Reed–Muller Error Correcting Codes

Ben Cooke

Abstract. This paper starts by defining key terms and operations used with Reed–Muller codes and binary numbers. Reed–Muller codes are then defined and encoding matrices are discussed. Finally, a method of decoding is explained and an example is given to clarify the method.

1. Introduction. Reed–Muller codes are some of the oldest error correcting codes. Error correcting codes are very useful in sending information over long distances or through channels where errors might occur in the message. They have become more prevalent as telecommunications have expanded and developed a use for codes that can self-correct. Reed–Muller codes were invented in 1954 by D. E. Muller and I. S. Reed. In 1972, a Reed–Muller code was used by Mariner 9 to transmit black and white photographs of Mars, see [**7**, p. 47]. Reed–Muller codes are relatively easy to decode, and first-order codes are especially efficient.

In Section 2, we discuss vector operations and describe how to form a vector from a polynomial. In Section 3, we introduce Reed–Muller codes and show how to form encoding matrices and encode messages. In Section 4, we give a method for decoding Reed–Muller encoded messages. Finally, we give an example of decoding an encoded message in Section 5.

2. Definition of Terms and Operations. The vector spaces used in this paper consist of strings of length 2^m, where m is a positive integer, of numbers in $\mathcal{F}_2 = \{0, 1\}$. The codewords of a Reed–Muller code form a subspace of such a space. Vectors can be manipulated by three main operations: addition, multiplication, and the dot product.

For two vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$, *addition* is defined by

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n),$$

where each x_i or y_i is either 1 or 0, and

$$1 + 1 = 0, \quad 0 + 1 = 1, \quad 1 + 0 = 1, \quad 0 + 0 = 0.$$

For example, if $\mathbf{x}$ and $\mathbf{y}$ are defined as $\mathbf{x} = (10011110)$ and $\mathbf{y} = (11100001)$, then the sum of $\mathbf{x}$ and $\mathbf{y}$ is

$$\mathbf{x} + \mathbf{y} = (10011110) + (11100001) = (01111111).$$

The *addition of a scalar* $a \in \mathcal{F}_2$ to vector $\mathbf{x}$ is defined by

$$a + \mathbf{x} = (a + x_1, a + x_2, \ldots, a + x_n).$$

The *complement* $\bar{\mathbf{x}}$ of a vector $\mathbf{x}$ is the vector equal to $1+\mathbf{x}$. An example of the addition of a constant to a vector is

$$1 + (000111) = (111000).$$

Multiplication is defined by the formula,

$$\mathbf{x} * \mathbf{y} = (x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n),$$

where each x_i and y_i is either 1 or 0 and

$$1 * 1 = 1,\ \ 0 * 1 = 0,\ \ 1 * 0 = 0,\ \ 0 * 0 = 0.$$

For example, using the same $\mathbf{x}$ and $\mathbf{y}$ above, the product of $\mathbf{x}$ and $\mathbf{y}$ is

$$\mathbf{x} * \mathbf{y} = (10011110) * (11100001) = (10000000).$$

The *multiplication of a constant* $\mathbf{a} \in \mathcal{F}_\mathbf{2}$ to vector $\mathbf{x}$ is defined by

$$a * \mathbf{x} = (a * \mathbf{x}_1, a * \mathbf{x}_2, \ldots, a * \mathbf{x}_n).$$

An example is $0 * (111001) = (000000)$. The *dot product* of $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$\mathbf{x} \cdot \mathbf{y} = x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n.$$

For example, using $\mathbf{x}$ and $\mathbf{y}$ from above,

$$\mathbf{x} \cdot \mathbf{y} = (10011110) \cdot (11100001) = 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 1.$$

All three of these operations require vectors with the same number of coordinates.

Vectors can be associated with Boolean polynomials. A *Boolean polynomial* is a linear combination of *Boolean monomials* with coefficients in $\mathcal{F}_2$. A Boolean monomial $\mathbf{p}$ in the variables $\mathbf{x}_1, \ldots, \mathbf{x}_m$, is an expression of the form,

$$\mathbf{p} = \mathbf{x}_1^{r_1}\mathbf{x}_2^{r_2} \cdots \mathbf{x}_m^{r_m} \text{ where } r_i \in \{0, 1, 2, \ldots\} \text{ and } 1 \le i \le m.$$

The *reduced form* $\mathbf{p}'$ of $\mathbf{p}$ is obtained by applying the rules,

$$\mathbf{x}_i\mathbf{x}_j = \mathbf{x}_j\mathbf{x}_i \text{ and } \mathbf{x}_i^2 = \mathbf{x}_i,$$

until the factors are distinct. The *degree* of $\mathbf{p}$ is the ordinary degree of $\mathbf{p}'$, which is the number of variables in $\mathbf{p}'$. A Boolean polynomial is in *reduced form* if each monomial is in reduced form. The *degree* of a Boolean polynomial $\mathbf{q}$ is the ordinary degree of its reduced form $\mathbf{q}'$.

