Chapter 1

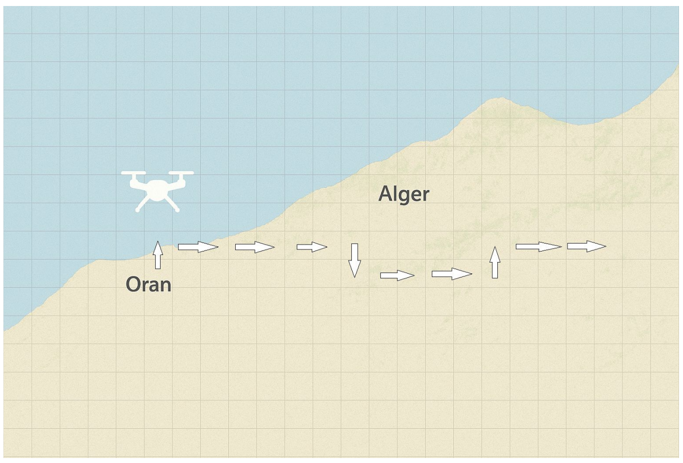
Introduction to Error-Correcting Codes and Motivation

1.1 An example : sending a path

Suppose that a drone and a control station have identical maps grid as shown in the following figure.

The control station can transmit data to drone of a safe route by which drone can avoid obstacles.

In this situation reliability is more important than speed of transmission.



If we encode the four directions North (N),South (S),E (East) ,West (W), the path to be sent is then

$$NEEESEENEE$$

Naive approach

We could encode the four directions North (N),South (S),E (East) ,West (W) using binary code, one code could be :

$$C_1 = \begin{cases} 00 = N \\ 01 = W \\ 10 = E \\ 11 = S \end{cases}$$

The path to be sent is then :

$$00101010111010001010$$

If we introduce an error

$$\underbrace{00}_{N}\underbrace{10}_{E}\underbrace{10}_{E}\underbrace{10}_{E}\underbrace{11}_{S}\underbrace{10}_{E}\underbrace{1\mathbf{1}}_{\mathbf{S}}\underbrace{00}_{N}\underbrace{10}_{E}\underbrace{10}_{E}$$

The receiver in unable to check for errors

Sending a path (Adding redundancy)

We could add redundancy in order to protect these message vectors against noise , consider the length 3 code C_2 as follows :

$$C_2 = \begin{cases} 000 = N \\ 011 = W \\ 101 = E \\ 110 = S \end{cases}$$

The path to be send is NEEESEENEE

$$\underbrace{000}\,\underbrace{101}\,\underbrace{101}\,\underbrace{101}\,\underbrace{110}\,\underbrace{101}\,\underbrace{1\mathbf{1}1}\,\underbrace{000}\,\underbrace{101}\,\underbrace{101}$$

If we introduce one error than we notice that 111 is not a valid code.

Notice that if we introduce one error in any position then it will be detected.

N	One error	Remark	W	One error	Remark
000	100	Not a code	011	111	Not a code
000	010	Not a code	011	001	Not a code
000	001	Not a code	011	010	Not a code

E	One error	Remark	E	One error	Remark
101	001	Not a code	110	010	Not a code
101	111	Not a code	110	100	Not a code
101	101	Not a code	110	111	Not a code

Notice that if we can detect one error we are unable to correct it for example 111 could be 101 or 011.

Error correction

In this example we consider the following code : $2^5 = 32 \;\; 4^3 = 64$

$$C_3 = \begin{cases} 00000 = N \\ 01101 = W \\ 10110 = E \\ 11011 = S \end{cases}$$

If a single error occurs in any codeword of C_3 we are able not only to detect it but actually to correct it, since the received vector will still be closer to the transmitted one than to any other.

$$\underbrace{00000}\underbrace{10110}\underbrace{10110}\underbrace{10110}\underbrace{11011}\underbrace{10110}\underbrace{10110}\underbrace{00000}\underbrace{10110}\underbrace{10110}$$

We will check this for 10110, others are similar

E	One error	Remark	Closest vestor
10110	**0**0110	Not a code	10110
10110	1**1**110	Not a code	10110
10110	10**0**10	Not a code	10110
10110	101**0**0	Not a code	10110
10110	1011**1**	Not a code	10110

1.1.1 Basic definitions

Definition 1 .

1- Let $\mathcal{A}$ be a finite set called an alphabet , en element of $\mathcal{A}$ is referred to as a letter.

2- A finite word is a sequence of elements of $\mathcal{A}$. The length of a finite word $u = u_0...u_{n-1} \in \mathcal{A}^n$ is $|u| = n$.

3- The set of finite words is given by $\mathcal{A}^*= \cup_{n=0}^{\infty}\mathcal{A}^n$. The set $\mathcal{A}^0 = \{\lambda\}$, where λ is the empty word.

4- We denote by $\mathcal{A}^{\mathbb{N}}$ the set of infinite words over $\mathcal{A}$ and by $\mathcal{A}^{\mathbb{Z}}$ the set of bi-infinite sequences over $\mathcal{A}$.

5- For two integers i, j with $i < j$ we denote by $x\,(i, j)$ the word $x_i...x_j$.

6- The length of an infinite word is denoted by ∞.

For an element of $\mathcal{A}^{\mathbb{Z}}$ we may use a decimal notation to avoid confusion. The first element to the right of the decimal point denoting position 0.

For example $x = ...000.100..$ of $\{\mathbf{0}, \mathbf{1}\}^{\mathbb{Z}}$ has a one at position zero $(x_0 = 1)$.

Definition 2 .

1- Let $u = u_0...u_{n-1}$ and $v = v_0...v_{m-1}$ be two finite words, the concatenation of the words

u and v is denoted by $uv = u_0 ... u_{n-1} v_0 ... v_{m-1}$.

