as the concatenation construction of codes, with | denoting the concatenation.

Proof. We use the boolean logic definition of the codewords. Let $\mathbf{f} \in \mathcal{R}(r + 1, m + 1)$. $\mathbf{f}$ can be written as

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

Where $\mathbf{g} \in \mathcal{R}(r+1,m)$ and $\mathbf{h} \in \mathcal{R}(r,m)$. Consider the associated vectors $\mathbf{f}, \mathbf{g}', \mathbf{h}'$ as polynomials over $v_1, \ldots, v_{m+1}$. Then, $\mathbf{g}' = (\mathbf{g}|\mathbf{g})$ and $\mathbf{h}' = (\mathbf{0}|\mathbf{h})$. [Problem7, [7].] Thus

$$\mathbf{f} = \mathbf{g}|\mathbf{g} + \mathbf{0}|\mathbf{h}$$

Thus we have shown that the vector in $\mathcal{R}(r+1,m+1)$ can be decomposed into vectors of $\mathcal{R}(r+1,m)$ and $\mathcal{R}(r,m)$. □

The generator matrix version of the above theorem is

$$G(r+1,m+1) = \begin{pmatrix} G(r+1,m) & G(r+1,m) \\ 0 & G(r,m) \end{pmatrix} \tag{5}$$

We can also choose the vectors of the generator matrix in a systematic way

$$G(1,m+1) = \begin{pmatrix} G(1,m) & G(1,m) \\ 0 & 1 \end{pmatrix} \tag{6}$$

This way, the columns of $G(1,m)$ are binary representations of numbers from $1 to 2*^m$ in descending order.

Another recursive construction: Let

$$R_1 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Then, $R_{n+1} = \begin{pmatrix} R_n & R_n \\ R_n & negR_n \end{pmatrix}$

2.4 Properties

Recursive definition naturally is very helpful for proving certain properties, particularly since it enables the use of induction.

The Reed-Muller codes are a nested family of codes - the codes of higher order contain those of the lower order.

Theorem 2.

$$\mathcal{R}(r,m) \subset \mathcal{R}(t,m) \text{ if } 0 \leq r \leq t \leq m$$

Proof. By Induction. Trivially true for $m = 1$. Let $\mathcal{R}(k,m-1) \subset \mathcal{R}(l,m-1)$ for all $0 \leq k \leq l < m$. Let $0 < i \leq j < m$. By the recursive definition, we get:

$$\mathcal{R}(i,m) = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}|\mathbf{u} \in \mathcal{R}(i,m-1), \mathbf{v} \in \mathcal{R}(i-1,m-1)\}$$

Induction hypothesis gives :

$$\subset \{(\mathbf{u}, \mathbf{u}+\mathbf{v}|\mathbf{u} \in \mathcal{R}(j,m-1), \mathbf{v} \in \mathcal{R}(j-1,m-1)\} = \mathcal{R}(j,m)$$

□

$$dim(\mathcal{R}(r,m)) = dim(\mathcal{R}(r,m-1)) + dim(\mathcal{R}(r-1,m-1)) \tag{7}$$

Theorem 3.

$$\mathcal{R}(m-2,m) \text{ is the extended binary hamming code}$$

Proof. □

We prove a general theorem

Theorem 4. *Let C_i be an $[n, k_i, d_i]$ code. Then the concatenated code defined by*

$$C = \{(\mathbf{u}, \mathbf{u}+\mathbf{v}) | \mathbf{u} \in C_1, \mathbf{v} \in C_2\}$$

has the parameters $[2n, k_1 + k_2, min2d_1, d_2]$ linear code.

Proof. Length: Clearly, since length of C_1 and C_2 is n each, concatenating produces vectors of length $2n$.

Dimension: The number of words in C is product of number of words in C_1 and C_2. because $(c_1, c_2) \to (c_1, c_1 + c_2)$ is a bijection. Thus, the dimension is

$$k = k_1 + k_2$$

Distance: We can split this into cases, depending on whether $c_1 = \mathbf{0}$ or $c_2 = \mathbf{0}$.

