ALGEBRAIC CODING THEORY

In computers and digital communication systems, information is almost always represented in a binary form as a sequence of bits each having the values 0 or 1. This sequence of bits is transmitted over a *channel* from a sender to a receiver. In some applications the channel is a storage medium like a CD, where the information is written to the medium at a certain time and retrieved at a later time. Due to physical limitations of the channel, some of the transmitted bits may be corrupted (the channel is *noisy*) and thus make it difficult for the receiver to reconstruct the information correctly.

In algebraic coding theory we are mainly concerned with developing methods for detecting and correcting errors that typically occur during transmission of information over a noisy channel. The basic technique to detect and correct errors is by introducing redundancy in the data that are to be transmitted. This is similar to communicating in a natural language in daily life. One can understand the information while listening to a noisy radio or talking on a bad telephone line due to the redundancy in the language.

For an example, suppose the sender wants to communicate one of 16 different messages to a receiver. Each message m can then be represented as a binary quadruple $m = (c_0, c_1, c_2, c_3)$. If the message (0101) is transmitted and the first position is corrupted such that (1101) is received, this leads to an

uncorrectable error since this quadruple represents a different valid message than the message that was sent across the channel. The receiver will have no way to detect and correct a corrupted message in general, since any quadruple represents a valid message.

Therefore, to combat errors the sender *encodes* the data by introducing redundancy into the transmitted information. If M messages are to be transmitted, the sender selects a subset of M binary n-tuples, where $M < 2^n$. Each of the M messages is encoded into one of the selected n-tuples. The set consisting of the M n-tuples obtained after encoding is called a binary (n, M) *code* and the elements are called *codewords.* The codewords are sent over the channel.

It is customary for many applications to let $M = 2^k$, such that each message can be represented uniquely by a k-tuple of information bits. To encode each message the sender can append $n - k$ parity bits depending on the message bits and use the resulting n bit codeword to represent the corresponding message.

A binary code C is called a linear code if the sum (modulo 2) of two codewords is again a codeword. This is always the case when the parity bits are linear combinations of the information bits. In this case, the code C is a vector space of dimension k over the binary field of two elements, containing $M = 2^k$ codewords, and is called an $[n, k]$ code. The main reason for using linear codes is that these codes have more algebraic structure and are therefore often easier to analyze and decode in practical applications.

The simplest example of a linear code is the $[n, n - 1]$ *even-weight code* (or parity-check code). The encoding consists of appending a single parity bit to the $n - 1$ information bits so that the codeword has an even number of ones. Thus the code consists of all 2^{n-1} possible n-tuples of even weight, where the *weight* of a vector is the total number of ones in its components. This code can detect all errors in an odd number of positions, since if such an error occurs the received vector will also have odd weight. The even-weight code, however, can only detect errors. For example, if (000 . . . 0) is sent and the first bit is corrupted, then (100 . . . 0) is received. Also, if (110 . . . 0) was sent and the second bit was corrupted, then (100 . . . 0) is received. Hence, there is no way the receiver can correct this single error or, in fact, any other error.

An illustration of a code that can correct any single error is shown in Fig. 1. The three circles intersect and divide the plane into seven finite areas and one infinite area. Each finite area contains a bit c_i for $i = 0, 1, \ldots, 6$. Each of the 16 possible messages, denoted by (c_0, c_1, c_2, c_3), is encoded into a codeword $(c_0, c_1, c_2, c_3, c_4, c_5, c_6)$, in such a way that the sum of the bits in each circle has an even parity.

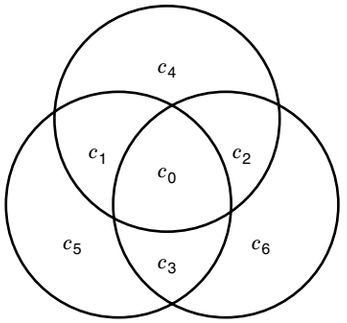

Figure 1. The message (c_0, c_1, c_2, c_3) is encoded into the codeword $(c_0, c_1, c_2, c_3, c_4, c_5, c_6)$, where c_4, c_5, c_6 are chosen such that there is an even number of ones within each circle.

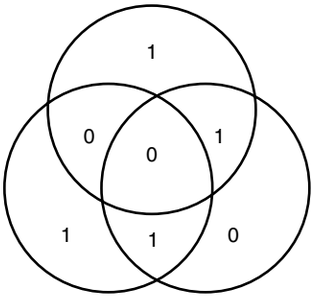

Figure 2. Example of the encoding procedure given in Fig. 1. The message (0011) is encoded into (0011110). Note that there is an even number of ones within each circle.

In Fig. 2, an example is shown of encoding the message (0011) into the codeword (0011110). Since the sum of two codewords also obeys the parity checks and thus is a codeword, the code is a linear [7, 4] code.

Suppose, for example, that the transmitted codeword is corrupted in the bit c_1 such that the received word is (0111110). Then, calculating the parity of each of the three circles, we see that the parity fails for the upper circle as well as for the leftmost circle while the parity of the rightmost circle is correct. Hence, from the received vector we can indeed conclude that bit c_1 is in error and should be corrected. In the same way, any single error can be corrected by this code.

LINEAR CODES

An (n, M) code is simply a set of M vectors of length n with components from a finite field $F_2 = \{0, 1\}$, where addition and multiplication are done modulo 2. For practical applications it is desirable that the code is provided with more structure. Therefore, linear codes are often preferred. A linear $[n, k]$ code C is a k-dimensional subspace C of F_2^n, where F_2^n is the vector space of n-tuples with coefficients from the finite field F_2.

A linear code C is usually described in terms of a generator matrix or a parity-check matrix. A *generator matrix* G of C is a $k \times n$ matrix whose row space is the code C. That is,

$$C = \{\boldsymbol{x}G | \boldsymbol{x} \in F_2^k\}$$

A *parity-check matrix* H is an $(n - k) \times n$ matrix such that

$$C = \{\boldsymbol{c} \in F_2^n | \boldsymbol{c}H^{\text{tr}} = \boldsymbol{0}\}$$

where H^{tr} denotes the transpose of H.

Example. The codewords in the code in the previous section are the vectors $(c_0, c_1, c_2, c_3, c_4, c_5, c_6)$ that satisfy the following

system of *parity-check equations:*

$$c_0 + c_1 + c_2 + c_4 = 0$$
$$c_0 + c_1 + c_3 + c_5 = 0$$
$$c_0 + c_2 + c_3 + c_6 = 0$$

where all additions are modulo 2. Each of the three parity-check equations correspond to one of the three circles.

The coefficient matrix of the parity-check equations is the parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

The code C is therefore given by

$$C = \{\boldsymbol{c} = (c_0, c_1, \cdots, c_6) | \boldsymbol{c}H^{\text{tr}} = \boldsymbol{0}\}$$

A generator matrix for the code in the previous example is given by

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

Two codes are *equivalent* if the codewords in one of the codes can be obtained by a fixed permutation of the positions in the codewords in the other code. If G (respectively, H) is a generator (respectively, parity-check) matrix of a code, then the matrices obtained by permuting the columns of these matrices in the same way give the generator matrix (respectively, parity-check) matrix of the permuted code.

The *Hamming distance* between $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$ and $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ in F_2^n is the number of positions in which they differ. That is,

$$d(\boldsymbol{x}, \boldsymbol{y}) = |\{i | x_i \neq y_i, 0 \leq i \leq n-1\}|$$

The Hamming distance has the properties required to be a metric:

1. $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ for all $\boldsymbol{x}, \boldsymbol{y} \in F_2^n$ and equality holds if and only if $\boldsymbol{x} = \boldsymbol{y}$.
2. $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$ for all $\boldsymbol{x}, \boldsymbol{y} \in F_2^n$.
3. $d(\boldsymbol{x}, \boldsymbol{z}) \geq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$ for all $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in F_2^n$.

For any code C one of the most important parameters is its *minimum distance,* defined by

$$d = \min\{d(\boldsymbol{x}, \boldsymbol{y}) | \boldsymbol{x} \neq \boldsymbol{y}, \boldsymbol{x}, \boldsymbol{y} \in C\}$$

The *Hamming weight* of a vector $\boldsymbol{x}$ in F_2^n is the number of nonzero components in $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1})$. That is,

$$w(\boldsymbol{x}) = |\{i | x_i \neq 0, 0 \leq i \leq n-1\}| = d(\boldsymbol{x}, \boldsymbol{0})$$

Note that since $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{0}) = w(\boldsymbol{x} - \boldsymbol{y})$ for a linear code C, it follows that

$$d = \min\{w(\boldsymbol{z}) | \boldsymbol{z} \in C, \boldsymbol{z} \neq \boldsymbol{0}\}$$

Therefore, finding the minimum distance of a linear code is equivalent to finding the minimum nonzero weight among all codewords in the code.

If $w(\boldsymbol{c}) = i$, then $\boldsymbol{c}H^{\text{tr}}$ is the sum of i columns of H. Hence, an alternative description of the minimum distance of a linear code is as follows: the smallest d such that there exists d linearly dependent columns in the parity-check matrix. In particular, to obtain a linear code of minimum distance at least three, it is sufficient to select the columns of a parity-check matrix to be distinct and nonzero.

Sometimes we include d in the notation and refer to an $[n, k]$ code with minimum distance d as an $[n, k, d]$ code. If t components are corrupted during transmission of a codeword, we say that t errors have occurred or that an error $\boldsymbol{e}$ of weight t has occurred [where $\boldsymbol{e} = (e_0, e_1, \ldots, e_{n-1}) \in F_2^n$, where $e_i = 1$ if and only if the ith component was corrupted—that is, if $\boldsymbol{c}$ was sent, $\boldsymbol{c} + \boldsymbol{e}$ was received].