2- We say that the word u is a factor of v and denote $u \sqsubseteq v$ if there is two words x, y such that $xuy = v$.

3- If $x = \lambda$ the word u is said a prefix of v and we denote it by $u \sqsubseteq_p v$.

4- If $y = \lambda$ the word u is said a suffix of v and we denote it by $u \sqsubseteq_s v$.

5- If u is a non empty finite word we denote by $u^\infty \in A^{\mathbb{N}}$ the infinite concatenation of u.

The set of finite words $\mathcal{A}^*$ equipped with the concatenation operation is a monoid. i.e. has the associative property and λ is the neutral element.

Definition 3 *The Hamming distance between two words x and y of $\mathcal{A}^n$ is the number of coordinates in which they are different :*

$$\begin{aligned} d_H(x, y) &= \sum_{i=1}^{n} \delta(x_i, y_i) \\ \delta(x_i, y_i) &= 1 \textit{ if } x_i \neq y_i \textit{ and } 0 \textit{ otherwise} \end{aligned}$$

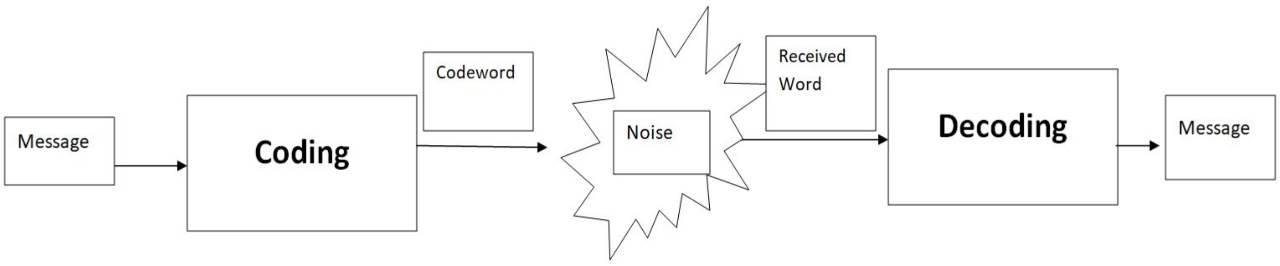
1.1.2 The coding problem

Let $\mathcal{A}$ be an alphabet, we want to send information written as words of constant length [1].

The transmission being subject to some form of alteration due to noise, the received word may be different from the original one.

Decoding is the process we apply to recover the original message.

[1] It is possible to use words of varying lenghts but for now we consider only the constant lenght cases.



Let $\mathcal{A}$ be an alphabet of q elements and suppose that we want to encode words of length k using words of length n.

A code C is a subset of $\mathcal{A}^n$ an element of C is often called a codeword or a vectorcode to distinguish it from any other element from $\mathcal{A}^n \backslash C$.

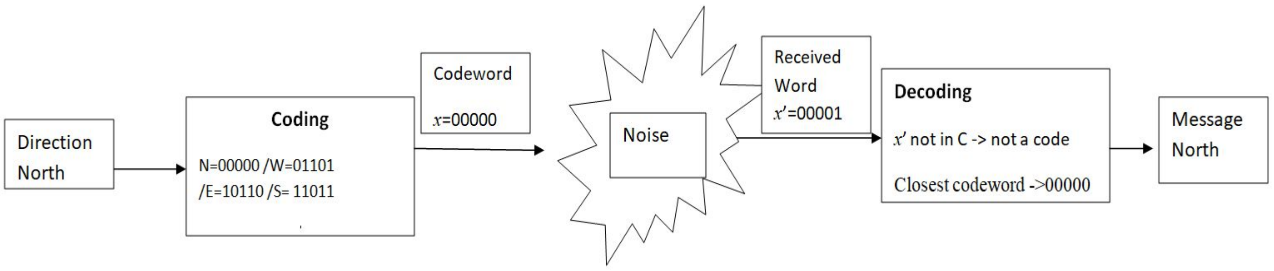
Let us go back to our example of sending a path.

The 4 elements to be coded are the four simple directions : N,E,S,W.

It is possible then to use any set with cardinality greater than 4. For example we may use $\{0,1\}^3$ or $\{0,1\}^5$.

We can then view the encoding problem as defining a map from $\{0,1\}^2$ to $\{0,1\}^n$ for $n > 2$.

Example 4 .



Notice from precedent examples that the closest codeword is the decoded word, this method is systematically used , a word which is not a codeword cannot be decoded if there are ambiguities.

Definition 5 *Let $C \subset \mathcal{A}^n$ be a code. We define the minimum distance over d_H by :*

$$d(C) = \min \{d_H(x, y) : x, y \in C, x \neq y\}$$

Proposition 6 .
1) A code C can detect up to s errors in any codeword if $d(C) \geq s+1$.
2) A code C can correct up to t errors in any codeword if $d(C) \geq 2t+1$.

Proof. 1) Suppose a codeword x is transmitted and that the number of errors is less than or equal to s.

Denote by x' the received word we have then $d_H(x, x') = s$ this means that $x' \notin C$.

2) Suppose $d(C) \geq 2t+1$ and suppose that a codeword x is transmitted and the vector x' received in which t or fewer errors have occurred so that $d(x, x') \leq t$.

If y is any codeword other than x then we should have $d(y, x') \geq t+1$. Otherwise $d(x, x') \leq d(x, y) + d(y, x') \leq 2t$ (contradiction)

So $x\prime$ is the nearest codeword to x and nearest neighbor decoding corrects the errors.
■

1.1.3 Hamming bound

The Hamming bound is a limit on the parameters of an arbitrary code. It gives an important limitation on the efficiency with which any error-correcting code can utilize the space in which its code words are embedded. A code that attains the Hamming bound is said to be a perfect code.

