Chapter 4

Block codes

Block coding involves associating with a data block $\mathbf{d}$ of k symbols coming from the information source, a block $\mathbf{c}$, called the codeword, of n symbols with $n \geq k$. The $(n-k)$ is the amount of redundancy introduced by the code. Knowledge of the coding rule at reception enables errors to be detected and corrected, under certain conditions. The ratio k/n is called the coding rate of the code.

The message symbols of the information $\mathbf{d}$ and of the codeword $\mathbf{c}$ take their values in a finite field $\mathbf{F}_q$ with q elements, called a Galois field, whose main properties are given in the appendix to this chapter. We shall see that for most codes, the symbols are binary and take their value in the field $\mathbf{F}_2$ with two elements (0 and 1). This field is the smallest Galois field.

The elementary addition and multiplication operations in field F_2 are resumed in Table 4.1.

a	b	$a + b$	ab
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 4.1 – Addition and multiplication in the Galois field $\mathbf{F}_2$

A block code of length n is an application g of the set $\mathbf{F}_q^k$ towards the set $\mathbf{F}_q^n$ that associates a codeword $\mathbf{c}$ with any block of data $\mathbf{d}$.

$$\begin{array}{rccc} g: & \mathbf{F}_q^k & \rightarrow & \mathbf{F}_q^n \\ & \mathbf{d} & \mapsto & \mathbf{c} = g(\mathbf{d}) \end{array}$$

The set of q^k codewords generally constitutes a very reduced subset of $\mathbf{F}_q^n$.

A block code with parameters (n, k), that we denote $C(n, k)$, is linear if the codewords are a vector subspace of $\mathbf{F}_q^n$, that is, if g is a linear application. A direct consequence of linearity is that the sum of two codewords is a codeword, and that the null word made up of n symbols at zero is always a codeword.

We will now consider linear block codes with binary symbols. Linear block codes with non binary symbols will be addressed later.

4.1 Block codes with binary symbols

In the case of a binary block code, the elements of $\mathbf{d}$ and $\mathbf{c}$ have values in $\mathbf{F}_2$. As g is a linear application, we will be able to describe the coding operation simply as the result of the multiplication of a vector of k symbols representing the data to be coded by a matrix representative of the code considered, called a code generator matrix.

4.1.1 Generator matrix of a binary block code

Let us denote $\mathbf{d} = [d_0 \cdots d_j \cdots d_{k-1}]$ and $\mathbf{c} = [c_0 \cdots c_j \cdots c_{n-1}]$ the data-word and the associated codeword. Expressing the vector $\mathbf{d}$ from a base $(\mathbf{e}_0, \ldots, \mathbf{e}_j, \ldots, \mathbf{e}_{k-1})$ of $\mathbf{F}_2^k$, we can write:

$$\mathbf{d} = \sum_{j=0}^{k-1} d_j \mathbf{e}_j \tag{4.1}$$

Taking into account the fact that application g is linear, the word $\mathbf{c}$ associated with $\mathbf{d}$ is equal to:

$$\mathbf{c} = g(\mathbf{d}) = \sum_{j=0}^{k-1} d_j g(\mathbf{e}_j) \tag{4.2}$$

Expressing the vector $g(\mathbf{e}_j)$ from a base $(\mathbf{e'}_0, \cdots, \mathbf{e'}_l, \cdots, \mathbf{e'}_{n-1})$ of $\mathbf{F}_2^n$, we obtain:

$$g(\mathbf{e}_j) = \sum_{l=0}^{n-1} g_{jl} \mathbf{e'}_l \tag{4.3}$$

The vectors $g(\mathbf{e}_j) = \mathbf{g}_j = (g_{j0} \cdots g_{jl} \cdots g_{j,n-1})$, $0 \leq j \leq k-1$ represent the k rows of matrix $\mathbf{G}$ associated with the linear application g.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \vdots \\ \mathbf{g_j} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = \begin{bmatrix} g_{0,0} & \cdots & g_{0,l} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{j,0} & \cdots & g_{j,l} & \cdots & g_{j,n-1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,l} & \cdots & g_{k-1,n-1} \end{bmatrix} \tag{4.4}$$

Matrix $\mathbf{G}$ with k rows and n columns, having its elements $g_{jl} \in \mathbf{F}_2$ is called a generator matrix of the code $C(n,k)$. It associates the codeword $\mathbf{c}$ with the block of data $\mathbf{d}$ by the matrix relation:

$$\mathbf{c} = \mathbf{dG} \tag{4.5}$$

The generator matrix of a block code is not unique. Indeed, by permuting the vectors of the base $(\mathbf{e'}_0, \ldots, \mathbf{e'}_l, \ldots, \mathbf{e'}_{n-1})$ or of the base $(\mathbf{e}_0, \ldots, \mathbf{e}_j, \ldots, \mathbf{e}_{k-1})$, we obtain a new generator matrix $\mathbf{G}$ whose columns or rows have also been permuted. Of course, the permutation of the columns or the rows of the generator matrix always produces the same set of codewords; what changes is the association between the codewords and the k-uplets of data.

Note that the rows of the generator matrix of a linear block code are independent codewords, and that they make up a base of the vector subspace generated by the code. The generator matrix of a linear block code is therefore of rank k. A direct consequence is that any family made up of k independent codewords can be used to define a generator matrix of the code considered.

Example 4.1

Let us consider a linear block code called the parity check code denoted $C(n,k)$, with $k = 2$ and $n = k + 1 = 3$ (for a parity check code, the sum of the symbols of a codeword is equal to zero). We have four codewords:

Dataword	Codeword
00	000
01	011
10	101
11	110

To write a generator matrix of this code, let us consider, for example, the canonical base of $\mathbf{F}_2^2$:

$$\mathbf{e}_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}, \mathbf{e}_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

and the canonical base of $\mathbf{F}_2^3$:

$$\mathbf{e'}_0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \mathbf{e'}_1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \mathbf{e'}_2 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

We can write:

$$g(\mathbf{e}_0) = [101] = 1.\mathbf{e'}_0 + 0.\mathbf{e'}_1 + 1.\mathbf{e'}_2$$
$$g(\mathbf{e}_1) = [011] = 0.\mathbf{e'}_0 + 1.\mathbf{e'}_1 + 1.\mathbf{e'}_2$$

A generator matrix of the parity check code is therefore equal to :

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

By permuting the first two vectors of the canonical base of $\mathbf{F}_2^3$, we obtain a new generator matrix of the same parity check code:

$$\mathbf{G}' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

In this example, we have just seen that the generator matrix of a block code is not unique. By permuting the rows or the columns of a generator matrix or by adding one or several other rows to a row, which means considering a new base in $\mathbf{F}_2^n$, it is always possible to write a generator matrix of a block code in the following form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{0,1} & \cdots & p_{0,l} & \cdots & p_{0,n-k} \\ 0 & 1 & \cdots & 0 & p_{1,1} & \cdots & p_{1,l} & \cdots & p_{1,n-k} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & 0 & \ldots & 1 & p_{k-1,1} & \ldots & p_{k-1,l} & \ldots & p_{k-1,n-k} \end{bmatrix} \quad (4.6)$$

where $\mathbf{I}_k$ is the identity matrix $k \times k$ and $\mathbf{P}$ a matrix $k \times (n-k)$ used to calculate the $(n-k)$ redundancy symbols.
Written thus, the generator matrix $\mathbf{G}$ is in a reduced form and produces codewords of the form:

$$\mathbf{c} = \begin{bmatrix} \mathbf{d} & \mathbf{d}P \end{bmatrix} \quad (4.7)$$

The code is therefore systematic. Following 4.7, the code is said to be systematic when there exist k indices $i_0, i_1, \ldots, i_{k-1}$, such that for any data word $\mathbf{d}$, the associated codeword $\mathbf{c}$ satisfies the relation:

$$c_{i_q} = d_q, \qquad q = 0, 1, \cdots, k-1.$$

4.1.2 Dual code and parity check matrix

Before tackling the notion of dual code, let us define the orthogonality between two vectors made up of n symbols. Two vectors $\mathbf{x} = [x_0 \cdots x_j \cdots x_{n-1}]$ and $\mathbf{y} = [y_0 \cdots y_j \cdots y_{n-1}]$ are orthogonal ($\mathbf{x} \perp \mathbf{y}$) if their scalar product denoted $\langle \mathbf{x}, \mathbf{y} \rangle$ is null.

$$\mathbf{x} \perp \mathbf{y} \Leftrightarrow \langle \mathbf{x}, \mathbf{y} \rangle = \sum_{j=0}^{n-1} x_j y_j = \mathbf{0}$$

With each linear block code $C(n,k)$, we can associate a dual linear block code that verifies that any word of the dual code is orthogonal to any word of the code $C(n,k)$. The dual of code $C(n,k)$ is therefore a vector subspace of $\mathbf{F}_2^n$ made up of 2^{n-k} codewords of n symbols. This vector subspace is the orthogonal

of the vector subspace made up of 2^k words of the code $C(n,k)$. It results that any word $\mathbf{c}$ of code $C(n,k)$ is orthogonal to the rows of the generator matrix $\mathbf{H}$ of its dual code

$$\mathbf{cH^T} = \mathbf{0} \tag{4.8}$$

where T indicates the transposition.

A vector $\mathbf{y}$ belonging to $\mathbf{F}_2^n$ is therefore a codeword of code $C(n,k)$ if, and only if, it is orthogonal to the codewords of its dual code, that is, if:

$$\mathbf{yH^T} = \mathbf{0}$$

The decoder of a code $C(n,k)$ can use this property to verify that the word received is a codeword and thus to detect the presence of errors. That is why matrix $\mathbf{H}$ is called the parity check matrix of code $C(n,k)$.

It is easy to see that the matrices $\mathbf{G}$ and $\mathbf{H}$ are orthogonal ($\mathbf{GH^T} = \mathbf{0}$). Hence, when the code is systematic and its generator matrix is of the form $\mathbf{G} = [\mathbf{I}_k \ \ \mathbf{P}]$, we have:

$$\mathbf{H} = [\mathbf{P^T} \ \mathbf{I}_{n-k}] \tag{4.9}$$

4.1.3 Minimum distance

Before recalling what the minimum distance of a linear block code is, let return to the notion of Hamming distance that measures the difference between two codewords. The Hamming distance, denoted d_H, is equal to the number of places where the two codewords have different symbols.

We can also define the Hamming weight, denoted w_H, of a codeword as the number of non-null symbols of this codeword. Thus, the Hamming distance between two codewords is also equal to the weight of their sum.

Example 4.2

Let there be two words $\mathbf{u} = [1101001]$ and $\mathbf{v} = [0101101]$. The Hamming distance between $\mathbf{u}$ and $\mathbf{v}$ is 2. Their sum $\mathbf{u} + \mathbf{v} = [1000100]$ has a Hamming weight 2.

The minimum distance $d_{\min}$ of a block code is equal to the smallest Hamming distance between its codewords.

$$d_{\min} = \min_{\mathbf{c} \neq \mathbf{c}'} d_H(\mathbf{c}, \mathbf{c}'), \qquad \forall \mathbf{c}, \mathbf{c}' \in C(n,k) \tag{4.10}$$

Taking into account the fact that the distance between two codewords is equal to the weight of their sum, the minimum distance of a block code is also equal to the minimum weight of its non-null codewords.

$$d_{\min} = \min_{\mathbf{c}\neq 0, \mathbf{c}\in C(n,k)} w_H(\mathbf{c}) \tag{4.11}$$

When the number of codewords is very high, searching for the minimum distance can be laborious. A first solution to get round this difficulty is to determine the minimum distance from the parity check matrix.

We have seen that $d_{\min}$ is equal to the minimum Hamming weight of the non-null codewords. Let us consider a codeword of weight $d_{\min}$. The orthogonality property $\mathbf{cH}^{\mathrm{T}} = \mathbf{0}$ implies that the sum of $d_{\min}$ columns of the parity check matrix is null. Thus $d_{\min}$ corresponds to the minimum number of linearly dependent columns of the parity check matrix.

A second solution to evaluate $d_{\min}$ is to use higher bounds of the minimum distance. A first bound can be expressed as a function of the k and n parameters of the code. For a linear block code whose generator matrix is written in the systematic form $\mathbf{G} = [\mathbf{I}_k \ \mathbf{P}]$, the $(n-k)$ columns of the matrix $\mathbf{I}_{n-k}$ of the parity check matrix ($\mathbf{H} = \left[\mathbf{P}^{\mathrm{T}} \ \mathbf{I}_{n-k}\right]$) being linearly independent, any column of $\mathbf{P}^{\mathrm{T}}$ can be expressed as at most a combination of these $(n-k)$ columns. The minimum distance is therefore upper bounded by:

$$d_{\min} \leq n - k + 1 \tag{4.12}$$

Another bound of the minimum distance, called the Plotkin bound, can be obtained by noting that the minimum distance is necessarily lower than the average weight of the non-null codewords. If we consider the set of codewords, it is easy to see that there are as many symbols at 0 as symbols at 1. Thus the sum of the weights of all the codewords is equal to $n2^{k-1}$. The number of non-null codewords being $2^k - 1$, the minimum distance can be upper bounded by:

$$d_{\min} \leq \frac{n2^{k-1}}{2^k - 1} \tag{4.13}$$

4.1.4 Extended codes and shortened codes

From a block code $C(n,k)$ with minimum distance $d_{\min}$ we can build a linear code $C(n+1,k)$ by adding to the end of each codeword a symbol equal to 1 (respectively to 0) if the codeword includes an odd (respectively even) number of 1s. This code is called an extended code and its minimum distance is equal to $d_{\min}+1$ if $d_{\min}$ is an odd number.

The parity check matrix $\mathbf{H}_e$ of an extended code is of the form:

$$\mathbf{H}_e = \begin{bmatrix} & & & 0 \\ & \mathbf{H} & & \vdots \\ & & & 0 \\ 1 & \cdots & 1 & 1 \end{bmatrix}$$

where $\mathbf{H}$ is the parity check matrix of code $C(n,k)$.

A systematic block code $C(n,k)$ with minimum distance $d_{\min}$ can be shortened by setting $s < k$ data symbols to zero. We thus obtain a systematic linear code $C(n-s,k-s)$. Of course the s symbols set to zero are not transmitted, but they are retained in order to calculate the $(n-k)$ redundancy symbols. The minimum distance of a shortened code is always higher than or equal to the distance of code $C(n,k)$.

4.1.5 Product codes

A product code is a code with several dimensions built from elementary codes. To illustrate these codes, let us consider a product code built from two systematic block codes $C_1(n_1,k_1)$ and $C_2(n_2,k_2)$.

Let there be a table with n_2 rows and n_1 columns. The k_2 first rows are filled with codewords of length n_1 generated by the code $C_1(n_1,k_1)$. The remaining (n_2-k_2) rows are filled by the redundancy symbols generated by the code $C_2(n_2,k_2)$; the k_2 symbols of each of the n_1 columns being the information bits of the code $C_2(n_2,k_2)$. We can show that the (n_2-k_2) rows of the table are codewords of code $C_1(n_1,k_1)$. It results that all the rows of the table are codewords of $C_1(n_1,k_1)$ and all the columns of the table are codewords of $C_2(n_2,k_2)$.

The parameters of the two-dimensional product code $C(n,k)$ with minimum distance $d_{\min}$ are equal to the product of the parameters of the elementary codes.

$$n = n_1 n_2 \qquad k = k_1 k_2 \qquad d_{\min} = d^1_{\min} d^2_{\min}$$

where $d^1_{\min}$ and $d^2_{\min}$ are the minimum distances of codes $C_1(n_1,k_1)$ and $C_2(n_2,k_2)$ respectively.

A two-dimensional product code can be seen as a double serial concatenation of two elementary codes (see Chapter 6). An encoder C_1 is fed with k_2 data blocks of length k_1 and it produces k_2 codewords of length n_1 that are written in rows in a matrix. The matrix is read column-wise and produces n_1 blocks of symbols of length k_2 that feed an encoder C_2. The latter in turn produces n_1 codewords of length n_2. Figure 4.1 illustrates the implementation of a two-dimensional product code built from two systematic block codes.