The *error-correcting capability* of a code is defined as

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

where $\lfloor x \rfloor$ denotes the largest integer $\leq x$.

A code with minimum distance d can correct all errors of weight t or less. This is due to the fact that if a codeword $\boldsymbol{c}$ is transmitted and an error $\boldsymbol{e}$ of weight $e \leq t$ occurs, the received vector $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ is closer in Hamming distance to the transmitted codeword $\boldsymbol{c}$ than to any other codeword. Therefore, decoding any received vector to the closest codeword corrects all errors of weight $\leq t$.

The code can also be used for error detection only. The code is able to detect all errors of weight $< d$ since if a codeword is transmitted and the error has weight $< d$, then the received vector is not another codeword.

The code can also be used for a combination of error correction and error detection. For a given $e \leq t$, the code can correct all errors of weight $\leq e$ and in addition detect all errors of weight at most $d - e - 1$. This is due to the fact that no vector in F_2^n can be at distance $\leq e$ from one codeword and at the same time at a distance $\leq d - e - 1$ from another codeword. Hence, the algorithm in this case is to decode a received vector to a codeword at distance $\leq e$ if such a codeword exists and otherwise detect an error.

If C is an $[n, k]$ code, the *extended code* C^{ext} is the $[n + 1, k]$ code defined by

$$C^{\text{ext}} = \left\{ (c_{\text{ext}}, c_0, c_1, \ldots, c_{n-1}) \,\middle|\, (c_0, c_1, \ldots, c_{n-1}) \in C, \; c_{\text{ext}} = \sum_{i=0}^{n-1} c_i \right\}$$

That is, each codeword in C is extended by one parity bit such that the Hamming weight of each codeword becomes even. In particular, if C has odd minimum distance d, then the minimum distance of C^{ext} is $d + 1$. If H is a parity-check matrix for C, then a parity-check matrix for C^{ext} is

$$\begin{pmatrix} 1 & 1 \\ \boldsymbol{0}^{\text{tr}} & H \end{pmatrix}$$

where $\boldsymbol{1} = (11 \ldots 1)$.

For any linear $[n, k]$ code C, the *dual code* $C^\perp$ is the $[n, n-k]$ code defined by

$$C^\perp = \{\boldsymbol{x} \in F_2^n | (\boldsymbol{x}, \boldsymbol{c}) = 0 \text{ for all } \boldsymbol{c} \in C\}$$

where $(\boldsymbol{x}, \boldsymbol{c}) = \sum_{i=0}^{n-1} x_i c_i$. We say that $\boldsymbol{x}$ and $\boldsymbol{c}$ are *orthogonal* if $(\boldsymbol{x}, \boldsymbol{c}) = 0$. Therefore, $C^\perp$ consists of all n-tuples that are orthogonal to all codewords in C and vice versa—that is, $(C^\perp)^\perp = C$. It follows that $C^\perp$ has dimension $n - k$ since it consists of all vectors that are solutions of a system of equations with coefficient matrix G of rank k. Hence, the parity-check matrix of $C^\perp$ is a generator matrix of C, and similarly the generator matrix of $C^\perp$ is a parity-check matrix of C. In particular, $GH^{\text{tr}} = O$ [the $k \times (n-k)$ matrix of all zeros].

Example. Let C be the $[n, n-1, 2]$ even-weight code where

$$G = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 1 & \cdots & 0 & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{pmatrix}$$

and

$$H = (1 \quad 1 \quad \cdots \quad 1 \quad 1 \quad 1)$$

Then $C^\perp$ has H and G as its generator and parity-check matrices, respectively. It follows that $C^\perp$ is the $[n, 1, n]$ *repetition code* consisting of the two codewords $(00 \cdots 000)$ and $(11 \cdots 111)$.

Example. Let C be the $[2^m - 1, 2^m - 1 - m, 3]$ code, where H contains all nonzero m-tuples as its columns. This is known as the *Hamming code*. In the case when $m = 3$, a parity-check matrix is already described in Eq. (1). Since all columns of the parity-check matrix are distinct and nonzero, the code has minimum distance at least 3. The minimum distance is indeed 3 since there exist three columns whose sum is zero, in fact the sum of any two columns of H equals another column in H for this particular code.

The dual code $C^\perp$ is the $[2^m - 1, m, 2^{m-1}]$ *simplex code* all of whose nonzero codewords have weight 2^{m-1}. This follows since the generator matrix has all nonzero vectors as its columns. In particular, taking any linear combination of rows, the number of columns with odd parity in the corresponding subset of rows equals 2^{m-1} (and the number with even parity is $2^{m-1} - 1$).

The extended code of the Hamming code is a $[2^m, 2^m - 1 - m, 4]$ code. Its dual code is a $[2^m, m+1, 2^{m-1}]$ code that is known as the first-order Reed-Muller code.

SOME BOUNDS ON CODES

The *Hamming bound* states that for any (n, M, d) code we have

$$M \sum_{i=0}^{e} \binom{n}{i} \le 2^n$$

where $e = \lfloor (d-1)/2 \rfloor$. This follows from the fact that the M spheres

$$S_{\boldsymbol{c}} = \{\boldsymbol{x} | d(\boldsymbol{x}, \boldsymbol{c}) \le e\}$$

centered at the codewords $\boldsymbol{c} \in C$ are disjoint and that each sphere contains

$$\sum_{i=0}^{e} \binom{n}{i}$$

vectors.

If the spheres fill the whole space, that is,

$$\bigcup_{\boldsymbol{c} \in C} S_{\boldsymbol{c}} = F_2^n$$

then C is called *perfect*. The binary *linear* perfect codes are as follows:

- The $[n, 1, n]$ repetition codes for all odd n
- The $[2^m - 1, 2^m - 1 - m, 3]$ Hamming codes H_m for all $m \ge 2$
- The $[23, 12, 7]$ Golay code G_{23}

We will return to the Golay code later.

GALOIS FIELDS

There exist finite fields, also known as Galois fields, with p^m elements for any prime p and any positive integer m. A Galois field of a given order p^m is unique (up to isomorphism) and is denoted by F_{p^m}.

For a prime p, let $F_p = \{0, 1, \ldots, p-1\}$ denote the integers modulo p with the two operations addition and multiplication modulo p.

To construct a Galois field with p^m elements, select a polynomial $f(x)$ with coefficients in F_p that is irreducible over F_p; that is, $f(x)$ cannot be written as a product of two polynomials with coefficients from F_p of degree ≥ 1 (irreducible polynomials of any degree m over F_p exist).

Let

$$F_{p^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_0 | a_0, \ldots, a_{m-1} \in F_p\}$$

Then F_{p^m} is a finite field when addition and multiplication of the elements (polynomials) are done modulo $f(x)$ and modulo p. To simplify the notations let α denote a zero of $f(x)$, that is, $f(\alpha) = 0$. If such an α exists, it can formally be defined as the equivalence class of x modulo $f(x)$. For coding theory, $p = 2$ is by far the most important case, and we assume this from now on. Note that for any $a, b \in F_{2^m}$,

$$(a+b)^2 = a^2 + b^2$$

Example. The Galois field F_{2^4} can be constructed as follows. Let $f(x) = x^4 + x + 1$ that is an irreducible polynomial over F_2. Then $\alpha^4 = \alpha + 1$ and

$$F_{2^4} = \{a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0 | a_0, a_1, a_2, a_3 \in F_2\}$$

Computing the powers of α, we obtain

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(\alpha + 1) = \alpha^2 + \alpha,$$
$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2,$$
$$\alpha^7 = \alpha \cdot \alpha^6 = \alpha(\alpha^3 + \alpha^2) = \alpha^4 + \alpha^3 = \alpha^3 + \alpha + 1$$

and, similarly, all higher powers of α can be expressed as a linear combination of α^3, α^2, α, and 1. In particular, $\alpha^{15} = 1$. We get the following table of the powers of α. In the table the polynomial $a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$ is represented as $a_3a_2a_1a_0$.

i	α^i	i	α^i	i	α^i
0	0001	5	0110	10	0111
1	0010	6	1100	11	1110
2	0100	7	1011	12	1111
3	1000	8	0101	13	1101
4	0011	9	1010	14	1001

Hence, the elements 1, α, α^2, . . ., α^{14} are all the nonzero elements in F_{2^4}. Such an element α that generates the nonzero elements of F_{2^m} is called a *primitive element* in F_{2^m}. An irreducible polynomial $g(x)$ with a primitive element as a root is called a primitive polynomial. Every finite field has a primitive element, and therefore the multiplicative subgroup of a finite field is cyclic.

All elements in F_{2^m} are roots of the equation $x^{2^m} + x = 0$. Let β be an element in F_{2^m}. It is important to study the polynomial $m(x)$ of smallest degree with coefficients in F_2 that has β as a zero. This polynomial is called the *minimal polynomial* of β over F_2.

First, observe that if $m(x) = \sum_{i=0}^{\kappa} m_i x^i$ has coefficients in F_2 and β as a zero, then

$$m(\beta^2) = \sum_{i=0}^{\kappa} m_i \beta^{2i} = \sum_{i=0}^{\kappa} m_i^2 \beta^{2i} = \left(\sum_{i=0}^{\kappa} m_i \beta^i \right)^2 = (m(\beta))^2 = 0$$

Hence, $m(x)$ has β, β^2, . . ., $\beta^{2^{\kappa-1}}$, as zeros, where κ is the smallest integer such that $\beta^{2^\kappa} = \beta$. Conversely, the polynomial with exactly these zeros can be shown to be a binary irreducible polynomial.

