Analysis of a reconstruction method based on multisets – Final Report

Yixuan Zhang
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
120010058@link.cuhk.edu.cn

***Abstract*—Polymer-based data storage provides people with a landmark method in storing data with greater capacity, higher density, and longer life. Compared with current data storage techniques, like Redundant Array of Independent Disks (RAID), it shows outstanding storing functions. However, the accuracy of the binary data recovered from polymers is sometimes relatively too low for practical usage. Transforming polymer information into binary data utilizing techniques in coding theory, enables accurate data reconstruction and error correction. In this article, we discuss in detail a known method that is able to reconstruct polymer-based data from their mass composition even if there are multiple errors in the mass composition. The method relies on a polynomial formulation of the problem that makes it possible to use classic tools from algebraic coding theory such as Reed-Solomon codes and cyclic codes.**

I. Introduction

Current digital storage systems are facing numerous problems such as conducting computations based on memory [1]. Scientists have come up with a number of molecular storage paradigms [2]–[4]. The non-volatile nature of nucleic acids, their low energy of operation in living cells, and a volumetric storage density far exceeding electronic-memory projections allow molecular storage to avoid such a problem [1]. As a potential storing technique, polymer data storage has been tried to take the place of hardware data storage. However, there are still some obstacles of polymer data storage that prevent it from being in common usage. One critical obstacle is the decoding error during the process of decoding from polymer to binary strings. Among the earlier work, the problem of *binary string reconstruction from its substring composition multiset* is addressed in [5]. The method discussed here is error-correcting reconstruction in symmetric case which leverages a polynomial formulation of the composition reconstruction problem first described in [5]. The method is optimized in [6] to deal with t symmetric composition errors. Our main goal is to analyze the method based on polynomials and try to do some improvements on its efficiency. We explain the basic idea of the method, illustrating how it works to do error correction successfully.

In the first part of this article, the method for error correction based on polynomial is introduced by discussing the principle of the transformation from binary strings to corresponding polynomials. In the second part, by discussing the theoretical feasibility of error reconstruction method in polynomial form, the method is indicated to be reliable. In order to introduce how the polynomial-based method is capable of correcting, we prove the following theorem, showing a key step of subsequent derivations of the method. Let $\mathbb{F}_q$ be a finite field of order q, where q is an odd prime. Let $\alpha \in \mathbb{F}_q$ be a primitive element of the field. For a polynomial $f(x) \in \mathbb{F}_q[x]$, let $\mathcal{R}(f)$ denote the set of its roots.

Theorem 1. ([7, Ch. 5]) Assume that $E(x) \in \mathbb{F}_q[x]$ has at most t nonzero coefficients. Then, $E(x)$ can be uniquely determined in $\mathcal{O}(n^2)$ time given $E(\alpha^t)$, $E(\alpha^{t-1})$, ..., $E(\alpha^1), E(\alpha^0), E(\alpha^{-1}), ..., E(\alpha^{-t+1}), E(\alpha^{-t})$.

The theorem indicates the feasibility given polynomials with at most t nonzero coefficients based on a primitive element α of given finite field $\mathbb{F}_q$. Giving polynomial form errors with at most t nonzero coefficients, the errors in the composition of the original string can be corrected and thus the original string can be reconstructed. The process of reconstruction utilizes Reed-Solomon codes and cyclic codes from coding theory. In [5], the upper bound of the number of redundant bits in the code reconstruction under symmetric multiple errors has been proved to be $156t^2 \log 8n$, where t is the number of symmetric errors and n is the length of binary string. Then the number of redundant bits is $\mathcal{O}(t^2 \log n)$ and its lower bound is still unknown. Thus, improving the method is an interesting direction for future studies.

II. Methodology

A. Polynomial formulation

In order to correct t symmetric composition errors while conducting binary strings reconstructions, we introduce a polynomial-based method for further usage. The method defines compositions of strings and their prefixes and extends to composition multisets which are constructed from compositions. The composition multiset of a string is obtained by writing out all its substrings of all possible lengths and representing each substring by its composition.

For a string $\mathbf{s} \in \{0,1\}^n$, let $P_{\mathbf{s}}(x,y)$ be a bivariate polynomial of degree n with coefficients in $\{0,1\}$ such that it contains exactly one term with total degree $i \in \{0,1,...,n\}$. If $\mathbf{s} = s_1..s_n$ and if $P_s(x,y)_i$ denotes the unique term of total degree i, then $P_s(x,y)_0 = 1$, and

Here we use x to denote 1 and y to denote 0 in composition multisets of substrings. We summarize prefixes from length

1 to $n-1$, construct composition, and transform it into polynomials. For example, for $\mathbf{s} = 0010$ we have $P_{\mathbf{s}}(x, y) = 1 + y + y^2 + xy^2 + xy^3$ and it is formed by all prefixes of string **s**. To prove the equation, we start with free coefficient 1 and expand to prefix with different lengths. By adding y to the free coefficient, prefix with length 1 is expressed. y^2, xy^2, and xy^3 is added to express prefixes with length 1, 2, and 3. Then the relation between left and right hand side can be proved.