4.1.6 Examples of binary block codes

Parity check code

This code uses a redundancy binary symbol ($n = k+1$) determined in such a way as to ensure the nullity of the modulo 2 addition of the symbols of each codeword.

$$\mathbf{c} = \left[\begin{array}{ccccccc} d_0 & d_1 & \cdots & d_{k-2} & d_{k-1} & c_{n-1} \end{array} \right] \quad \text{with} \quad c_{n-1} = \sum_{j=0}^{k-1} d_j$$

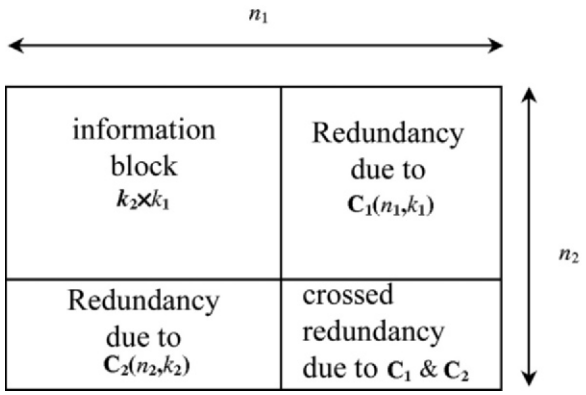

Figure 4.1 – Product code resulting from the serial concatenation of two systematic block codes.

where $\mathbf{d} = \begin{bmatrix} d_0 & d_1 & \cdots & d_{k-1} \end{bmatrix}$ represents the dataword. The minimum distance of this code is 2.

Example 4.3

A generator matrix $\mathbf{G}$ of this code for $n = 5$, $k = 4$ is equal to:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_4 & \mathbf{P} \end{bmatrix}$$

and the parity check matrix $\mathbf{H}$ is reduced to one vector.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_1 \end{bmatrix}$$

Repetition code

For this code with parameters $k = 1$ and $n = 2m + 1$, each bit coming from the information source is repeated an odd number of times. The minimum distance of this code is $2m + 1$. The repetition code $C(2m + 1, 1)$ is the dual code of the parity check code $C(2m + 1, 2m)$.

Example 4.4

The generator matrix and the parity check matrix of this code, for $k = 1$, $n = 5$, can be the following:

$$\mathbf{G} = \left[\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{I}_1 & \mathbf{P} \end{array} \right]$$

$$\mathbf{H} = \left[\begin{array}{ccccc} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_4 \end{array} \right]$$

Hamming code

For a Hamming code, the columns of the parity check matrix are the binary representations of the numbers from 1 to n. Each column being made up of $m = (n-k)$ binary symbols, the parameters of the Hamming code are therefore:

$$n = 2^m - 1 \qquad k = 2^m - m - 1$$

The columns of the parity check matrix being made up of all the possible combinations of $(n-k)$ binary symbols except $(00\cdots0)$, the sum of two columns is equal to one column. The minimum number of linearly dependent columns is 3. The minimum distance of a Hamming code is therefore equal to 3, whatever the value of parameters n and k.

Example 4.5

Let there be a Hamming code with parameter $m = 3$. The codewords and the datawords are then made up of $n = 7$ and $k = 4$ binary symbols respectively. The parity check matrix can be the following:

$$\mathbf{H} = \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{P}^{\mathrm{T}} & \mathbf{I}_3 \end{array} \right]$$

and the corresponding generator matrix is equal to:

$$\mathbf{G} = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{I}_4 & \mathbf{P} \end{array} \right]$$

Maximum length code

The columns of the generator matrix of a maximum length code are the binary representations of the numbers from 1 to n. The parameters of this code are therefore $n = 2^m - 1, k = m$ and we can show that its minimum distance is 2^{k-1}. The maximum length code with parameters $n = 2^m - 1, k = m$ is the dual code of the Hamming code with parameters $n = 2^m - 1, k = 2^m - m - 1$, that is, the generator matrix of the one is the parity check matrix of the other.

Hadamard code

The codewords of a Hadamard code are made up of the rows of a Hadamard matrix and of its complementary matrix. A Hadamard matrix has n rows and n columns (n even) whose elements are 1s and 0s. Each row differs from the other rows at $n/2$ positions. The first row of the matrix is made up only of 0, the other rows having $n/2$ 0 and $n/2$ 1.
For $n = 2$, the Hadamard matrix is of the form:

$$\mathbf{M}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

From a $\mathbf{M}_n$ matrix we can generate a $\mathbf{M}_{2n}$ matrix.

$$\mathbf{M}_{2n} = \begin{bmatrix} \mathbf{M}_n & \mathbf{M}_n \\ \mathbf{M}_n & \overline{\mathbf{M}}_n \end{bmatrix}$$

where $\overline{\mathbf{M}}_n$ is the complementary matrix of $\mathbf{M}_n$, that is, where each element at 1 (respectively at 0) of $\mathbf{M}_n$ becomes an element at 0 (respectively at 1) for $\overline{\mathbf{M}}_n$.

Example 4.6

If $n = 4$ $\mathbf{M}_4$ and $\overline{\mathbf{M}}_4$ have the form:

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad \overline{\mathbf{M}}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

The rows of $\mathbf{M}_4$ and $\overline{\mathbf{M}}_4$ are the codewords of a Hadamard code with parameters $n = 4$, $k = 3$ and with minimum distance equal to 2. In this particular case, the Hadamard code is a parity check code.

More generally, the rows of matrices $\mathbf{M}_n$ and $\overline{\mathbf{M}}_n$ are the codewords of a Hadamard code with parameters $n = 2^m, k = m+1$ and with minimum distance $d_{\min} = 2^{m-1}$.

Reed-Muller codes

A Reed-Muller code (RM) of order r and with parameter m, denoted $\mathrm{RM}_{r,m}$, has codewords of length $n = 2^m$ and the datawords are made up of k symbols with:

$$k = 1 + \begin{pmatrix} m \\ 1 \end{pmatrix} + \cdots + \begin{pmatrix} m \\ r \end{pmatrix}, \quad \text{with} \quad \begin{pmatrix} N \\ q \end{pmatrix} = \frac{N!}{q!\,(N-q)!}$$

where $r < m$. The minimum distance of an RM code is $d_{\min} = 2^{m-r}$.
The generator matrix of an RM code of order r is built from the generator matrix of an RM code of order $r-1$ and if $\mathbf{G}^{(r,m)}$ represents the generator matrix of the Reed-Muller code of order r and with parameter m, it can be obtained from $\mathbf{G}^{(r-1,m)}$ by the relation:

$$\mathbf{G}^{(r,m)} = \begin{bmatrix} \mathbf{G}^{(r-1,m)} \\ \mathbf{Q}_r \end{bmatrix}$$

where $\mathbf{Q}_r$ is a matrix with dimensions $\begin{pmatrix} m \\ r \end{pmatrix} \times n$.

By construction, $\mathbf{G}^{(0,m)}$ is a row vector of length n whose elements are equal to 1. The matrix $\mathbf{G}^{(1,m)}$ is obtained by writing on each column the binary representation of the index of the columns (from 0 to $n-1$). For example, for $m = 4$, the matrix $\mathbf{G}^{(1,m)}$ is given by:

$$G^{(1,4)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Matrix $\mathbf{Q}_r$ is obtained simply by considering all the combinations of r rows of $\mathbf{G}^{(1,m)}$ and by obtaining the product of these vectors, component by component. The result of this multiplication constitutes a row of $\mathbf{Q}_r$. For example, for the combination having the rows of $\mathbf{G}^{(1,m)}$ with indices i_1, i_2, $\ldots$, i_r, the j-th coefficient of the row thus obtained is equal to $G^{(1,m)}_{i_1,j} G^{(1,m)}_{i_2,j} \cdots G^{(1,m)}_{i_r,j}$, the multiplication being carried out in the field $\mathbf{F}_2$. For example, for $r = 2$, we obtain:

$$\mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We can show that the code $\mathrm{RM}_{m-r-1,m}$ is the dual code of the code $\mathrm{RM}_{r,m}$, that is, the generator matrix of code $\mathrm{RM}_{m-r-1,m}$ is the parity check matrix of

code $\mathrm{RM}_{r,m}$. For some values of r and m, the generator matrix of code $\mathrm{RM}_{r,m}$ is also its parity check matrix. We then say that code $\mathrm{RM}_{r,m}$ is *self dual*. Code $\mathrm{RM}_{1,3}$, for example, is a *self dual* code.

4.1.7 Cyclic codes

Cyclic codes are the largest class of linear block codes. Their relatively easy implementation, from shift registers and logical operators, has made them attractive and widely-used codes.

Definition and polynomial representation

A linear block code $C(n,k)$ is cyclic if, for any codeword $\mathbf{c} = \begin{bmatrix} c_0 & c_1 & \cdots & c_{n-1} \end{bmatrix}$, $\mathbf{c}_1 = \begin{bmatrix} c_{n-1} & c_0 & \cdots & c_{n-2} \end{bmatrix}$, obtained by circular shift to the right of a symbol of $\mathbf{c}$, is also a codeword. This definition of cyclic codes means that any circular shift to the right of j symbols of a codeword, gives another codeword.
For cyclic codes, we use a polynomial representation of the codewords and of the datawords. Thus, with codeword $\mathbf{c}$ we associate the polynomial $c(x)$ of degree $n-1$.

$$c(x) = c_0 + c_1 x + \cdots + c_j x^j + \cdots + c_{n-1} x^{n-1}$$

and with dataword $\mathbf{d}$ the polynomial $d(x)$ of degree $k-1$.

$$d(x) = d_0 + d_1 x + \cdots + d_j x^j + \cdots + d_{k-1} x^{k-1}$$

where d_j and c_j take their values in $\mathbf{F}_2$.
Multiplying $c(x)$ by x,

$$xc(x) = c_0 x + c_1 x^2 + \cdots + c_j x^{j+1} + \cdots + c_{n-1} x^n$$

then dividing $xc(x)$ by $x^n + 1$, we obtain:

$$xc(x) = (x^n + 1)c_{n-1} + c_1(x)$$

where $c_1(x)$ is the remainder of the division of $xc(x)$ by $x^n + 1$ with:

$$c_1(x) = c_{n-1} + c_0 x + \cdots c_j x^{j+1} + \cdots c_{n-2} x^{n-1}$$

We can note that $c_1(x)$ corresponds to the codeword $\mathbf{c}_1 = (c_{n-1} c_0 \ldots c_j \ldots c_{n-2})$.
Using the same method as above, we obtain:

$$x^j c(x) = (x^n + 1)q(x) + c_j(x) \tag{4.14}$$

where $c_j(x)$ is also a codeword obtained by j circular shifts to the right of the symbols of $c(x)$.

The codewords of a cyclic code are multiples of a normalized polynomial $g(x)$ of degree $(n-k)$ called a generator polynomial .

$$g(x) = g_0 + g_1 x + \cdots + g_j x^j + \cdots + x^{n-k}$$

where g_j takes its values in $\mathbf{F}_2$. The generator polynomial of a cyclic code is a divisor of $x^n + 1$. There exists a polynomial $h(x)$ of degree k such that equation (4.15) is satisfied.

$$g(x)h(x) = x^n + 1 \tag{4.15}$$

The product $d(x)g(x)$ is a polynomial of degree lower than or equal to $n-1$, so it can represent a codeword. The polynomial $d(x)$ having 2^k realizations, $d(x)g(x)$ enables 2^k codewords to be generated. Let us denote $d_l(x)$ the l-th realization of $d(x)$ and $c_l(x)$ the polynomial representation of the associated codeword. We can write:

$$c_l(x) = d_l(x)g(x) \tag{4.16}$$

We will now show that the codewords satisfying relation (4.16) satisfy the properties of cyclic codes. To do so, we re-write relation (4.14) in the form:

$$c_j(x) = (x^n + 1)q(x) + x^j c(x) \tag{4.17}$$

Since $c(x)$ represents a codeword, there exists a polynomial $d(x)$ of degree at most $k-1$, such that $c(x) = d(x)g(x)$. Using (4.15), we can therefore express (4.17) in another way:

$$c_j(x) = g(x)[h(x)q(x) + x^j d(x)] \tag{4.18}$$

The codewords $c_j(x)$ are therefore multiples of the generator polynomial, and they can be generated from the $d_j(x)$ by applying relation (4.16).

• *Generator polynomial of the dual code of $C(n,k)$*

The dual code of a cyclic block code is also cyclic. Polynomial $h(x)$ of degree k can be used to build the dual code of $C(n,k)$. The reciprocal polynomial $\tilde{h}(x)$ of $h(x)$ is defined as follows:

$$\tilde{h}(x) = x^k h(x^{-1}) = 1 + h_{k-1}x + h_{k-2}x^2 + \cdots + h_1 x^{k-1} + x^k$$

We can write (4.15) differently:

$$[x^{n-k}g(x^{-1})][x^k h(x^{-1})] = x^n + 1 \tag{4.19}$$

The polynomial $\tilde{h}(x)$ is also a divisor of $x^n + 1$; it is the generator polynomial of a $C^\perp = C(n, n-k)$ code that is the dual code of $C(n,k)$.

Note: the code of generator polynomial $h(x)$ is equivalent to dual code $C^\perp$. The vector representation of the codewords generated by $h(x)$ corresponds to the reversed vector representation of the codewords of $C^\perp$.

$$\begin{array}{lcl} C^\perp, \text{ generated by } \tilde{h}(x) & \leftrightarrow & \text{Code generated by } h(x) \\ \tilde{\mathbf{c}} = \left[\begin{array}{cccc} c_0 & c_1 & \cdots & c_{n-1} \end{array}\right] & \leftrightarrow & \mathbf{c} = \left[\begin{array}{cccc} c_{n-1} & \cdots & c_1 & c_0 \end{array}\right] \end{array}$$

• *Generator matrix of a cyclic code*

From the generator polynomial $g(x)$ it is possible to build a generator matrix $\mathbf{G}$ of code $C(n,k)$. We recall that the k rows of the matrix $\mathbf{G}$ are made up of k linearly independent codewords. These k codewords can be obtained from a set of k independent polynomials of the form:

$$x^j g(x) \qquad j = k-1, k-2, \ldots, 1, 0.$$

Let $d(x)$ be the polynomial representation of any dataword. The k codewords generated by the polynomials $x^j g(x)$ have the expression:

$$\mathbf{c}_j(x) = x^j g(x) d(x) \qquad j = k-1, k-2, \cdots, 1, 0$$

and the k rows of the matrix $\mathbf{G}$ have for their elements the binary coefficients of the monomials of $\mathbf{c}_j(x)$.

Example 4.7

Let $C(7,4)$ be the generator polynomial code $g(x) = 1 + x^2 + x^3$. Let us take $d(x) = 1$ for the dataword. The 4 rows of the generator matrix $\mathbf{G}$ are obtained from the 4 codewords $\mathbf{c}_j(x)$.

$$\begin{aligned}
\mathbf{c}_3(x) &= x^3 + x^5 + x^6 \\
\mathbf{c}_2(x) &= x^2 + x^4 + x^5 \\
\mathbf{c}_1(x) &= x + x^3 + x^4 \\
\mathbf{c}_0(x) &= 1 + x^2 + x^3
\end{aligned}$$

A generator matrix of the code $C(7,4)$ is equal to:

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Cyclic code in systematic form

When the codewords are in systematic form, the data coming from the information source are separated from the redundancy symbols. The codeword $c(x)$ associated with dataword $d(x)$ is then of the form:

$$c(x) = x^{n-k} d(x) + v(x) \tag{4.20}$$

where $v(x)$ is the polynomial of degree at most equal to $n-k-1$ associated with the redundancy symbols.

Taking into account the fact that $c(x)$ is a multiple of the generator polynomial and that the addition and the subtraction can be merged in $\mathbf{F}_2$, we can then write:

$$x^{n-k}d(x) = q(x)g(x) + v(x)$$

$v(x)$ is therefore the remainder of the division of $x^{n-k}d(x)$ by the generator polynomial $g(x)$. The codeword associated with dataword $d(x)$ is equal to $x^{n-k}d(x)$ increased by the remainder of the division of $x^{n-k}d(x)$ by the generator polynomial.

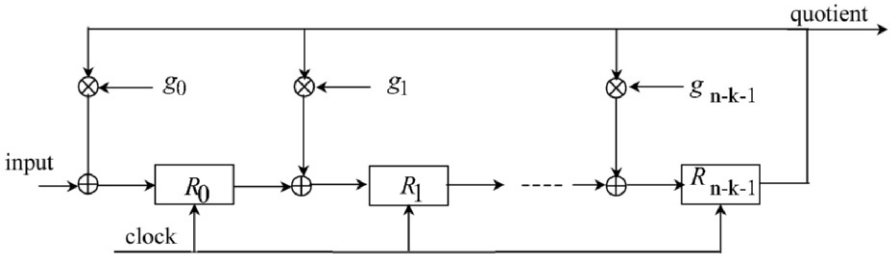

Figure 4.2 – Schematic diagram of a circuit divisor by $g(x)$.