Example. We will find the minimal polynomial of all the elements in F_{2^4}. Let α be a root of $x^4 + x + 1 = 0$; that is, $\alpha^4 = \alpha + 1$. The minimal polynomials over F_2 of α^i for $0 \le i \le 14$ are denoted $m_i(x)$. Observe by the preceding argument that $m_{2i}(x) = m_i(x)$, where the indices are taken modulo 15. It follows that

$$m_0(x) = (x + \alpha^0) = x + 1,$$
$$m_1(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4)(x + \alpha^8) = x^4 + x + 1,$$
$$m_3(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^{12})(x + \alpha^9) = x^4 + x^3 + x^2 + x + 1,$$
$$m_5(x) = (x + \alpha^5)(x + \alpha^{10}) = x^2 + x + 1,$$
$$m_7(x) = (x + \alpha^7)(x + \alpha^{14})(x + \alpha^{13})(x + \alpha^{11}) = x^4 + x^3 + 1,$$
$$m_9(x) = m_3(x),$$
$$m_{11}(x) = m_7(x),$$
$$m_{13}(x) = m_7(x)$$

To verify this, one simply computes the coefficients and uses the preceding table of F_{2^4} in the computations. For example,

$$m_5(x) = (x + \alpha^5)(x + \alpha^{10}) = x^2 + (\alpha^5 + \alpha^{10})x + \alpha^5 \cdot \alpha^{10} = x^2 + x + 1$$

This also leads to a factorization into irreducible polynomials:

$$x^{2^4} + x = x \prod_{j=0}^{14} (x + \alpha^j) = x(x + 1)(x^2 + x + 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^4 + x^3 + 1) = x m_0(x) m_1(x) m_3(x) m_5(x) m_7(x)$$

In fact, it holds in general that $x^{2^m} + x$ is the product of all irreducible polynomials over F_2 of degree that divides m.

Let $C_i = \{i2^j \pmod n \mid j = 0, 1, \ldots\}$, which is called the *cyclotomic coset* of $i \pmod n$. Then the elements of the cyclotomic coset $C_i \pmod{2^m - 1}$ correspond to the exponents of the zeros of $m_i(x)$. That is,

$$m_i(x) = \prod_{j \in C_i} (x - \alpha^j)$$

The cyclotomic cosets $\pmod n$ are important in the next section when cyclic codes of length n are discussed.

CYCLIC CODES

Many good linear codes that have practical and efficient decoding algorithms have the property that a cyclic shift of a codeword is again a codeword. Such codes are called cyclic codes.

We can represent the set of n-tuples over F_2^n as polynomials of degree $< n$ in a natural way. The vector $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$ is represented as the polynomial $c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$. A *cyclic shift*

$$\sigma(\boldsymbol{c}) = (c_{n-1}, c_0, c_1, \ldots, c_{n-2})$$

of $\boldsymbol{c}$ is then represented by the polynomial

$$\sigma(c(x)) = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1} = x(c_{n-1}x^{n-1} + c_0 + c_1x + \cdots + c_{n-2}x^{n-2}) + c_{n-1}(x^n + 1) \equiv xc(x) \pmod{x^n + 1}$$

Example. Rearranging the columns in the parity-check matrix of the [7, 4] Hamming code in Eq. (1), an equivalent code is obtained with parity-check matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{2}$$

This code contains 16 codewords, which are represented next in polynomial form:

$$\begin{aligned}
1000110 \leftrightarrow \quad & x^5 + x^4 + 1 = && (x^2 + x + 1)g(x)\\
0100011 \leftrightarrow \quad & x^6 + x^5 + x = && (x^3 + x^2 + x)g(x)\\
1010001 \leftrightarrow \quad & x^6 + x^2 + 1 = && (x^3 + x + 1)g(x)\\
1101000 \leftrightarrow \quad & x^3 + x + 1 = && g(x)\\
0110100 \leftrightarrow \quad & x^4 + x^2 + x = && xg(x)\\
0011010 \leftrightarrow \quad & x^5 + x^3 + x^2 = && x^2g(x)\\
0001101 \leftrightarrow \quad & x^6 + x^4 + x^3 = && x^3g(x)\\
0010111 \leftrightarrow \quad & x^6 + x^5 + x^4 + x^2 = && (x^3 + x^2)g(x)\\
1001011 \leftrightarrow \quad & x^6 + x^5 + x^3 + 1 = && (x^3 + x^2 + x + 1)g(x)\\
1100101 \leftrightarrow \quad & x^6 + x^4 + x + 1 = && (x^3 + 1)g(x)\\
1110010 \leftrightarrow \quad & x^5 + x^2 + x + 1 = && (x^2 + 1)g(x)\\
0111001 \leftrightarrow \quad & x^6 + x^3 + x^2 + x = && (x^3 + x)g(x)\\
1011100 \leftrightarrow \quad & x^4 + x^3 + x^2 + 1 = && (x + 1)g(x)\\
0101110 \leftrightarrow \quad & x^5 + x^4 + x^3 + x = && (x^2 + x)g(x)\\
0000000 \leftrightarrow \quad & 0 = && 0\\
1111111 \leftrightarrow \quad & x^6 + x^5 + \cdots + x + 1 = && (x^3 + x^2 + 1)g(x)
\end{aligned}$$

By inspection it is easy to verify that any cyclic shift of a codeword is again a codeword. Indeed, the 16 codewords in the code are $\mathbf{0}$, $\mathbf{1}$ and all cyclic shifts of (1000110) and (0010111). The unique nonzero polynomial in the code of lowest possible degree is $g(x) = x^3 + x + 1$, and $g(x)$ is called the *generator polynomial* of the cyclic code. The code consists of all polynomials $c(x)$ that are multiples of $g(x)$. Note that the degree of $g(x)$ is $n - k = 3$ and that $g(x)$ divides $x^7 + 1$ since $x^7 + 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$.

The code therefore has a simple description in terms of the set of code polynomials as

$$C = \{c(x) | c(x) = u(x)(x^3 + x + 1), \deg(u(x)) < 4\}$$

This situation holds in general for any cyclic code.

For any cyclic $[n, k]$ code C, we have

$$C = \{c(x) | c(x) = u(x)g(x), \deg(u(x)) < k\}$$

for a polynomial $g(x)$ of degree $n - k$ that divides $x^n + 1$.

We can show this as follows: Let $g(x)$ be the generator polynomial of C, which is the nonzero polynomial of smallest degree r in the code C. Then the cyclic shifts $g(x)$, $xg(x)$, $\cdots$, $x^{n-r-1}g(x)$ are codewords as well as any linear combination $u(x)g(x)$, where $\deg(u(x)) < n - r$. These are the only 2^{n-r} codewords in the code C, since if $c(x)$ is a codeword then

$$c(x) = u(x)g(x) + s(x), \text{where} \quad \deg(s(x)) < \deg(g(x))$$

By linearity, $s(x)$ is a codeword and therefore $s(x) = 0$ since $\deg(s(x)) < \deg(g(x))$ and $g(x)$ is the nonzero polynomial of smallest degree in the code. It follows that C is as described previously. Since C has 2^{n-r} codewords, it follows that $n - r = k$; that is, $\deg(g(x)) = n - k$.

Finally, we show that $g(x)$ divides $x^n + 1$. Let $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$ be a nonzero codeword shifted such that $c_{n-1} = 1$. Then a cyclic shift of $c(x)$ given by $\sigma(c(x)) = c_{n-1} + c_0x + c_1x + \cdots + c_{n-2}x^{n-1}$ is also a codeword and

$$\sigma(c(x)) = xc(x) + \vartheta(x^n + 1)$$

Since both of the codewords $c(x)$ and $\sigma(c(x))$ are divisible by $g(x)$, it follows that $g(x)$ divides $x^n + 1$.

Since the generator polynomial of a cyclic code divides $x^n + 1$, it is important to know how to factor $x^n + 1$ into irreducible polynomials. Let n be odd. Then there is an integer m such that $2^m \equiv 1 \pmod{n}$ and there is an element $\alpha \in F_{2^m}$ of order n [if ω is a primitive element of F_{2^m}, then α can be taken to be $\alpha = \omega^{(2^m-1)/n}$].

We have

$$x^n + 1 = \prod_{i=0}^{n-1} (x + \alpha^i)$$

Let $m_i(x)$ denote the minimal polynomial of α^i; that is, the polynomial of smallest degree with coefficients in F_2 and having α^i as a zero. The generator polynomial $g(x)$ can be written as

$$g(x) = \prod_{i \in I} (x + \alpha^i)$$

where I is a subset of $\{0, 1, \ldots, n - 1\}$, called the *defining set* of C with respect to α. Then $m_i(x)$ divides $g(x)$ for all $i \in I$. Further, $g(x) = \prod_{j=1}^{l} m_{i_j}(x)$ for some $i_1, i_2, \ldots, i_l$.