Case: $\mathbf{c_2} = \mathbf{0}$

$$wt((c_1, c_1 + c_2)) = wt(c_1, c_1) = 2wt(c_1) = 2d_1$$

Case: $\mathbf{c_2} \neq \mathbf{0} wt((c_1, c_1 + c_2)) = wt(c_1) + wt(c_1 + c_2) \geq wt(c_1) + wt(c_2) - wt(c_1) = wt(c_2) = d_2$

Thus, the minimum distance of the code, which is equal to the weight of the minimum weight vector is $min2d_1, d_2$

□

Theorem 5. *Minimum distance, $d = 2^{m-r}$*

Proof. Let $\mathbf{f} \in \mathcal{R}(r+1, m)$ and $\mathbf{g} \in \mathcal{R}(r, m)$. By theorem **??** . Let $d_1 = 2^{m-r-1}$ and $d_2 = 2^{m-r}$. By 4, $d = min\{2d_1, d_2\} = 2^{m-r}$.

The standard proof of this by induction is found everywhere - see [7], [**?**]. □

Lemma 1. *Every codeword in $\mathcal{R}(1, m)$ (except $\mathbf{1}, \mathbf{0}$) has weight 2^{m-1}*

Proof. This can also be proved by using the randomization lemma for boolean functions as done in [7]. However, if we notice that

$$\mathcal{R}(1, m) = \{\mathbf{u} | \mathbf{u} : \mathbf{u} \in \mathcal{R}(1, m-1)\}\{(\mathbf{u}, \vec{1} + \mathbf{u}), \mathbf{u} \in \mathcal{R}(1, m-1)\}$$

We use induction on m to complete the proof. □

Theorem 6.

$$\mathcal{R}(m-r-1, m) = \mathcal{R}(r, m)^{\perp}$$

Proof. Let $\mathbf{a} \in \mathcal{R}(m-r-1, m)$, and $\mathbf{b} \in \mathcal{R}(r, m)$. Then a and b are polynomials of degree $m-r-1$ and r respectively. $\mathbf{ab} \in \mathcal{R}(m-2, m)$, and since $\mathcal{R}(m-2, m)$ contains all the even weight vectors, $a \cdot b \equiv 0 (mod 2)$. Thus, $\mathcal{R}(m-r-1, m) \subset \mathcal{R}(r, m)^{\perp}$. But by considering the dimensions we get

$$dim(\mathcal{R}(m-r-1, m)) + dim(\mathcal{R}(r, m) = 2^m = n$$

This implies the relation. □

Theorem 7. *The dual code $\mathcal{R}(1, m)^{\perp}$ is the extended binary Hamming code $H(m)$*

Proof. TODO. this is important. From singa book □

3 $\mathcal{R}(1,3)$

1	1	1	1	1	1	1	1	1
v_1	0	0	0	0	1	1	1	1
v_2	0	0	1	1	0	0	1	1
v_3	0	1	0	1	0	1	0	1
$v_1 + v_2$	0	0	1	1	1	1	0	0
$v_1 + v_3$	0	1	1	0	0	1	1	0
$v_2 + v_3$	0	1	0	1	1	0	1	0
$v_1 + v2 + v_3$	0	1	1	0	1	0	0	1
0	0	0	0	0	0	0	0	0
$1 + v_1$	1	1	1	1	0	0	0	0
$1 + v_2$	1	1	0	0	1	1	0	0
$1 + v_3$	1	0	1	0	1	0	1	0
$1 + v_1 + v2$	1	1	0	0	0	0	1	1
$1 + v_1 + v3$	1	0	0	1	1	0	0	1
$1 + v_2 + v_3$	1	0	1	0	0	1	0	1
$1 + v_1 + v_2 + v_3$	1	0	0	1	0	1	1	0

(8)

4 G(2,3)

1	1	1	1	1	1	1	1	1
v_1	0	0	0	0	1	1	1	1
v_2	0	0	1	1	0	0	1	1
v_3	0	1	0	1	0	1	0	1
$v_1 \cdot v_2$	0	0	0	0	0	0	1	1
$v1 \cdot v_3$	0	0	0	0	0	1	0	1
$v2 \cdot v_3$	0	0	0	1	0	0	0	1

(9)

5 Bounds

$A_q(n,d)$ is the largest integer M such that there exists a q-ary (n,M,d) code where M is the number of codewords. A q-ary (n,M,d) code C is called optimal if $M = A_q(n,d)$.

The Reed-Muller codes satisfy the *Plotkin bound*.

