Introduction to algebraic theory of linear error correcting codes (incomplete version)

Robert Rolland

September 29, 2009

Contents

Chapter 1

Basic facts

1.1 Introduction

The following notes are an approach for beginners of linear block coding theory. We only tackle a little bit of the theory. In particular we don't speak about the probabilistic aspects, we don't study all the numerous type of codes etc ...

1.2 Simple example

Let us suppose that we want to transmit the message

$$01101$$

and let us in fact transmit the message three times

$$011010110101101.$$

By majority logic, **if no more than one error** occurs during the transmission process, we can recover the true message.

If two errors occur during the transmission, it is not always possible to recover the true message. For example the wrong message (2 errors)

$$001010010101101,$$

decoded by majority logic, gives the wrong result

$$00101.$$

But it is always possible to detect errors.

If three errors or more occur during the transmission, it is not always possible to detect errors. For example the wrong message (3 errors)

$$001010010100101$$

is considered as an errorfree message.

We say that this code

- **corrects one error,**
- **detects two errors.**

The method involved here is to send a longer message than the initial one and to use the redundancy of the encoded message to compute the most probable initial message.

1.3 Alphabet, basic structures

If you want to represent information in a suitable form, allowing an easy processing, you have to choose a set of symbols (the alphabet) with a rich structure, to write this information. The finite fields are very convenient for such a task.

1.4 Linear algebraic aspects of linear codes

1.4.1 Generalities

Let $\mathbb{F}_q$ be the finite field with q elements where $q = p^m$ is a power of a prime number.

Let E be a k-dimensional vector space over the field $\mathbb{F}_q$, with a fixed basis $(e_1, e_2, \cdots, e_k)$.

Let F be a n-dimensional vector space over the field $\mathbb{F}_q$, with a fixed basis $(f_1, f_2, \cdots, f_n)$. We suppose $n > k$.

Let l be a linear injection from E into F. The subspace $C = l(E)$ is a code.

For transmission purpose we use this situation in the following way.

$$\begin{tikzcd}
\mathbb{F}_q^k \arrow[r, "l"] & \mathbb{F}_q^n \arrow[r, rightsquigarrow] & \mathbb{F}_q^n \\
 & C = l(\mathbb{F}_q^k) &
\end{tikzcd}$$

$$u = (u_1, \cdots, u_k) \qquad x = (x_1, \cdots, x_n) \qquad y = (y_1, \cdots, y_n)$$

u is the **raw** word,
$x \in C$ is the **encoded** word,
$y = x + e$ is the **received** word and e the error.
If $y \in C$ you assume that probably $y = x$. When y is not in C, errors occur. In this case the problem is to recover x from y. To do this we try to find the **nearest word** in some sense of x wich is in C. Computing u from x is an another problem which is not the problem of error correction but just the resolution of the linear system $l(u) = x$.

So now, we are focused on the problem of recovering x from y.

1.4.2 Hamming distance, correction ability, parameters

To formalize the problem we define an adapted notion of distance between words of F.
Let us define the **Hamming distance** Δ on F by

$$\Delta(x, y) = Card\{i \mid x_i \neq y_i\}.$$

As an obvious application, if x is en encoded message and y the received message, we claim that $\Delta(x, y)$ errors occur.

The dimension n of F is called the **length of the code**. The dimension k of C (or of E) is called the **dimension of the code**.

To avoid confusion between the codewords, each word of C must be far from the other words of C. We define the **minimal distance** of the code

$$d(C) = min\{\Delta(x, x') \mid x \neq x', x \in C, x' \in C\}.$$

Taking into account the linear structure of F we define the **weight** of a word

$$w(x) = Card\{i \mid x_i \neq 0\},$$

So that

$$\Delta(x,y) = w(x-y),$$

and

$$d(C) = inf\{w(x) \mid x \neq 0, x \in C\}.$$

The **parameters** of a code C are its **length**, its **dimension**, its **minimal distance**. A code with length n, dimension k and minimal distance d is called a $[n,k,d]$-code.

Clearly we try to obtain codes with large minimal distance and low rate of increase of the word length. To go into more details we define the **rate of the code** or the **efficiency of the code**

$$R = \frac{k}{n},$$

and the relative distance of the code

$$\delta = \frac{d}{n}.$$

We are now in position to compute the correction ability of a code.