We can therefore describe the cyclic code in alternative equivalent ways as

$$\begin{aligned}
&C = \{c(x) | m_i(x) \text{ divides } c(x), \text{ for all } i \in I\},\\
&C = \{c(x) | c(\alpha^i) = 0, \text{ for all } i \in I\},\\
&C = \{\boldsymbol{c} \in F_2^n | \boldsymbol{c}H^{\text{tr}} = \mathbf{0}\}
\end{aligned}$$

where

$$H = \begin{pmatrix}
1 & \alpha^{i_1} & \alpha^{2i_1} & \cdots & \alpha^{(n-1)i_1}\\
1 & \alpha^{i_2} & \alpha^{2i_2} & \cdots & \alpha^{(n-1)i_2}\\
\vdots & \vdots & \vdots & \ddots & \vdots\\
1 & \alpha^{i_l} & \alpha^{2i_l} & \cdots & \alpha^{(n-1)i_l}
\end{pmatrix}$$

The encoding for cyclic codes is usually done in one of two ways. Let $u(x)$ denote the information polynomial of degree $< k$. The two ways are as follows:

1. Encode into $u(x)g(x)$.
2. Encode into $c(x) = x^{n-k}u(x) + s(x)$, where $s(x)$ is the polynomial such that
   - $s(x) \equiv x^{n-k}u(x) \pmod{g(x)}$ [thus $g(x)$ divides $c(x)$]
   - $\deg(s(x)) < \deg(g(x))$

The last of these two methods is systematic; that is, the last k bits of the codeword are the information bits.

BCH CODES

An important task in coding theory is to design codes with a guaranteed minimum distance d that correct all errors of Hamming weight $\lfloor (d - 1)/2 \rfloor$. Such codes were designed independently by Bose and Ray-Chaudhuri (1960) and by Hocquenghem (1959) and are known as BCH codes. To construct a BCH code of *designed distance* d, the generator polynomial

is chosen to have $d-1$ consecutive powers of α as zeros

$$\alpha^b, \alpha^{b+1}, \cdots, \alpha^{b+d-2}$$

That is, the defining set I with respect to α contains a set of $d-1$ consecutive integers (mod n). The parity-check matrix of the BCH code is

$$H = \begin{pmatrix} 1 & \alpha^b & \alpha^{2b} & \cdots & \alpha^{(n-1)b} \\ 1 & \alpha^{b+1} & \alpha^{2(b+1)} & \cdots & \alpha^{(n-1)(b+1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{b+d-2} & \alpha^{2(b+d-2)} & \cdots & \alpha^{(n-1)(b+d-2)} \end{pmatrix}$$

To show that this code has minimum distance at least d, it is sufficient to show that any $d-1$ columns are linear independent. Suppose there is a linear dependency between the $d-1$ columns corresponding to $\alpha^{i_1 b}, \alpha^{i_2 b}, \ldots, \alpha^{i_{d-1} b}$. In this case the $(d-1) \times (d-1)$ submatrix obtained by retaining these columns in H has determinant

$$\begin{vmatrix} \alpha^{i_1 b} & \alpha^{i_2 b} & \cdots & \alpha^{i_{d-1} b} \\ \alpha^{i_1 (b+1)} & \alpha^{i_2 (b+1)} & \cdots & \alpha^{i_{d-1} (b+1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^{i_2 (b+d-2)} & \alpha^{i_2 (b+d-2)} & \cdots & \alpha^{i_{d-1} (b+d-2)} \end{vmatrix}$$

$$= \alpha^{b(i_1+i_2+\ldots+i_{d-1})} \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \cdots & \alpha^{i_{d-1}} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha^{(d-2)i_1} & \alpha^{(d-2)i_2} & \cdots & \alpha^{(d-2)i_{d-1}} \end{vmatrix}$$

$$= \alpha^{b(i_1+i_2+\ldots+i_{d-1})} \prod_{k<r} (\alpha^{i_k} - \alpha^{i_r}) \neq 0$$

since the elements $\alpha^{i_1}, \alpha^{i_2}, \cdots, \alpha^{i_{d-1}}$ are distinct (the last equality follows from the fact that the last determinant is a Vandermonde determinant). It follows that the BCH code has minimum Hamming distance at least d.

If $b = 1$, which is often the case, the code is called a *narrow-sense* BCH code. If $n = 2^m - 1$, the BCH code is called a *primitive* BCH code. A binary single error-correcting primitive BCH code is generated by $g(x) = m_1(x)$. The zeros of $g(x)$ are α^{2^i}, $i = 0, 1, \ldots, m-1$. The parity-check matrix is

$$H = (1 \quad \alpha^1 \quad \alpha^2 \quad \ldots \quad \alpha^{2^m-2})$$

This code is equivalent to the Hamming code since α is a primitive element of F_{2^m}.

To construct a binary double error-correcting primitive BCH code, we let $g(x)$ have α, α^2, α^3, α^4 as zeros. Therefore, $g(x) = m_1(x)m_3(x)$ is a generator polynomial of this code. The parity-check matrix of a double error-correcting BCH code is

$$H = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{2^m-2} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(2^m-2)} \end{pmatrix}$$

In particular, a binary double-error correcting BCH code of length $n = 2^4 - 1 = 15$ is obtained by selecting

$$\begin{aligned} g(x) &= m_1(x)m_3(x) \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) \\ &= x^8 + x^7 + x^6 + x^4 + 1 \end{aligned}$$

Similarly, a binary triple-error correcting BCH code of the same length is obtained by choosing the generator polynomial

$$\begin{aligned} g(x) &= m_1(x)m_3(x)m_5(x) \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \end{aligned}$$

The main interest in BCH codes is due to the fact that they have a very fast and efficient decoding algorithm. We describe this later.

AUTOMORPHISMS

Let C be a binary code of length n. Consider a permutation π of the set $\{0, 1, \ldots, n-1\}$; that is, π is a one-to-one function of the set of coordinate positions onto itself.

For a codeword $\boldsymbol{c} \in C$, let

$$\pi(\boldsymbol{c}) = (c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(n-1)})$$

That is, the coordinates are permuted by the permutation π. If

$$\{\pi(\boldsymbol{c}) | \boldsymbol{c} \in C\} = C$$

then π is called an *automorphism* of the code C.

Example. Consider the following (nonlinear code):

$$C = \{101, 011\}$$

The actions of the six possible permutations on three elements are given in the following table. The permutations that are automorphisms are marked by a star.

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi((101))$	$\pi((011))$	
0	1	2	101	011	★
0	2	1	110	011	
1	0	2	011	101	★
1	2	0	011	110	
2	0	1	110	101	
2	1	0	101	110	

In general, the set of automorphisms of a code C is a group, the *Automorphism group* Aut(C). We note that

$$\sum_{i=0}^{n-1} x_i y_i = \sum_{i=0}^{n-1} x_{\pi(i)} y_{\pi(i)}$$

and so $(\boldsymbol{x}, \boldsymbol{y}) = 0$ if and only if $(\pi(\boldsymbol{x}), \pi(\boldsymbol{y})) = 0$. In particular, this implies that

$$\text{Aut}(C) = \text{Aut}(C^\perp)$$

That is, C and $C^\perp$ have the same automorphism group.

For a *cyclic* code C of length n, we have by definition $\sigma(\boldsymbol{c}) \in C$ for all $\boldsymbol{c} \in C$, where $\sigma(i) \equiv i - 1 \pmod{n}$. In particular, $\sigma \in \text{Aut}(C)$. For n odd, the permutation δ defined by $\delta(j) =$

$2j \pmod n$ is also contained in the automorphism group. To show this it is easier to show that $\delta^{-1} \in \mathrm{Aut}(C)$. We have

$$\begin{aligned} \delta^{-1}(2j) &= j && \text{for } j = 0, 1, \ldots, (n-1)/2, \\ \delta^{-1}(2j+1) &= (n+1)/2 + j && \text{for } j = 0, 1, \ldots, (n-1)/2 - 1 \end{aligned}$$

Let $g(x)$ be a generator polynomial for C, and let $\sum_{i=0}^{n-1} c_i x^i = a(x)g(x)$. Since $x^n \equiv 1 \pmod{x^n + 1}$, we have

$$\begin{aligned} \sum_{i=0}^{n-1} c_{\delta^{-1}(i)} x^i &\equiv \sum_{j=0}^{(n-1)/2} c_j x^{2j} + \sum_{j=0}^{(n-1)/2-1} c_{(n+1)/2+j} x^{2j+1+n} \\ &= \sum_{j=0}^{(n-1)/2} c_j x^{2j} + \sum_{j=(n+1)/2}^{n-1} c_j x^{2j} \\ &= a(x^2)g(x^2) = (a(x^2)g(x))g(x), \quad (\mathrm{mod}\, x^n + 1) \end{aligned}$$

and so $\delta^{-1}(\boldsymbol{c}) \in C$; that is, $\delta^{-1} \in \mathrm{Aut}(C)$ and so $\delta \in \mathrm{Aut}(C)$.

The automorphism group $\mathrm{Aut}(C)$ is *transitive* if for each pair (i, j) there exists a $\pi \in \mathrm{Aut}(C)$ such that $\pi(i) = j$. More general, $\mathrm{Aut}(C)$ is t-fold transitive if, for distinct $i_1, i_2, \ldots, i_t$ and distinct $j_1, j_2, \ldots, j_t$, there exists a $\pi \in \mathrm{Aut}(C)$ such that $\pi(i_1) = j_1, \pi(i_2) = j_2, \ldots, \pi(i_t) = j_t$.

Example. Any cyclic $[n, k]$ code has a transitive automorphism group since σ repeated s times, where $s \equiv i - j \pmod n$, maps i to j.

Example. The (nonlinear) code $C = \{101, 011\}$ was considered previously. Its automorphism group is not transitive since there is no automorphism π such that $\pi(0) = 2$.

