Report of Project No.2: (63,42) Reed-Solomon Code

January 24, 2021

1 Outline

- Introduction.
- Implement detail.
- Appendix C++ Code.

2 Introduction

The purpose of this report is to summarize the work of Project 2, which aims to produce a program to implement the (63,42) Reed-Solomon decoding algorithm. The Reed-Solomon code is similar to BCH code, but each element of codeword is in the $GF(2^m)$. Moreover, in this project, the elements are in $GF(2^6)$ which can represented by $[a_5a_4a_3a_2a_0]$ with $a_5\alpha^5 + a_4\alpha^4 + a_3\alpha^3 + a_2\alpha^2 + a_1\alpha^1 + a_0$, where the α is a primitive element in $GF(2^6)$ and satisfying $\alpha^6 + \alpha + 1 = 0$. For convenience, we use range 0 to 63 integer to represent each 6 bits and modulo $\alpha^6 + \alpha + 1$ also equal to modulo 67.

The encode process is not required in this project, but I also implement it for directly check the result of decoding. After a encode process, a 42 length information sequence encode into a 63 length codeword. For prove the correcting errors ability, I modify some element of the codeword into other element, and modify some element into erasure, which be denoted by "*". Then, the modified codeword be the "receive sequence " as the input of decoding process. The main of decoding process is an Euclid's algorithm with a specify stop condition. The result of this Euclid's algorithm is the solution of key equation, and then I use $Time - Domain\ Completion$ to find the error vector and find the origin codeword. Note that, the code can only decode t_0 erasures and t_1 errors, which satisfy $t_0 + 2t_1 \leq 21$. The system structure as Figure. 2.

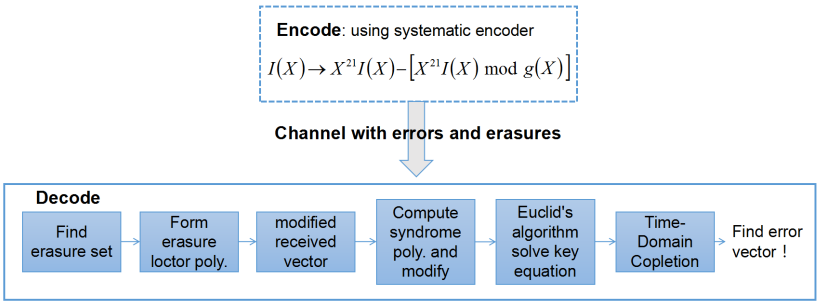

Figure 1: System model.

2.1 Implement detail

As Figure.2.1, the implement can divided into four main components: operations in $GF(2^6)$, operations of polynomial, encode, decode (Euclid's algorithm, Time-Domain complement)

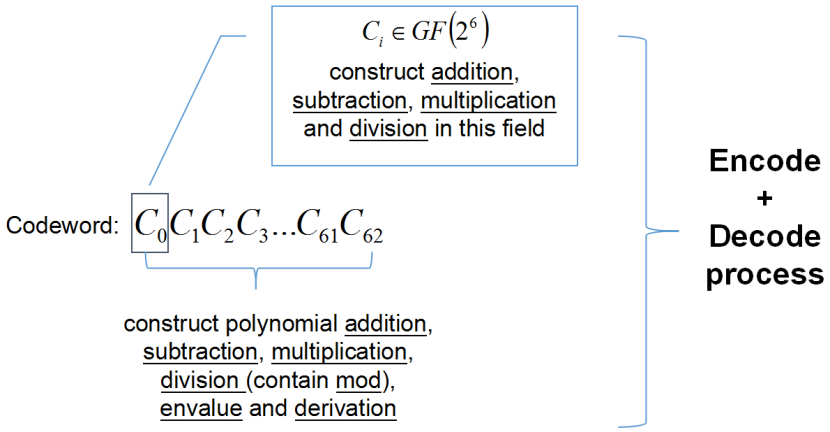

Figure 2: Implement structure.