[Plotkin Bound] If $C = [n,k,d]$ code,

$$d \leq \frac{n2^{k-1}}{2^k - 1}$$

Proof. Counting in two ways:

$$D = \sum_{\mathbf{u} \in C} \sum_{\mathbf{v} \in C} d(u,v)$$

Since $d(u,v) \geq d$, $D \geq 2^k(2^k - 1)d$.

Let D_{1i} denote the number of codewords with a 1 in their i-th component.

$$D = \sum_{i \in n} \sum_{j=\{0,1\}} D_{ji}(2^k - D_{ji})$$

Since $\sum D_{ji} = 2^k$, we have:

$$D = n2^{2k} - \sum_{i \in n} \sum_{j=\{0,1\}} D_{ji}^2$$

$$\sum_{j=0,1} D_{ji}^2 \geq 2^{2k-1}, \qquad i \in n$$

Combining the upper and the lower bounds, we obtain:

$$2^k(2^k - 1)d \leq D \leq n2$$

□

It can be verified that $\mathcal{R}(1, m)$ codes satisfy this bound. This can also be shown by using the fact proved in **??** that $\mathcal{R}(1, m)$ are equidistant codes, and using that result in the proof of the plotkin bound.

5.1 Geometries

Some properties of the Reed-Muller codes can be best stated using finite geometries. A comprehensive source of the relation between the code and finite geometry is [**?**] and [7].

Lemma 2. $\mathbf{f} \in \mathcal{R}(r, m)$ *is the incidence vector of the set* S*, then* $\mathbf{hf} \in \mathcal{R}(r + 1, m)$ *is the incidence vector of* $S \cup H$

Theorem 8. *Let* f *be a minimum weight codeword of* $\mathcal{R}(r, m)$*, and* $f = \chi(S)$*. Then* S *is an* $(m - r)$ *dimensional flat in* $AG(m, 2)$*.*

*From [**?**].* Let H be a hyperplane in $AG(m, 2)$ and let H' be the parallel hyperplane.

By the previous lemma, $S \cap H$ and $S \cap H'$ are in $\mathcal{R}(r + 1, m)$ and thus contain 0 or $\geq 2^{m-r-1}$ points. (Minimum weight , **??**). But since $|S| = 2^{m-r} = |S \cap H| + |S \cap H'|$, we have

$$|SH| = 0, 2^{m-r-1}, \text{ or } 2^{m-r}$$

Using the rothschild-vanlint theorem [**?**], we get that S is an $(m - r)$-dimensional flat in $AG(m, 2)$. □

The converse of this theorem also holds:

Theorem 9. *The incidence vector of any* $(m - r)$ *dimensional flat in* $AG(m, 2)$ *is a codeword in* $\mathcal{R}(r, m)$*.*

Proof. Every $(m - r)$ dimensional flat in $AG(m, 2)$ can be represented by r linearly independent linear equations of the form

$$\sum_{j \in m} a_{ij} c_j = b_i, \qquad i \in r$$

The elements of the flats are the vectors $c = (c_0, \ldots, c_m)$ satisfying these equations. We thus need to show that $c \in \mathcal{R}(r, m)$.

$$\sum_{j \in m} a_{ij} c_j + b_i + 1 = 1, \qquad i \in r$$

Since we are dealing with F_2, we can combine all the t equations to get

$$\prod_{i \in r} (\sum_{j \in m} a_{ij} c_j + b_i + 1) = 1$$

This is because if one of the original equations does not hold then one of the right hand sides must be zero, which makes the product in the second equation become zero. Conversely if, c is a solution, then each of the

equations must be satisfied since the product is one. The last equation is also a polynomial of degree r, we get that $c \in \mathcal{R}(r, m)$.

Since an affine space of rank $(m - r)$ contains 2^{m-r} points, we get that the incidence vector is the minimum weight vector in $\mathcal{R}(r, m)$.

□

Theorem 10. *The number of codewords of minimum weight in $\mathcal{R}(r, m)$ is*

$$A_2^{m-r} = 2^r \prod_{i=0}^{m-r-1} \frac{2^{m-i} - 1}{2^{m-r-i} - 1}$$

A lemma that will help us in decoding:

Lemma 3. *In an affine geometry $AG(m, 2)$, each r-flat is constained in exactly $2^{m-r} - 1$ flats of dimension $r + 1$. These $r + 1$-flats intersect pairwise exactly in the elements of the given r-flat.*

Proof. Let V be an r-dimensional vector subspace of F_2^m. $W = (V, w)$ is an $r+1$ dimensional space containing V. We can choose w in $2^{r+1} - 2^r$ different ways such that the same vector space W is obtained. Thus there are $\frac{2^m - 2^r}{2^{r+1} - 2^r} = 2^{m-r} - 1$ linear subspaces W of dimension $r + 1$ containing V. Any two of these subspaces are going to intersect exactly in V.