For any codeword $x \in \mathcal{C}$ the balls

$$B_t(x) = \{y \in \mathcal{A}^n : d_H(x, y) \leq t\}$$

We will count the number of elements of $B_t(x)$

Number of different letters	Number of possibilities
0	1
1	$(\|\mathcal{A}\| - 1) C_n^1$
2	$(\|\mathcal{A}\| - 1)^2 C_n^2$
...	...
t	$(\|\mathcal{A}\| - 1)^t C_n^t$

Hence

$$|B_t(x)| = 1 + (|\mathcal{A}| - 1) C_n^1 + \dots (|\mathcal{A}| - 1)^t C_n^t = \sum_{k=0}^{t} (|\mathcal{A}| - 1)^k C_n^k$$

We have then

$$|\mathcal{C}| \sum_{k=0}^{t} (|\mathcal{A}| - 1)^k C_n^k \leq |\mathcal{A}|^n$$

The last inequality is called the Hamming bound on a code.

1.1.4 Algorithmic efficiency and algorithmic complexity

There might be several algorithms or programs for the same problem, the concept of complexity allows us to compare several methods to evaluate the least expensive in computation time.The arithmetic cost of a method is the number of arithmetic operations necessary for its completion; it is also called arithmetic complexity. We use the Landau notation O to indicate the asymptotic behavior of complexity.

Let us study some basic examples.

1.1.5 Power evaluation

1. **Naive method.**

   We want to write a program to compute x^n for given values of x and n.

   A first approach is using successive multiplication :

$$x^n = \underbrace{x \times x \times ..... \times x}_{(n-1)\ times}$$

   The cost of this method is $(n-1)$ multiplications so the complexity is $\mathcal{O}(n)$.

2. **Dichotomic exponentiation.**

   The dichotomic exponentiation is summarized by the following formula :

$$x^n = \begin{cases} \left(x^{\frac{n}{2}}\right)^2 & if\ n \text{ is even} \\ x\left(x^{\frac{n-1}{2}}\right)^2 & if\ n \text{ is odd} \end{cases}$$

Example 7 *For example $x^{16} = (x^8)^2 = \left((x^4)^2\right)^2 = \left(\left(((x^2))^2\right)^2\right)^2$ and we need 4 multiplications in order to compute x^{16} instead of 15 operations for the naive method.*

The cost of the exponentiation method may be evaluated using the following method

.

Denote by $CostDicho\left(n\right)$ the cost to evaluate x^n.

If $n = 2^m$ is a pure power of 2 then we have :

$$\begin{aligned} CostDicho\left(n\right) &= CostDicho\left(\frac{n}{2}\right) + 1 \\ &= CostDicho\left(\frac{n}{4}\right) + 2 \\ &= .... \\ &= CostDicho\left(1\right) + m \end{aligned}$$

So the cost is $\mathcal{O}\left(\ln_2\left(n\right)\right)$

1.1.6 Polynomial evaluation : Horner's method

The Horner algorithm aims to avoid the evaluation of successive powers of x, the idea consists of a repeated evaluation of a polynomial of degree 1, thus a polynomial P of degree n will be evaluated according to the following technique:

$$\left| \begin{aligned} P\left(x\right) &= a_n x^n + a_{n-1}x^{n-1}.... + a_3x^3 + a_2x^2 + a_1x + a_0 \\ &= \left(a_nx^{n-1} + a_{n-1}x^{n-2}.... + a_3x^2 + a_2x + a_1\right)x + a_0 \\ &\qquad .... \\ &= \left(\left(\left(\underbrace{\left(\underbrace{a_nx + a_{n-1}}_{P1}\right)x + a_{n-2}....}_{P2} + a_3\right)x + a_2\right)x + a_1\right)x + a_0 \end{aligned} \right.$$

This may be denoted by the following formula :

$$\begin{cases} P_0 = a_n \\ P_i = P_{i-1}x + a_{n-i} \qquad 1 \leq i \leq n \end{cases}$$

The cost of this method can be evaluated as follows :

$$\left| \begin{array}{ll} P_1 = P_0 x + a_{n-1} & 1\ addition + 1\ multiplication \\ P_2 = P_1 x + a_{n-2} & 1\ addition + 1\ multiplication \\ P_3 = P_2 x + a_{n-3} & 1\ addition + 1\ multiplication \\ \ldots\ldots\ldots\ldots\ldots\ldots & 1\ addition + 1\ multiplication \\ P_n = P_{n-1} x + a_0 & 1\ addition + 1\ multiplication \end{array} \right.$$

The cost of the Horner's method is then n multiplications + n additions or $2n$ operations. The complexity is then $\mathcal{O}(n)$.

Chapter 2

Linear codes

2.1 Introduction

In this chapter we suppose that the alphabet $\mathcal{A}$ has a field structure, this allows us to treat $\mathcal{A}^n$ as a linear space.

A linear code is simply any subspace of $\mathcal{A}^n$. As a consequence of the linearity the word $0_{\mathcal{A}}^n$ is always a codeword.

Definition 8 *Let $x = x_0...x_{n-1} \in \mathcal{A}^n$ the weight of the word x denoted by $w(x)$ is the number of letters of x different from $0_{\mathcal{A}}$.*

$$w(u) = |\{0 \leq i \leq n-1 : x_i \neq 0_{\mathcal{A}}\}|$$

From the definition we have $w(x) = 0$ if and only if x is 0_n, the weight function check the triangular inequality $w(x+y) \leq w(x) + w(y)$, however the weight function is not a norm.

There is an interesting relation between the minimum distance of a code and the concept of weight.

$$d_H(x, y) = d_H(x-y, y-y) = d_H(x-y, 0_{\mathcal{A}}) = w(x-y)$$

Definition 9 *Let us denote by $w(C)$ the smallest of the weights of the non-zero codewords of C*

$$w(C) = \min \{ w(x), x \in C \backslash \{0^n_{\mathcal{A}}\} \}$$

Proposition 10 *Let C be a linear code and let $w(C)$ be the smallest of the weights of the non-zero codewords of C. Then $d(C) = w(C)$.*

2.1.1 Generator matrix

As a linear code C is a linear space we can found a generator matrix.