Example 4.8

To illustrate the computation of a codeword written in systematic form, let us take the example of a $C(7,4)$ code of generator polynomial $g(x) = 1+x+x^3$ and let us determine the codeword $c(x)$ associated with message $d(x) = 1+x^2+x^3$, that is:

$$c(x) = x^3d(x) + v(x)$$

The remainder of the division of $x^3d(x)$ by $g(x) = 1 + x + x^3$ being equal to 1, codeword $\mathbf{c}(x)$ associated with dataword $d(x)$ is:

$$c(x) = 1 + x^3 + x^5 + x^6$$

Thus, with data block $\mathbf{d}$, made up of 4 binary information symbols, is associated codeword $\mathbf{c}$ with:

$$\mathbf{d} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \rightarrow \mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

To obtain the generator matrix, it suffices to encode

$$d(x) = \quad 1, \quad x, \quad x^2, \quad x^3.$$

We obtain:

$d(x)$	$c(x)$
1	$1 + x + x^3$
x	$x + x^2 + x^4$
x^2	$1 + x + x^2 + x^5$
x^3	$1 + x^2 + x^6$

and thus the generator matrix in a systematic form:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We can verify that for

$$\mathbf{d} = \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix},$$

the matrix product $\mathbf{dG}$ does give

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Implementation of an encoder

We have just seen that the encoder must carry out the division of $x^{n-k}d(x)$ by the generator polynomial $g(x)$ then add the remainder $v(x)$ of this division to $x^{n-k}d(x)$. This operation can be done using only shift registers and adders in field $\mathbf{F}_2$. As the most difficult operation to carry out is the division of $x^{n-k}d(x)$ by $g(x)$, let us first examine the schematic diagram of a divisor by $g(x)$ shown in Figure 4.2. The circuit divisor is realized from a shift register with $(n-k)$ memories denoted R_i and the same number of adders. The shift register is initialized to zero and the k coefficients of the polynomial $x^{n-k}d(x)$ are introduced sequentially into the circuit divisor. After k clock pulses, we can verify that the result of the division is available at the output of the circuit divisor, as well as the remainder $v(x)$ which is in the shift register memories.

The schematic diagram of the encoder shown in Figure 4.3, uses the circuit divisor of Figure 4.2. The multiplication of $d(x)$ by x^{n-k}, corresponding to a simple shift, is realized by introducing polynomial $d(x)$ at the output of the shift register of the divisor.

The k data coming from the information source are introduced sequentially into the encoder (switch I in position 1) that carries out the division of $x^{n-k}d(x)$ by $g(x)$. Simultaneously, the k data coming from the information source are also transmitted. Once this operation is finished, the remainder $v(x)$ of the division is in the $(n-k)$ shift register memories. Switch I then moves to position 2, and the $(n-k)$ redundancy symbols are sent to the output of the encoder.

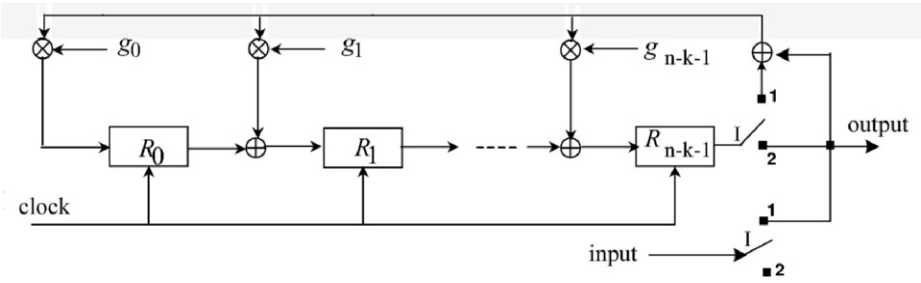

Figure 4.3 – Schematic diagram of an encoder for a cyclic code.

BCH codes

Bose-Chaudhuri-Hocquenghem codes, called BCH codes, enable cyclic codes to be built systematically correcting at least t errors in a block of n symbols, that is, codes whose minimum distance $d_{\min}$ is at least equal to $2t+1$.

To build a BCH code, we set t or equivalently d, called the constructed distance of the code and we determine its generator polynomial $g(x)$. The code obtained has a minimum distance $d_{\min}$ that is always higher than or equal to the constructed distance.

Primitive BCH code

The generator polynomial $g(x)$ of a primitive BCH code constructed over a Galois field $\mathbf{F}_q$ with $q = 2^m$ elements, with a constructed distance d has $(d-1)$ roots of the form: $\alpha^l, \cdots, \alpha^{l+j}, \cdots, \alpha^{l+d-2}$, where $2t+1$ is a primitive element of Galois field $\mathbf{F}_q$ and l an integer. The BCH code is said to be primitive since the roots of its generator polynomial are powers of α, a primitive element of $\mathbf{F}_q$. We will see later that it is possible to build non-primitive BCH codes.

Generally, parameter l is set to 0 or 1 and we show that for a primitive BCH code exponent $(l+d-2)$ of root α^{j+d-2} must be even. When $l=0$, the constructed distance is therefore necessarily even, that is, equal to $2t+2$ for a code correcting t errors. When $l=1$, the constructed distance is odd, that is, equal to $2t+1$ for a code correcting t errors.

- *Primitive BCH code with $l=1$*

  The generator polynomial of a primitive BCH code correcting at least t errors (constructed distance $2t+1$) therefore has $\alpha, \cdots, \alpha^j, \cdots, \alpha^{2t}$ as roots. We show that the generator polynomial $g(x)$ of a primitive BCH code is equal to:

$$g(x) = \text{S.C.M.}\,(m_\alpha(x), \cdots, m_{\alpha^i}(x), \cdots, m_{\alpha^{2t}}(x))$$

where $m_{\alpha^i}(x)$ is the minimal polynomial with coefficients in field $\mathbf{F}_2$ associated with α^j, and S.C.M. is the Smallest Common Multiple.

It is shown in the appendix that a polynomial with coefficients in $\mathbf{F}_2$ having α^j as its root also has α^{2j} as its root. Thus, the minimal polynomials $m_{\alpha^i}(x)$ and $m_{\alpha^{2i}}(x)$ have the same roots. This remark enables us to simplify the writing of generator polynomial $g(x)$.

$$g(x) = \text{S.C.M.}\,(m_\alpha(x), m_{\alpha^3}(x), \cdots, m_{\alpha^{2t-1}}(x)) \tag{4.21}$$

The degree of a minimal polynomial being lower than or equal to m, degree $(n-k)$ of the generator polynomial of a primitive BCH code correcting at least t errors, is therefore lower than or equal to mt. Indeed, $g(x)$ is at most equal to the product of t polynomials of degree lower than or equal to m.

The parameters of a primitive BCH code constructed over a Galois field $\mathbf{F}_q$ with a constructed distance $d = 2t+1$ are therefore the following:

$$n = 2^m - 1; k \geq 2^m - 1 - mt; d_{\min} \geq 2t+1$$

When $t = 1$ a primitive BCH code is a Hamming code. The generator polynomial of a Hamming code, equal to $m_\alpha(x)$, is therefore a primitive polynomial.

Example 4.9

Let us determine the generator polynomial of a BCH code having parameters $m = 4$ and $n = 15$, $t = 2$ and $l = 1$. To do this, we will use a Galois field with $q = 2^4$ elements built from a primitive polynomial of degree $m = 4(\alpha^4 + \alpha + 1)$. The elements of this field are given in the appendix.

We must first determine the minimal polynomials $m_\alpha(x)$ and $m_{\alpha^3}(x)$ associated with elements α and α^3 respectively of field $\mathbf{F}_{16}$.

We have seen in the appendix that if α is a root of polynomial $m_\alpha(x)$ then $\alpha^2, \alpha^4, \alpha^8$ are also roots of this polynomial (raising α to the powers of 16, 32 etc. gives, modulo $\alpha^4+\alpha+1$, elements $\alpha, \alpha^2, \alpha^4, \alpha^8$). We can therefore write:

$$m_\alpha(x) = (x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)$$

Developing the expression of $m_\alpha(x)$ we obtain:

$$m_\alpha(x) = [x^2 + x(\alpha^2+\alpha) + \alpha^3][x^2 + x(\alpha^8+\alpha^4) + \alpha^{12}]$$

Using the binary representations of the elements of field $\mathbf{F}_{16}$, we can show that $\alpha^2+\alpha = \alpha^5$ and that $\alpha^4+\alpha^8 = \alpha^5$ (we recall that the binary additions

are done modulo 2 in the Galois field). We then continue the development of $m_\alpha(x)$ and finally we have:

$$m_\alpha(x) = x^4 + x + 1$$

For the computation of $m_{\alpha^3}(x)$, the roots to take into account are $\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24} = \alpha^9$ ($\alpha^{15} = 1$), and the other powers of α^3 ($\alpha^{48}, \alpha^{96}, \cdots$) give the previous roots again. The minimal polynomial $m_{\alpha^3}(x)$ is therefore equal to:

$$m_{\alpha^3}(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^{12})(x + \alpha^9)$$

which after development and simplification gives:

$$m_{\alpha^3}(x) = x^4 + x^3 + x^2 + x + 1$$

The S.C.M. of polynomials $m_\alpha(x)$ and $m_{\alpha^3}(x)$ is obviously equal to the product of these two polynomials since they are irreducible and thus, the polynomial generator is equal to:

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$$

Developing this, we obtain:

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

Finally the parameters of this BCH code are:

$$m = 4; n = 15; n - k = 8; k = 7; t = 2$$

The numerical values of parameters (n, k, t) of the main BCH codes and the associated generator polynomials have been put table form and can be found in [4.2]. As an example, we give in Table 4.2 the parameters and the generator polynomials, expressed in octals, of some BCH codes with error correction capability $t = 1$ (Hamming codes).

Note : $g(x) = 13$ in octals gives 1011 in binary, that is, $g(x) = x^3 + x + 1$

- *Primitive BCH code with $l = 0$*

  The generator polynomial of a primitive BCH code correcting at least t errors (constructed distance $d = 2t + 2$) has $(2t + 1)$ roots of the form: $\alpha^0, \alpha^1, \cdots, \alpha^j, \cdots, \alpha^{2t}$; that is, one root more (α^0) than when $l = 1$. Taking into account the fact that the minimal polynomials $m_{\alpha^j}(x)$ and $m_{\alpha^{2j}}(x)$ have the same roots, generator polynomial $g(x)$ is equal to:

$$g(x) = \text{S.C.M.}(m_{\alpha^0}(x), m_{\alpha^1}(x), m_{\alpha^3}(x), \cdots, m_{\alpha^{2t-1}}(x))$$

n	k	t	g(x)
7	4	1	13
15	11	1	23
31	26	1	45
63	57	1	103
127	120	1	211
255	247	1	435
511	502	1	1021
1023	1013	1	2011
2047	2036	1	4005
4095	4083	1	10123

Table 4.2 – Parameters of some Hamming codes.

1. Parity check code

   Let us consider a BCH code with $l = 0$ and $t = 0$. Its generator polynomial, $g(x) = (x + 1)$ has only one root $\alpha^0 = 1$. This code uses only one redundancy symbol and the $c(x)$ words of this code satisfy the condition:

   $$c(\alpha^0) = c(1) = 0$$

   This code, which is cyclic since $(x + 1)$ divides $(x^n + 1)$, is a parity check code with parameters $n = k + 1, k, t = 0$. Thus, every time we build a BCH code by selecting $l = 0$, we introduce into the generator polynomial a term in $(x+1)$ and the codewords are of even weight.

2. Cyclic Redundancy Code (CRC)

   Another example of a BCH code for which $l = 0$, is the CRC used for detecting errors. A CRC has a constructed distance of 4 ($t = 1$) and its generator polynomial, from above, is therefore equal to:

   $$g(x) = (x + 1)m_\alpha(x)$$

   α being a primitive element, $m_\alpha(x)$ is a primitive polynomial and thus the generator polynomial of a CRC is a code equal to the product of $(x + 1)$ by the generator polynomial of a Hamming code.

   $$g_{\mathrm{CRC}}(x) = (x + 1)g_{\mathrm{Hamming}}(x)$$

   The parameters of a CRC are therefore:

   $$n = 2^m - 1; (n - k) = m + 1; k = 2^m - m - 2$$

Code	m	$g(x)$
CRC-12	12	14017
CRC-16	16	300005
CRC-CCITT	16	210041
CRC-32	32	40460216667

Table 4.3 – generator polynomials of some codes CRC.

The most widely-used CRC codes have the parameters $m = 12$, 16, 32 and their generator polynomials are given, in octals, in Table 4.3. Note: $g(x) = 14017$ in octals corresponds to 1 100 000 001 111 in binary, that is:

$$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$$

Non-primitive BCH code

The generator polynomial of a non-primitive BCH code (with $l = 1$) correcting at least t errors (constructed distance $d = 2t + 1$) has $2t$ roots of the form: $\beta, \beta^2, \beta^3, \ldots, \beta^{2t}$ where β is a non-primitive element of a Galois field $\mathbf{F}_q$. Taking into account the fact that the minimal polynomials $m_{\beta^j}(x)$ and $m_{\beta^{2j}}(x)$ have the same roots, the generator polynomial of a non-primitive BCH code is equal to:

$$g(x) = \text{S.C.M.}(m_\beta(x), m_{\beta^3}(x).....m_{\beta^{2t-1}}(x))$$

We can show that length n of the words of a non-primitive BCH code is no longer of the form $2^m - 1$ but is equal to p, where p is the exponent of β such that $\beta^p = 1$ (p is the order of β). A Galois field $\mathbf{F}_q$ has non-primitive elements if $2^m - 1$ is not prime. The non-primitive elements are then of the form $\beta = \alpha^\lambda$ where λ is a divisor of $2^m - 1$ and α is a primitive element of the field.

Example 4.10

Let there be a Galois field $\mathbf{F}_q$ with $m = 6$ and $q = 64$. The quantity $2^m - 1 = 63$ is not equal to a prime number; it is divisible by 3, 7, 9, 21 and 63. The non-primitive elements of this field are therefore $\alpha^3, \alpha^7, \alpha^9, \alpha^{21}, \alpha^{63} = 1$. Let us build, for example, a non-primitive BCH code having an error correction capability at least equal to $t = 2$ on field $\mathbf{F}_{64}$ and let us take $\beta = \alpha^3$ as the non-primitive element. We have two minimal polynomials to calculate $m_\beta(x)$ and $m_{\beta^3}(x)$. Taking into account the fact that $\beta^{21} = \alpha^{63} = 1$, the roots of these polynomials are:

$$m_\beta(x) : \text{roots } \beta, \beta^2, \beta^4, \beta^8, \beta^{16}, \beta^{32} = \beta^{11}$$
$$m_{\beta^3}(x) : \text{roots } \beta^3, \beta^6, \beta^{12}$$

The generator polynomial of this code is equal to:

$$g(x) = m_{\beta}(x)m_{\beta^3}(x)$$

which, after development and simplification, gives:

$$g(x) = x^9 + x^8 + x^7 + x^5 + x^4 + x + 1$$

The parameters of this non-primitive BCH code are:

$$n = 21; (n-k) = 9; k = 12$$

- *Golay code*

  Among non-primitive BCH codes, the most well-known is certainly the Golay code constructed over a Galois field $\mathbf{F}_q$ with $m = 11, q = 2048$. Noting that $2^m - 1 = 2047 = 23 \times 89$, the non-primitive element used to build a Golay code is $\beta = \alpha^{89}$. The computation of the generator polynomial of this code constructed on field $\mathbf{F}_{2048}$ leads to the following expression:

  $$g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$$

  We can show that the minimum distance $d_{\min}$ of a Golay code is 7 and thus, its correction capability is 3 errors in a block of 23 binary symbols ($\beta^{23} = \alpha^{2047} = 1$). The parameters of a Golay code are therefore:

  $$n = 23; (n-k) = 11; k = 12; t = 3$$

  Note that the reciprocal polynomial of $g(x)$, equal to $\tilde{g}(x) = x^{11}g(x^{-1})$ also enables a Golay code to be produced.

  $$\tilde{g}(x = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$$

4.2 Block codes with non-binary symbols

4.2.1 Reed-Solomon codes

Reed-Solomon or RS codes are the most well-known and the most widely-used codes having non-binary symbols. For codes with non-binary symbols the coefficients c_j of the codewords and d_j of the datawords take their value in a Galois field $\mathbf{F}_q$ with $q = 2^m$ elements. Thus, each symbol of these codes can be encoded on m binary symbols. Reed-Solomon codes being cyclic codes, they are generated by a generator polynomial $g(x)$ divisor of $x^n + 1$ whose coefficients g_j $j = 0, 1, \cdots, n-k-1$ also take their value in the Galois field $\mathbf{F}_q$.