Example. Let C be the $[9, 3]$ code generated by the matrix

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

This is a cyclic code and we will determine its automorphism group. The all zero and the all one vectors in C are transformed into themselves by any permutation. The vectors of weight 3 are the rows of the generator matrix and the vectors of weight 6 are the complements of these vectors. Hence, we see that π is an automorphism if and only if it leaves the set of the three rows of the generator matrix invariant, that is, if and only if the following conditions are satisfied:

$$\begin{aligned} \pi(0) \equiv \pi(3) \equiv \pi(6) \quad (mod\ 3), \\ \pi(1) \equiv \pi(4) \equiv \pi(7) \quad (mod\ 3), \\ \pi(2) \equiv \pi(5) \equiv \pi(8) \quad (mod\ 3). \end{aligned}$$

Note that the two permutations σ and δ defined previously satisfy these conditions, as they should. They are listed explicitly in the following table

i	0	1	2	3	4	5	6	7	8
$\sigma(i)$	8	0	1	2	3	4	5	6	7
$\delta(i)$	0	2	4	6	8	1	3	5	7

The automorphism group is transitive since the code is cyclic, but not doubly transitive. For example, there is no automorphism π such that $\pi(0) = 0$ and $\pi(3) = 1$ since 0 and 1 are not equivalent modulo 3. A simple counting argument shows that $\mathrm{Aut}(C)$ has order 1296: First choose $\pi(0)$; this can be done in 9 ways. There are then 2 ways to choose $\pi(3)$ and $\pi(6)$. Next choose $\pi(1)$; this can be done in 6 ways. There are again 2 ways to choose $\pi(4)$ and $\pi(7)$. Finally, there are $3 \cdot 2$ ways to choose $\pi(2)$, $\pi(5)$, $\pi(8)$. Hence, the order is $9 \cdot 2 \cdot 6 \cdot 2 \cdot 3 \cdot 2 = 1296$.

Example. Consider the extended Hamming code H_m^{ext}. The positions of the codewords correspond to the elements of F_{2^m} and are permuted by the affine group

$$\mathrm{AG} = \{\pi \,|\, \pi(x) = ax + b, a, b \in F_{2^m}, a \neq 0\}$$

This is the automorphism group of H_m^{ext}. It is double transitive.

THE WEIGHT DISTRIBUTION OF A CODE

Let C be a binary linear $[n, k]$ code. As we noted before,

$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{x} - \boldsymbol{y}, \boldsymbol{0}) = w(\boldsymbol{x} - \boldsymbol{y})$$

If $\boldsymbol{x}, \boldsymbol{y} \in C$, then $\boldsymbol{x} - \boldsymbol{y} \in C$ by the linearity of C. In particular, this means that the set of distances from a fixed codeword to all the other codewords is independent of which codeword we fix; that is, the code looks the same from any codeword. In particular, the set of distances from the codeword $\boldsymbol{0}$ is the set of *Hamming weights* of the codewords. For $i = 0, 1, \ldots, n$, let A_i denote the number of codewords of weight i. The sequence

$$A_0, A_1, A_2, \ldots, A_n$$

is called the *weight distribution* of the code C. The corresponding polynomial

$$A_C(z) = A_0 + A_1 z + A_2 z^2 + \cdots + A_n z^n$$

is known as the *weight enumerator polynomial* of C.

The polynomials $A_C(z)$ and $A_{C^\perp}(z)$ are related by the fundamental MacWilliams identity:

$$A_{C^\perp}(z) = 2^{-k}(1+z)^n A_C\left(\frac{1-z}{1+z}\right)$$

Example. The $[2^m - 1, m]$ simplex code has the weight enumerator polynomial $1 + (2^m - 1)z^{2^{m-1}}$. The dual code is the $[2^m - 1, 2^m - 1 - m]$ Hamming code with weight enumerator polynomial

$$\begin{aligned} &2^{-m}(1+z)^{2^m-1}\left(1 + (2^m - 1)\left(\frac{1-z}{1+z}\right)^{2^{m-1}}\right) \\ &\quad = 2^{-m}(1+z)^{2^m-1} + (1 - 2^{-m})(1-z)^{2^{m-1}}(1+z)^{2^{m-1}-1} \end{aligned}$$

For example, for $m = 4$, we get the weight distribution of the [15, 11] Hamming code:

$$1 + 35z^3 + 105z^4 + 168z^5 + 280z^6 + 435z^7 + 435z^8 + 280z^9 + 168z^{10} + 105z^{11} + 35z^{12} + z^{15}$$

Consider a binary linear code C that is used purely for error detection. Suppose a codeword $\boldsymbol{c}$ is transmitted over a binary symmetric channel with bit error probability p. The probability of receiving a vector $\boldsymbol{r}$ at distance i from $\boldsymbol{c}$ is $p^i(1-p)^{n-i}$, since i positions are changed (each with probability p) and $n - i$ are unchanged (each with probability $1 - p$). If $\boldsymbol{r}$ is not a codeword, then this will be discovered by the receiver. If $\boldsymbol{r} = \boldsymbol{c}$, then no errors have occurred. However, if $\boldsymbol{r}$ is another codeword, then an undetectable error has occurred. Hence, the *probability of undetected error* is given by

$$\begin{aligned} P_{\text{ue}}(C, p) &= \sum_{\boldsymbol{c}' \neq \boldsymbol{c}} p^{d(\boldsymbol{c}', \boldsymbol{c})}(1-p)^{n-d(\boldsymbol{c}', \boldsymbol{c})} \\ &= \sum_{\boldsymbol{c}'' \neq \boldsymbol{0}} p^{w(\boldsymbol{c}'')}(1-p)^{n-w(\boldsymbol{c}'')} \\ &= \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \\ &= (1-p)^n A_C\left(\frac{p}{1-p}\right) - (1-p)^n \end{aligned}$$

From the MacWilliams identity we also get

$$P_{\text{ue}}(C^{\perp}, p) = 2^{-k} A_C(1-2p) - (1-p)^n$$

Example. For the $[2^m - 1, 2^m - 1 - m]$ Hamming code H_m, we get

$$P_{\text{ue}}(H_m, p) = 2^{-m}(1 + (2^m - 1)(1-2p)^{2^{m-1}}) - (1-p)^{2^{m-1}}$$

More information on the use of codes for error detection can be found in the book by Kløve and Korzhik (see Reading List).

THE BINARY GOLAY CODE

The Golay code G_{23} has received much attention. It is practically useful and has a number of interesting properties. The code can be defined in various ways. One definition is that G_{23} is the cyclic code generated by the irreducible polynomial

$$x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$$

which is a factor of $x^{23} + 1$ over F_2. Another definition is the following: Let H denote the [7, 4] Hamming code and let H^* be the code whose codewords are the reversed of the codewords of H. Let

$$C = \{(\boldsymbol{u} + \boldsymbol{x}, \boldsymbol{v} + \boldsymbol{x}, \boldsymbol{u} + \boldsymbol{v} + \boldsymbol{x}) | \boldsymbol{u}, \boldsymbol{v} \in H^{\text{ext}}, \boldsymbol{x} \in (H^*)^{\text{ext}}\}$$

where H^{ext} is the [8, 4] extended Hamming code and $(H^*)^{\text{ext}}$ is the [8, 4] extended H^*. The code C is a [24, 12, 8] code. Puncturing the last position, we get a [23, 12, 7] code that is (equivalent to) the Golay code.

The weight distribution of G_{23} is given by the following table:

i	A_i
0, 23	1
7, 16	253
8, 15	506
11, 12	1288

The automorphism group $\text{Aut}(G_{23})$ of the Golay code is the Mathieu group M_{23}, a simple group of order $10200960 = 2^7 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 23$, which is four-fold transitive.

Much information about G_{23} can be found in the book by MacWilliams and Sloane (see Reading List).

DECODING

Suppose that a codeword $\boldsymbol{c}$ from the $[n, k]$ code C was sent and that an error $\boldsymbol{e}$ occurred during the transmission over the noisy channel. Based on the received vector $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$, the receiver has to make an estimate of what was the transmitted codeword. Since error patterns of lower weight are more probable than error patterns of higher weight, the problem is to estimate an error $\hat{\boldsymbol{e}}$ such that the weight of $\hat{\boldsymbol{e}}$ is as small as possible. He will then decode the received vector $\boldsymbol{r}$ into $\hat{\boldsymbol{c}} = \boldsymbol{r} + \hat{\boldsymbol{e}}$.

If H is a parity-check matrix for C, then $\boldsymbol{c}H^{\text{tr}} = \boldsymbol{0}$ for all codewords $\boldsymbol{c}$. Hence,

$$\boldsymbol{r}H^{\text{tr}} = (\boldsymbol{c} + \boldsymbol{e})H^{\text{tr}} = \boldsymbol{c}H^{\text{tr}} + \boldsymbol{e}H^{\text{tr}} = \boldsymbol{e}H^{\text{tr}} \tag{3}$$

The vector

$$\boldsymbol{s} = \boldsymbol{e}H^{\text{tr}}$$

is known as the *syndrome* of the error $\boldsymbol{e}$; Eq. (3) shows that $\boldsymbol{s}$ can be computed from $\boldsymbol{r}$. We now have the following outline of a decoding strategy:

1. Compute the syndrome $\boldsymbol{s} = \boldsymbol{r}H^{\text{tr}}$.
2. Estimate an error $\hat{\boldsymbol{e}}$ of smallest weight corresponding to the syndrome $\boldsymbol{s}$.
3. Decode to $\hat{\boldsymbol{c}} = \boldsymbol{r} + \hat{\boldsymbol{e}}$.