Definition 11 *A generator matrix G is a matrix of dimension $(n \times k)$ such that :*

$$C = \left\{ G.x : x \in \mathcal{A}^k \right\}$$

Example 12 *We want to encode elements from $\{0,1\}^3$ using the generator matrix G :*

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

Each codeword has the form

$$G.x = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 + x_3 \\ x_1 + x_3 \\ x_1 \\ x_3 \\ x_1 + x_2 \end{pmatrix}$$

Hence the code C is given by :

$$C = \left\{ (x_1 + x_2 + x_3, x_1 + x_3, x_1, x_3, x_1 + x_2), (x_1, x_2, x_3) \in \{0,1\}^3 \right\}$$

Remark 13 *Some references uses the notation $x.G$ to denote Generator, in this case the generator matrix is imply the transpose of the one from the notation $G.x$ and the x is supposed a vector of dimension $(1 \times k)$*

Example 14 *Consider the following linear code generated by G'*

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

Each codeword has the form

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 + x_3 & x_1 + x_3 & x_1 & x_3 & x_1 + x_2 \end{pmatrix}$$

Hence the code C is given by :

$$C = \left\{ (x_1 + x_2 + x_3, x_1 + x_3, x_1, x_3, x_1 + x_2), (x_1, x_2, x_3) \in \{0,1\}^3 \right\}$$

2.1.2 Equivalence of linear codes

Two $(n \times k)$ matrices generate equivalent linear codes if one matrix can be obtained from the other by elementary operations

1. Permutation of the rows.

2. Multiplication of a row by a non-zero scalar.

3. Addition of a. scalar multiple of one row to another.

4. Permutation of the columns.

5. Multiplication of any column by a non-zero scalar.

2.1.3 Gaussian elimination over finite fields

The Gauss method still works over finite fields.

Example 15 *Solve the following linear system in Z_5 using Gaussian elimination :*

$$\begin{cases} x + 2y + 2z = 3 \\ 2x + z = 4 \\ 3x + y + 3z = 1 \end{cases} \Leftrightarrow \begin{bmatrix} 1 & 2 & 2 & 3 \\ 2 & 0 & 1 & 4 \\ 3 & 1 & 3 & 1 \end{bmatrix}$$

$$\begin{cases} l_2 \leftarrow l_2 + 3l_1 \\ l_3 \leftarrow l_3 + 2l_1 \end{cases} \rightarrow \begin{bmatrix} 1 & 2 & 2 & 3 \\ 2+3 & 6 & 1+6 & 4+9 \\ 3+2 & 1+4 & 3+4 & 1+6 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & 2 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$

$$\begin{cases} x + 2y + 2z = 3 \\ y + 2z = 3 \\ 2z = 2 \end{cases} \rightarrow \begin{cases} x + 2y + 2z = 3 \\ y = 3 - 2z = 1 \\ z = 1 \end{cases}$$

$$\rightarrow \begin{cases} x = 3 - 2y - 2z = 3 - 2 - 2 = 4 \\ y = 3 - 2z = 1 \\ z = 1 \end{cases}$$

Remark 16 *There is no need to minimize roundoff errors when you work over finite fields.*

2.2 Standard form

Any generator matrix G can be transformed to the standard form $\begin{pmatrix} I_k & A \end{pmatrix}$ $\left(\text{resp. } \begin{matrix} I_k \\ A \end{matrix}\right)$

where I_k is the $k \times k$ identity matrix, and A is a $k \times (n-k)$ matrix. (resp. A a $(n-k) \times k$ version)

Example 17 $\begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$

$$\left\{\begin{array}{l} l_2 \leftarrow l_2 + l_1 \\ l_3 \leftarrow l_3 + l_1 \end{array}\right. \quad \rightarrow \quad \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\left\{\begin{array}{l} l_1 \leftarrow l_1 + l_3 \\ l_2 \leftarrow l_2 + l_1 \end{array}\right. \quad \rightarrow \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$\{l_1 \leftarrow l_1 + l_2 \quad \rightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

2.2.1 Dual code and parity check matrix

Definition 18 *A matrix H is said to be a parity check matrix if*

$$Hx = 0_n, \forall x \in C$$

Proposition 19 *Suppose that $C \subset \mathcal{A}^n$ is a linear code with a generator matrix is stand-*

ard form $(I_k|\, G)$ *then the matrix* $(-G^t|\, I_r)$ *is a parity check matrix for* C *where* $r = n-k$.

Proof. Exercise. ■

Remark 20 *The parity check matrix is a generator matrix of the dual code* $C^\perp$.

Definition 21 *Suppose that* $C \subset \mathcal{A}^n$ *is a code of length* n, *define the dual code* $C^\perp$ *by :*

$$C^\perp = \{x \in \mathcal{A}^n : x.y = 0_{\mathcal{A}}, \forall y \in C\}$$

Definition 22 *A linear code is self dual if* $C^\perp = C$.

Example 23 *1) Suppose* $C = \{0000, 1100, 0011, 1111\}$ *then* $C^\perp = C$.

2) Suppose $C = \{000, 110, 011, 101\}$ *then* $C^\perp = \{000, 111\}$.

Proposition 24 *For any* (n,k) *code* C *we have* $\left(C^\perp\right)^\perp = C$.