The generator polynomial of a Reed-Solomon code, with a constructed distance d has $(d-1)$ roots $\alpha^l, \cdots, \alpha^{l+j}, \cdots, \alpha^{l+d-2}$ where α is a primitive element of Galois field $\mathbf{F}_q$. It therefore has the expression:

$$g(x) = \text{S.C.M.}(m_{\alpha^l}(x), \cdots, m_{\alpha^{l+j}}(x), \cdots, m_{\alpha^{l+d-2}}(x))$$

where $m_{\alpha^{l+j}}$ is the minimal polynomial associated with the α^{l+j} element of field $\mathbf{F}_q$.

Using the results of the appendix on the minimal polynomials with coefficients in $\mathbf{F}_q$, the minimal polynomial $m_{\alpha^{l+j}}$ has only one root α^{l+j}.

$$m_{\alpha^{j+i}}(x) = (x + \alpha^{j+i})$$

The generator polynomial of a Reed-Solomon code is therefore of the form:

$$g(x) = (x + \alpha^j)(x + \alpha^{j+1})...(x + \alpha^{j+i})...(x + \alpha^{j+d-2})$$

In general, parameter j is set to 0 or 1 like for binary BCH codes. The generator polynomial of a Reed-Solomon code, of degree $(n - k)$, has $(d - 1)$ roots, that is $n - k = d - 1$. Its constructed distance is therefore equal to:

$$d = n - k + 1$$

For a block code k $C(n, k)$ the minimum distance $d_{\min}$ being lower than or equal to $n - k + 1$, the minimum distance of a Reed-Solomon code is therefore always equal to its constructed distance. A code whose minimum distance is equal to $n - k + 1$ is called a maximum distance code.
The parameters of a Reed-Solomon code correcting t errors in a block of n q-ary symbols are therefore:

$$n = q - 1; \quad n - k = d_{\min} - 1 = 2t; \quad k = n - 2t$$

Example 4.11

Let us determine the generator polynomial of a Reed-Solomon code built from a Galois field with 16 elements having a correction capability of $t = 2$ errors. The minimum distance of this code is therefore $d_{\min} = 5$. Taking for example $l = 1$, the generator polynomial of this code is therefore of the form:

$$g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)$$

Developing the expression above, we obtain:

$$g(x) = \lfloor x^2 + x(\alpha + \alpha^2) + \alpha^3 \rfloor \lfloor x^2 + x(\alpha^3 + \alpha^4) + \alpha^7 \rfloor$$

Using the binary representations of the elements of field $\mathbf{F}_{16}$ (Appendix), the polynomial $g(x)$ after development and simplification is equal to:

$$g(x) = x^4 + \alpha^3 x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10}$$

4.2.2 Implementing the encoder

The schematic diagram of an encoder for Reed-Solomon codes is quite similar to that of an encoder for cyclic codes with binary symbols, but the encoder must now carry out multiplications between q-ary symbols and memorize q-ary symbols.

As an example, we have shown in Figure 4.4 the schematic diagram of the encoder for the Reed-Solomon code treated in the example above.

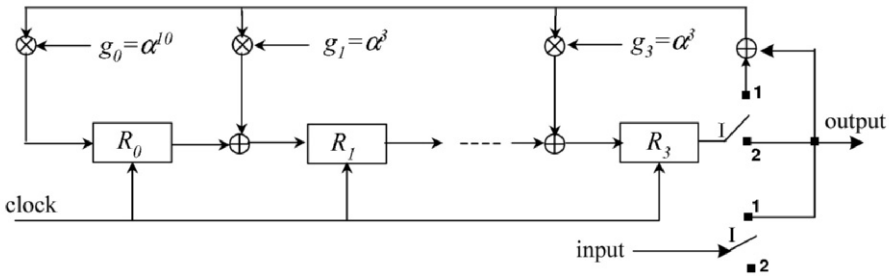

Figure 4.4 – Schematic diagram of the encoder for the RS code (15,11).

4.3 Decoding and performance of codes with binary symbols

4.3.1 Error detection

Considering a binary symmetric transmission channel, the decoder receives binary symbols assumed to be perfectly synchronized with the encoder. This means that the splitting into words having n symbols at the input of the decoder corresponds to the splitting used by the encoder. Thus, in the absence of errors, the decoder sees codewords at its input.

Let us assume that codeword $\mathbf{c}$ is transmitted by the encoder and let $\mathbf{r}$ be the word of n symbols received at the input of the decoder. Word $\mathbf{r}$ can always be written in the form:

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

where $\mathbf{e}$ is a word whose non-null symbols represent the errors. A non-null symbol of $\mathbf{e}$ indicates the presence of an error in the corresponding position of $\mathbf{c}$.

Errors are detected by using the orthogonality property of the parity check matrix with the codewords and calculating the quantity $\mathbf{s}$ called the error syndrome.

$$\mathbf{s} = \mathbf{r}\mathbf{H}^{\mathbf{T}} = (\mathbf{c} + \mathbf{e})\mathbf{H}^{\mathbf{T}} = \mathbf{e}\mathbf{H}^{\mathbf{T}}$$

Syndrome **s** is null if, and only if, **r** is a codeword. A non-null syndrome implies the presence of errors. However, it should be noted that a null syndrome does not necessarily mean absence of errors since **r** can belong to the set of codewords even though it is different from **c**. For this to occur, it suffices for word **e** to be a codeword. Indeed, for a linear block code, the sum of two codewords is another codeword.

Finally, let us note that for any linear block code, there are configurations of non-detectable errors.

Detection capability

Let $\mathbf{c}_j$ be the transmitted codeword and $\mathbf{c}_l$ its nearest neighbour. We have the following inequality:

$$d_H(\mathbf{c}_j, \mathbf{c}_l) \geqslant d_{\min}$$

Introducing the received word **r**, we can write:

$$d_{\min} \leq d_H(\mathbf{c}_j, \mathbf{c}_l) \leq d_H(\mathbf{c}_j, \mathbf{r}) + d_H(\mathbf{c}_l, \mathbf{r})$$

and thus all the errors can be detected if the Hamming distance between **r** and $\mathbf{c}_l$ is higher than or equal to 1, that is, if **r** is not merged with $\mathbf{c}_l$.

The detection capability of a $C(n, k)$ code with minimum distance $d_{\min}$ is therefore equal to $d_{\min} - 1$.

Probability of non-detection of errors

Considering a block code $C(n, k)$ and a binary symmetric channel with error probability p, the probability of non-detection of the errors P_{nd} is equal to:

$$P_{nd} = \sum_{j=d_{\min}}^{n} A_j p^j (1-p)^{n-j} \tag{4.22}$$

where A_j is the number of codewords with weight j.

Examining the hypothesis of a completely degraded transmission, that is, of an error probability of $p = 1/2$ on the channel, and taking into account the fact that for any block code we have:

$$\sum_{j=d_{\min}}^{n} A_j = 2^k - 1$$

(the -1 in the above expression corresponds to the null codeword), probability P_{nd} is equal to:

$$P_{nd} = \frac{2^k - 1}{2^n} \cong 2^{-(n-k)}$$

The detection of errors therefore remains efficient whatever the error probability on the transmission channel if the number of redundancy symbols $(n-k)$ is large enough. The detection of errors is therefore not very sensitive to error statistics.

When erroneous symbols are detected, the receiver generally asks the source to send them again. To transmit this re-transmission request, it is then necessary to have a receiver source link, called a return channel. The data rate on the return channel being low (a priori, requests for retransmission are short and few in number), we can always arrange it so that the error probability on this channel is much lower than the error probability on the transmission channel. Thus, the performance of a transmission system using error detection and repetition does not greatly depend on the return channel.

In case of error detection, the emission of the source can be interrupted to enable the retransmission of the corrupted information. The data rate is therefore not constant, which can present problems in some cases.

4.3.2 Error correction

Error correction involves looking for the transmitted codeword $\mathbf{c}$ given the received word $\mathbf{r}$. Two strategies are possible. The first one corresponds to a received word $\mathbf{r}$ at the input of the decoder made up of binary symbols (the case of a binary symmetric channel) and the second, to a received word $\mathbf{r}$ made up of analogue symbols (the case of a Gaussian channel). In the first case, we speak of hard input decoding whereas in the second case we speak of soft input decoding. We will now examine these two types of decoding, already mentioned in Chapter 1.

Hard decoding

- *Maximum a posteriori likelihood decoding*

For hard decoding the received word $\mathbf{r}$ is of the form:

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

where $\mathbf{c}$ and $\mathbf{e}$ are words with binary symbols.

Maximum *a posteriori* likelihood decoding involves looking for the codeword $\hat{\mathbf{c}}$ such that:

$$\Pr\{\hat{\mathbf{c}} \,|\mathbf{r}\} > \Pr\{\mathbf{c}_i \,|\mathbf{r}\} \quad \forall\, \mathbf{c}_i \neq \hat{\mathbf{c}} \in C(n,k)$$

Using Bayes' rule and assuming that all the codewords are equiprobable, the above decision rule can also be written:

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow \Pr(\mathbf{r} \,|\mathbf{c} = \mathbf{c}_i) > \Pr(\mathbf{r} \,|\mathbf{c} = \mathbf{c}_j), \forall \mathbf{c}_j \neq \mathbf{c}_i \in C(n,k)$$

Again taking the example of a binary symmetric channel with error probability p and denoting $d_H(\mathbf{r}, \hat{\mathbf{c}})$ the Hamming distance between $\mathbf{r}$ and $\hat{\mathbf{c}}$, the decision

rule is:

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow p^{d_H(\mathbf{c_i},\mathbf{r})}\,(1-p)^{n-d_H(\mathbf{c}_i,\mathbf{r})} > p^{d_H(\mathbf{c}_j,\mathbf{r})}\,(1-p)^{n-d_H(\mathbf{c}_j,\mathbf{r})}\ ,\ \forall \mathbf{c}_j \neq \mathbf{c}_i$$

Taking the logarithm of the two parts of the above inequality and considering $p < 0.5$, the decision rule of the maximum *a posteriori* likelihood can finally be written:

$$\hat{\mathbf{c}} = \mathbf{c}_i \Leftrightarrow d_H(\mathbf{r}, \mathbf{c}_i) \leq d_H(\mathbf{r}, \mathbf{c}_j), \quad \forall\, \mathbf{c}_j \neq \mathbf{c}_i \in C(n,k)$$

If two or several codewords are the same distance from $\mathbf{r}$, the codeword $\hat{\mathbf{c}}$ is chosen arbitrarily among the codewords equidistant from $\mathbf{r}$.

This decoding procedure which is optimal, that is, which minimizes the probability of erroneous decoding, becomes difficult to implement when the number of codewords becomes large, which is often the case for the widely-used block codes.

• *Decoding from the syndrome*

To get around this difficulty, it is possible to perform the decoding using syndrome $\mathbf{s}$. We recall that the syndrome is a vector of dimension $(n-k)$ that depends solely on the error configuration $\mathbf{e}$. For a binary symbol block code, the syndrome has 2^{n-k} configurations, which is generally much lower than the 2^k codewords.

To decode from the syndrome, we use a table with n rows and two columns. We write respectively in each row of the first column the null syndrome $\mathbf{s}$ (all the symbols are at zero, no errors) then the syndromes $\mathbf{s}$ corresponding to the configuration of an error then two errors etc. until the n rows are filled. All the configurations of the syndromes of the first column must be different. In the second column, we write the error configuration associated with each syndrome of the first column.

For a received word $\mathbf{r}$ we calculate the syndrome $\mathbf{s}$ then, using the table, we deduce the error word $\mathbf{e}$. Finally, we add word $\mathbf{e}$ to $\mathbf{r}$ and we obtain the most likely codeword.

Example 4.12

Let us consider a code $C(7,4)$ with a parity check matrix $\mathbf{H}$ with:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

This code has 16 codewords but only 8 configurations for the syndrome as indicated in Table 4.4.

Syndrome **s**	Error word **e**
000	0000000
001	0000001
010	0000100
011	0001000
100	0000100
101	0010000
110	0100000
111	1000000

Table 4.4: Syndromes and corresponding error words for a $C(7,4)$ code.

Let us assume that the codeword transmitted is $\mathbf{c} = [0101101]$ and that the received word $\mathbf{r} = [0111101]$ has an error in position 3. The syndrome is then equal to $\mathbf{s} = [101]$ and, according to the table, $\mathbf{e} = [0010000]$. The decoded codeword is $\hat{\mathbf{c}} = \mathbf{r} + \mathbf{e} = [0101101]$ and the error is corrected.

If the number of configurations of the syndrome is still too high to apply this decoding procedure, we use decoding algorithms specific to certain classes of codes but that, unfortunately, do not always exploit the whole correction capability of the code. These algorithms will be presented below.

• *Correction power*

Let $\mathbf{c}_j$ be the codeword transmitted and $\mathbf{c}_l$ its nearest neighbour. We have the following inequality:

$$d_H(\mathbf{c}_j, \mathbf{c}_l) \geqslant d_{\min}$$

Introducing the received word $\mathbf{r}$ and assuming that the minimum distance $d_{\min}$ is equal to $2t+1$ (integer t), we can write:

$$2t + 1 \leq d_H(\mathbf{c}_j, \mathbf{c}_l) \leq d_H(\mathbf{c}_j, \mathbf{r}) + d_H(\mathbf{c}_l, \mathbf{r})$$

We see that if the number of errors is lower than or equal to t, $\mathbf{c}_j$ is the most likely codeword since it is nearer to $\mathbf{r}$ than to $\mathbf{c}_l$ and thus the t errors can be corrected. If the minimum distance is now $(2t+2)$, using the same reasoning, we arrive at the same error correction capability. In conclusion, the correction capability of a linear block code with minimum distance $d_{\min}$ with hard decoding is equal to:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor \tag{4.23}$$

where $\lfloor x \rfloor$ is the whole part of x rounded down (for example $\lfloor 2.5 \rfloor = 2$).

• *Probability of erroneous decoding of a codeword*

For a linear block code $C(n,k)$ of error correction capability t, the codeword transmitted will be wrongly decoded if there are $t+j$ errors, $j=1,2,\cdots,n-t$, in the received word $\mathbf{r}$. For a binary symmetric channel of probability p, the probability $P_{e,\text{word}}$ of performing an erroneous decoding of the transmitted codeword is upper bounded by:

$$P_{e,\text{word}} < \sum_{j=t+1}^{n} \left(\begin{array}{c} n \\ j \end{array} \right) p^j (1-p)^{n-j} \tag{4.24}$$

We can also determine the binary error probability $P_{e,\text{bit}}$ on the information data after decoding. In presence of erroneous decoding, the maximum *a posteriori* likelihood decoder adds at most t errors by choosing the codeword with the minimum distance from the received word. The error probability is therefore bounded by:

$$P_{e,\text{bit}} < \frac{1}{n} \sum_{j=t+1}^{n} (j+t) \left(\begin{array}{c} n \\ j \end{array} \right) p^j (1-p)^{n-j} \tag{4.25}$$

If the transmission is performed with binary phase modulation (2-PSK, 4-PSK), probability p is equal to:

$$p = \frac{1}{2} \text{erfc} \sqrt{\frac{RE_b}{N_0}}$$

where R is the coding rate, E_b the energy received per transmitted information bit and N_0 the unilateral power spectral density of the noise. Figure 4.5 shows the binary error probability and word error probability after algebraic decoding for the (15,7) BCH code. The modulation is 4-PSK and the channel is Gaussian. The higher bounds expressed by (4.24) and (4.25) respectively are also plotted.

Soft decoding

Considering a channel with additive white Gaussian noise and binary phase modulation transmission (2-PSK or 4-PSK), the components $r_j, \quad j=0,1,\cdots,n-1$ of the received word $\mathbf{r}$ have the form:

$$r_j = \sqrt{E_s}\tilde{c}_j + b_j, \quad \tilde{c}_j = 2c_j - 1$$

where $c_j = 0,1$ is the symbol in position j of codeword $\mathbf{c}$, $\tilde{c}_j$ is the binary symbol associated with c_j, E_s is the energy received per transmitted symbol and b_j is white Gaussian noise, with zero mean and variance equal to σ_b^2.