Theorem 1.4.1 *Let C be a code with minimal distance $d(C)$. Then C corrects*

$$t = \left[\frac{d(C)-1}{2}\right]$$

errors.
That is, if during a transmission the number of errors is $\leq t$, we can recover the true transmitted word x from the wrong received word y because there is only one x which is at minimum hamming distance from y.
This number t is optimum,

Proof. Let x the transmitted message and y the received one. Let $x' \in C$ with $x' \neq x$. Then $\Delta(x,x') \geq d$. But by triangle inequality

$$\Delta(y,x') \geq |\Delta(x,x') - \Delta(x,y)|$$

so,

$$\Delta(y,x') \geq d(C) - t > t.$$

Let us remark that if in the transmission process the number of errors is $> t$ and if x is a word in C such that $w(x) = d(C)$, it is possible to receive a word which is nearest from 0 than from x, and the correction is impossible because $0 \in C$.

1.4.3 Definition of a code

We have seen in the previous paragraph that the correcting process depends only on the space F and its subspace C. The space E and the linear function l are just a mean to encode the raw words. The encoder, that is the space E, the choosen basis of E, the linear injection l, can be changed. Different encoders can use the same code.

Definition 1.4.1 *A linear code is a subspace of a finite dimensional vector space F, over a finite field $\mathbb{F}_q$, in wich a basis $(f_1, f_2, \cdots, f_n)$ is fixed.*

Remark: when a basis of F is fixed we have a natural isomorphism from F into $\mathbb{F}_q^n$. So we can define a linear code to be a subspace of an $\mathbb{F}_q^n$.

1.4.4 General schemes for encoding and decoding

Generator matrix

Let F be a space of dimension n over $\mathbb{F}_q$ with a fixed basis $(f_1, f_2, \cdots, f_n)$ and C a subspace of F of dimension k (a code of length n and dimension k).

Let us suppose that we have an encoder for this code, that is a space E of dimension k, a basis $(e_1, e_2, \cdots, e_k)$ of E and l a linear injection from E into F such that $C = l(E)$.
The transposed matrix G of the matrix of l relative to the given basis is a **generator matrix** of C. If $u = (u_1, u_2, \cdots, u_k)$ is a raw word, the corresponding encoded word is

$$x = uG.$$

Remark: in coding theory, we use to write the components of a word in a row and not in a column, so for a linear application we have to transpose the matrix.

The matrix G has k rows and n columns.

Example: An Hamming code on $\mathbb{F}_2$.

$$E = \mathbb{F}_2^4 \quad F = \mathbb{F}_2^7,$$

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

If $u = (u_1, u_2, u_3, u_4)$ and $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ such that $x = uG$, then

$$x_1 = u_1, \quad x_2 = u_2, \quad x_3 = u_3, \quad x_4 = u_4,$$

$$x_5 = u_2 + u_3 + u_4,\ x_6 = u_1 + u_3 + u_4,\ x_7 = u_1 + u_2 + u_4.$$

Check application, parity check matrix

A linear application h from F into a linear space F' such that $C = Ker(h)$ is call a **check application**. When we receive a word y we verifie that this word is a codeword by computing $h(y)$ (called the **syndrom**). If $h(y) = 0$ we assume that the word is errorfree.
It is easy to see that the rank of such an h is

$$rg(h) = n - k = r.$$

The number r is called the **redondancy**.
The matrix H of h when F is endowed with the given basis and F' with any basis is called a **parity check matrix**.

Linear algebra characterization of the minimal distance

Let h a check application of a code $C \subset F$ and $(f_1, f_2, \cdots, f_n)$ the given basis of F.

Theorem 1.4.2 *The number d is the minimal distance of the code C if and only if*

a) There is a family of d vectors in the family $(h(f_i))_i$ which is linearly dependant.

b) Each family with $d-1$ vectors extracted from the family $(h(f_i))_i$ is free.

Proof. Let d the minimal distance of the code. There is a codeword of weight d

$$x = x_{i_1} f_{i_1} + \cdots + x_{i_d} f_{i_d}.$$

Hence

$$h(x) = x_{i_1} h(f_{i_1}) + \cdots + x_{i_d} h(f_{i_d}).$$

But x is a codeword, so h(x)=0. Hence $h(f_{i_1}), \cdots, h(f_{i_d})$ are linearly dependant. Now if there is a linear decomposition

$$\lambda_{i_1} h(f_{i_1}) + \cdots + \lambda_{i_{d-1}} h(f_{i_{d-1}}) = 0,$$

we have

$$\lambda_{i_1} f_{i_1} + \cdots + \lambda_{i_{d-1}} f_{i_{d-1}} \in C.$$

But the minimal weight for a non-null codeword is d. So $\lambda_{i_j} = 0$ for all j.