$V + x$ is a contained in $W + x$. Also the collection of all these $W + x$ flats intersect pairwise in exactly $V + x$. □

6 Weight distribution

For the codes of order $r \geq 2$, determining the complete weight distribution is hard. (impossible). For $r = 1$, as shown **??**, the weight of all words is 2^{m-1}.

Considerable work has been done to establish the distribution for $\mathcal{R}(2, m)$. We state the classic results (for proofs see [7]) :

[6]

There are at least$2^{mrr(r1)}$.minimum weight codewords in $\mathcal{R}(r, m)$.

(10)

7 cyclic decoding

In addition, it is next shown that a cyclic RM (r, m) code is simpler to decode. The argument is as follows. It is shown in Chapter 3 that in a cyclic code C , if (v1 , v2 , . . . , vn) is a code word of C , then its cyclic shift (vn , v1 , . . . , vn 1) is also a code word of C . Therefore, once a particular position can be corrected with ML decoding, all the other positions can also be corrected with the same algorithm (or hardware circuit) by cyclically shifting received code words, until all n positions have been tried.

$\mathcal{R}(,)$

8 Decoding

As mentioned earlier, the Reed-Muller family of codes came into prominence because of the ease of decoding. In this section we present a variety of decoding algorithms. Several very efficient algorithms have been known specially for the Reed-Muller codes of the first order.

8.1 Majority Logic Decoding

Majority Logic Decoding is the oldest and simplest method to decode the Reed-Muller codes. [1] Some terminology is needed for a clearer understanding: Let C be an arbitrary code. Then the *parity-check* is simply a codeword in the orthogonal code $C^{\perp}$. Define the *support-vector* of a vector $\vec{v}$ as the coordinates where it is non-zero.For example, the support-vector of 01010100 is $(2, 4, 6)$. Suppose we are given J parity checks and a coordinate position, i, such that the intersection of all the J supports is precisely $\{i\}$. The J parity checks essentially 'vote' for the position i , and the result of the majority for the parity of i is the value. A collection of such parity checks described above is said to be *orthogonal* to i. If each of the coordinates of the code has a collection of J parity checks orthogonal to it, then the code is capable of correcting up-to $J/2$ errors. Another definition of orthogonality is: A set of parity check equations is called orthogonal on the i coordinate if x_i appears in each equation and no other x_j appears more than once in the set.

If there are J parity checks on every co-ordinate, the code can correct $J/2$ errors.

Proof. Let m_i be the message bit at coordinate i. x_i is the received bit. Let $m_i = 0$ Since there are J orthogonal parity equations voting for x_i, if there are fewer than $J/2$ errors, then the result is unchanged. Therefore atleast $J/2$ equations are 0 and thus $x_i = 0$. If $m_i = 1$ Still less than $J/2$ equations are affected, and we get the right result. □

Note: Ties can be broken in any way consistently. Either 0, etc.

The number of errors that can be corrected by one-step majority logic decoding, E_1 is at most $n - 1/2(d' - 1)$ where d' is the minimum distance of the dual code.

Proof: The dual code gives the parity check equations. Consider the 1st coordinate, then the parity checks orthogonal will have the form

$$111100000100011111$$

Since there are J orthogonal equations there will be J such vectors. Therefore by the definition of orthogonality the coordinate positions in the dual code vectors cannot overlap, hence $J \leq n - 1/(d' - 1)$. We get the desired result by using the previous theorem.

8.2 L-step decoding

We can extend the idea of step-decoding using orthogonal parity checks by generalizing for parity check equations to be orthogonal on a set of coordinates.

Definition: A set of parity checks $S_1, S_2, \ldots$ is orthogonal on coordinates $i, j, ..$ if the sum $x_i + x_j + \ldots$ appears in each S but no other x_p appears more than once in the set.