An example of a Boolean polynomial in reduced form with degree three is

$$\mathbf{q} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_2\mathbf{x}_3\mathbf{x}_4.$$

See [**6**, pp. 268–9].

We can now describe how to associate a boolean monomial in m variables to a vector with 2^m entries. The degree-zero monomial is $\mathbf{1}$, and the degree-one monomials are $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$. First we define the vectors associated with these monomials. The vector associated with the monomial $\mathbf{1}$ is simply a vector of length 2^m, where every entry of the vector is 1. So, in a space of size 2^3, the vector associated with $\mathbf{1}$ is (11111111).

The vector associated with the monomial $\mathbf{x}_1$ is 2^{m-1} ones, followed by 2^{m-1} zeros. The vector associated with the monomial $\mathbf{x}_2$ is 2^{m-2} ones, followed by 2^{m-2} zeros, then another 2^{m-2} ones, followed by another 2^{m-2} zeros. In general, the vector associated with a monomial $\mathbf{x}_i$ is a pattern of 2^{m-i} ones followed by 2^{m-i} zeros, repeated until 2^m values have been defined. For example, in a space of size 2^4, the vector associated with $\mathbf{x}_4$ is (1010101010101010).

To form the vector for a monomial $\mathbf{x}_1^{r_1}, \mathbf{x}_2^{r_2}, \ldots$, first put the monomial in reduced form. Then multiply the vectors associated with each monomial $\mathbf{x}_i$ in the reduced form. For example, in a space with $m = 3$, the vector associated with the monomial $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ can be found by multiplying

$$(11110000) * (11001100) * (10101010),$$

which gives (10000000).

To form the vector for a polynomial, simply reduce all of the monomials in the polynomial, and find the vectors associated with each of the monomials. Then, add all the vectors associated with each of these monomials together to form the vector associated with the polynomial. This gives us a bijection between reduced polynomials and vectors. From now on, we will treat the reduced polynomial and the vector associated with that polynomial interchangeably.

3. Reed–Muller Code and Encoding Matrices. An r^{th} *order Reed–Muller code* $\Re(r,m)$ is the set of all binary strings (vectors) of length $n = 2^m$ associated with the Boolean polynomials $\mathbf{p}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ of degree at most r. The 0^{th} order Reed–Muller code $\Re(0,m)$ consists of the binary strings associated with the constant polynomials $\mathbf{0}$ and $\mathbf{1}$; that is,

$$\Re(0,m) = \{\mathbf{0}, \mathbf{1}\} = \mathbf{Rep}(2^m).$$

Thus, $\Re(0,m)$ is just a repetition of either zeros or ones of length 2^m. At the other extreme, the m^{th} order Reed–Muller code $\Re(m,m)$ consists of all binary strings of length 2^m, see [**6**, pp. 270–71].

To define the encoding matrix of $\Re(r,m)$, let the first row of the encoding matrix be $\mathbf{1}$, the vector length 2^m with all entries equal to 1. If r is equal to 0, then this row is the only one in the encoding matrix. If r is equal to 1, then add m rows corresponding to the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ to the $\Re(0,m)$ encoding matrix.

To form a $\Re(r,m)$ encoding matrix where r is greater than 1, add $\binom{m}{r}$ rows to the $\Re(r-1,m)$ encoding matrix. These added rows consist of all the possible reduced degree r monomials that can be formed using the rows $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_m}$. For example, when $m = 3$ we have

$$\begin{bmatrix} \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \mathbf{x}_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{x}_2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ \mathbf{x}_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad \Re(1,3).$$

The rows $\mathbf{x}_1\mathbf{x}_2 = 11000000$, $\mathbf{x}_1\mathbf{x}_3 = 10100000$, and $\mathbf{x}_2\mathbf{x}_3 = 10001000$ are added to form

$$\left[\begin{array}{lcccccccc}
\mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\mathbf{x}_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\mathbf{x}_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\mathbf{x}_1 x_2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2 x_3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right] \quad \Re(2,3).$$