Reciprocally, if b) occurs, a non-null word of weight $\leq d-1$ cannot be in C. If a) occurs, a codeword of weight d exists.

Example: Let us consider again the Hamming code on $\mathbb{F}_2$.

$$E = \mathbb{F}_2^4 \quad F = \mathbb{F}_2^7,$$

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

We can verify that

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

is a parity check matrix. Two columns of H are always independant and there exist three columns dependant. So the minimal distance is 3. This code corrects 1 error.

The singleton bound

Theorem 1.4.3 *Let C be a linear $[n,k,d]$-code. Then*

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

Proof. To prove the theoreme, let h be a parity check matrix. We have

$$dim(Ker(h)) + dim(Im(h)) = n.$$

Hence

$$dim(Im(h)) = n - k.$$

But each family extracted from $(h(f_1), \cdots, h(f_n))$ with $d-1$ vectors is free, so that $d - 1 \leq n - k$.

This bound can be reached. Codes reaching this bound are called **maximal distance separable codes** (MDS codes).

Systematic encoder

Let F be a linear space with a given basis $(f_1, \cdots, f_n)$. Let C be a code of dimension k in F.

Theorem 1.4.4 *We can choose a vector space E of dimension k, a basis $(e_1, \cdots, e_k)$ of E, a linear bijective map l from E onto C and a permutation of the basis of F such that the corresponding generator matrix is*

$$G = [I_k \ A],$$

where I_k is the unit matrix of size k.

Proof. As a consequence of the incomplete basis theorem, it is possible (with an eventual permutation of the basis of F) to write

$$F = C \bigoplus D,$$

where

$$D = [f_{k+1}, \cdots, f_n].$$

Hence,

$$\begin{array}{ll} f_1 = c_1 + d_1 & c_1 \in C, d_1 \in D \\ f_2 = c_2 + d_2 & c_2 \in C, d_2 \in D \\ \vdots & \vdots \\ f_k = c_k + d_k & c_k \in C, d_k \in D. \end{array}$$

The family $(c_1, \cdots, c_k)$ is a basis of C.

Let $E = C$, $e_i = c_i$ and l the identity map. The generator matrix associated to this encoder is in the needed form.

Such an encoder is called a **systematic encoder**. It is a very nice situation to dispose of a systematic encoder because if the raw message is

$$(u_1, u_2, \cdots, u_k),$$

the encoded message is

$$(u_1, u_2, \cdots, u_k, x_{k+1}, \cdots, x_n).$$

Theorem 1.4.5 *If $G = [I_k \ A]$ is the generator matrix of a systematic encoder for C, the matrix*

$$H = [-A^t \ I_{n-k}]$$

is a parity check matrix for C.

Proof. With the notations of the previous theorem's proof, let h be the projection from $F = C \bigoplus D$ onto D with kernel C. The matrix of the check function h is $H = [-A^t \ I_{n-k}]$.

General decoding scheme

We give here a decoding scheme for an abstract linear code. This algorithm is intractable for large codes.

Let $x \in C$ be a codeword and $y \in F$ the transmitted word. Hence $y = x + e$ where e is the error. Let h be a check map. The syndrome of y is $h(y)$.

$$h(y) = h(x) + h(e) = h(x).$$

Let $\mathcal{F} = F/Ker(h) = F/C$. We can write

$$\mathcal{F} = \{Y_1, Y_2, \cdots, Y_s\}$$

where

$$Y_i = \{e_i, y_{i,2}, y_{i,3}, \cdots, y_{i,r}\}$$

and where e_i is one element of minimal weight among those in the same coset Y_i.

So, if the received message is y, we have to compute the syndrome $h(y)$, then find the class Y_i of y and then choose the head e_i of this class to be the most probable occuring error. Hence we decode by $x = y - e_i$.

This algorithm needs a long preprocessing: write down all the classes Y_i and for each class find a head and the common syndrome of the elements.

1.4.5 Dual codes

To define and study **dual codes** we first recall some notions on vector space duality.