Except for prefixes, we also use $S_{\mathbf{s}}(x, y)$ to express composition multisets of the string **s**, with similar logic as above. As an example, for $\mathbf{s} = 0010$ we have

$$C(\mathbf{s}) = \{0, 0, 1, 0, 0^2, 01, 01, 0^21, 0^21, 0^31\}$$

and

$$S_{\mathbf{s}}(x, y) = x + 3y + 2xy + y^2 + 2xy^2 + xy^3,$$

where the first two terms of $S_{\mathbf{s}}(x, y)$ indicate that it contains one substring 1 and three substrings 0. The following three terms indicate that it contains two substrings constructed by one 0 and one 1, one substring constructed by two 0s, two substrings constructed by one 1 and two 0s. For longer strings and their composition multisets, similar logic could be used and following terms could be interpreted similarly.

From [5], we introduce Equation 1

$$P_{\mathbf{s}}(x, y)P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y}) = (n + 1) + S_{\mathbf{s}}(x, y) + S_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y}) \quad (1)$$

Proof. For each component except 1 in $P_{\mathbf{s}}(x, y)$ denoted by $x^i y^{n-i}$ for the prefix with length i, there is paired component in $P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})$ represented by $x^{-i}y^{i-n}$. From $P_{\mathbf{s}}(x, y)P_s(\frac{1}{x}, \frac{1}{y})$, we can gain n 1s from n pairs of variables above if there is a string **s** with length n.

Given the length of $P_{\mathbf{s}}$ is $n + 1$, let $P_{\mathbf{s}}(k)$ represents the k^{th} term of $P_{\mathbf{s}}$, where k is from 1 to $n + 1$. Let $P_{\mathbf{s}}(k-)$ represents the polynomial containing 1^{st} to $k - 1^{th}$ terms of $P_{\mathbf{s}}$ and $P_{\mathbf{s}}(k+)$ represents the polynomial containing $k + 1^{th}$ to $n + 1^{th}$ terms of $P_{\mathbf{s}}$. The multiplication would be $1 + S_{\mathbf{s}}(x, y) + S_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})$. 1 from $1 \cdot 1$, $S_{\mathbf{s}}(x, y)$ from $\sum P_{\mathbf{s}}(x, y)(k)P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})(k-)$, $S_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})$ from $\sum P_{\mathbf{s}}(x, y)(k)P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})(k+)$.

Further we conclude that

$$\sum_{k=1}^{n+1} P_{\mathbf{s}}(x, y)(k)P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})(k-) = \sum_{k=1}^{n+1} P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})(k)P_{\mathbf{s}}(x, y)(k+)$$

Then the equation holds. □

We use a square matrix to represent the pairing result of a string **s**=0010, where we have $P_{\mathbf{s}}(x, y) = 1+y+y^2+xy^2+xy^3$ and $P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y}) = 1 + \frac{1}{y} + \frac{1}{y^2} + \frac{1}{xy^2} + \frac{1}{xy^3}$,

$$\mathbf{M}(\mathbf{s}) = \begin{bmatrix} 1 & y & y^2 & xy^2 & xy^3 \\ \frac{1}{y} & 1 & y & xy & xy^2 \\ \frac{1}{y^2} & \frac{1}{y} & 1 & x & xy \\ \frac{1}{xy^2} & \frac{1}{xy} & \frac{1}{x} & 1 & y \\ \frac{1}{xy^3} & \frac{1}{xy^2} & \frac{1}{xy} & \frac{1}{y} & 1 \end{bmatrix} \quad (2)$$

In the matrix, we can find out that the matrix is divided into two parts by the diagonal constructed by 1 from upper left to lower right. Since each string **s** with length n will have $P_{\mathbf{s}}(x, y)$ and $P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})$ with lengths equal to $n+1$, the $(n+1) \times (n+1)$ matrix have the diagonal with length $n+1$. The sum of components in the diagonal is $n+1$ which matches $n+1$ at the right-hand side of the equation. The upper triangular matrix contains the same components as $P_{\mathbf{s}}(x, y)$ and the lower triangular matrix contains the same components as $P_{\mathbf{s}}(\frac{1}{x}, \frac{1}{y})$.