• *Maximum a posteriori likelihood decoding*

Decoding using the maximum *a posteriori* likelihood criterion means searching for codeword $\hat{\mathbf{c}}$ such that:

$$\hat{\mathbf{c}} = \mathbf{c} \Leftrightarrow \Pr\{\mathbf{c}\,|\mathbf{r}\} > \Pr\{\mathbf{c}'\,|\mathbf{r}\}, \quad \forall \mathbf{c} \neq \mathbf{c}' \in C(n,k)$$

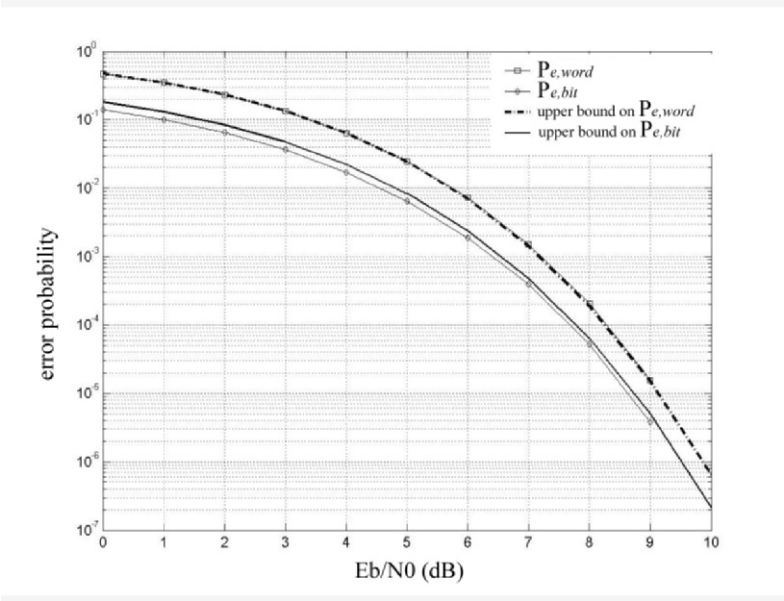

Figure 4.5 – Performance of the algebraic decoding of the (15,7) BCH code. 4-PSK transmission on a Gaussian channel.

Using the Bayes' rule and assuming all the codewords equiprobable, the above inequality can also be written:

$$\hat{\mathbf{c}} = \mathbf{c} \text{ if } p(\mathbf{r}\,|\mathbf{c}\,) > p(\mathbf{r}\,|\mathbf{c}'\,), \qquad \forall\, \mathbf{c} \neq \mathbf{c}' \in C(n,k) \tag{4.26}$$

where $p(\mathbf{r}\,|\mathbf{c}\,)$ is the probability density function of observation $\mathbf{r}$ conditionally to codeword $\mathbf{c}$.

For a Gaussian channel, probability density function $p(\mathbf{r}\,|\mathbf{c}\,)$ is equal to:

$$p(\mathbf{r}\,|\mathbf{c}\,) = \left(\frac{1}{\sqrt{2\pi}\sigma_b}\right)^n \exp\left(-\frac{1}{2\sigma_b^2}\sum_{j=0}^{n-1}(r_j - \sqrt{E_s}\tilde{c}_j)^2\right)$$

where σ_b^2 is the variance of the noise.

Replacing the two probability density functions by their respective expressions in inequality (4.26) and after some basic computation, we obtain:

$$\hat{\mathbf{c}} = \mathbf{c} \Leftrightarrow \sum_{j=0}^{n-1} r_j c_j > \sum_{j=0}^{n-1} r_j c'_j, \quad \forall \mathbf{c} \neq \mathbf{c}' \in C(n,k)$$

The decoded codeword is the one that maximizes the scalar product $\langle \mathbf{r}, \mathbf{c} \rangle$. We could also show that the decoded codeword is the one that minimizes the square of the Euclidean distance $\left\|\mathbf{r} - \sqrt{E_s}\tilde{\mathbf{c}}\right\|^2$.

This decoding procedure is applicable when the number of codewords is not too high. In the presence of a large number of codewords we can use a Chase algorithm whose principle is to apply the above decoding procedure and restrict the search space to a subset of codewords.

- *Chase algorithm*

The Chase algorithm is a sub-optimal decoding procedure that uses the maximum *a posteriori* likelihood criterion but considers a very reduced subset of codewords. To determine this subset of codewords, the Chase algorithm works in the following way.

- Step 1: The received word $\mathbf{r}$, made up of analogue symbols, is transformed into a word with binary symbols $\mathbf{z}_0 = (z_{00} \cdots z_{0j} \cdots z_{0n-1})$ by thresholding,

$$z_{0j} = \text{sgn}(r_j)$$

with the following convention:

$$\begin{aligned} \text{sgn}(x) &= 1 \text{ if } x \geqslant 0 \\ &= 0 \text{ if } x < 0 \end{aligned}$$

The binary word $\mathbf{z}_0$ is then decoded by a hard decision algorithm other than the maximum *a posteriori* likelihood algorithm (we will present algorithms for decoding block codes later). Let $\mathbf{c}_0$ be the codeword obtained.

- Step 2: Let $j_1, j_2, \cdots, j_t$ be the positions of the least reliable symbols, that is, such that the $|r_j|$ amplitudes are the smallest.

$2^t - 1$ words $\mathbf{e}_i$ are built by forming all the non-null binary combinations possible on positions $j_1, j_2, \cdots, j_t$. On the positions other than $j_1, j_2, \cdots, j_t$, the symbols of $\mathbf{e}_i$ are set to zero. Recall that t is the correction capability of the code.

- Step 3: Each of the $2^t - 1$ words $\mathbf{e}_i$ is used to define the words $\mathbf{z}_i$ with:

$$\mathbf{z}_i = \mathbf{z}_0 + \mathbf{e}_i$$

A hard decoder processes the words $\mathbf{z}_i$ to obtain at most $2^t - 1$ codewords $\mathbf{c}_i$. Note that the word at the output of the algebraic decoder is not always a codeword and only codewords will be considered when applying the decision criterion.

- Step 4: The maximum *a posteriori* likelihood rule is applied to the subset of the codewords $\mathbf{c}_i$ created in the previous step.

Example 4.13

Let there be a code $C(n,k)$ with correction capability $t=3$. The subset of the codewords is made up of 8 codewords, 7 of which are elaborated from the words $\mathbf{e}_i$. Words $\mathbf{e}_i$ are words of length n whose components are null except possibly those with indices j_1, j_2 and j_3 (see the table below).

i	$e_{i,j_1}\ e_{i,j_2}\ e_{i,j_3}$
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

• *Probability of erroneous decoding of a codeword*

Let us assume that the transmitted codeword is $\mathbf{c}_0 = (c_{01} \cdots c_{0j} \cdots c_{0n-1})$ and let $\mathbf{r}_0 = (r_0 \cdots r_j \cdots r_{n-1})$ be the received word with:

$$r_j = \sqrt{E_s}\tilde{c}_{0j} + b_j$$

Codeword $\mathbf{c}_0$ will be wrongly decoded if:

$$\sum_{j=0}^{n-1} r_j c_{0,j} < \sum_{j=0}^{n-1} r_j c_{l,j} \qquad \forall\, \mathbf{c}_l \neq \mathbf{c}_0 \in C(n,k)$$

The code being linear, we can, without loss of generality, assume that the codeword transmitted is the null word, that is, $c_{0,j}=0$ for $j=0,1,\cdots,n-1$.

The probability of erroneous decoding $P_{e,\text{word}}$ of a codeword is then equal to:

$$P_{e,\text{word}} = \Pr\left(\sum_{j=0}^{n-1} r_j c_{1,j} > 0 \text{ or } \ldots \sum_{j=0}^{n-1} r_j c_{l,j} > 0 \text{ or } \ldots\right)$$

Probability $P_{e,\text{word}}$ can be upper bounded by a sum of probabilities and, after some standard computation, it can be written in the form:

$$P_{e,\text{word}} \leq \frac{1}{2}\sum_{j=2}^{2^k} \text{erfc}\sqrt{w_j \frac{E_s}{N_0}}$$

where w_j is the Hamming weight of the j-th codeword.

Assuming that code $C(n,k)$ has A_w codewords of weight w, probability $P_{e,\text{word}}$ can again be written in the form:

$$P_{e,\text{word}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} A_w \text{erfc}\sqrt{w\frac{E_s}{N_0}} \tag{4.27}$$

Introducing the energy E_b received per bit of information transmitted, probability $P_{e,\text{word}}$ can finally be upper bounded by:

$$P_{e,\text{word}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} A_w \text{erfc}\sqrt{w\frac{RE_b}{N_0}} \tag{4.28}$$

where R is the coding rate.
We can also establish an upper bound of the binary error probability on the information symbols after decoding.

$$P_{e,\text{bit}} < \frac{1}{2} \sum_{w=d_{\min}}^{n} \frac{w}{n} A_w \text{erfc}\sqrt{w\frac{RE_b}{N_0}} \tag{4.29}$$

To calculate probabilities $P_{e,\text{word}}$ and $P_{e,\text{bit}}$ we must know the number A_w of codewords of weight w. For extended BCH codes the quantities A_w are given in [4.1].

As an example, Table 4.5 gives the A_w quantities for three extended Hamming codes.

n	k	$d_{\min}$	A_4	A_6	A_8	A_{10}	A_{12}	A_{14}	A_{16}
8	4	4	14	-	1	-	-	-	-
16	11	4	140	448	870	448	140	-	1
32	26	4	1240	27776	330460	2011776	7063784	14721280	18796230

Table 4.5 – A_w for three extended Hamming codes.

For the code (32,26) the missing A_w quantities are obtained from the relation $A_w = A_{n-w}$ for $0 \leq w \leq n/2$, $n/2$ even.

The A_w quantities for non-extended Hamming codes can be deduced from those of extended codes by resolving the following system of equations:

$$\begin{aligned} (n+1)A_{w-1} &= wA_w^{\text{extended}} \\ wA_w &= (n+1-w)A_{w-1} \end{aligned}$$

where n is the length of the words of the non-extended code.

For the Hamming code (7,4), for example, the A_w quantities are:

$$\begin{aligned} 8A_3 &= 4A_4^{\text{extended}} \quad A_3 = 7 \\ 4A_4 &= 4A_3 \quad A_4 = 7 \\ 8A_7 &= 8A_8^{\text{extended}} \quad A_7 = 1 \end{aligned}$$

Weight	A_w (23,12)	A_w (24,12)
0	1	1
7	253	0
8	506	759
11	1288	0
12	1288	2576
15	506	0
16	253	759
23	1	0
24	0	1

Table 4.6 – A_w for extended Golay and Golay codes.

For Golay and extended Golay codes, the A_w quantities are given in Table 4.6.

With a high signal to noise ratio, error probability $P_{e,\text{word}}$ is well approximated by the first term of the series:

$$P_{e,\text{word}} \cong \frac{1}{2} A_{d_{\min}} \text{erfc} \sqrt{\frac{R d_{\min} E_b}{N_0}} \quad \text{if} \quad \frac{E_b}{N_0} >> 1 \tag{4.30}$$

The same goes for error probability $P_{e,\text{bit}}$ on the information symbols.

$$P_{e,\text{bit}} \cong \frac{d_{\min}}{n} P_{e,\text{word}} \quad \text{if} \quad \frac{E_b}{N_0} >> 1 \tag{4.31}$$

In the absence of coding, the error probability on the binary symbols is equal to:

$$p = \frac{1}{2} \text{erfc} \sqrt{\frac{E_b}{N_0}}$$

As seen in Section 1.5, comparing the two expressions of the binary error probability with and without coding, we observe that the signal to noise ratio E_b/N_0 is multiplied by $Rd_{\min}$ in the presence of coding. If this multiplying coefficient is higher than 1, the coding acts as an amplifier of the signal to noise ratio whose asymptotic gain is approximated by

$$G_a = 10 \log(R d_{\min})(\text{dB})$$

To illustrate these bounds, let us again take the example of the (15,7) BCH code transmitted on a Gaussian channel with 4-PSK modulation. In Figure 4.6, we show the evolution of the binary error probability and word error probability obtained by simulation from the sub-optimal Chase algorithm (4 non-reliable positions). We also show the first two terms of the sums appearing in the bounds given by (4.28) and (4.29). As a reference, we have also plotted the binary error probability curve of a 4-PSK modulation without coding.

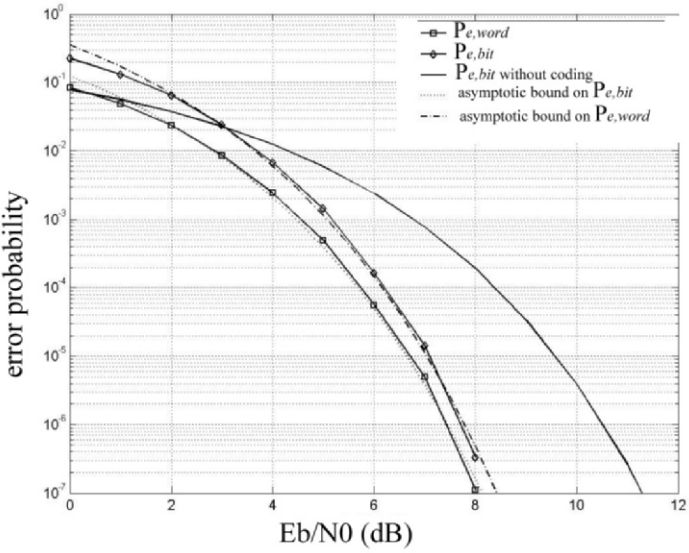

Figure 4.6 – Performance of the soft input decoding of the (15,7) BCH code. 4-PSK transmission on a Gaussian channel.

4.4 Decoding and performance of codes with non-binary symbols

4.4.1 Hard input decoding of Reed-Solomon codes

Hard input decoding algorithms make it possible to decode Reed-Solomon (RS) codes and BCH codes with binary symbols. We begin by presenting the principle of decoding RS codes then we treat the case of BCH codes using binary symbols as a particular case of decoding RS codes.

Assuming that $c(x)$ is the transmitted codeword, then for a channel with discrete input and output, the received word can always be written in the form:

$$r(x) = c(x) + e(x)$$

with:

$$e(x) = e_0 + e_1 x + \cdots + e_j x^j + \cdots + e_{n-1} x^{n-1}, \quad e_j \in \mathbf{F}_q \quad \forall\, j$$

When $e_j \neq 0$ there is an error in position j.

It was seen above that the generator polynomial of an RS code or of a BCH code (with $l = 1$) correcting t errors had the roots $\alpha, \cdots, \alpha^j, \cdots, \alpha^{2t}$ and that the

codewords were multiples of the generator polynomial. Thus, for any codeword, we can write:

$$c(\alpha^i) = 0; \forall\, i = 1, 2, \cdots, 2t$$

Decoding RS codes and binary BCH codes can be performed from a vector with $2t$ components $\mathbf{S} = [S_1 \cdots S_j \cdots S_{2t}]$, called a *syndrome*.

$$S_j = r(\alpha^j) = e(\alpha^j), \quad j = 1, 2, \cdots, 2t \tag{4.32}$$

When the components of vector $\mathbf{S}$ are all null, there are no errors or, at least, no detectable errors. When some components of the vector $\mathbf{S}$ are non-null, errors are present that, in certain conditions, can be corrected.

In the presence of t transmission errors, the error polynomial $e(x)$ is of the form:

$$e(x) = e_{n_1} x^{n_1} + e_{n_2} x^{n_2} + \cdots + e_{n_t} x^{n_t}$$

where the e_{n_l} are non-null coefficients taking their value in the field $\mathbf{F}_q$.
The components S_j of syndrome $\mathbf{S}$ are equal to:

$$S_j = e_{n_1}(\alpha^j)^{n_1} + \cdots + e_{n_l}(\alpha^j)^{n_l} + \cdots + e_{n_t}(\alpha^j)^{n_t}$$

Putting $Z_l = \alpha^{n_l}$ and, to simplify the notations $e_{n_l} = e_l$, the component S_j of the syndrome is again equal to:

$$S_j = e_1 Z_1^j + \cdots + e_l Z_l^j + \cdots + e_t Z_t^j \tag{4.33}$$

To determine the position of the transmission errors it is therefore sufficient to know the value of quantities $Z_l; j = 1, 2, \cdots, t$ then, in order to correct the errors, to evaluate coefficients $e_l; l = 1, 2, \cdots, t$.

The main difficulty in decoding RS codes or binary BCH codes is determining the position of the errors. Two methods are mainly used to decode RS codes or binary BCH codes: Peterson's direct method and the iterative method using the Berlekamp-Massey algorithm or Euclid algorithm .

4.4.2 Peterson's direct method

Description of the algorithm for codes with non-binary symbols

This method is well adapted for decoding RS codes or binary BCH codes correcting a low number of errors, typically 1 to 3. Indeed, the complexity of this method increases as the square of the correction capability of the code, whereas for the iterative method, the complexity increases only linearly with the correction capability of the code.

To determine the position of the errors let us introduce a polynomial $\sigma_d(x)$ called the *error locator* polynomial whose roots are exactly the quantities Z_l.

$$\sigma_d(x) = \prod_{l=1}^{t} (x + Z_l)$$

Developing this expression, the polynomial $\sigma_d(x)$ is again equal to:

$$\sigma_d(x) = x^t + \sigma_1 x^{t-1} + \cdots + \sigma_j x^{t-j} + \cdots + \sigma_t$$

where the coefficients σ_j are functions of the quantities Z_l.