The hard part is, of course, step 2.

For any vector $\boldsymbol{x} \in F_2^n$, the set $\{\boldsymbol{x} + \boldsymbol{c} \mid \boldsymbol{c} \in C\}$ is a *coset* of C. All the elements of the coset have the same syndrome—namely, $\boldsymbol{x}H^{\text{tr}}$. There are 2^{n-k} cosets, one for each syndrome in F_2^{n-k}, and the set of cosets is partition of F_2^n. We can rephrase step 2 as follows: Find a vector $\boldsymbol{e}$ of smallest weight in the coset with syndrome $\boldsymbol{s}$.

Example. Let C be the [6, 3, 3] code with parity-check matrix

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

A *standard array* for C is the following array (the eight columns to the right):

000	111000	001011	010101	011110	100110	101101	110011	111000
110	100000	101011	110101	111110	000110	001101	010011	011000
101	010000	011011	000101	001110	110110	111101	100011	101000
011	001000	000011	011101	010110	101110	100101	111011	110000
100	000100	001111	010001	011010	100010	101001	110111	111100
010	000010	001001	010111	011100	100100	101111	110001	111010
001	000001	001010	010100	011111	100111	101100	110010	111001
111	100001	101010	110100	111111	000111	001100	010010	011001

Each row in the array is a listing of a coset of C; the first row is a listing of the code itself. The vectors in the first column have minimal weight in their cosets and are known as *coset leaders*. The choice of coset leader may not be unique. For example, in the last coset there are three vectors of minimal weight. Any entry in the array is the sum of the codeword at the top of the column and the coset leader (at the left in the row). Each vector of F_2^6 is listed exactly once in the array. The standard array can be used for decoding: Locate $\boldsymbol{r}$ in the array and decode to the codeword at the top of the corresponding column (that is, the coset leader is assumed to be the error pattern). However, this is not a practical method; except for small n, the standard array of 2^n entries is too large to store (also locating $\boldsymbol{r}$ may be a problem). A step in simplifying the method is to store a table of coset leaders corresponding to the 2^{n-k} syndromes. In the preceding table this is illustrated by listing the syndromes at the left. Again this is a possible alternative only if $n - k$ is small. For carefully designed codes, it is possible to *compute* $\boldsymbol{e}$ from the syndrome. The simplest case is single errors: If $\boldsymbol{e}$ is an error pattern of weight 1, where the 1 is in the ith position, then the corresponding syndrome is the ith column of H; hence, from H and the syndrome we can determine i.

Example. Let H be the $m \times (2^m - 1)$ parity-check matrix where the ith column is the binary expansion of the integer i for $i = 1, 2, \ldots, 2^m - 1$. The corresponding $[2^m - 1, 2^m - 1 - m, 3]$ Hamming code corrects all single errors. Decoding is done as follows: Compute the syndrome $\boldsymbol{s} = (s_0, s_1, \ldots, s_{m-1})$. If $\boldsymbol{s} \neq \boldsymbol{0}$, then correct position $i = \sum_{j=0}^{m-1} s_j 2^j$.

Example. Let

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(n-1)} \end{pmatrix}$$

where $\alpha \in F_{2^m}$ and $n = 2^m - 1$. This is the parity-check matrix for the double error-correcting BCH code. It is convenient to have a similar representation of the syndromes:

$$\boldsymbol{s} = (S_1, S_3) \quad \text{where} \quad S_1, S_3 \in F_{2^m}$$

Depending on the syndrome, there are several cases:

1. If no errors have occurred, then clearly $S_1 = S_3 = 0$.
2. If a single error has occurred in the ith position (that is, the position corresponding to α^i), then $S_1 = \alpha^i$ and $S_3 = \alpha^{3i}$. In particular, $S_3 = S_1^3$.
3. If two errors have occurred in positions i and j, then

$$S_1 = \alpha^i + \alpha^j, \quad S_3 = \alpha^{3i} + \alpha^{3j}$$

This implies that $S_1^3 = S_3 + \alpha^i\alpha^j S_1 \neq S_3$. Furthermore, $x_1 = \alpha^{-i}$ and $x_2 = \alpha^{-j}$ are roots of the equation

$$1 + S_1 x + \frac{S_1^3 + S_3}{S_1} x^2 = 0 \tag{4}$$

This gives the following procedure to correct two errors:

- Compute S_1 and S_3.
- If $S_1 = S_3 = 0$, then assume that no errors have occurred.
- Else, if $S^3 = S_1^3 \neq 0$, then one error has occurred in the ith position determined by $S_1 = \alpha^i$.
- Else (if $S_3 \neq S_1^3$), consider the equation

$$1 + S_1 x + (S_1^3 + S_3)/S_1 x^2 = 0$$

If the equation has two roots α^{-i} and α^{-j}, then errors have occurred in positions i and j.

Else (if the equation has no roots in F_{2^m}), then more than two errors have occurred.

Similar explicit expressions (in terms of the syndrome) for the coefficients of an equation with the error positions as roots can be found for t error-correcting BCH codes when $t = 3$, $t = 4$, etc., but they become increasingly complicated. However, there is an efficient algorithm for determining the equation, and we describe this is some detail next.

Let α be a primitive element in F_{2^m}. A parity-check matrix for the primitive t error-correcting BCH code is

$$H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^3 & \alpha^6 & \cdots & \alpha^{3(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t-1} & \alpha^{2(2t-1)} & \cdots & \alpha^{(2t-1)(n-1)} \end{pmatrix}$$

where $n = 2^m - 1$. Suppose errors have occurred in positions $i_1, i_2, \ldots, i_\tau$, where $\tau \leq t$. Let $X_j = \alpha^{i_j}$ for $j = 1, 2, \ldots, \tau$. The *error locator polynomial* $\Lambda(x)$ is defined by

$$\Lambda(x) = \prod_{j=1}^{\tau} (1 + X_j x) = \sum_{l=0}^{\tau} \lambda_l x^l$$

The roots of $\Lambda(x) = 0$ are X_j^{-1}. Therefore, if we can determine $\Lambda(x)$, then we can determine the locations of the errors. Expanding the expression for $\Lambda(x)$, we get

$$\begin{aligned} \lambda_0 &= 1, \\ \lambda_1 &= X_1 + X_2 + \cdots + X_\tau, \\ \lambda_2 &= X_1X_2 + X_1X_3 + X_2X_3 + \cdots + X_{\tau-1}X_\tau, \\ \lambda_3 &= X_1X_2X_3 + X_1X_2X_4 + X_2X_3X_4 \\ &\quad + \cdots + X_{\tau-2}X_{\tau-1}X_\tau, \\ &\vdots \\ \lambda_\tau &= X_1X_2 \cdots X_\tau, \\ \lambda_l &= 0 \text{ for } l > \tau \end{aligned}$$

Hence λ_l is the lth elementary symmetric function of $X_1, X_2, \ldots, X_\tau$.

From the syndrome we get $S_1, S_3, \ldots, S_{2t-1}$, where

$$\begin{aligned} S_1 &= X_1 + X_2 + \cdots + X_\tau, \\ S_2 &= X_1^2 + X_2^2 + \cdots + X_\tau^2, \\ S_3 &= X_1^3 + X_2^3 + \cdots + X_\tau^3, \\ &\vdots \\ S_{2t} &= X_1^{2t} + X_2^{2t} + \cdots + X_\tau^{2t} \end{aligned}$$

Further,

$$S_{2r} = X_1^{2r} + X_2^{2r} + \cdots + X_\tau^{2r} = (X_1^r + X_2^r + \cdots + X_\tau^r)^2 = S_r^2$$

for all r. Hence, from the syndrome we can determine the polynomial

$$S(x) = 1 + S_1 x + S_2 x^2 + \cdots + S_{2t} x^{2t}$$

The *Newton equations* are a set of relations between the power sums S_r and the symmetric functions λ_l—namely,

$$\sum_{j=0}^{l-1} S_{l-j}\lambda_j + l\lambda_l = 0 \text{ for } l \geq 1$$

Let

$$\Omega(x) = S(x)\Lambda(x) = \sum_{l \geq 0} \omega_l x^\ell \tag{5}$$

Since $\omega_l = \sum_{j=0}^{l-1} S_{l-j}\lambda_j + \lambda_l$, the Newton equations imply that

$$\omega_l = 0 \text{ for all odd } l, 1 \leq l \leq 2t-1 \tag{6}$$

The *Berlekamp–Massey algorithm* is an algorithm that, given $S(x)$, determines the polynomial $\Lambda(x)$ of smallest degree such that Eq. (6) is satisfied, where the ω_l are defined by Eq. (5). The idea is, for $r = 0, 1, \ldots, t$, to determine polynomials $\Lambda^{(r)}$ of lowest degree such that

$$\omega_l^{(r)} = 0 \text{ for all odd } l, 1 \leq l \leq 2r-1$$

where

$$\sum_{l \geq 0} \omega_l^{(r)} x^l = S(x)\Lambda^{(r)}(x)$$

For $r = 0$, we can clearly let $\Lambda^{(0)}(x) = 1$. We proceed by induction. Let $0 \leq r < t$, and suppose that polynomials $\Lambda^{(\rho)}(x)$ have been constructed for $0 \leq \rho \leq r$. If $\omega_{2r+1}^{(r)} = 0$, then we can choose

$$\Lambda^{(r+1)}(x) = \Lambda^{(r)}(x)$$

If, on the other hand, $\omega_{2r+1}^{(r)} \neq 0$, then we modify $\Lambda^{(r)}(x)$ by adding another suitable polynomial. There are two cases to consider. First, if $\Lambda^{(r)}(x) = 1$ [in which case $\Lambda^{(r)}(x) = 1$ for $0 \leq \tau \leq r$], then

$$\Lambda^{(r+1)}(x) = 1 + \omega_{2r+1}^{(r)} x^{2r+1}$$

will have the required property. If $\Lambda^{(r)}(x) \neq 1$, then there exists a maximal positive integer $\rho < r$ such that $\omega_{2\rho+1}^{(\rho)} \neq 0$ and we add a suitable multiple of $\Lambda^{(\rho)}$:

$$\Lambda^{(r+1)}(x) = \Lambda^{(r)}(x) + \omega_{2r+1}^{(r)} (\omega_{2\rho+1}^{\rho})^{-1} x^{2r-2\rho} \Lambda^{(\rho)}(x)$$