Let F be a n dimensional vector space over $\mathbb{F}_q$ and let $(f_1, \cdots, f_n)$ be a fixed basis of F.
Let us denote by ϕ the bilinear form on $F \times F$ defined by $\phi(x, y) = \sum_{i=1}^{n} x_i y_i =< x, y >$, where $x = \sum x_i f_i$ and $y = \sum y_i f_i$.
Let T be the linear application from F into the algebraic dual F^* of F defined by

$$T(y) = \phi_y,$$

where

$$\phi_y(x) = \phi(x, y) =< x, y > .$$

Lemma 1.4.1 *T is a linear isomorphism from F onto F^*. That is, for all $g \in F^*$ there is one and only one $y \in F$ such that*

$$g(x) =< x, y > .$$

Proof. Let us compute the kernel of the linear application T. $T(y) = \phi_y = 0$ if and only if for all x in F, we have $\phi_y(x) = 0$, that is $\sum x_i y_i = 0$. From the particular values $x = (0, \cdots, 1, \cdots, 0)$ we get y_i=0.
Then $Ker(T) = \{0\}$ and T is injective. But $dim(F) = dim(F^*)$, so that T is one to one.

Let C a subspace of F. Let us define $C^\perp$ to be

$$C^\perp = \{y \in F \mid \phi_y(x) = 0 \ \forall x \in C\}.$$

We get the following lemma.

Lemma 1.4.2

$$C \subset C^{\perp\perp}.$$

Proof. If $x \in C$ then for all y in $C^\perp$ we have $< x, y >= 0$, so that $x \in C^{\perp\perp}$.

Lemma 1.4.3 *If $C_1 \subset C_2$ then $C_2^\perp \subset C_1^\perp$.*

Proof. If $y \in C_2^\perp$ then for all x in C_2 and in particular for all x in C_1 we have $< x, y >= 0$, so that $y \in C_1^\perp$.

Lemma 1.4.4

$$C = C^{\perp\perp}.$$

Proof. Let $x \notin C$. Let us complete x in a basis of F adding a basis $(a_1, a_2, \cdots, a_k)$ of C and eventually others vectors. We obtain a basis

$$(x, a_1, a_2, \cdots, a_k, a_{k+1}, \cdots, a_{n-1}).$$

Now, let ψ the linear form defined on F by

$$\psi(\lambda_0 x + \lambda_1 a_1 + \cdots + \lambda_{n-1} a_{n-1}) = \lambda_0$$

Let us remark that in particular $\psi(x) = 1$.
We know by 1.4.1 that there exist one and only one y such that for all u,

$$\psi(u) =< u, y > .$$

From the definition, if $c \in C$ then $\psi(c) = 0$. So that $< c, y >= 0$. We conclude that $y \in C^\perp$ and $< x, y >= 1$, so that $x \notin C^{\perp\perp}$. The lemma follows.

Lemma 1.4.5 *Let $(y_1, \cdots, y_s)$ a free family in F, then the linear map θ from F in $\mathbb{F}_q^s$ defined by*

$$\theta(x) = (< x, y_1 >, < x, y_2 >, \cdots, < x, y_s >)$$

is onto.

Proof. The rank of θ is the rank s of the vector family $(y_1, \cdots, y_s)$.
Now we precise the notion of dual code.

Definition 1.4.1 *The dual code of C is $C^\perp$.*

Theorem 1.4.6 *If H is a parity check matrix for C, H is a generator matrix for $C^\perp$ and if G is a generator matrice for C, G is a parity check matrix for $C^\perp$.*

Proof. Let $(y_1, \cdots, y_s)$ a basis of $C^{\perp}$) and h the linear map defined on F by

$$h(x) = (< x, y_1 >, \cdots, < x, y_s >).$$

So, $x \in Ker(h)$ if and only if for all $i < x, y_i >= 0$ that is, if and only if for all $y \in C^{\perp}$ we have $< x, y >= 0$. Then

$$Ker(h) = C^{\perp\perp} = C.$$

Moreover, h is onto, hence $s = n - k$. Now, let h^t the transposed of h. For all $u \in \mathbb{F}_q^s$ and for all $y \in F$,

$$< h^t(u), y >=< u, h(y) > .$$

If $y \in C$ we have $h(y) = 0$ and $< h^t(u), y >= 0$. Hence $h^t(u) \in C^{\perp}$. Moreover, h^t is injective

$$h^t(u) = 0 \Leftrightarrow \forall y \in F, < u, h(y) >= 0 \Leftrightarrow \forall z \in \mathbb{F}_q^s, < u, z >= 0 \Leftrightarrow u = 0.$$

Hence $Im(h^t) = C^{\perp}$.