The number of errors which can be corrected by an L-step majority decoding scheme, $E_l \leq n/d' - 1/2$

Proof. Let the $i - th$ parity check involve a_i coordinates (besides the l). Since these checks correspond to the codewords in the dual code, we have

$$l + a_i \geq d'$$

$a_i + a_j \geq d'(11)$

We also have

$$S = \sum_{i=1}^{J} a_i \leq n - l$$

Thus,

$$Jl + S \geq Jd'$$

(J-1)S $\geq \binom{J}{2} d'$Eliminating l and S, we get

$$J \leq 2n/d' - 1$$

□

8.3 Reed Decoding Algorithm

The Reed decoding algorithm is a majority logic decoding scheme for decoding Reed-Muller codes of any order.

Each codeword from $\mathcal{R}(r,m)$ is an incidence vector that denes a subspace in in which the nonzero coordinates of the codeword are points lying in the subspace. Based upon this geometry we can nd equations for orthogonal checksums as follows. To nd a set of orthogonal checksums that provide an estimate for the k-th order message bit $m_{i_1,i_2,\ldots,i_k} correspondingtothebasisvectorsv_{i_1}v_{i_2}\ldots v_{ik}$.

Let the corresponding subspace be S. Let T be the complementary subspace of S where $j_1 j_2 \ldots j_{m-k} = 1,2,ldots,m-i_1,i_2,\ldots,i_k$. The incidence vector of T is v. ¡todo¿

Define a *Translate* of a subspace with respect to a point to be the subspace obtained by adding the binary representation of the point to each element of the subspace.

If there are no errors, the message bit m_b is given by

$$m_b = \sum_{P \in U_i} x_P \quad i = 1,2,\ldots,2^{m-r}$$

Proof. A codeword can be written as

$$\mathbf{v} = \sum_{t=i_1 i_2 \ldots i_s} a_t \mathbf{x_{i_1} x_{i_2} \ldots x_{i_s}}$$

The sum is over all the subsets $\{i_1, i_2 ldots i s_s\}$ of $\{1 \ldots m\}$ of size atmost r. Summing up $\mathbf{v}$ over all the translates gives:

$$\sum_{P \in U_i} x_P = \sum_t a_t \sum_{P \in U_i} \mathbf{x_{i_1} x_{i_2} \ldots x_{i_s}}$$

Let $N(U_i,t)$ be the number of points in the intersection of U_i and the subspace with the incidence vector $\mathbf{x_{i_1} x_{i_2} \ldots x_{i_s}}$.

$$\sum_{P \in U_i} x_P = \sum_t a_t N(U_i,t)$$

Since the intersection of two subspaces is a subspace and the number of points in an affine subspace of rank r is 2^r. Let V and W intersect in the subspace

$$\mathbf{x_{j_1} x_{j_2} \ldots x_{j_{m-r}} x_{i_1} x_{i_2} \ldots x_{i_s}}$$

□

For a detailed algorithm refer to [**?**] and [**?**]

8.4 Geometrical

We can use the finite geometry construction of the codes to help in the decoding.

8.5 Hadamard Transforms

$of a binary vector \mathbf{v} is a vector with 0 replaced by 1 and 1 by$ -1.

Lemma 4. *The Orthogonal code $\mathcal{O}_m$ with real vectors is equivalent to the Hadamard matrix H_m.*

Proof. Let the elements of this real-vectorized orthogonal code be v.

Claim 1.

$$v_i, v_j \in \mathcal{R}(1,m) \implies v_i \cdot v_j = 0$$

Proof. We can prove using induction on the basis vectors. Since the code is a linear combination, the result follows. □

This is also the definition of the hadamard matrix. □

$$\mathcal{R}(1,m) = \begin{pmatrix} H_m \\ -H_m \end{pmatrix} \tag{12}$$

Keeping note that the code has been converted to the real-vector form, maximizing $F(v)H_m$ is equivalent to finding the least-distance neighbour of $\mathbf{v}$. This would entail doing a simple vector product with each row of the hadamard matrix. If the correlation is negative, the absolute value is considered since the correlation of the negation of the corresponding row of the hadamard matrix is .

There are n such rows, and we need $O(n^2)$ operations. However, by using the *Fast Hadamard Transform* , we can speed it up to $O(nlogn)$.

Theorem 11.

$$H_{2^m} = M_{2^m}^{(1)} M_{2^m}^{(2)} \dots M_{2^m}^{(m)} \text{ where } M_{2^m}^{(i)} = I_{2^{m-i}} \otimes H_2 \oplus I_{2^{i-1}}, \qquad 1 \le i \le m$$

Proof. By simple induction on m. □

This is the fastest classical decoding technique and was employed in the *Green Machine* decoder for the Mariner-9 space mission.