We note that this implies that

$$\Lambda^{(r+1)}(x)S(x) = \sum_{l \geq 0} w_l^{(r)} x^l + \omega_{2r+1}^{(r)} (\omega_{2\rho+1}^{(\rho)})^{-1} \sum_{l \geq 0} \omega_l^{(\rho)} x^{l+2r-2\rho}$$

Hence for odd l we get

$$\omega_l^{(r+1)} = \begin{cases} \omega_l^{(r)} = 0 \\ \qquad \text{for } 1 \leq l \leq 2r-2\rho-1, \\ \omega_l^{(r)} + \omega_{2r+1}^{(r)} (\omega_{2\rho+1}^{(\rho)})^{-1} \omega_{l-2r+2\rho}^{(\rho)} = 0 + 0 = 0 \\ \qquad \text{for } 2r-2\rho+1 \leq l \leq 2r-1, \\ \omega_{2r+1}^{(r)} + \omega_{2r+1}^{(r)} (\omega_{2\rho+1}^{(\rho)})^{-1} \omega_{2\rho+1}^{(\rho)} = \omega_{2r+1}^{(r)} + \omega_{2r+1}^{(r)} = 0 \\ \qquad \text{for } l = 2r+1 \end{cases}$$

We now formulate these ideas as an algorithm (in a Pascal-like syntax). In each step we keep the present $\Lambda(x)$ [the superscript (r) is dropped] and the modifying polynomial [$x^{2r-2\rho-1}$ or $(\omega_{2\rho+1}^{(\rho)})^{-1} x^{2r-2\rho-1} \Lambda^{(\rho)}(x)$], which we denote by $B(x)$.

Berlekamp–Massey algorithm in the binary case

```
Input: t and S(x).

Λ(x) := 1;   B(x) := 1;
for r := 1 to t do
begin
      ω := coefficient of x^(2r-1) in S(x)Λ(x);
      if ω = 0  then B(x) := x²B(x)
                else [Λ(x), B(x)] := [Λ(x) +
                 ωxB(x), xΛ(x)/ω]
end;
```

The assignment following the `else` is two assignments to be done in parallel; the new $\Lambda(x)$ and $B(x)$ are computed from the old ones.

The Berlekamp–Massey algorithm determines the polynomial $\Lambda(x)$. To find the roots of $\Lambda(x) = 0$, we try all possible elements of F_{2^m}. In practical applications, this can be efficiently implemented using shift registers (usually called the *Chien search*).

Example. We consider the [15, 7, 5] double-error correcting BCH code; that is, $m = 4$ and $t = 2$. As a primitive element, we choose α such that $\alpha^4 = \alpha + 1$. Suppose that we have received a vector with syndrome $(S_1, S_3) = (\alpha^4, \alpha^5)$. Since $S_3 \neq S_1^3$, at least two errors have occurred. Equation (4) becomes

$$1 + \alpha^4 x + \alpha^{10} x^2 = 0$$

which has the zeros α^{-3} and α^{-7}. We conclude that the received vector has two errors (namely, in positions 3 and 7).

Now consider the Berlekamp–Massey algorithm for the same example. First we compute $S_2 = S_1^2 = \alpha^8$ and $S_4 = S_2^2 =$

α. Hence

$$S(x) = 1 + \alpha^4 x + \alpha^8 x^2 + \alpha^5 x^3 + \alpha x^4$$

The values of r, ω, $\Lambda(x)$, and $B(x)$ after each iteration of the for-loop in the Berlekamp–Massey algorithm are shown in the following table:

r	ω	$\Lambda(x)$	$B(x)$
		1	1
1	α^4	$1 + \alpha^4 x$	$\alpha^{11} x$
2	α^{14}	$1 + \alpha^4 x + \alpha^{10} x^2$	$\alpha x + \alpha^5 x^2$

Hence, $\Lambda(x) = 1 + \alpha^4 x + \alpha^{10} x^2$ (as before).

Now consider the same code with syndrome of received vector $(S_1, S_3) = (\alpha, \alpha^9)$. Since $S_3 \neq S_1^3$, at least two errors have occurred. We get

$$\Lambda(x) = 1 + \alpha x + x^2$$

However, the equation $1 + \alpha x + x^2 = 0$ does not have any roots in F_{2^4}. Hence, at least three errors have occurred, and the code is not able to correct them.

REED–SOLOMON CODES

In the previous sections we have considered binary codes where the components of the codewords belong to the finite field $F_2 = \{0, 1\}$. In a similar way we can consider codes with components from any finite field F_q.

The *Singleton bound* states that for any $[n, k, d]$ code with components from F_q, we have

$$d \leq n - k + 1$$

A code for which $d = n - k + 1$ is called *maximum distance separable* (MDS). The only binary MDS codes are the trivial $[n, 1, n]$ repetition codes and $[n, n - 1, 2]$ even-weight codes. However, there are important nonbinary MDS codes (in particular, the Reed–Solomon codes, which we now will describe).

Reed–Solomon codes are t error-correcting cyclic codes with symbols from a finite field F_q, even though they can be constructed in many different ways. They can be considered as the simplest generalization of BCH codes. Since the most important case for applications is $q = 2^m$, we consider this case here. Each symbol is then an element in F_{2^m} and can be considered as an m-bit symbol.

The construction of a cyclic Reed–Solomon code is as follows: Let α be a primitive element of F_{2^m}. Since $\alpha^i \in F_{2^m}$ for all i, the minimal polynomial of α^i over F_{2^m} is just $x + \alpha^i$. The generator polynomial of a (primitive) t error-correcting Reed–Solomon code of length $2^m - 1$ has $2t$ consequtive powers of α as zeros:

$$\begin{aligned} g(x) &= \prod_{i=0}^{2t-1} (x + \alpha^{b+i}) \\ &= g_0 + g_1 x + \cdots + g_{2t-1} x^{2t-1} + x^{2t} \end{aligned}$$

The code has the following parameters:

Block length: $n = 2^m - 1$
Number of parity-check symbols: $n - k = 2t$
Minimum distance: $d = 2t + 1$

Thus, the Reed–Solomon codes satisfy the Singleton bound with equality $n - k = d - 1$. That is, they are MDS codes.

The weight distribution of the Reed–Solomon code is (for $i \geq d$)

$$A_i = \binom{n}{i} \sum_{j=0}^{i-d} (-1)^j \binom{i}{j} (2^{m(i-d-j+1)} - 1)$$

The encoding of Reed–Solomon codes is similar to the encoding of binary cyclic codes. The decoding is similar to the decoding of binary BCH codes with one added complication. Using a generalization of the Berlekamp–Massey algorithm, we determine the polynomials $\Lambda(x)$ and $\Omega(x)$. From $\Lambda(x)$ we can determine the locations of the errors. In addition, we have to determine the value of the errors (in the binary case the values are always 1). The value of the error at location X_j can easily be determined using $\Omega(x)$ and $\Lambda(x)$; we omit further details.

NONLINEAR CODES FROM CODES OVER Z_4

In the previous sections we have mainly considered binary *linear* codes; that is, codes where the sum of two codewords is again a codeword. The main reason has been that the linearity greatly simplified construction and decoding of the codes.

A binary nonlinear (n, M, d) code C is simply a set of M binary n-tuples with pairwise distance at least d, but without any further imposed structure. In general, to find the minimum distance of a nonlinear code one has to compute the distance between all pairs of codewords. This is, of course, more complicated than for linear codes, where it suffices to find the minimum weight among all the nonzero codewords. The lack of structure in a nonlinear code also makes it quite difficult to decode in an efficient manner.

There are, however, some advantages to nonlinear codes. For given values of length n and minimum distance d, it is sometimes possible to construct nonlinear codes with more codewords than is possible for linear codes. For example, for $n = 16$ and $d = 6$ the best linear code has dimension $k = 7$ (i.e., it contains 128 codewords). The code of length 16 obtained by extending the double-error-correcting primitive BCH code has these parameters.

In 1967, Nordstrom and Robinson found a nonlinear code with parameters $n = 16$ and $d = 6$ containing $M = 256$ codewords, which has twice as many codewords as the best linear code for the same values of n and d.

In 1968, Preparata generalized this construction to an infinite family of codes having parameters

$$(2^{m+1}, 2^{2^{m+1}-2m-2}, 6), m \text{ odd}, m \geq 3$$

A few years later, in 1972, Kerdock gave another generalization of the Nordstrom–Robinson code and constructed another infinite class of codes with parameters

$$(2^{m+1}, 2^{2m+2}, 2^m - 2^{(m-1)/2}), m \text{ odd}, m \geq 3$$

The Preparata code contains twice as many codewords as the extended double-error-correcting BCH code and is optimal in the sense of having the largest possible size for the given length and minimum distance. The Kerdock code has twice as

many codewords as the best known linear code. In the case $m = 3$ the Preparata code and the Kerdock codes both coincide with the Nordstrom–Robinson code.