From the expression of S_j we can build a non-linear system of $2t$ equations.

$$S_j = \sum_{i=1}^{t} e_i Z_i^j, \quad j = 1, 2, \cdots, 2t$$

The quantities $Z_l, l = 1, \cdots, t$ being the roots of the error locator polynomial $\sigma_d(x)$, we can write:

$$\sigma_d(Z_l) = Z_l^t + \sum_{j=1}^{t} \sigma_j Z_l^{t-j} = 0, \quad l = 1, 2, \cdots, t \tag{4.34}$$

Multiplying the two parts of this expression by the same term $e_l Z_l^q$, we obtain:

$$e_l Z_l^{t+q} + \sum_{j=1}^{t} \sigma_j e_l Z_l^{t+q-j} = 0, \quad l = 1, 2, \cdots, t \tag{4.35}$$

Summing relations (4.35) for l from 1 to t and taking into account the definition of component S_j of syndrome $\mathbf{S}$, we can write:

$$S_{t+q} + \sigma_1 S_{t+q-1} + \cdots + \sigma_j S_{t+q-j} + \cdots + \sigma_t S_q = 0, \quad \forall\, q \tag{4.36}$$

For an RS code correcting one error ($t = 1$) in a block of n symbols, syndrome $\mathbf{S}$ has two components S_1 and S_2. Coefficient σ_1 of the error locator polynomial is determined from relation (4.36) by making $t = 1$ and $q = 1$.

$$S_2 + \sigma_1 S_1 = 0 \rightarrow \sigma_1 = \frac{S_2}{S_1} \tag{4.37}$$

In the same way, for an RS code correcting two errors ($t = 2$) in a block of n symbols, the syndrome has four components S_1, S_2, S_3, S_4. Using relation (4.36) with $t = 2$ and $q = 1, 2$ we obtain the following system with two equations:

$$\begin{aligned} \sigma_1 S_2 + \sigma_2 S_1 &= S_3 \\ \sigma_1 S_3 + \sigma_2 S_2 &= S_4 \end{aligned}$$

Resolving this system of two equations enables us to determine coefficients σ_1 and σ_2 of the error locator polynomial.

$$\begin{aligned} \sigma_1 &= \tfrac{1}{\Delta_2} \left[S_1 S_4 + S_2 S_3 \right] \\ \sigma_2 &= \tfrac{1}{\Delta_2} \left[S_2 S_4 + S_3^2 \right] \end{aligned} \tag{4.38}$$

where Δ_2 is the determinant of the system with two equations.

$$\Delta_2 = S_2^2 + S_1 S_3$$

Finally, for an RS code correcting three errors ($t = 3$), the relation (4.36) with $t = 3$ and $q = 1, 2, 3$ leads to the following system of three equations:

$$\begin{aligned} \sigma_1 S_3 + \sigma_2 S_2 + \sigma_3 S_1 &= S_4 \\ \sigma_1 S_4 + \sigma_2 S_3 + \sigma_3 S_2 &= S_5 \\ \sigma_1 S_5 + \sigma_2 S_4 + \sigma_3 S_3 &= S_6 \end{aligned}$$

The resolution of this system enables us to determine coefficients σ_1, σ_2 and σ_3 of the error locator polynomial.

$$\begin{aligned} \sigma_1 &= \tfrac{1}{\Delta_3}\left[S_1 S_3 S_6 + S_1 S_4 S_5 + S_2^2 S_6 + S_2 S_3 S_5 + S_2 S_4^2 + S_3^2 S_4\right] \\ \sigma_2 &= \tfrac{1}{\Delta_3}\left[S_1 S_4 S_6 + S_1 S_5^2 + S_2 S_3 S_6 + S_2 S_4 S_5 + S_3^2 S_5 + S_3 S_4^2\right] \\ \sigma_3 &= \tfrac{1}{\Delta_3}\left[S_2 S_4 S_6 + S_2 S_5^2 + S_3^2 S_6 + S_4^3\right] \end{aligned} \qquad (4.39)$$

where Δ_3 is the determinant of the system with three equations.

$$\Delta_3 = S_1 S_3 S_5 + S_1 S_4^2 + S_2^2 S_5 + S_3^3$$

Implementation of Peterson's decoder for an RS code with parameter $t = 3$

1. Calculate the $2t$ syndromes S_j: $S_j = r(\alpha^j)$
2. Determine the number of errors:
   - Case (a) $S_j = 0$, $\forall\, j$: no detectable error.
   - Case (b) $\Delta_3 \neq 0$: presence of three errors.
   - Case (c) $\Delta_3 = 0$ and $\Delta_2 \neq 0$: presence of two errors.
   - Case (d) $\Delta_3 = \Delta_2 = 0$ and $S_1 \neq 0$: presence of one error.
3. Calculate the error locator polynomial $\sigma_d(x)$
   - Case (b) Use (4.39)
   - Case (c) Use (4.38)
   - Case (d) Use (4.37)
4. Look for the roots of $\sigma_d(x)$ in field $\mathbf{F}_q$
5. Calculate the error coefficients e_i

- Case (b)

$$e_i = \frac{1}{\Delta}\left[S_1(Z_k^2 Z_p^3 + Z_k^3 Z_p^2) + S_2(Z_k^3 Z_p + Z_k Z_p^3) + S_3(Z_k^2 Z_p + Z_k Z_p^2)\right], \quad k \neq p \neq i, (i,k,p) \in \{1,2,3\}^3$$

$$\Delta = \sum_{\substack{1 \leq i_1, i_2, i_3 \leq 3 \\ i_1 + i_2 + i_3 = 6 \\ i_1 \neq i_2 \neq i_3}} Z_1^{i_1} Z_2^{i_2} Z_3^{i_3}$$

- Case (c)

$$e_i = \frac{S_1 Z_p + S_2}{Z_i(Z_1 + Z_2)}, \; p \neq i, \; (i,p) \in \{1,2\}^2$$

- Case (d)

$$e_1 = \frac{S_1^2}{S_2}$$

6. Correct the errors: $\hat{c}(x) = r(x) + e(x)$

Example 4.14

To illustrate the decoding of an RS code using the direct method, we now present an example considering an RS code correcting up to three errors ($t = 3$) and having the following parameters:

$$m = 4 \quad q = 16 \quad n = 15 \quad n - k = 6$$

Let us assume, for example, that the transmitted codeword is $c(x) = 0$ and that the received word has two errors.

$$r(x) = \alpha^7 x^3 + \alpha^3 x^6$$

1. Calculate the components of the syndrome

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_5 = \alpha^{22} + \alpha^{33} = \alpha^4 \\ S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} & S_6 = \alpha^{25} + \alpha^{39} = \alpha^{13} \end{array}$$

2. Determine the number of errors

$$\Delta_3 = 0 \qquad \Delta_2 = \alpha^8$$

Δ_3 being null and $\Delta_2 \neq 0$, we have two errors.

3. Calculate coefficients σ_1 and σ_2 of the error locator polynomial.

$$\begin{aligned}\sigma_1 &= \frac{1}{\Delta_2}\left[S_1S_4 + S_2S_3\right] = \frac{\alpha^{19}+\alpha^{17}}{\alpha^8} = \alpha^{11} + \alpha^9 = \alpha^2\\ \sigma_2 &= \frac{1}{\Delta_2}\left[S_2S_4 + S_3^2\right] = \frac{\alpha^{12}+\alpha^{22}}{\alpha^8} = \alpha^4 + \alpha^{14} = \alpha^9\end{aligned}$$

The error locator polynomial is therefore equal to:

$$\sigma_d(x) = x^2 + \alpha^2 x + \alpha^9$$

4. Look for the two roots of the error locator polynomial.

Looking through the elements of field $\mathbf{F}_{16}$ we find that α^3 and α^6 cancel the polynomial $\Lambda(x)$. The errors therefore concern the terms in x^3 and in x^6 of word $r(x)$.

5. Calculate the error coefficients e_1 and e_2.

$$\begin{aligned}e_1 &= \frac{S_1Z_2+S_2}{Z_1Z_2+Z_1^2} = \frac{\alpha^{19}+\alpha^6}{\alpha^9+\alpha^6} = \frac{\alpha^{12}}{\alpha^5} = \alpha^7\\ e_2 &= \frac{S_1Z_1+S_2}{Z_1Z_2+Z_2^2} = \frac{\alpha^{16}+\alpha^6}{\alpha^9+\alpha^{12}} = \frac{\alpha^{11}}{\alpha^8} = \alpha^3\end{aligned}$$

6. Correct the errors

$$c(x) = (\alpha^7x^3 + \alpha^3x^6) + (\alpha^7x^3 + \alpha^3x^6) = 0$$

The transmitted codeword is the null word; the two errors have therefore been corrected.

Simplification of Peterson's algorithm for binary codes

For BCH codes with binary symbols it is not necessary to calculate the coefficients e_j. Indeed, as these coefficients are binary, they are necessarily equal to 1 in the presence of an error in position j. The computation of coefficients σ_j can also be simplified by taking into account the fact that for a code with binary symbols we have:

$$S_{2j} = e(\alpha^{2j}) = \left[e(\alpha^j)\right]^2 = S_j^2$$

For a BCH code with binary symbols correcting up to $t = 3$ errors, taking into account the previous remark and using the expressions of the three coefficients σ_j of the error locator polynomial, we obtain:

$$\begin{aligned}\sigma_1 &= S_1\\ \sigma_2 &= \frac{S_1^2S_3+S_5}{S_1^3+S_3}\\ \sigma_3 &= (S_1^3 + S_3) + S_1\frac{S_1^2S_3+S_5}{S_1^3+S_3}\end{aligned} \tag{4.40}$$

For a BCH code with binary symbols correcting up to $t = 2$ errors, also taking into account the previous remark and using the expressions of the two coefficients σ_j of the error locator polynomial, we obtain:

$$\begin{array}{rcl} \sigma_1 & = & S_1 \\ \sigma_2 & = & \frac{S_3+S_1^3}{S_1} \end{array} \tag{4.41}$$

Finally, in the presence of an error $\sigma_2 = \sigma_3 = 0$ and $\sigma_1 = S_1$.

Example 4.15

Let us consider a BCH code correcting two errors ($t = 2$) in a block of $n = 15$ symbols of a generator polynomial equal to:

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1$$

Let us assume that the transmitted codeword is $c(x) = 0$ and that the received word $r(x)$ has two errors.

$$r(x) = x^8 + x^3$$

There are three steps to the decoding: calculate syndrome $\mathbf{S}$, determine the coefficients σ_l of the error locator polynomial and search for its roots in field $\mathbf{F}_{16}$.

1. Calculate syndrome $\mathbf{S}$: we only need to calculate the odd index components S_1 and S_3 of syndrome $\mathbf{S}$. Using the binary representations of the elements of field $\mathbf{F}_{16}$ given in the appendix, and taking into account the fact that $\alpha^{15} = 1$, we have:

$$\begin{array}{l} S_1 = r(\alpha) = \alpha^8 + \alpha^3 = \alpha^{13} \\ S_3 = r(\alpha^3) = \alpha^{24} + \alpha^9 = \alpha^9 + \alpha^9 = 0 \end{array}$$

2. Determine the coefficients σ_1 and σ_2 of the error locator polynomial. Using the expressions of coefficients σ_1 and σ_2, we obtain:

$$\begin{array}{l} \sigma_1 = S_1 = \alpha^{13} \\ \sigma_2 = \frac{S_3+S_1^3}{S_1} = S_1^2 = \alpha^{26} = \alpha^{11} \quad (\alpha^{15} = 1) \end{array}$$

and the error locator polynomial is equal to:

$$\sigma_d(x) = x^2 + \alpha^{13}x + \alpha^{11}$$

3. Search for the roots of the error locator polynomial in field $\mathbf{F}_{16}$. By trying all the elements of field $\mathbf{F}_{16}$, we can verify that the roots of the error locator polynomial are α^3 and α^8. Indeed, we have

$$\sigma(\alpha^3) = \alpha^6 + \alpha^{16} + \alpha^{11} = \alpha^6 + \alpha + \alpha^{11} = 1100 + 0010 + 1110 = 0000$$

$$\sigma(\alpha^8) = \alpha^{16} + \alpha^{21} + \alpha^{11} = \alpha + \alpha^6 + \alpha^{11} = 0010 + 1100 + 1110 = 0000$$

The transmission errors concern the terms x^8 and x^3 of received word $r(x)$. The transmitted codeword is therefore $c(x) = 0$ and the two errors have been corrected.

The reader can verify that in the presence of a single error, $r(x) = x^j$; $0 \leq j \leq (n-1)$, the correction is still performed correctly since:

$$S_1 = \alpha^j;\ S_3 = \alpha^{3j};\ \sigma_1 = \alpha^j;\ \sigma_2 = 0;\ \sigma_d(x) = x(x + \sigma_1)$$

and the error locator polynomial has one sole root $\sigma_1 = \alpha^j$.

Chien algorithm

To search for the error locator polynomial roots in the case of codes with binary symbols, we can avoid going through all the elements of field $\mathbf{F}_q$ by using Chien's iterative algorithm.
Dividing polynomial $\sigma_d(x)$ by x^t, we obtain:

$$\tilde{\sigma}_d(x) = \frac{\sigma_d(x)}{x^t} = 1 + \sigma_1 x^{-1} + \cdots + \sigma_j x^{-j} + \cdots + \sigma_t x^{-t}$$

The roots of polynomial $\sigma_d(x)$ that are also the roots of $\tilde{\sigma}_d(x)$ have the form α^{n-j} where $j = 1, 2, \ldots, n-1$ and $n = q - 1$.
Thus α^{n-j} is a root of $\tilde{\sigma}_d(x)$ if:

$$\sigma_1 \alpha^{-n+j} + \cdots + \sigma_p x^{-np+jp} + \cdots + \sigma_t x^{-nt+jt} = 1$$

Taking into account the fact that $\alpha^n = 1$, the condition to satisfy in order for α^{n-j} to be a root of the error locator polynomial is:

$$\sum_{p=1}^{t} \sigma_p \alpha^{jp} = 1; \quad j = 1, 2, \cdots, (n-1) \tag{4.42}$$

Chien's algorithm has just tested whether condition (4.42) is satisfied using the circuit shown in Figure 4.7.
This circuit has a register with t memories initialized with the t coefficients σ_j of the error locator polynomial and a register with n memories that stocks symbols r_j; $j = 0, 1, \cdots, (n-1)$ of word $r(x)$. At the first clock pulse, the circuit performs the computation of the left-hand part of expression (4.42) for $j = 1$. If the result of this computation is equal to 1, α^{n-1} is a root of the error locator polynomial and the error that concerned symbol r_{n-1} is then corrected. If the result of this computation is equal to 0, no correction is performed. At the end of this first phase, the σ_j coefficients contained in the t memories of the register are replaced by $\sigma_j \alpha^j$. At the second clock pulse the circuit again

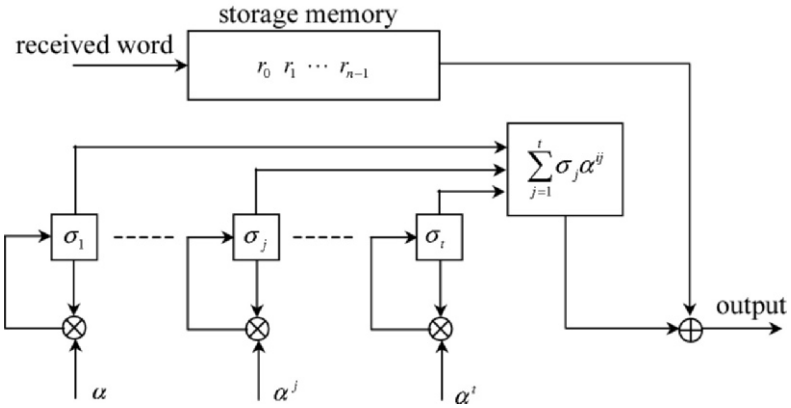

Figure 4.7 – Schematic diagram of the circuit implementing the Chien algorithm.

performs the computation of the left-hand part of expression (4.42) for $j = 2$. If the result of this computation is equal to 1, α^{n-2} is a root of the error locator polynomial and the error that concerned symbol r_{n-2} is then corrected. The algorithm continues in the same way for the following clock pulses.