Now let g a linear encoder from $\mathbb{F}_q^k$ into F for C. We can write

$$g^t(y) = 0 \Leftrightarrow \forall v, < g^t(y), v >= 0 \Leftrightarrow \forall v, < y, g(v) >= 0,$$

$$g^t(y) = 0 \Leftrightarrow \forall x \in C, < y, x >= 0.$$

Hence

$$g^t(y) = 0 \Leftrightarrow y \in c^{\perp}.$$

Chapter 2

Concrete linear codes built with polynomial functions

2.1 Concrete codes

The vector space structure is not sufficient to dispose of a nice decoding process. The general decoding scheme is intractable for large codes. So, we define codes related with stronger structures, for example codes related with spaces of functions. In that way we get new tools: multiplication of functions, evaluation of functions, Fourier transform, etc ... These tools help us to simplify the encoding and the decoding processes.

2.2 Order one Reed Muller codes

2.2.1 Definition, parameters, encoding

Let F be the space of functions from $G = \{0,1\}^m$ into $\{0,1\}$. Let us choose the evaluation basis of F, that is the basis given by the family $(e_a)_{a\in\{0,1\}^m}$ where

$$e_a(x) = \left\{ \begin{array}{lll} 1 & if & x = a \\ 0 & if & x \neq a \end{array} \right. ,$$

ordered by the value

$$val(a) = \sum_{i=1}^{m} a_i 2^{i-1},$$

of a.
In this basis, $f \in F$ is given by

$$f = \sum_{a} f(a) e_a,$$

that is by the row vector $(f(a))_{a \in \{0,1\}^m}$.

The **order one reed-Muller code** is the subsbace C of F of the polynomial functions with m indeterminates and degree ≤ 1. Such a function can be written

$$g(x_1, x_2, \cdots, x_m) = u_0 + u_1 x_1 + \cdots + u_m x_m.$$

The length of this code is 2^m, its dimension is $k = m + 1$.

Let $f = (f(a))_{a \in \{0,1\}^m} \in F$ anf $N(f)$ the number of zeroes of f. The weight of f is given by

$$W(f) = 2^m - N(f).$$

Then, if g is a non-constant polynomial function in C, $N(g)$ is the cardinality of an hypersurface,

$$N(g) = 2^{m-1},$$

and

$$W(g) = 2^m - 2^{m-1} = 2^{m-1}.$$

So the weight repartition is:
one word of weight 0 (the word $g = 0$),
one word of weight 2^m (the word $g = 1$)
the others ($2^k - 2$ words) of weight 2^{m-1}.

In particular, the minimal distance is $d = 2^{m-1}$.

It is easy to describe an **encoding process:**
starting from a raw word $(u_0, u_1, \cdots, u_m)$, we consider the polynomial function $g(x_1, \cdots, x_m) = u_0 + u_1 x_1 + \cdots u_m x_m$ and then we compute the encoded word

$$(g(0,0,\cdots,0), g(1,0,\cdots,0), g(0,1,0,\cdots,0), g(1,1,0,\cdots,0), \cdots, g(1,1,\cdots,1)).$$

2.2.2 Decoding

Hadamard transform

The Hadamard transform is the Fourier transform related to the group $\{0,1\}^m$. The characters of this group are the Walsh functions

$$\chi_u(v) = (-1)^{<u,v>},$$

where $u = (u_1, u_2, \cdots, u_m)$, $v = (v_1, v_2, \cdots, v_m)$ and $< u, v >= u_1 v_1 + \cdots + u_m v_m$. The Hadamard transform of a complex valued function ϕ is defined by

$$\widehat{\phi}(v) = \sum_u \phi(u)(-1)^{<u,v>}.$$

We know that

$$\phi(u) = \frac{1}{2^m} \sum_v \widehat{\phi}(v)(-1)^{<u,v>}.$$

Remark: the elements $u, v \in \{0,1\}^m$ can be considered as numbers between 0 and $2^m - 1$, using binary representation. We will use the two interpretations.

Let us explain now how to compute the Hadamard transform of ϕ by the Fast Hadamard Transform algorithm.