Finally, the row $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 = 10000000$ is added to form

$$\left[\begin{array}{lcccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\mathbf{x}_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\mathbf{x}_3 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\mathbf{x}_1\mathbf{x}_2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_1\mathbf{x}_3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2\mathbf{x}_3 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right] \quad \Re(3,3).$$

Another example of a Reed–Muller encoding matrix is

$$\left[\begin{array}{lcccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\mathbf{x}_1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{x}_3 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\mathbf{x}_4 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\mathbf{x}_1\mathbf{x}_2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_1\mathbf{x}_3 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_1\mathbf{x}_4 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2\mathbf{x}_3 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_2\mathbf{x}_4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{x}_3\mathbf{x}_4 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right] \quad \Re(2,4).$$

Encoding a message using Reed–Muller code $\Re(r,m)$ is straightforward. Take the code we are using to be $\Re(r,m)$. Its dimension is

$$k = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}.$$

In other words, the encoding matrix has k rows, see [**6**, pp. 271–72]. We send messages in blocks of length k. Let $\mathbf{m} = (m_1, m_2, \ldots m_k)$ be a block, the encoded message $\mathbf{M}_c$ is

$$\mathbf{M}_c = \sum_{i=l}^{k} m_i R_i$$

where $\mathbf{R}_i$ is a row of the encoding matrix of $\Re(r,m)$.

For example, using $\Re(\mathbf{1},\mathbf{3})$ to encode $\mathbf{m} = (0110)$ gives

$$0 * (11111111) + 1 * (11110000) + 1 * (11001100) + 0 * (10101010) = (00111100).$$

Similarly, using $\Re(2,4)$ to encode $\mathbf{m} = (10101110010)$ gives (0011100100000101).

4. Decoding Reed–Muller. Decoding Reed–Muller encoded messages is more complex than encoding them. The theory behind encoding and decoding is based on the *distance* between vectors. The distance between any two vectors is the number of places in the two vectors that have different values. The distance between any two codewords in $\Re(r,m)$ code is 2^{m-r}. The basis for Reed–Muller encoding is the assumption that the closest codeword in $\Re(r,m)$ to the received message is the original encoded message. Thus for e errors to be corrected in the received message, the distance between any two of the codewords in $\Re(r,m)$ must be greater than $2e$.

The decoding method used is not very efficient, but is straightforward to implement. It checks each row of the encoding matrix and uses majority logic to determine whether that row was used in forming the encoding message. Thus, it is possible to determine what the error-less encoded message was and what the original message was. This method of decoding is given by the following algorithm: Apply Steps 1 and 2 below, to each row of the matrix, starting from the bottom and working upwards.

Step 1. Choose a row in the $\Re(r,m)$ encoding matrix. Find $\mathbf{2^{m-r}}$ characteristic vectors (this process is described below) for that row, and then take the dot product of each of those rows with the encoded message.

Step 2. Take the majority of the values of the dot products and assign that value to the coefficient of the row.

Step 3. After doing Steps 1 and 2 for each row except the top row from the bottom of the matrix up, multiply each coefficient by its corresponding row and add the resulting vectors to form $\mathbf{M_y}$. Add this result to the received encoded message. If the resulting vector has more ones than zeros, then the top row's coefficient is 1, otherwise it is 0. Adding the top row, multiplied by its coefficient, to $\mathbf{M_y}$ gives the original encoded message. Thus, we can identify the errors. The vector formed by the sequence of coefficients starting from the top row of the encoding matrix and ending with the bottom row is the original message.

To find the characteristic vectors of any row of the matrix, take the monomial r associated with the row of the encoding matrix. Then, take $\mathbf{E}$ to be the set of all $\mathbf{x}_i$ that are not in the monomial r, but are in the encoding matrix. The *characteristic vectors* are the vectors corresponding to the monomials in $\mathbf{x}_i$ and $\bar{\mathbf{x}}_i$, such that exactly one of $\mathbf{x}_i$ or $\bar{x}_i$ is in each monomial for all $\mathbf{x}_i$ in $\mathbf{E}$. For example, the last row of the encoding matrix $\Re(2,4)$ is associated with $\mathbf{x}_3\mathbf{x}_4$, so the characteristic vectors correspond to the following combinations of $\mathbf{x}_1, \mathbf{x}_2, \bar{\mathbf{x}}_1$, and $\bar{\mathbf{x}}_2 : \mathbf{x}_1\mathbf{x}_2, \mathbf{x}_1\bar{\mathbf{x}}_2, \bar{\mathbf{x}}_1\mathbf{x}_2, \bar{\mathbf{x}}_1\bar{\mathbf{x}}_2$. These characteristic vectors have the property that the dot product is zero with all the rows in $\Re(r,m)$ except the row to which the characteristic vectors correspond.