4.4.3 Iterative method

Decoding RS codes or binary BCH codes with the iterative method uses two polynomials, error locator polynomial $\Lambda(x)$ and error evaluator polynomial $\Gamma(x)$. These two polynomials are defined respectively by:

$$\Lambda(x) = \prod_{j=1}^{t} (1 + Z_j x) \tag{4.43}$$

$$\Gamma(x) = \sum_{i=1}^{t} e_i Z_i x \frac{\Lambda(x)}{1 + Z_i x} \tag{4.44}$$

The error locator polynomial whose roots are Z_j^{-1} enables the position of the errors to be determined and the error evaluator polynomial enables the value of the error e_j to be determined. Indeed, taking into account the fact that $\Lambda(Z_j^{-1}) = 0$, the polynomial $\Gamma(x)$ taken in Z_j^{-1} is equal to:

$$\begin{aligned} \Gamma(Z_j^{-1}) &= e_j \prod_{p \neq j} (1 + Z_p Z_j^{-1}) \\ &= e_j Z_j^{-1} \Lambda'(Z_j^{-1}) \end{aligned}$$

where $\Lambda'(x) = \frac{d\Lambda}{dx}(x)$.

The value of error e_j is then given by the Forney algorithm:

$$e_j = Z_j \frac{\Gamma(Z_j^{-1})}{\Lambda'(Z_j^{-1})} \tag{4.45}$$

Introducing the polynomial $S(x)$ defined by:

$$S(x) = \sum_{j=1}^{2t} S_j x^j \tag{4.46}$$

we can show that:

$$\Lambda(x)S(x) \equiv \Gamma(x) \text{ modulo } x^{2t+1} \tag{4.47}$$

This relation is called *the key equation* for decoding a cyclic code.
To determine polynomials $\Lambda(x)$ and $\Gamma(x)$ two iterative algorithms are mainly used, the Berlekamp-Massey algorithm and Euclid's algorithm.

Berlekamp-Massey algorithm for codes with non-binary symbols

Computation of polynomials $\Lambda(x)$ and $\Gamma(x)$ using the Berlekamp-Massey algorithm is performed iteratively. It requires two intermediate polynomials denoted $\Theta(x)$ and $\Omega(x)$. The algorithm has $2t$ iterations. Once the algorithm has terminated, the Chien algorithm must be implemented to determine the roots Z_j^{-1} of $\Lambda(x)$ and consequently the position of the errors. Next, the Forney algorithm expressed by (4.45) enables the value of the errors e_j to be calculated.

Initial conditions:

$$\begin{array}{rclrcl} L_0 & = & 0 & & & \\ \Lambda^{(0)}(x) & = & 1 & \Theta^{(0)}(x) & = & 1 \\ \Gamma^{(0)}(x) & = & 0 & \Omega^{(0)}(x) & = & 1 \end{array}$$

Recursion: $1 \leq p \leq 2t$

$$\Delta_p = \sum_j \Lambda_j^{(p-1)} S_{p-j}$$

$$\begin{aligned} \delta_p &= 1 \text{ if } \Delta_p \neq 0 \text{ and } 2L_{p-1} \leq p-1 \\ &= 0 \text{ otherwise} \end{aligned}$$

$$L_p = \delta_p(p - L_{p-1}) + (1-\delta_p)L_{p-1}$$

$$\begin{bmatrix} \Lambda^{(p)} & \Gamma^{(p)} \\ \Theta^{(p)} & \Omega^{(p)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta_p x \\ \Delta_p^{-1}\delta_p & (1-\delta_p)x \end{bmatrix} \begin{bmatrix} \Lambda^{(p-1)} & \Gamma^{(p-1)} \\ \Theta^{(p-1)} & \Omega^{(p-1)} \end{bmatrix}$$

Termination :

$$\begin{aligned} \Lambda(x) &= \Lambda^{(2t)}(x) \\ \Gamma(x) &= \Gamma^{(2t)}(x) \end{aligned}$$

Example 4.16

To illustrate the decoding of an RS code using the Berlekamp-Massey algorithm, let us consider an RS code correcting up to two errors ($t = 2$) and having the following parameters:

$$m = 4; \quad q = 16; \quad n = 15; \quad n - k = 4$$

Let us assume, for example, that the transmitted codeword is $c(x) = 0$ and that the received word has two errors.

$$r(x) = \alpha^7 x^3 + \alpha^3 x^6$$

The set of calculations performed to decode this RS code will be done in field $\mathbf{F}_{16}$ whose elements are given in the appendix.

1. Calculate syndrome $\mathbf{S} = (S_1, S_2, S_3, S_4)$

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \end{array}$$

   The polynomial $S(x)$ is therefore equal to:

$$S(x) = \alpha^{13} x + \alpha^6 x^2 + \alpha^{11} x^3 + \alpha^6 x^4$$

2. Calculate polynomials $\Lambda(x)$ and $\Gamma(x)$ from the Berlekamp-Massey algorithm

p	Δ_p	δ_p	L_p	$\Lambda^p(x)$	$\Theta^p(x)$	$\Gamma^p(x)$	$\Omega^p(x)$
0			0	1	1	0	1
1	α^{13}	1	1	$1 + \alpha^{13} x$	α^2	$\alpha^{13} x$	0
2	α	0	1	$1 + \alpha^8 x$	$\alpha^2 x$	$\alpha^{13} x$	0
3	α^{10}	1	2	$1 + \alpha^8 x + \alpha^{12} x^2$	$\alpha^5 + \alpha^{13} x$	$\alpha^{13} x$	$\alpha^3 x$
4	α^{10}	0	2	$1 + \alpha^2 x + \alpha^9 x^2$	$\alpha^5 x + \alpha^{13} x^2$	$\alpha^{13} x + \alpha^{13} x^2$	$\alpha^3 x^2$

   In the table above, all the calculations are done in field $\mathbf{F}_{16}$ and take into account the fact that $\alpha^{15} = 1$.

   The error locator and error evaluator polynomials are:

$$\begin{array}{rcl} \Lambda(x) & = & 1 + \alpha^2 x + \alpha^9 x^2 \\ \Gamma(x) & = & \alpha^{13} x + \alpha^{13} x^2 \end{array}$$

   We can verify that the key equation for the decoding has been satisfied. Indeed, we do have:

$$\Lambda(x) S(x) = \alpha^{13} x + \alpha^{13} x^2 + \alpha^4 x^5 + x^6 \equiv \alpha^{13} x + \alpha^{13} x^2 = \Gamma(x) \text{ modulo } x^5$$

3. Search for the roots of the error locator polynomial

   By looking through all the elements of field $\mathbf{F_{16}}$ we find that α^{12} and α^9 are roots of polynomial $\Lambda(x)$. The errors are therefore in position $x^3(\alpha^{-12} = \alpha^3)$ and $x^6(\alpha^{-9} = \alpha^6)$ and error polynomial $e(x)$ is equal to:

$$e(x) = e_3 x^3 + e_6 x^6$$

4. Calculate error coefficients e_j (4.45).

$$\begin{aligned} e_3 &= \alpha^3 \frac{\alpha^6}{\alpha^2} = \alpha^7 \\ e_6 &= \alpha^6 \frac{\alpha^{14}}{\alpha^2} = \alpha^3 \end{aligned}$$

   Error polynomial $e(x)$ is therefore equal to:

$$e(x) = \alpha^7 x^3 + \alpha^3 x^6$$

   and the estimated codeword is $\hat{c}(x) = r(x) + e(x) = 0$. The two transmission errors are corrected.

Euclid's algorithm

Euclid's algorithm enables us to solve the key equation for decoding, that is, to determine polynomials $\Lambda(x)$ and $\Gamma(x)$.

Initial conditions:

$$R_{-1}(x) = x^{2t};\ R_0(x) = S(x);\ U_{-1}(x) = 0;\ U_0(x) = 1$$

Recursion:
calculate $Q_j(x)$, $R_{j+1}(x)$ and $U_{j+1}(x)$ from the two following expressions:

$$\begin{aligned} \frac{R_{j-1}(x)}{R_j(x)} &= Q_j(x) + \frac{R_{j+1}(x)}{R_j(x)} \\ U_{j+1}(x) &= Q_j(x)U_j(x) + U_{j-1}(x) \end{aligned}$$

When $\deg(U_j) \leq t$ and $\deg(R_j) \leq t$ then:

$$\begin{aligned} \Lambda(x) &= U_{j+1}(x) \\ \Gamma(x) &= R_{j+1}(x) \end{aligned}$$

Example 4.17

Let us again take the RS code used to illustrate the Berlekamp-Massey algorithm. Assuming that the received word is always $r(x) = \alpha^7 x^3 + \alpha^3 x^6$ when the transmitted codeword is $c(x) = 0$, the decoding algorithm is the following:

1. Calculate syndrome $\mathbf{S} = (S_1, S_2, S_3, S_4)$

$$\begin{array}{ll} S_1 = \alpha^{10} + \alpha^9 = \alpha^{13} & S_3 = \alpha^{16} + \alpha^{21} = \alpha^{11} \\ S_2 = \alpha^{13} + \alpha^{15} = \alpha^6 & S_4 = \alpha^{19} + \alpha^{27} = \alpha^6 \end{array}$$

Polynomial $S(x)$ is therefore equal to:

$$S(x) = \alpha^{13}x + \alpha^6 x^2 + \alpha^{11}x^3 + \alpha^6 x^4$$

2. Calculate polynomials $\Lambda(x)$ and $\Gamma(x)$ from Euclid's algorithm (the calculations are performed in field $\mathbf{F_{16}}$ whose elements are given in the appendix).

$$\begin{array}{rcl|rcl} j & = & 0 & j & = & 1 \\ R_{-1}(x) & = & x^5 & R_0(x) & = & S(x) \\ R_0(x) & = & S(x) & R_1(x) & = & \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^{12} x \\ Q_0(x) & = & \alpha^9 x + \alpha^{14} & Q_1(x) & = & \alpha x + \alpha^5 \\ R_1(x) & = & \alpha^5 x^3 + \alpha^{13} x^2 + \alpha^{12} x & R_2(x) & = & \alpha^{14} x^2 + \alpha^{14} x \\ U_1(x) & = & \alpha^9 x + \alpha^{14} & U_2(x) & = & \alpha^{10} x^2 + \alpha^3 x + \alpha \end{array}$$

We can verify that $\deg(U_2(x)) = 2$ is lower than or equal to t ($t = 2$) and that $\deg(R_2(x)) = 2$ is lower than or equal to t. The algorithm is therefore terminated and polynomials $\Lambda(x)$ and $\Gamma(x)$ respectively have the expression:

$$\begin{array}{l} \Lambda(x) = U_2(x) = \alpha + \alpha^3 x + \alpha^{10} x^2 = \alpha(1 + \alpha^2 x + \alpha^9 x^2) \\ \Gamma(x) = R_2(x) = \alpha^{14} x + \alpha^{14} x^2 = \alpha(\alpha^{13} x + \alpha^{13} x^2) \end{array}$$

We can verify that the key equation for the decoding is satisfied and that the two polynomials obtained are identical, to within one coefficient α, to those determined using the Berlekamp-Massey algorithm.

The roots of the polynomial $\Lambda(x)$ are therefore $1/\alpha^3$ and $1/\alpha^6$, and error polynomial $e(x)$ is equal to:

$$e(x) = \alpha^7 x^3 + \alpha^3 x^6$$

Calculating coefficients e_j by a transform

It is possible to calculate the coefficients e_j; $j = 0, 1, \cdots, (n-1)$ of error polynomial $e(x)$ without determining the roots of the error locator polynomial $\Lambda(x)$. To do this, we introduce the *extended syndrome* $S^*(x)$ defined by:

$$S^*(x) = \Gamma(x)\frac{1 + x^n}{\Lambda(x)} = \sum_{j=1}^{n} S_j x^j \tag{4.48}$$

Coefficient e_j is null (no errors) if α^{-j} is not a root of error locator polynomial $\Lambda(x)$. In this case, we have $S^*(\alpha^{-j}) = 0$ since $\alpha^{-jn} = 1$ (recall that $n = q - 1$ and $\alpha^{q-1} = 1$).
A contrario if α^{-j} is a root of the locator polynomial, coefficient e_j is non-null (presence of an error) and $S^*(\alpha^{-j})$ is of the form 0/0. This indetermination can be removed by calculating the derivation of the numerator and the denominator of expression (4.48).

$$S^*(\alpha^{-i}) = \Gamma(\alpha^{-i}) \frac{n\alpha^{-j(n-1)}}{\Lambda'(\alpha^{-j})}$$

Using Equation (4.45) and taking into account the fact that $\alpha^{-j(n-1)} = \alpha^j$ and that $na = a$ for n odd in a Galois field, coefficient e_j is equal to:

$$e_j = S^*(\alpha^{-j}) \tag{4.49}$$

The extended syndrome can be computed from polynomials $\Lambda(x)$ and $\Gamma(x)$ using the following relation deduced from expression (4.48).

$$\Lambda(x)S^*(x) = \Gamma(x)(1 + x^n) \tag{4.50}$$

Coefficients S_j of the extended syndrome are identical to those of syndrome $S(x)$ for j from 1 to $2t$ and are determined by cancelling the coefficients of the x^j terms in the product $\Lambda(x)S^*(x)$, for j from $2t + 1$ to n.

Example 4.18

Again taking the example of the RS code ($q = 16$; $n = 15$; $k = 11$; $t = 2$) used to illustrate the Berlekamp-Massey algorithm let us determine the extended syndrome.

$$S^*(x) = \sum_{j=1}^{15} S_j x^j$$

with:

$$\begin{array}{ll} S_1 = \alpha^{13} & S_3 = \alpha^{11} \\ S_2 = \alpha^6 & S_4 = \alpha^6 \end{array}$$

Equation (4.50) provides us with the following relation:

$$S(x) + \alpha^2 x S(x) + \alpha^9 x^2 S(x) = \alpha^{13}(x + x^2 + x^{16} + x^{17})$$

$$\begin{array}{l} S_1 x + (\alpha^2 S_1 + S_2)x^2 \\ + \sum_{k=3}^{15} (\alpha^9 S_{k-2} + \alpha^2 S_{k-1} + S_k)x^k \\ +(\alpha^2 S_{15} + \alpha^9 S_{14})x^{16} + \alpha^9 S_{15} x^{17} \quad = \quad \alpha^{13}(x + x^2 + x^{16} + x^{17}) \end{array}$$

From this there results the recurrence relation:

$$S_k = \alpha^2 S_{k-1} + \alpha^9 S_{k-2}, \quad k = 3, 4, \cdots, 15$$

We thus obtain the coefficients of the extended syndrome:

$$S_5 = \alpha^4, S_6 = \alpha^{13}, S_7 = \alpha^6, S_8 = \alpha^{11}, S_9 = \alpha^6, S_{10} = \alpha^4$$
$$S_{11} = \alpha^{13}, S_{12} = \alpha^6, S_{13} = \alpha^{11}, S_{14} = \alpha^6, S_{15} = \alpha^4$$

Another way to obtain the extended polynomial involves dividing $\Gamma(x)(1 + x^n)$ by $\Lambda(x)$ by increasing power orders.
The errors being with monomials x^3 and x^6, let us calculate coefficients e_3 and e_6.

$$\begin{aligned} e_3 &= S^*(\alpha^{12}) = \alpha^2 + \alpha^4 + \alpha^{10} + \alpha^7 = \alpha^7 \\ e_6 &= S^*(\alpha^9) = \alpha^4 + \alpha^7 = \alpha^3 \end{aligned}$$

The values found for coefficients e_3 and e_6 are obviously identical to those obtained in example 4.16. We can verify that the other e_j coefficients are all null.

Berlekamp-Massey algorithm for binary cyclic codes

For binary BCH codes the Berlekamp-Massey algorithm can be simplified since it is no longer necessary to determine the error evaluator polynomial, and since it is possible to show that the Δ_j terms are null for j even. This implies:

$$\begin{aligned} &\delta_{2p} = 0 \\ &L_{2p} = L_{2p-1} \\ &\Lambda^{(2p)}(x) = \Lambda^{(2p-1)}(x) \\ &\Theta^{(2p)}(x) = x\Theta^{(2p-1)}(x) \end{aligned}$$

Hence the algorithm in t iterations:
Initial conditions:

$$\begin{aligned} L_{-1} &= 0 \\ \Lambda^{(-1)}(x) &= 1 \quad \Theta^{(-1)}(x) = x^{-1} \end{aligned}$$

Recursion: $0 \leq p \leq t - 1$

$$\Delta_{2p+1} = \sum_j \Lambda_j^{(2p-1)} S_{2p+1-j}$$

$$\begin{aligned} \delta_{2p+1} &= 1 \quad \text{if } \Delta_{2p+1} \neq 0 \text{ and } L_{2p-1} \leq p \\ &= 0 \quad \text{if not} \end{aligned}$$

$$L_{2p+1} = \delta_{2p+1}(2p + 1 - L_{2p-1}) + (1 - \delta_{2p+1})L_{2p-1}$$

$$\begin{bmatrix} \Lambda^{(2p+1)} \\ \Theta^{(2p+1)} \end{bmatrix} = \begin{bmatrix} 1 & \Delta_{2p+1}x^2 \\ \Delta_{2p+1}^{-1}\delta_{2p+1} & (1 - \delta_{2p+1})x^2 \end{bmatrix} \begin{bmatrix} \Lambda^{(2p-1)} \\ \Theta^{(2p-1)} \end{bmatrix}$$

Termination:

$$\Lambda(x) = \Lambda^{(2t-1)}(x)$$

Example 4.19

Again taking the BCH code that was used to illustrate the computation of the error locator polynomial with the direct method, let us assume that the received word is $r(x) = x^8 + x^3$ when the transmitted codeword is $c(x) = 0$.