We must evaluate

$$P(x) = \sum_{y=0}^{2^r-1} a_y (-1)^{<x,y>}$$

where $x = (x_1, x_2, ..., x_r)$ et $y = (y_1, y_2, ..., y_r)$. Let us define

$$P_0(u) = \sum_{v=0}^{2^{r-1}-1} a_u (-1)^{<u,v>}$$

and

$$P_1(u) = \sum_{v=0}^{2^{r-1}-1} a_{u+2^{r-1}} (-1)^{<u,v>}$$

where $u = (u_1, u_2, ..., u_{r-1})$ et $v = (v_1, v_2, ..., v_{r-1})$. We can write

$$P(x) = P_0(\tau(x)) + (-1)^{x_r} P_1(\tau(x))$$

where $\tau(x) = (x_1, x_2, ..., x_{r-1})$.

Let us consider the case $n = 2^3 = 8$. We obtain

$$P_{000}(X) = a_0, P_{001}(X) = a_1, P_{010}(X) = a_2, P_{011}(X) = a_3$$

$$P_{100}(X) = a_4, P_{101}(X) = a_5, P_{110}(X) = a_6, P_{111}(X) = a_7$$

and the fast Hadamard transform is done following the figure 2.1.

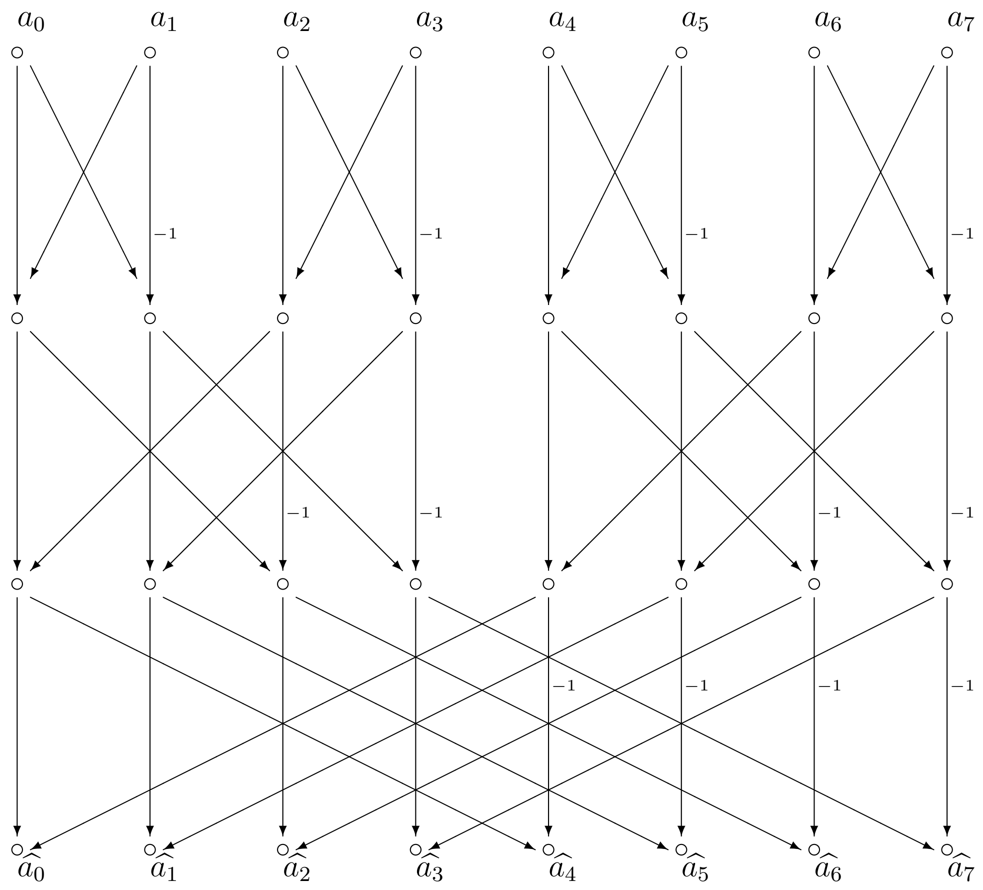

Figure 2.1: La FHT sur 8 points

Using Hadamard transform for decoding

Let f be the received word, f is a function from $\{0,1\}^m$ into $\{0,1\}$. Let ϕ the function from $\{0,1\}^m$ into $\mathbb{C}$ defined by