2.2 Operations in $GF(2^6)$

- Addition: the addition is equivalent to the exclusive XOR of 6 binary bits of two number.
- Subtraction: because the coefficient of α in the $GF(2)$, the subtraction same as addition, equivalent to the XOR.
- Multiplication: We can build a table of the value of α^i, $for\ i = 1...62$, the multiplication equivalent the order addition while modulo 63, i.e. $\alpha^i \times \alpha^j = \alpha^{i+j\ mod\ 63}$. The modulo 63 because $\alpha^{63} = 1$.
- Division: the division equivalent to times the multiplication inverse (also need modulo 63), i.e.$\alpha^i/\alpha^j = \alpha^i \times \alpha^{-j} = \alpha^{i-j\ mod\ 63}$.

2.3 Operations of Polynomial

The operations of polynomial with each coefficient in above filed contain: addition, subtraction, multiplication, division, modulo, envalue and derivation.

- Addition: the degree of the result of addition operation is same as the largest one of two polynomial degree, and change to every two corresponding coefficient addition in the field.
- Subtraction: Similar to addition because the addition and subtraction in the filed are same.
- Multiplication: in multiplication, refer to compute multiple by hand, I use 2 loops go from low order term to high order term then do multiplication and addition in the filed.

- Division and modulo: use long division, we can get quotient and remainder, and define

$$remainder = dividend - quotient \times divisor$$

. the long division starting at the highest order and in each iteration calculate coefficient and order of quotient, then we multiple the divisor by quotient and substrate.

- Envalue: use Horner's rule, assume $u(x) = u_n x^n + u_{n-1}x^{n-1} + ... + u_1 x + u_0$, the envalue can be solved by loop, i.e. $((((u_n x + u_{n-1})x + u_{n-1})x + u_{n-2})x... + u_1)x + u_0$.

- Derivation: the formal derivation of polynomial $u(x) = u_n x^n + u_{n-1}x^{n-1} + ... + u_1 x + u_0$ be defined as $u'(x) = \frac{u_n}{n}x^{n-1} + \frac{u_{n-1}}{n-1}x^{n-2} + ... + u_1$ where $\underbrace{n = 1 + 1... + 1}_{n}$.

2.3.1 Encode

I use the systemic encoder in this project, which following the equation:

$$I(x) \implies x^{21}I(x) - [x^{21}I(x) \ mod \ g(x)]$$

which can bring the information sequence to the high order in the codeword i.e. $C_0, ..., C_{20}$ are the parity-check characters and $C_{21}, ..., C_{62}$ are information characters. This process use the multiplication, subtraction and modulo of polynomial.

2.3.2 Decode

As Figure 2, we can think of decoding as the following parts:

- First remark the index and form the *erasure set* $I_0$and find the *erasure locator* $\sigma_0(x)$ which can be formed by polynomial multiplication. Then we can get the number of erasure t_0 and modified received vector R'.

- Compute the syndrome polynomial $S(x)$ by $S_J = \sum_{i=0}^{62} R'_i \alpha^{ij}$ for $j = 1, 2, ...21$, where use the operate in the field, moreover if $ij > 63$, use $ij \ mod \ 63$. Then we can get the modified syndrome polynomial $S_0(x) = \sigma_0(x)S(x) \ mod \ x^{21}$.

- The key equation to be solved becomes $\sigma_1(x)S_0 \equiv \omega(X) \ (mod \ x^{21}$. and we can compute $\mu = \lfloor 21 - t_0/2 \rfloor$ and $\nu = \lceil 21 + t_0/2 \rceil - 1$.

- implement Euclid's algorithm by recursive and the stop condition is

$$deg(r_j(x)) \leq \nu$$

$$deg((v_j(x)) \leq \mu$$

then set

$$\sigma_1(x) = v_j(x), \ \omega(x) = r_j(x)$$

- Time-Domain complement refer to the sudo code and note that if $t_0 + 2t_1 > 21$, the decoder can not decode the code.

Then, we can obtain an error vector E from the decoder and then we can recover the code by $C = R - E$ which only need subtraction of polynomial. At this point, the implementation of the entire system is complete