1. Syndrome **S** has four components.

$$\begin{aligned} S_1 &= r(\alpha) = \alpha^8 + \alpha^3 = \alpha^{13} \\ S_3 &= r(\alpha^3) = \alpha^{24} + \alpha^9 = 0 \\ S_2 &= S_1^2 = \alpha^{26} = \alpha^{11} \\ S_4 &= S_2^2 = \alpha^{22} = \alpha^7 \end{aligned}$$

   Polynomial $S(x)$ is equal to:

$$S(x) = \alpha^{13}x + \alpha^{11}x^2 + \alpha^7 x^4$$

2. Calculate polynomial $\Lambda(x)$ from the Berlekamp-Massey algorithm

p	Δ_{2p+1}	δ_{2p+1}	L_{2p+1}	$\Lambda^{2p+1}(x)$	$\Theta^{2p+1}(x)$
-1			0	1	x^{-1}
0	α^{13}	1	1	$1 + \alpha^{13}x$	α^2
1	α^9	1	2	$1 + \alpha^{13}x + \alpha^{11}x^2$	$\alpha^6 + \alpha^4 x$

   Note that polynomial $\Lambda(x)$ obtained is identical to that determined using the direct method. The roots of $\Lambda(x)$ are $1/\alpha^3$ and $1/\alpha^8$, and the errors therefore concern terms x^3 and x^8. The estimated codeword is $\hat{c}(x) = 0$.

Euclid's algorithm for binary codes

Example 4.20

Let us again take the decoding of the (15,7) BCH code. The received word is $r(x) = x^8 + x^3$.

$$\begin{array}{ccccccc} j & = & 0 & & j & = & 1 \\ R_{-1}(x) & = & x^5 & & R_0(x) & = & S(x) \\ R_0(x) & = & S(x) & & R_1(x) & = & \alpha^4 x^3 + \alpha^6 x^2 \\ Q_0(x) & = & \alpha^8 x & & Q_1(x) & = & \alpha^3 x + \alpha^5 \\ R_1(x) & = & \alpha^4 x^3 + \alpha^6 x^2 & & R_2(x) & = & \alpha^{13} x \\ U_1(x) & = & \alpha^8 x & & U_2(x) & = & \alpha^{11} x^2 + \alpha^{13} x + 1 \end{array}$$

We can verify that $\deg(U_2(x)) = 2$ is lower than or equal to t ($t = 2$) and that the degree of R_2 is lower than or equal to t. The algorithm is therefore terminated and polynomial $\Lambda(x)$ has the expression:

$$\Lambda(x) = U_2(x) = 1 + \alpha^{13}x + \alpha^{11}x^2$$
$$\Gamma(x) = R_2(x) = \alpha^{13}x$$

For a binary BCH code it is not necessary to use the error evaluator polynomial to determine the value of coefficients e_3 and e_8. However, we can verify that:

$$e_3 = \alpha^3 \frac{\Gamma(\alpha^{-3})}{\Lambda'(\alpha^{-3})} = 1$$
$$e_8 = \alpha^8 \frac{\Gamma(\alpha^{-8})}{\Lambda'(\alpha^{-8})} = 1$$

The decoded word is therefore $\hat{c}(x) = r(x) + e(x) = 0$ and the two errors have been corrected.

4.4.4 Hard input decoding performance of Reed-Solomon codes

Recall that for a Reed-Solomon code, the blocks of information to encode and the codewords are made up of k and $n = q - 1$ ($q = 2^m$) q-ary symbols respectively. The probability $P_{e,\text{word}}$ of having a wrong codeword after hard decoding can be upper bounded by:

$$P_{e,\text{word}} \leq \sum_{j=t+1}^{n} \binom{n}{j} p_s^j (1 - p_s)^{n-j} \tag{4.51}$$

where p_s is the error probability per q-ary symbol on the transmission channel and t is the code correction capability in number of q-ary symbols.
When a codeword is wrongly decoded, the error probability per corresponding $P_{e,\text{symbol}}$ symbol after decoding is upper bounded by:

$$P_{e,\text{symbol}} \leq \frac{1}{n} \sum_{j=t+1}^{n} (j + t) \binom{n}{j} p_s^j (1 - p_s)^{n-j} \tag{4.52}$$

The binary error probability after decoding is obtained from the error probability per symbol, taking into account that a symbol is represented by m bits:

$$P_{e,\text{bit}} = 1 - (1 - P_{e,\text{symbol}})^{\frac{1}{m}}$$

At high signal to noise ratio, we can approximate the binary error probability after decoding:

$$P_{\text{e,bit}} \cong \frac{1}{m} P_{\text{e,symbol}} \quad \frac{E_b}{N_0} >> 1 \ .$$

Bibliography

[4.1] R.H. Morelos-Zaragoza. *The Art of Error Correcting Coding*. John Wiley & sons, 2005.

[4.2] J. G. Proakis. *Digital Communications*. McGraw-Hill, New-York, 4th edition, 2000.

Appendix

Notions about Galois fields and minimal polynomials

Definition

A Galois field with $q = 2^m$ elements denoted $\mathbf{F}_q$, where m is a positive integer is defined as a polynomial extension of the field with two elements $(0, 1)$ denoted $\mathbf{F}_2$. The polynomial $\varphi(x)$ used to build field $\mathbf{F}_q$ must be

- irreducible, that is, non factorizable in $\mathbf{F}_2$ (in other words, 0 and 1 are not roots of $\varphi(x)$),
- of degree m,
- and with coefficients in $\mathbf{F}_2$.

The elements of a Galois field $\mathbf{F}_q$ are defined modulo $\varphi(x)$ and thus, each element of this field can be represented by a polynomial with degree at most equal to $(m - 1)$ and with coefficients in $\mathbf{F}_2$.

Example 1

Consider an irreducible polynomial $\varphi(x)$ in the field $\mathbf{F}_2$ of degree $m = 2$.

$$\varphi(x) = x^2 + x + 1$$

This polynomial enables a Galois field to be built with 4 elements. The elements of this field $\mathbf{F}_4$ are of the form:

$$a\alpha + b \quad \text{where} \quad a, b \in \mathbf{F_2}$$

that is:

$$\mathbf{F}_4 : \{0,\ 1,\ \alpha,\ \alpha + 1\}$$

We can see that if we raise element α to successive powers 0, 1 and 2 we obtain all the elements of field $\mathbf{F}_4$ with the exception of element 0. Indeed, α^2 is still equal to $(\alpha+1)$ modulo $\varphi(\alpha)$. Element α is called the *primitive element* of field $\mathbf{F}_4$.

The elements of field $\mathbf{F}_4$ can also be represented in binary form:

$$\mathbf{F}_4 : \{00,\ 01,\ 10,\ 11\}$$

The binary couples correspond to the four values taken by coefficients a and b.

Primitive element of a Galois field

We call the primitive element of a Galois field $\mathbf{F}_q$, an element of this field that, when it is raised to successive powers $0, 1, 2, \cdots, (q-2)$; $q = 2^m$, makes it possible to retrieve all the elements of the field except element 0. Every Galois field has at least one primitive element. If α is a primitive element of field $\mathbf{F}_q$ then, the elements of this field are:

$$\mathbf{F}_q = \left\{0, \alpha^0, \alpha^1, \cdots, \alpha^{q-2}\right\} \text{ with } \alpha^{q-1} = 1$$

Note that in such a Galois field the "-" sign is equivalent to the "+" sign, that is:

$$-\alpha^j = \alpha^j \quad \forall j \in \{0, 1, \cdots, (q-2)\}$$

Observing that $2\alpha^j = 0$ modulo 2, we can always add the zero quantity $2\alpha^j$ to $-\alpha^j$ and we thus obtain the above equality.

For example, for field $\mathbf{F}_4$ let us give the rules that govern the addition and multiplication operations. All the operations are done modulo 2 and modulo $\alpha^2 + \alpha + 1$.

+	**0**	**1**	α	α^2
0	0	1	α	α^2
1	1	0	$1+\alpha=\alpha^2$	$1+\alpha^2=\alpha$
α	α	$1+\alpha=\alpha^2$	0	$\alpha+\alpha^2=1$
α^2	α^2	$1+\alpha^2=\alpha$	$\alpha+\alpha^2=1$	0

Table 4.7 – Addition in field $\mathbf{F}_4$.

Minimal polynomial with coefficients in $\mathbf{F_2}$ associated with an element of a Galois field $\mathbf{F}_q$

The minimal polynomial $m_\beta(x)$ with coefficients in $\mathbf{F_2}$ associated with any element β of a Galois field $\mathbf{F}_q$, is a polynomial of degree at most equal to

$\times$	$\mathbf{0}$	$\mathbf{1}$	α	α^2
$\mathbf{0}$	0	0	0	0
$\mathbf{1}$	0	1	α	α^2
α	0	α	α^2	$\alpha^3 = 1$
α^2	0	α^2	$\alpha^3 = 1$	$\alpha^4 = \alpha$

Table 4.8 – Multiplication in field $\mathbf{F}_4$.

$m = \log_2(q)$, having β as a root. This polynomial is unique and irreducible in $\mathbf{F_2}$. If β is a primitive element of Galois field $\mathbf{F}_q$ then polynomial $m_\beta(x)$ is exactly of degree m. Note that a polynomial with coefficients in $\mathbf{F_2}$ satisfies the following property:

$$[f(x)]^2 = f(x^2) \Rightarrow [f(x)]^{2^p} = f(x^{2^p})$$

So, if β is a root of polynomial $f(x)$ then $\beta^2, \beta^4, \cdots$ are also roots of this polynomial. The minimal polynomial with coefficients in $\mathbf{F}_2$ having β as a root can then be written in the form:

$$m_\beta(x) = (x+\beta)(x+\beta^2)(x+\beta^4)\cdots$$

If β is a primitive element of $\mathbf{F}_q$, the minimal polynomial with coefficients in $\mathbf{F_2}$ being of degree m, it can also be written:

$$m_\beta(x) = (x+\beta)(x+\beta^2)(x+\beta^4)\cdots(x+\beta^{2^{m-1}})$$

Example 2

Let us calculate the minimal polynomial associated with the primitive element α of Galois field $\mathbf{F}_4$.

$$\mathbf{F}_4 : \left\{0,\, 1,\, \alpha,\, \alpha^2\right\}$$

The minimal polynomial associated with element α therefore has α and α^2 ($m = 2$) as roots, and can be expressed:

$$m_\alpha(x) = (x+\alpha)(x+\alpha^2) = x^2 + x(\alpha+\alpha^2) + \alpha^3$$

Taking into account the fact that $\alpha^3 = 1$ and that $\alpha + \alpha^2 = 1$ in field $\mathbf{F}_4$, the polynomial $m_\alpha(x)$ is thus equal to:

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

Minimal polynomial with coefficients in $\mathbf{F}_q$ associated with an element in a Galois field $\mathbf{F}_q$

The minimal polynomial $m_\beta(x)$, with coefficients in the Galois field $\mathbf{F}_q$ associated with an element $\beta = \alpha^j$ (α a primitive element of field $\mathbf{F}_q$) of this field, is the lowest degree polynomial having β as a root.
Recalling that for a polynomial with coefficients in $\mathbf{F}_q$, we can write:

$$[f(x)]^q = f(x^q) \Rightarrow [f(x)]^{q^p} = f(x^{q^p})$$

Then if β is a root of polynomial $f(x)$, $\beta^q, \beta^{q^2}, \cdots$ are also roots of this polynomial.
Since in field $\mathbf{F}_q$ $\alpha^{q-1} = 1$, then $\beta^{q^p} = (\alpha^j)^{q^p} = \alpha^j = \beta$ and, thus, minimal polynomial $m_\beta(x)$ is simply equal to:

$$m_\beta(x) = x + \beta$$

These results on minimal polynomials are used to determine the generator polynomials of particular cyclic codes (BCH and Reed-Solomon).

Primitive polynomials

A polynomial with coefficients in $\mathbf{F}_2$ is primitive if it is the minimal polynomial associated with a primitive element of a Galois field. A primitive polynomial is thus irreducible in $\mathbf{F}_2$ and consequently can be used to build a Galois field. When a primitive polynomial is used to build a Galois field, all the elements of the field are obtained by raising the primitive element, the root of the primitive polynomial, to successively increasing powers. As the main primitive polynomials are listed in the literature, the construction of a Galois field with $q = 2^m$ elements can then be done simply by using a primitive polynomial of degree m. Table 4.9 gives some primitive polynomials.

To end this introduction to Galois fields and minimal polynomials, let us give an example of a Galois field with $q = 16$ ($m = 4$) elements built from the primitive polynomial x^4+x+1. This field is used to build generator polynomials of BCH and Reed-Solomon codes and to decode them. The elements of this field are:

$$\mathbf{F}_{16} = \left\{0,\, 1,\, \alpha,\, \alpha^2,\, \alpha^3 \cdots \alpha^{14}\right\}$$

where α is a primitive element of $\mathbf{F}_{16}$. With these 16 elements, we can also associate a polynomial representation and a binary representation. The polynomial representation of an element of this field is of the form:

$$a\alpha^3 + b\alpha^2 + c\alpha + d$$

where a, b, c and d are binary coefficients belonging to $\mathbf{F}_2$.

Degree of the polynomial	Primitive polynomial
2	$\alpha^2 + \alpha + 1$
3	$\alpha^3 + \alpha + 1$
4	$\alpha^4 + \alpha + 1$
5	$\alpha^5 + \alpha^2 + 1$
6	$\alpha^6 + \alpha + 1$
7	$\alpha^7 + \alpha^3 + 1$
8	$\alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
9	$\alpha^9 + \alpha^4 + 1$
10	$\alpha^{10} + \alpha^3 + 1$

Table 4.9 – Examples of primitive polynomials

Galois field $\mathbf{F}_{16}$ being made up of 16 elements, the binary representation of an element of this field is done with the help of 4 binary symbols belonging to $\mathbf{F}_2$. These 4 symbols are equal to the values taken by coefficients a, b, c and d respectively.

Elements of the field	Polynomial representation	Binary representation
0	0	0 0 0 0
1	1	0 0 0 1
α	α	0 0 1 0
α^2	α^2	0 1 0 0
α^3	α^3	1 0 0 0
α^4	$\alpha + 1$	0 0 1 1
α^5	$\alpha^2 + \alpha$	0 1 1 0
α^6	$\alpha^3 + \alpha^2$	1 1 0 0
α^7	$\alpha^3 + \alpha + 1$	1 0 1 1
α^8	$\alpha^2 + 1$	0 1 0 1
α^9	$\alpha^3 + \alpha$	1 0 1 0
α^{10}	$\alpha^2 + \alpha + 1$	0 1 1 1
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	1 1 1 0
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	1 1 1 1
α^{13}	$\alpha^3 + \alpha^2 + 1$	1 1 0 1
α^{14}	$\alpha^3 + 1$	1 0 0 1

Table 4.10 – Different representations of the elements of Galois field $\mathbf{F}_{16}$

Example 3

Some calculations in field $\mathbf{F}_{16}$ are given in table 4.11 for addition, in table 4.12 for multiplication and in table 4.13 for division.

$+$	α^2	α^4
α^8	$0100 + 0101 = 0001 = 1$	$0011 + 0101 = 0110 = \alpha^5$
α^{10}	$0100 + 0111 = 0011 = \alpha^4$	$0011 + 0111 = 0100 = \alpha^2$

Table 4.11 – Addition in $\mathbf{F}_{16}$

$\times$	α^2	α^6
α^8	α^{10}	α^{14}
α^{14}	$\alpha^{16} = \alpha$ as $\alpha^{15} = 1$	$\alpha^{20} = \alpha^5$ as $\alpha^{15} = 1$

Table 4.12 – Multiplication in $\mathbf{F}_{16}$

$\div$	α^2	α^{12}
α^8	$\alpha^{-6} = \alpha^9$ as $\alpha^{15} = 1$	α^4
α^{14}	$\alpha^{-12} = \alpha^3$ as $\alpha^{15} = 1$	$\alpha^{-2} = \alpha^{13}$ as $\alpha^{15} = 1$

Table 4.13 – Division in $\mathbf{F}_{16}$