$$\phi(u) = (-1)^{f(u)}.$$

We have

$$\widehat{\phi}(u) = \sum_v (-1)^{f(v)}(-1)^{<u,v>},$$

$$\widehat{\phi}(u) = \sum_v (-1)^{f(v)+<u,v>}.$$

Hence $\widehat{\phi}(u)$ is the number of 0 minus the number of 1 in the binary vector

$$f + \sum_{i=1}^{m} u_i X_i,$$

so that,

$$\widehat{\phi}(u) = 2^m - 2dist(f, \sum_{i=1}^{m} u_i X_i),$$

and

$$dist(f, \sum_{i=1}^{m} u_i X_i) = \frac{1}{2}(2^m - \widehat{\phi}(u)).$$

In an analogous way we obtain also

$$dist(f, 1 + \sum_{i=1}^{m} u_i X_i) = \frac{1}{2}(2^m + \widehat{\phi}(u)).$$

As a consequence we have the following decoding algorithm.

Compute $\widehat{\phi}(u)$ for all u (that is compute the Hadamard transform).
Choose u such that $|\widehat{\phi}(u)|$ is maximum.
If $\widehat{\phi}(u) \geq 0$ the closest codeword from f is $\sum_{i=1}^{m} u_i X_i$.
If $\widehat{\phi}(u) < 0$ the closest word from f is $1 + \sum_{i=1}^{m} u_i X_i$.

An example

Let $m = 3$. The received word is $f = (0, 1, 1, 1, 0, 1, 1, 0)$. So we compute $\phi = (1, -1, -1, -1, 1, -1, -1, 1)$. The Hadamard transform is performed in 3 stages:

$$\begin{array}{rrrrrrrr}
1 & -1 & -1 & -1 & 1 & -1 & -1 & 1 \\
0 & 2 & -2 & 0 & 0 & 2 & 0 & -2 \\
-2 & 2 & 2 & 2 & 0 & 0 & 0 & 4 \\
-2 & 2 & 2 & 6 & -2 & 2 & 2 & -2
\end{array}$$

then the maximum is 6 with position's index $3 = (1, 1, 0)$ (the first index is $0 = (0, 0, 0)$). The raw word is $(0, 1, 1, 0)$ ($f = x_1 + x_2$).

2.3 Reed Solomon codes

2.3.1 Definition

Let $\mathbf{F}_q$ be the finite field with $q = 2^m$ elements and α a primitive element. We set $n = q - 1$. If $k < n$ we define

$$g(X) = (X - \alpha)(X - \alpha^2) \cdots (X - \alpha^{n-k}).$$

Now, the polynomial spaces will be endowed with the usual monomial basis. A polynomial is given by its polynomial writing or by the sequence of its coefficients.
Let F be the n dimensional space of polynomials s with degree$\leq n - 1$ and with coefficients in $\mathbf{F}_q$. The code C is the subspace of F of all multiples of $g(X)$.

The polynomial $g(X)$ is of degree $n-k$. We consider C the space of multiples of $g(X)$ with degree $\leq n - 1$. This space is isomorphic to the space of polynomial with degree $\leq (n - 1) - (n - k) = k - 1$. Hence, the dimension of C is k.

2.3.2 A systematic encoder for C

Let E the space of polynomials of degree $\leq k - 1$ and coefficients in $\mathbf{F}_q$. To such a polynomial $u(X) = (u_0, u_1, \cdots, u_{k-1})$ we associate the polynomial

$a(X) = (a_0, a_1, \cdots, a_{n-1})$ defined by

$$a(X) = r(X) + X^{n-k}u(X)$$

where $-r(X)$ $(= r(X))$ is the remainder of the euclidian division of $X^{n-k}u(X)$ by $g(X)$.

The euclidian division of $X^{n-k}u(X)$ by $g(X)$ gives us $X^{n-k}u(X) = v(X)g(X) + r(X)$. So, $a(X) = v(X)g(X)$, that is a multiple of $g(X)$. The application which maps $u(X)$ on $a(X)$ is an injective linear map from the space of degree $\leq k-1$ polynomials onto C.

The generator matrix begins by a block which is the unity matrix of order k. The encoder is systematic.