Proof. Clearly $C \subset \left(C^\perp\right)^\perp$ since every vector in C is orthogonal to every vector in $C^\perp$.
We have also $\dim \left(C^\perp\right)^\perp = n - (n-k) = k = \dim(C)$ and so $\left(C^\perp\right)^\perp = C$. ■

2.3 Decoding of linear codes

2.3.1 Cosets and Lagrange's theorem

Definition 25 *Let* H *be a subgroup of a group* G *and* g *be any element of* G. *The left coset* gH *is defined by :*

$$g + H = \{g + h : h \in H\}$$

Remark 26 *The coset h+H is simply the subgroup H.*

Proposition 27 *Let H be a subrgoup of G then the relation R on G defined by*

$$x\mathcal{R}y \Leftrightarrow y - x \in H$$

is an equivalence relation.

Proof. Exercise. ■

Proposition 28 *Let H be a subgroup of G and R be the equivalence relation $\mathcal{R}$ defined in the previous proposition. Then the equivalence class of an element $g \in G$ is the left coset $g + H$.*

Proof. Exercise. ■

Corollary 29 *Let H be a subgroup of G. Then two left cosets $x + H$ and $y + H$ of H in G are either equal or disjoint and each element of G is in some left coset of H.*

Example 30 *Let G be the additive group of integers $\mathbb{Z}$ and let n be any positive integer.*

Proposition 31 *Let H be a subgroup of a group G. For any element $g \in G$, there is a bijection between H and $g + H$.*

Proof. Define the map $f : H \to g + H$ by $f(h) = g + h$ using the properties of a group structures this map is bijective. ■

Theorem 32 (Lagrange) *Let G be a group with a finite number of elements and let H be a subgroup of G. Then the number r of distinct left cosets of H is equal to $|G| / |H|$. In particular both $|H|$ and r divide $|G|$.*

Proof. The equivalence classes under the relation R partition G. Each equivalence class is a left coset and each left coset has the same number of elements as H.

It follows that if r is the number of distinct left cosets then $|G| = r|H|$ ■

Definition 33 *The number of distinct left cosets of a subgroup H in a group G is the index of H in G. It is usually denoted by $|G : H|$.*

2.3.2 Cosets and standard array

Definition 34 *Suppose that $C \subset \mathcal{A}^n$ is a code of length n and consider $a \in \mathcal{A}^n$. Define the set $a + C$ by*

$$a + C = \{a + x : x \in C\}$$

the set $a + C$ is called a coset of C.

Proposition 35 *Properties*

1- If $a + C$ is a coset of C and $b \in a + C$ then $b + C = a + C$.

2- Every element of$\mathcal{A}^n$ is in some coset of C.

3- Every coset contains exactly $|\mathcal{C}|$ elements.

4-Two cosets either are disjoint or coincide.

Definition 36 *The vector having minimum weight in a coset is called the coset leader.*

A standard array for a code C is an array of all the vectors in $\mathcal{A}^n$ in which the first row consists of the code C on the extreme left and the other rows are the cosets $a + C$ each arranged in corresponding order, with the coset leader on the left.

A standard array may be constructed as follows:

1. Step 1 : List the codewords of C, starting with 0, as the first row.

2. Step 2 Choose any vector a_1 not in the first row of minimum weight. List the coset $a_1 + C$ as the second row starting from a_1.

3. Step k From those vectors not in rows 1 until $k - 1$, choose a_k of minimum weight and list the coset $a_k + C$ as in Step 2 to get the third row.

Example 37 *Consider the code generated by the matrix G*

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

The code C is hence given by $G \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1 + x_2 \\ x_2 \\ x_1 \end{pmatrix}$

$$C = \left\{ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix}^t, \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \end{pmatrix}^t, \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix}^t, \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \end{pmatrix}^t \right\}$$

$$\begin{array}{ccccc}
 & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \end{pmatrix} & \left(1 \;\; 1 \;\; 0 \;\; 1 \right. \\
+\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \end{pmatrix} & \left(1 \;\; 1 \;\; 0 \;\; 1 \right. \\
+\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix} & \left(1 \;\; 1 \;\; 0 \;\; 0 \right. \\
+\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \end{pmatrix} & \left(1 \;\; 1 \;\; 1 \;\; 1 \right. \\
+\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \end{pmatrix} & \left(1 \;\; 0 \;\; 0 \;\; 1 \right. \\
+\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \end{pmatrix} & \left(0 \;\; 1 \;\; 0 \;\; 1 \right. \\
+\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \end{pmatrix} & \left(0 \;\; 0 \;\; 0 \;\; 1 \right. \\
+\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \end{pmatrix} & \left(0 \;\; 1 \;\; 0 \;\; 0 \right.
\end{array}$$

Proposition 38 *Let $\mathcal{C}$ be a linear code with distance d, if x is a vector such that*

$$w(x) \leq \left[\frac{d-1}{2} \right]$$

then x is a unique element of minimum weight in its coset of $\mathcal{C}$ and hence is always a coset leader in a standard array for $\mathcal{C}$.

Proof. Suppose $w(x) \leq \left[\frac{d-1}{2}\right]$ and x is not a unique vector of minimum weight in its coset C_i.

Then there exists some vector $y \neq x \in C_i$ such that $w(x) \leq w(y)$.

Since x and y are in the same coset C_i such that $w(y) \leq w(x)$

$$w(x-y) \leq w(x) + w(y) \leq \left[\frac{d-1}{2}\right] + \left[\frac{d-1}{2}\right] \leq d-1$$

This contradicts the distance d of the code. ■

2.3.3 Syndromes

Definition 39 *Let H be a parity-check matrix for an (n,k) code over F, for $x \in \mathcal{A}^n$ the syndrome s of x is defined by $s = Hx$ (or $x.H$)*

Proposition 40 *Let H be a parity-check matrix for a linear code C. Then two words x and y are in the same coset of C if and only if they have the same syndrome $(i.e\ Hx = Hy)$*

Proof. If x and y are in the same coset of C, then $x = l + c_i$ and $y = l + c_j$ for some codewords c_i and c_j and some coset leader l.

$$Hx = H(l + c_i) = Hl + Hc_i = Hl = Hl + Hc_j = Hy$$

Conversely, suppose that $Hx = Hy$ then $Hx - Hy = 0_n$ hence $x - y$ is a codeword.