For a detailed look into the construction of the decoder, refer [7]

8.6 List Decoding

A list decoding procedure for a code C is a function which for given a $\mathbf{u} \in F_q^n$ and $e > 0$ outputs all vC such that $(u, v) \le e$. Alternately, given a decoding radius T, it should produce a list (set) of codewords which are distance less than T. List decoding is an old technique given by P.Elias in 1950 [**?**], but has recently found significant attention.

The existence of a good, fast list decoding algorithm for the first order Reed-Muller codes was given by goldreich and levin [**?**], [**?**]. However they do not give a usable algorithm. Recently, Kabatiansky, Dumer and Tavernier [5], [3] have presented a simple list decoding algorithm for $\mathcal{R}(1,m)$ capable of correcting $n(\frac{1}{2} - \epsilon)$ errors in $O(n\epsilon^3)$. This is significantly better than the $\frac{n}{4}$ correcting capability of the Hadamard transform **??**, which uses $O(nlogn)$ time.

The algorithm

Johnson Bound [8] If C is an [n,k,d] code with $d \ge (1/2 - \epsilon)$, then

$$\forall u \in F_q^n | \{v \in C : \Delta(u,v) \le (1/2 - \sqrt{\epsilon})\} | leq 1/\epsilon$$

Proof: (from lec10.ps)

8.7 RM(1,m) decoding algorithm

Let $\mathbf{y}$ be a received vector and $L_\epsilon(y) = \{f \in \mathcal{R}(1,m) : d(\mathbf{y}, \mathbf{f}) \le n(\frac{1}{2} - \epsilon)\}$ be the desired list. The algorithm works recursively by nding on the i-th step a list $L_\epsilon^i(y)$ of 'candidates' which should (but may not) coincide with i-prex of some $f(x1, ..., xm) = f0 + f1x1 + ... + fmxmL(y)$

The main idea is to approximate the Hamming distance between the received vector y and an arbitrary "propagation" of a candidate $c(i)(x1, ..., xm) = c1x1 + ... + cixi$ by the sum of Hamming distances over all i-dimensional "facets" of the m-dimensional Boolean cube.

The i-th prefix of a boolean function is defined as an i-variate linear Boolean funtion:

$$c^{(i)}(x_1, x_2, ldots, x_i) = c_1 x_1 + ldots + c_i x_i$$

The algorithm essentially proceeds by considering more components of the candidates and in the process adds/removes candidates from the list.

Thus, in phase i, we use

$$L_{\epsilon}^{(i-1)}(y) = \{c^{(i-1)}(x_1, x_2, ldots, x_i)\}$$

to obtain

$$L_{\epsilon}^{(i)}(y) = \{c^{(i1)}(x_1, x_2, ldots, x_i) = c^{(i-1)} + c_i x_i\}$$

Note that x_i can be any of the remaining $m - i$ basis vectors, and hence we get 2^{m-i} 'extensions' of $c^{(i)}$ for each $c^{(i-1)}$. This is an exponentially growing list, and we consider how to limit it's size next.

Given a 2^m length vector, we consider its coordinates to be the points in an m-dimensional Boolean cube. Cosider an i-dimensional *facet* of this cube as

$$S_\alpha = \{(x_1, \ldots x_i, \alpha_{i+1} \ldots \alpha_m\} \text{where} \alpha = (\alpha_{i+1}, \ldots, \alpha_m)$$

The prefixes $(x_1, \ldots x_i)$ run through F_2^i, and we get a set of coordinates.

Thus, each facet is a particular subset of the coordinates of the vector. Let the *restriction* of a vector $\mathbf{y}$ to a given facet S_α be denoted by $\mathbf{y}_\alpha$. The restriction to a facet yields the vector values only on those coordinates.

Keeping note of the fact that we have to compute the inner product of the received vector with each of the codewords , let $\mathbf{v_\alpha^{(i)}} \equiv \mathbf{y}_\alpha \mathbf{c^{(i)}}$

Define our distance function as

$$\Delta(\mathbf{y}, \mathbf{c^{(i)}}) = \sum_\alpha |\mathbf{y}_\alpha \mathbf{c^{(i)}}| = \sum_\alpha |\mathbf{v_\alpha^{(i)}}|$$

The central idea of the algorithm is to calculate the Δ for each candidate $\mathbf{c^{(i)}}$ and use this function as an upperbound .