2.3.3 Another encoder for C

Let $u(X) = (u_0, u_1, \cdots, u_{k-1})$ a raw message. We set $u_k = u_{k+1} = \cdots = u_{n-1} = 0$ et $U = (u_0, u_1, \cdots, u_{n-1})$. Let $a(X) = (a_0, a_1, \cdots, a_{n-1})$ be the sequence having U for Mattson transform. So,

$$u_j = \hat{a}_j = \sum_{i=0}^{n-1} a_i \alpha^{-ij} = a(\alpha^{-j}) \quad 0 \leq j \leq n-1$$

$$a_i = \sum_{j=0}^{n-1} u_j \alpha^{ij} = \sum_{j=0}^{k-1} u_j \alpha^{ij} = u(\alpha^i) \quad 0 \leq i \leq n-1$$

Then the values a_i are the values of the polynomial $u(X) = \sum_{j=0}^{k-1} u_i X^j$ at the points α^i.

The inverse Mattson Solomon transform shows that the u_i are the values taken by $a(X)$ at the points α^{n-i}. But $u_k = u_{k+1} = \cdots = u_{n-1} = 0$, so, $a(X)$ is 0 at the points $\alpha, \alpha^2, \cdots, \alpha^{n-k}$. Then $a(X)$ is a multiple of $g(X)$. It is easy now to show that the map which transformes $u(x)$ in $a(X)$ is a linear bijection from the space of polynomials with degree $\leq k-1$ onto C.

Let us compute the minimal distance of C. We know that $a_i = u(\alpha^i)$. So each nul a_i comes from a root of $u(X)$. But the degree of $u(X)$ is $\leq k-1$, so that the number of coefficients a_i which are zero is at most $k-1$. Then $d \geq n-k+1$. Using the Singleton bound Singleton $(d \leq n-k+1)$ we conclude that $d = n-k+1$.

2.3.4 Decoding

Decoding Reed-Solomon codes was not so easy than decoding Reed-Muller codes. But Reed-Solomon codes are better and very interesting to encode compressed files. The better decoding algorithm was known as the Berlekamp Massey algorithm. But we dispose now of a new fine algorithm: Sudan algorithm. Reed-Solomon codes are often used in CIRC (cross-interleaved Reed-Solomon code) mode. Decoding such two cascaded codes is easier and faster.

Chapter 3

Families of codes

3.1 Codes domain

Let $\delta = \frac{d}{n}$ be the relative distance of a code and $R = \frac{k}{n}$ its rate. So to each code we can associate a point in the plane (δ, R). The Singleton bound determines a part of the plane where is situated the point (δ, R):
in the square $(0,0), (1,0), (1,1), (0,1)$, under the line $\delta + R = 1 + \frac{1}{n}$.

Let $(C_i)_{i>0}$ be a family of $[n_i, k_i, d_i]$-codes. We say that it is a **good family** if

$$lim n_i = +\infty,$$

$$lim \frac{k_i}{n_i} > 0,$$

$$lim \frac{d_i}{n_i} > 0.$$

Let us denote by U_q the set of $(a, b) \in [0,1]^2$ such that it exists a sequence $(C_i)_i$ of $[n_i, k_i, d_i]$-codes with

$$lim n_i = +\infty,$$

$$lim \frac{k_i}{n_i} = a,$$

$$lim \frac{d_i}{n_i} = b.$$

3.2 Bounds

Theorem 3.2.1 *(Manin) It exists a continuous function α_q from $[0,1]$ into $[0,1]$ such that*

$$U_q = \{(x,y) \in [0,1]^2 \mid y \leq \alpha_q(x)\}.$$

Moreover

$$\alpha_q(x) \leq sup(1 - \frac{1}{q}x, 0).$$

The Manin function is not explicitely known and the regularity of the function is not known (just the continuity).

The **Varshamov-Gilbert** bound is a lower bound for this function.

Theorem 3.2.2 *For all $x \in [0, \frac{q-1}{q}]$, we have*

$$\alpha_q \geq \beta_q$$

where

$$\beta_q(x) = 1 - xlog_q(q-1) + xlog_q(x) + (1-x)log_q(1-x) \quad if \quad x \neq 0,$$

$$\beta_q(0) = 1.$$

Remark: $\beta_q(\frac{q-1}{q}) = 0$.

Reed-Muller codes are not good families of codes because $lim\frac{k_i}{n_i} = 0$. Reed-Solomon codes are not good families of codes because n_i is bounded by q and cannot grows to infinity.

So the problem is now to built good families of codes with a very good limit point that is a limit point which is between the Manin function and the Varshamov-Gilbert function. This problem has been solved by some families of algebraic geometric codes (Tsfasman-Vladut-Zink and also Ihara).