We have then $x - y = c_i$ for some $1 \leq i \leq |C|$ hence $x = y + c_i$ and x and y are in the same coset. ■

Example 41 *Suppose we have a linear code with a parity check matrix*

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

The syndromes are given by the following table :

Coset leader	*Syndrome*
000000	*000*
000001	*100*
000010	*010*
000100	*001*
001000	*110*
010000	*011*
100000	*101*
001100	*111*

Proposition 42 *For every code with parity check matrix H, the minimum distance d equals the size of the smallest subset of columns of H that are linearly dependent.*

Proof. We need to show that the minimum weight of a non zero codeword in C is the minimum number of linearly dependent columns.

Let t be the minimum number of linearly dependent columns in H.

Let $c \neq 0_{A^k} \in C$ be a codeword with $w(c) = d$.
By definition of the parity check matrix $Hc = 0_{A^k}$ and by matrix multiplication this gives us that $\sum_{i=1}^{n} c_i H^i = 0_{A^k}$ where H^i is the i^{th} column of the matrix H.

So for Hc to be the zero vector we need all H_i with non zero coefficients are linearly dependent.

This means that $d \geq t$ as the columns corresponding to non-zero entries in c are one instance of linearly dependent columns.

For the other direction, consider the minimum set of columns from H that are linearly dependent.

This implies that there exists non zero elements $c'_{i_1}, ..., c'_{i_t} \in \mathcal{A}$ such that

$$c'_{i_1} H^{i_1} + ... + c'_{i_t} H^{i_t} = 0$$

Now extend $c'_{i_1}, ..., c'_{i_t}$ to the vector c' such that $c'_j = 0$ for $j \notin \{i_1, ...i_t\}$.
We have then $Hc' = 0$ and thus c' is a codeword with a weight t thus $d \leq t$. ■

2.4 Some special codes

2.4.1 The $(7, 4, 3)$ Hamming code

The binary Hamming code is generated by the following generating matrix :

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

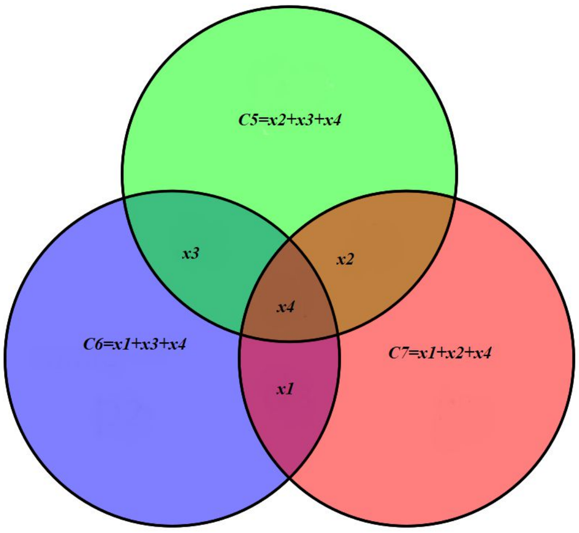
hence each word (x_1, x_2, x_3, x_4) is associated to the codeword

$$(x_1, x_2, x_3, x_4, x_2 + x_3 + x_4, x_1 + x_3 + x_4, x_1 + x_2 + x_4)$$

If we denote by $c_1, ..., c_7$ the successive letters of the alphabet of the coded word, we have the following property :

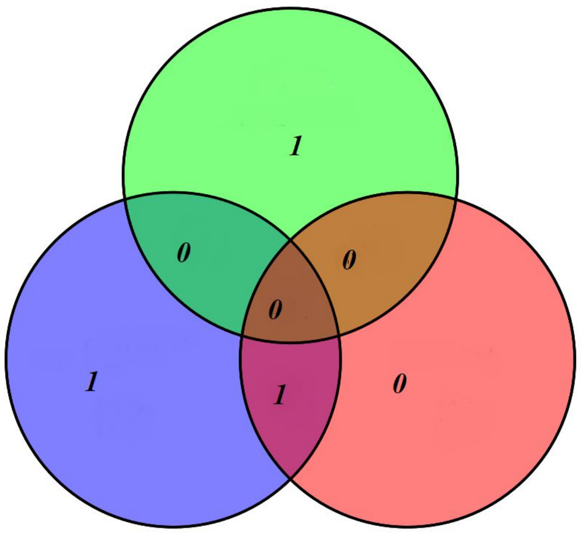
$$\begin{cases} c_2 + c_3 + c_4 + c_5 = 0 \\ c_1 + c_3 + c_4 + c_6 = 0 \\ c_1 + c_2 + c_4 + c_7 = 0 \end{cases}$$

This code has the following nice geometric interpretation :



The sum of elements of each circle should be 0, let us explain how to use this scheme with an example.

Example 43 *Suppose we received the message 1000110*



Here we have a problem with the green circle because the sum of the four elements inside is different from 0.
We have also a problem with the salmon circle however the blue circle is correct.
If we suppose that only one error occurred then it should be at the intersection of the two circle hence we have to correct x_2.

2.4.2 The general binary Hamming code

Let r be a positive integer, define the parity check matrix H_r as the matrix where each column H_r^i is the binary representation of i, for $1 \leq i \leq 2^r - 1$.