The Preparata and Kerdock codes are *distance invariant.* This means that the distance distribution from a given codeword to all the other codewords is independent of the given codeword. In particular, since they contain the all-zero codeword, their weight distribution equals their distance distribution.

In general, there is no natural way to define the dual code of a nonlinear code, and thus the MacWilliams identities have no meaning for nonlinear codes. However, one can define the weight enumerator polynomial $A(z)$ of a nonlinear code in the same way as for linear codes and compute its *formal dual* $B(z)$ from the MacWilliams identities:

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

The polynomial $B(z)$ obtained in this way has no simple interpretation. In particular, it may have coefficients that are noninteger or even negative. For example, if $C = \{(110), (101), (111)\}$, then $A(z) = 2z^2 + z^3$ and $B(z) = (3 - 5z + z^2 + z^3)/3$.

An observation that puzzled the coding theory community for a long time was that the weight enumerator of the Preparata code $A(z)$ and the weight enumerator of the Kerdock code $B(z)$ satisfied the MacWilliams identities, and in this sense these nonlinear codes behaved like dual linear codes.

Hammons, Kumar, Calderbank, Sloane, and Solé (*IEEE Trans. Information Theory* **40**: 301–319, 1994) gave a significantly simpler description of the family of Kerdock codes. They constructed a linear code over $Z_4 = \{0, 1, 2, 3\}$, which is an analog of the binary first-order Reed–Muller code. This code is combined with a mapping called the Gray map that maps the elements of Z_4 into binary pairs. The Gray map ϕ is defined by

$$\phi(0) = 00, \phi(1) = 01, \phi(2) = 11, \phi(3) = 10$$

The Lee weight of an element in Z_4 is defined by

$$w_L(0) = 0, w_L(1) = 1, w_L(2) = 2, w_L(3) = 1$$

Extending ϕ in a natural way to a map ϕ: $Z_4^n \to Z_2^{2n}$, one observes that ϕ is a distance preserving map from Z_4^n (under the Lee metric) to Z_2^{2n}, (under the Hamming metric).

A linear code over Z_4 is a subset of Z_4^n such that any linear combination of two codewords is again a codeword. From a linear code $\mathscr{C}$ of length n over Z_4, one obtains a binary code $C = \phi(\mathscr{C})$ of length $2n$ by replacing each component in a codeword in $\mathscr{C}$ by its image under the Gray map. This code is usually nonlinear.

The minimum Hamming distance of C equals the minimum Lee distance of $\mathscr{C}$ and is equal to the minimum Lee weight of $\mathscr{C}$ since $\mathscr{C}$ in linear over Z_4.

Example. To obtain the Nordstrom–Robinson code, we will construct a code over Z_4 of length 8 and then apply the Gray map.

Let $f(x) = x^3 + 2x^2 + x + 3 \in Z_4[x]$. Let β be a zero of $f(x)$; that is, $\beta^3 + 2\beta^2 + \beta + 3 = 0$. Then we can express all powers of β in terms of 1, β, and β^2, as follows:

$$\begin{aligned}
\beta^3 &= 2\beta^2 + 3\beta + 1 \\
\beta^4 &= 3\beta^2 + 3\beta + 2 \\
\beta^5 &= \beta^2 + 3\beta + 3 \\
\beta^6 &= \beta^2 + 2\beta + 1 \\
\beta^7 &= 1
\end{aligned}$$

Consider the code $\mathscr{C}$ over Z_4 with generator matrix given by

$$\begin{aligned}
G &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \beta & \beta^2 & \beta^3 & \beta^4 & \beta^5 & \beta^6 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 2 & 3 & 1 \\ 0 & 0 & 1 & 0 & 3 & 3 & 3 & 2 \\ 0 & 0 & 0 & 1 & 2 & 3 & 1 & 1 \end{bmatrix}
\end{aligned}$$

where the column corresponding to β^i is replaced by the coefficients in its expression in terms of 1, β, and β^2. Then the Nordstrom–Robinson code is the Gray map of $\mathscr{C}$.

The dual code $\mathscr{C}^\perp$ of a code $\mathscr{C}$ over Z_4 is defined similarly as for binary linear codes, except that the inner product of the vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ with components in Z_4 is defined by

$$(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} x_i y_i \pmod 4$$

The dual code $\mathscr{C}^\perp$ of $\mathscr{C}$ is then

$$\mathscr{C}^\perp = \{\boldsymbol{x} \in Z_4^n (\boldsymbol{x}, \boldsymbol{c}) = 0 \text{ for all } \boldsymbol{c} \in Z_4^n\}$$

For a linear code $\mathscr{C}$ over Z_4, there is a MacWilliams relation that determines the Lee weight distribution of the dual code $\mathscr{C}^\perp$ from the Lee weight distribution of $\mathscr{C}$. Therefore, one can compute the relation between the Hamming weight distributions of the nonlinear codes $C = \phi(\mathscr{C})$ and $C_\perp = \phi(\mathscr{C}^\perp)$, and it turns out that the MacWilliams identities hold.

Hence, to find nonlinear binary codes related by the MacWilliams identities, one can start with a pair of Z_4-linear dual codes and apply the Gray map. For any odd integer $m \geq 3$, the Gray map of the code $\mathscr{K}_m$ over Z_4 with generator matrix

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & \beta & \beta^2 & \cdots & \beta^{2^m-2} \end{bmatrix}$$

is the binary nonlinear $(2^{m+1}, 2^{2m+2}, 2^m - 2^{(m-1)/2})$ Kerdock code. The Gray map of $\mathscr{K}_m^\perp$ has the same weight distribution as the $(2^{m+1}, 2^{2^{m+1}-2m-2}, 6)$ Preparata code. It is, however, not identical to the Preparata code and is therefore denoted the "Preparata" code. Hence the Kerdock code and the "Preparata" code are the Z_4-analogy of the first-order Reed–Muller code and the extended Hamming code, respectively.

Hammons, Kumar, Calderbank, Sloane, and Solé also showed that the binary code defined by $C = \phi(\mathscr{C})$, where $\mathscr{C}$ is

the quaternary code with parity-check matrix given by

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & \beta & \beta^2 & \cdots & \beta^{2^m-2} \\ 0 & 2 & 2\beta^3 & 2\beta^6 & \cdots & 2\beta^{3(2^m-2)} \end{bmatrix}$$

is a binary nonlinear $(2^{m+1}, 2^{2^{m+1}-3m-2}, 8)$ code whenever $m \geq 3$ is odd. This code has the same weight distribution as the Goethals code, which is a nonlinear code that has four times as many codewords as the comparable linear extended triple-error-correcting primitive BCH code. The code $C_\perp = \phi(\mathscr{C}^\perp)$ is identical to a binary nonlinear code that was constructed in a much more complicated way by Delsarte and Goethals more than 20 years ago.

To analyze codes obtained from codes over Z_4 in this manner, one is led to study Galois rings instead of Galois fields. Similar to a Galois field, a Galois ring can be defined as $Z_{p^e}[x]/(f(x))$, where $f(x)$ is a monic polynomial of degree m that is irreducible modulo p. The richness in structure of the Galois rings has led to several recently discovered good nonlinear codes that have an efficient and fast decoding algorithm.

Reading List

R. Blahut, *The Theory and Practice of Error Control Codes.* Reading, MA: Addison-Wesley, 1983.

R. Hill, *A First Course in Coding Theory.* Oxford: Clarendon Press, 1986.

T. Kløve and V. I. Korzhik, *Error-Detecting Codes,* Boston, MA: Kluwer Academic, 1995.

R. Lidl and H. Niederreiter, *Finite Fields,* vol. 20 of *Encyclopedia of Mathematics and Its Applications.* Reading, MA: Addison-Wesley, 1983.

S. Lin and D. J. Costello, Jr., *Error Control Coding, Fundamentals and Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes,* Amsterdam: North-Holland, 1977.

W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes.* Cambridge, MA: MIT Press, 1972.

M. Purser, *Introduction to Error-Correcting Codes,* Boston, MA: Artech House, 1995.

J. H. van Lint, *Introduction to Coding Theory,* New York: Springer-Verlag, 1982.

H. van Tilborg, *Error-Correcting Codes—A First Course,* Lund: Studentlitteratur, 1993.

S. A. Vanstone and P. C. van Oorschot, *An Introduction to Error-Correcting Codes with Applications,* Boston, MA: Kluwer Academic, 1989.

TOR HELLESETH TORLEIV KLØVE
University of Bergen

ALGEBRA, LINEAR. See LINEAR ALGEBRA.

ALGEBRA, PROCESS. See PROCESS ALGEBRA.

ALGORITHMIC DIFFERENTIATION. See AUTOMATIC DIFFERENTIATION.

ALGORITHMS. See DIVIDE AND CONQUER METHODS.

ALGORITHMS AND DATA STRUCTURES. See DATA STRUCTURES AND ALGORITHMS.

ALGORITHMS FOR BACKTRACKING. See BACKTRACKING.

ALGORITHMS FOR RECURSION. See RECURSION.

ALGORITHMS, GENETIC. See GENETIC ALGORITHMS.

ALGORITHMS, MULTICAST. See MULTICAST ALGORITHMS.

ALGORITHMS, ONLINE. See ONLINE OPERATION.