Then in each step i, we use the function $\Delta(\mathbf{y}, \mathbf{c^{(i)}})$ and construct the list of prefixes which are under the threshold

$$L_{\epsilon}^{(i)}(y) = \{c^{(i1)} : \Delta(\mathbf{y}, \mathbf{c^{(i)}}) \geq n\epsilon\}$$

The final key idea is to calculate $\mathbf{v_\alpha^{(i)}}$ recursively. S_α, the i-dimensional facet can be written as two $(i-1)$-dimensional subfacets

$$S_{0,\alpha} = \{(x_1, \ldots x_{i-1}, 0, \alpha_{i+1} \ldots \alpha_m\}$$

$\mathrm{S}_{1,\alpha} = \{(x_1, \ldots x_{i-1}, 1, \alpha_{i+1} \ldots \alpha_m\}$

Since $c^{(i)} = c^{(i-1)} + c_i x_i$, we can write :

$$\mathbf{v_\alpha^{(i)}} = \mathbf{v_{0,\alpha}^{(i)}} + (-1)^{c_i} \mathbf{v_{1,\alpha}^{(i-1)}}$$

The algorithm runs for m phases.

This algorithm is similar to the fast hadamard transform described earlier **??**. The key difference is that in the fast hadamard transform, all the 2^m vector products are computed, albiet in an efficient manner. Here, we perform the distance-check after each phase to limit the number of candidates.

For a proof of correctness of the algorithm , see [5]. Analysis of running times and a probabilistic version is also presented.

8.8 general bounds

[**?**]

8.9 Other decoding algorithms

Besides the algorithms presented above, there exists a large number of algorithms. Majority logic algorithms are: Some run in $O(n)$ time. There are hard and soft decision algorithms as well.

9 Generalized Reed-Mullercodes

10 Applications

The Reed-Muller codes typically find application in communication and complexity theory. In this section a few representative examples are presented.

10.1 Communication

The oldest and original use. Reed-Muller codes were used in the 1969 Mariner-9 deep space probe to the IEEE 802.11b standard for Wireless Local Area Networks (WLANs) [4].

10.2 Testing Low-degree polynomials

[?]

Using $\mathcal{R}(1, m)$ codes to test whether a binary function is a low-degree polynomial. The functions are mapped to the Reed-Muller codes, and the properties are used to prove the bounds on the number of queries needed.

10.3 Sidenkov cryptanalysis

The Sidenkov public-key system, (a variant of the McEliece cryptosystem) uses Reed-Muller codes instead of Goppa codes because of the efficient decoding [?]. A cryptanalysis attack also uses the properties of Reed-Muller codes to break the cryptographic code [?]. The uniqueness result proved earlier is a central feature in the cryptographic attack. [?]

10.4 Side Channel attacks

[?]

[?]

10.5 Other Applications

[?] , [?], [].

11 Conclusion

12 References

References

[1] EF Assmus and JD Key. *Designs and their Codes*. Cambridge Univ Pr, 1994.

[2] M.K. Behbahani, GB Khosrovshahi, and B. Tayfeh-Rezaie. Reed-Muller Codes and Incidence Matrices.

[3] I. Dumer, G. Kabatiansky, C. Tavernier, and F. Le Plessis Robinson. On complexity of decoding Reed-Muller codes within their code distance. In *Proc. Eleventh International Workshop on Algebraic and Combinatorial Coding Theory*, pages 82–85, 2008.

[4] Matteo Frigo Jon Feldman, Ibrahim Abou-Faycal. A noise-adaptive algorithm for first-order reed-muller decoding.

[5] G. Kabatiansky and C. Tavernier. List decoding of first order Reed-Muller codes II. In *Tenth International Workshop on Algebraic and Combinatorial Coding Theory, ACCT*, pages 131–134, 2006.

[6] T. Kasami and N. Tokura. On the weight structure of Reed-Muller codes. *IEEE Transactions on Information Theory*, 16(6):752–759, 1970.

[7] N. Sloane and F. MacWilliams. The Theory of Correcting Codes, 1979.

[8] David Zuckerman. Lecture notes cs295t, lecture 10.