5. Example. If the original message is $\mathbf{m} = (0110)$ using $\Re(1,3)$, then the encoded message is $\mathbf{M_c} = (00111100)$. Because the distance in $\Re(1,3)$ is $2^{3-1} = 4$, this code can correct one error. Let the encoded message after the error be $\mathbf{M_e} = (10111100)$. The characteristic vectors of the last row $\mathbf{x}_3 = (10101010)$ are $\mathbf{x}_1\mathbf{x}_2$, $\mathbf{x}_1\bar{\mathbf{x}}_2$, $\bar{\mathbf{x}}_1\mathbf{x}_2$, and $\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2$.

The vector associated with $\mathbf{x}_1$ is (11110000), so $\bar{\mathbf{x}}_1 = (00001111)$. The vector associated with $\mathbf{x}_2$ is (11001100), so $\bar{\mathbf{x}}_2 = (00110011)$. Therefore, we have $\mathbf{x}_1\mathbf{x}_2 = (11000000)$, $\mathbf{x}_1\bar{\mathbf{x}}_2 = (00110000)$, $\bar{\mathbf{x}}_1\mathbf{x}_2 = (00001100)$, and $\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2 = (00000011)$. Taking the dot products of these vectors with $\mathbf{M_e}$, we get

$$(11000000) \cdot (10111100) = 1, \qquad (00110000) \cdot (10111100) = 0,$$
$$(00001100) \cdot (10111100) = 0, \qquad (00000011) \cdot (10111100) = 0.$$

We can conclude that the coefficient of $\mathbf{x_3}$ is 0.

Doing the same for the second to last row of the matrix, $\mathbf{x_2} = (11001100)$, we get the characteristic vectors $\mathbf{x_1}x_3$, $\mathbf{x_1}\bar{\mathbf{x}}_3$, $\bar{\mathbf{x}}_1\mathbf{x_3}$, and $\bar{\mathbf{x}}_1\bar{\mathbf{x}}_3$. These vectors are (10100000), (01010000), (00001010), and (00000101), respectively. Taking the dot products of these vectors with $\mathbf{M_e}$, we get

$$\begin{array}{ll} (10100000)\cdot(10111100)=0, & (01010000)\cdot(10111100)=1, \\ (00001010)\cdot(10111100)=1, & (00000101)\cdot(10111100)=1. \end{array}$$

So, we can conclude that the coefficient of $\mathbf{x_2}$ is 1. Doing the same for the second row of the matrix $\mathbf{x_1} = (11110000)$, we get

$$\begin{array}{ll} (10001000)\cdot(10111100)=0, & (00100010)\cdot(10111100)=1, \\ (01000100)\cdot(10111100)=1, & (00010001)\cdot(10111100)=1. \end{array}$$

We can conclude that the coefficient for $\mathbf{x_1}$ is also 1.

If we add $0*(10101010)$ and $1*(11001100)$ and $1*(11110000)$ we get $\mathbf{M_y}$, which is equal to (00111100). Then we see that the sum of $\mathbf{M_y}$ and $\mathbf{M_e}$ is equal to

$$(00111100)+(10111100)=(10000000).$$

This message has more zeros than ones, so the coefficient of the first row of the encoding matrix is zero. Thus we can put together coefficients for the four rows of the matrix, 0,1,1, and 0, and see that the original message was (0110). We can also see that the error was in the first place of the error-free message $\mathbf{M_c} = (00111100)$.

References

[**1**] Arazi, Benjamin, "A Commonsense Approach to the Theory of Error Correcting Codes," MIT Press, 1988.

[**2**] Hoffman, D. G. *et al.,* "Coding Theory; the Essentials," Marcel Dekker Inc., 1991.

[**3**] Mann, Henry B., "Error Correcting Codes," John Wiley and Sons, 1968.

[**4**] Purser, Michael, "Introduction to Error-Correcting Codes," Artech House Inc., Norwood, MA, 1995.

[**5**] Reed, Irving S., "A Class of Multiple-Error-Correcting Codes and Decoding Scheme," MIT Lincoln Laboratory, 1953.

[**6**] Roman, Steven, "Coding and Information Theory," Springer-Verlag, 1992.

[**7**] Van Lindt, J. H., "Introduction to Coding Theory," Springer-Verlag, 1982.