Example 44 *For example, for the case $r = 3$ we have $2^r - 1 = 7$*

	$1\|_2$	$2\|_2$	$3\|_2$	$4\|_2$	$5\|_2$	$6\|_2$	$7\|_2$
$Coff\ 2^2$	0	0	0	1	1	1	1
$Coff\ 2^1$	0	1	1	0	0	1	1
$Coff\ 2^0$	1	0	1	0	1	0	1

Hence the code is given by :

$$\begin{cases} c_2 + c_3 + c_4 + c_5 = 0 \\ c_1 + c_3 + c_4 + c_6 = 0 \\ c_1 + c_2 + c_4 + c_7 = 0 \end{cases}$$

Proposition 45 *The general binary Hamming code* $(2^r - 1, 2^r - r - 1, 3)$ *has distance 3.*

Proof. No two columns in H are linearly dependent.
If they were we would have $H^i + H^j = 0$ but this is impossible since they differ in at least one bit.
(binary representations of integers $i \neq j$) Thus the distance is at least 3.
It is as at most 3 since $H^1 + H^2 + H^3 = 0$. ■

2.4.3 Decoding single error linear codes

Let H be a parity check matrix for a linear code C. Suppose that our channel has a high probability to introduce only one error.

So a single error correcting code is supposed to be enough for this kind of channels.

Let us denote by r the received word, c the codeword and e the error introduced.

We have then $r = c + e$, using the parity check matrix we obtain :

$$Hr = H(c+e) = \underbrace{Hc}_{0_{F^n}} + He = He$$

As our channel is supposed to introduce only one error hence the weight of e is equal to one. Then there exist an index i and a scalar $\alpha \in F$ such that $He = \alpha h_i$ where h_i is the $t-th$ column of H.

Using this remark we can give an algorithm for decoding single error codes including Hamming code.

Algorithm 46 *Let H be the partity-check matrix and let r be the received word.*

1) Compute Hr

2) If $Hr = 0_{F^n}$ then r is the transmitted codeword.

3) If $Hr = s \neq 0_{F^n}$ then compare s with the columns of H.

4) If there is some i such that $s = \alpha h_i$ then e is the n-tuple with α in position i and $0's$ elsewhere, correct r to $c = r - e$.

5) Otherwise more than one error has occurred.

Example 47 *Suppose that we want to encode the information* (1010)
Hence the encoded word is given by (1010101), *suppose now that the received word with one error is* $r = (1\mathbf{1}10101)$
Let us compute Hr.

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

Hence Hr corresponds to the column 2 of the matrix H.
So the error was made at position 2 of the received word and it is enough to change the value at position 2

$$r = (1\mathbf{1}10101) \to c = (1\mathbf{0}10101)$$

2.4.4 Reed Solomon code

Background (Polynomial interpolation)

Let $\left\{(x_i, y_i)_{0\leq i\leq n}\right\}$ a set of $n+1$ pairs the main interpolation problem is to find a polynomial P such that

$$P(x_i) = y_i : 0 \leq i \leq n$$

Proposition 48 *For $(n+1)$ distinct points (x_i, y_i) there is a unique polynomial of degree n satisfying $P(x_i) = y_i : 0 \leq i \leq n$.*

Remark 49 *By distinct points we mean that $x_i, 0 \leq i \leq n$. are distinct.*

Lagrange polynomials

For a set $\{(x_i, y_i)\}_{0\leq i\leq n}$ of $n+1$ points, we construct the Lagrange polynomials

$$L_k\left(x\right) = \prod_{\substack{j=0..n \\ j\neq k}} \frac{x - x_j}{x_k - x_j} : k = 0.....n$$

Lagrange polynomials have the following property :

$$\begin{cases} L_k\left(x_k\right) = 1 \quad \forall k \\ L_k\left(x_j\right) = 0 \quad \forall k \neq j \end{cases}$$

Proposition 50 *Let $\{(x_i, y_i)\}_{0\leq i\leq n}$ be a set of of $n+1$ points. Then the unique interpolation polynomial is given by :*

$$P\left(x\right) = \sum_{k=0}^{n} y_k L_k\left(x\right)$$

Interpolation problem over finite fields

Proposition 51 *A polynomial of degree n with coefficients in a field F has at most n roots.*

Proposition 52 *Every function defined from a finite field to itself is a polynomial*

Proof. Suppose f is a function defined from a finite field to itself, we have then

$$f(x_i) = y_i : 1 \leq i \leq p^n$$

Considering this as an interpolation problem there is a unique polynomial of degree $m \leq p^n - 1$ that interpolates this data ■

2.4.5 Reed Solomon code

Reed–Solomon codes are a group of error-correcting codes that were introduced in 1960. They are used in many applications, such as DVDs and QR codes.

The Reed Solomon code is described via its encoder mapping.

Fix integers $k \leq n \leq q$ and n distinct elements $x_1, x_2, ...x_n \in GF(q)$.

Suppose we want to encode the message $a_1...a_k$ compute $P(x)$ the interpolation polynomial of $(x_i, a_i)_{1 \leq i \leq k}$.

The encoded message is then

$$(P(x_1), ...P(x_k), P(x_{k+1}), ...P(x_n))$$

Hence the number of redundant bits is $n - k$.

Example 53 *We want to send the following message :* $[3, 2, 4]$ *using the alphabet* $\mathbb{Z}/5\mathbb{Z}$ *We start by computing the interpolation polynomial of the points* $(0, 3), (1, 2), (3, 4)$.

We can use Lagrange method :

$$\begin{aligned}
L_0(x) &= \frac{(x-1)(x-2)}{(0-1)(0-2)} = 3x^2+x+1 \\
L_1(x) &= \frac{(x)(x-2)}{(1)(1-2)} = 4x^2+2x \\
L_2(x) &= \frac{(x)(x-1)}{(2)(2-1)} = 3x^2+2x
\end{aligned}$$

$$P(x) = 3\left(3x^2+x+1\right)+2\left(4x^2+2x\right)+4\left(3x^2+2x\right) = 29x^2+15x+3 = 4x^2+3$$

Now we replace the values 3 and 4 in $P(x)$ *and we have* $P(3)=4$ *and* $P(4)=2$.
The encoded message is then $[3,2,4,4,2]$

2.4.6 Reed Solomon code used for erasures correction

The Reed Solomon is particularly suited to recover data lost to erasures

Example 54 *Suppose we received the following message* $[\blacksquare,2,\blacksquare,4,2]$
Our received message has two erasures.
We start by computing the interpolation polynomial of the points $,(1,2),(3,4),(4,2)$.
We can use Lagrange method :

$$\begin{aligned}
L_0(x) &= \frac{(x-3)(x-4)}{(1-3)(1-4)} = x^2+3x+2 \\
L_1(x) &= \frac{(x-1)(x-4)}{(3-1)(3-4)} = 2x^2+3 \\
L_2(x) &= \frac{(x-1)(x-3)}{(4-1)(4-3)} = 2x^2+2x+1
\end{aligned}$$

$$\begin{aligned}
P(x) &= 2\left(x^2+3x+2\right)+4\left(2x^2+3\right)+2\left(2x^2+2x+1\right) \\
&= 4x^2+3
\end{aligned}$$

Now we replace the values 0 and 2 in $P(x)$ *and we have* $P(0)=3$ *and* $P(2)=4$.

The decoded message is then $[3,2,4]$.

Proposition 55 *The Reed Solomon code is linear.*

Proof. The proof is straightforward using definition and construction of Reed Solomon code. ■

2.4.7 Minimum distance of Reed Solomon code

Proposition 56 *The minimum distance of a Reed Solomon code is* $d = n - k + 1$.

Proof. Let $(P_k(x_1), P_k(x_2), ..., P_k(x_n))$ be a codeword.

As P is of degree $k-1$ it has at most $k-1$ roots hence :

$$w(P_k(x_1), P_k(x_2), ..., P_k(x_n)) \geq n - k + 1$$

So $d \geq n - k + 1$ ■

Corollary 57 *The Reed Solomon code match the Singleton bound.*

2.4.8 Decoding algorithm for RS code (original version)

The original decoding algorithm used the majority rule.

Suppose we received a code of lenght n containing e errors and we want to decode it.

Pick up all possible k uples of received words and then compute the associated polynomial interpolation, compute again the values corresponding to that polynomial.

We will end up with C_n^k possible values and consider the major one as the most probable one.

This method need at to run C_n^k polynomial interpolation and hence is not algorithmically efficient.

2.4.9 Decoding algorithm for RS code (Berlekamp Welch Algorithm)

Definition 58 (Polynomial error locator) *Consider the n pairs* $\{(x_i, y_i)\}_{1\leq i\leq n}$ *where* $(y_i)_{1\leq i\leq n}$ *is the received message of an RS code and* $P(x)$ *the generator polynomial. Define* $E(x) = \prod\limits_{P(x_i)\neq y_i} (x - x_i)$ *the polynomial having as roots only the points* x_i *corresponding to the transmission errors .*$E(x)$ *is called the Polynomial error locator.*

Consider now the polynomial $Q(x) = E(x)P(x)$ this polynomial has the following property

$$Q(x_i) = y_i E(x_i) : 1 \leq i \leq n$$

Notice that if x_i corresponds to an error transmission then we have $Q(x_i) = 0 = y_i E(x_i) = 0$ and if x_i corresponds to a correct transmission then by definition we have $Q(x_i) = y_i E(x_i)$.

The main idea of the Berlekamp Welch algorithm is to try to find a polynomial Q and a polynomial E such that

$$Q(x_i) = y_i E(x_i) : 1 \leq i \leq n$$

Notice that this algorithm may fail.

Algorithm 59 (Berlekamp Welch) .
Input : $n \geq k \geq 1$, $0 < e < \frac{n-k+1}{2}$ *and n pairs* $\{(x_i, y_i)\}_{1\leq i\leq n}$
Output : Polynomial $P(x)$ *of degree at most* $k-1$ *or* ***fail***
Step 1 : Compute a non zero polynomial $E(x)$ *of degree* e *and a polynomial* $Q(x)$ *of degree at most* $e+k-1$ *such that*

$$y_i E(x_i) = Q(x_i) : 1 \leq i \leq n$$

If such polynomial do not exist output fail.

*Step 2 : If $E(x)$ does not divide $Q(x)$ then output **fail** else compute $P(x) = \frac{Q(x)}{E(x)}$.*
*If $d_H((y_1, ..., y_n), (P(x_1), ...P(x_n))) > e$ then output **fail** else output $P(x)$.*

Example 60 *Suppose we are using a Reed Solomon code with $k = 2$ and $n = 5$ hence our code is of distance 3 and can correct one error.*
Suppose we received the following message

$$\begin{array}{cccccc} x & 1 & 2 & 3 & 4 & 5 \\ P(x) & 2 & 4 & 3 & 7 & 6 \end{array}$$

Define the error locator polynomial $E(x) = 1 + ex$ and the polynomial $Q(x) = a_0 + a_1x + a_2x^2$

$$\begin{cases} 2(1+e) = a_0 + a_1 + a_2 \\ 4(1+2e) = a_0 + a_1 2 + a_2 2^2 \\ 3(1+3e) = a_0 + a_1 3 + a_2 3^2 \\ 7(1+4e) = a_0 + a_1 4 + a_2 4^2 \\ 6(1+5e) = a_0 + a_1 5 + a_2 5^2 \end{cases}$$

$$\begin{cases} 2 + 2e = a_0 + a_1 + a_2 \\ 4 + e = a_0 + 2a_1 + 4a_2 \\ 3 + 2e = a_0 + 3a_1 + 2a_2 \\ 0 = a_0 + 4a_1 + 2a_2 \\ 6 + 5e = a_0 + 5a_1 + 4a_2 \end{cases}$$