For any given bivariate polynomial $f(x, y)$, denote $f^*(x, y)$ to be its reciprocal polynomial which defined as

$$f^*(x, y) = x^{\deg_x(f)}y^{\deg_y(f)}f(\frac{1}{x}, \frac{1}{y})$$

where $\deg_x(f)$ denotes the x-degree of $f(x, y)$ and $\deg_y(f)$ denotes the y-degree of $f(x, y)$. Thus we can rewrite (1) as:

$$P_{\mathbf{s}}(x, y)P^*_{\mathbf{s}}(x, y) = x^{d_x}y^{d_y}(n+1+S_{\mathbf{s}}(x, y)) + S^*_{\mathbf{s}}(x, y). \quad (3)$$

Given $\widetilde{C}(s)$ representing composition multiset with t errors, we use $S_{\mathbf{s}}(x, y)$ to denote the polynomial form of $C(s)$ and $\widetilde{S}_{\mathbf{s}}(x, y)$ to denote $\widetilde{C}(s)$. We use $E(x, y)$ to denote the difference between $\widetilde{S}_{\mathbf{s}}(x, y)$ and $S_{\mathbf{s}}(x, y)$, represented as:

$$\widetilde{S}_{\mathbf{s}}(x, y) = S_{\mathbf{s}}(x, y) + E(x, y) \quad (4)$$

where $E(x, y)$ has at most 2t nonzero coefficients.

The above are some basic concepts and equations that transform composition multisets into polynomial forms. More importantly, we obtain a polynomial $E(x, y)$ that represents the difference multiset $C(s)$ and $\widetilde{C}(s)$. The knowledge will be used in the proof of Theorem 1.

B. Reed-Solomon codes

We use the following definitions of polynomial from [8] for the ease of understanding.

Definition II.1. A polynomial over a variable X and a finite field $\mathbb{F}_q$ is given by a finite sequence $(f_0, f_1, ..., f_n)$ with $f_i \in \mathbb{F}_q$ and is denoted by $F(x) = \sum_{i=0}^{n} f_i x^i$. The *degree* of $F(x)$, denoted *deg(F)*, is the largest index i such that $f_i \neq 0$.

For example, $6x^4 + 2x^3 + x^2 + x + 4$ is a polynomial over $\mathbb{F}_7$ of $deg(F) = 4$.

Recalling the definition of finite field, we know that any given polynomial $F(x)$ over a prime finite field $\mathbb{F}_n$ with $deg(F)$, $\max(f_0, f_1, ..., f_i) = n-1$. By the definition of finite field, over $\mathbb{F}_n$, $n = 0$, so for addition and multiplication of polynomials, coefficients after calculation are executed with $\bmod n$ operation. For example, adding x and $1+x$ over finite field $\mathbb{F}_2$, we obtain $x + (1+x) = x \cdot (1+1) + 1 \cdot (1+0) = 1$.

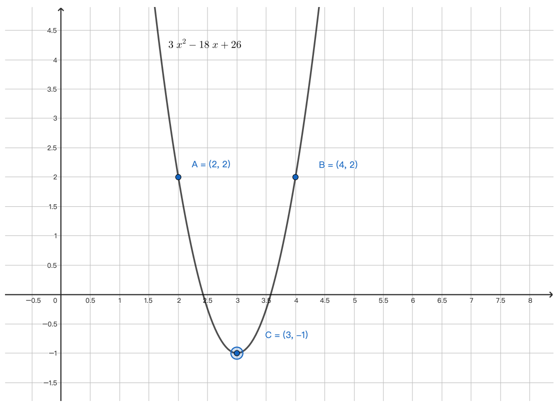

Figure 1. degree-2 function constructed by 3 points

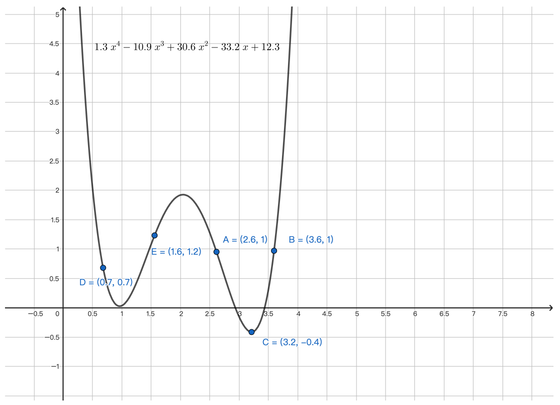

Figure 2. degree-4 function constructed by 5 points

Since coefficient of x is 2, which we have $2 \bmod 2 = 0$, we have $2x = 0$.

Now we use the definition of Reed-Solomon codes from [8] to see how polynomials can be used in Definition II.1

Definition II.2. Let $\mathbb{F}_q$ be a finite field, and choose n and k satisfying $k \leq n \leq q$. Fix a sequence $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$ of n distinct elements (also called evaluation points) from $\mathbb{F}_q$. We define an encoding function for Reed-Solomon code $RS_q[\boldsymbol{\alpha}, k] : \mathbb{F}_q^k \to \mathbb{F}_q^n$ as follows. Map a message $\mathbf{m} = (m_0, m_1, ..., m_{k-1})$ with $m_i \in \mathbb{F}_q$ to the degree $k-1$ polynomial.

$$\mathbf{m} \mapsto f_{\mathbf{m}}(X),$$

where

$$f_{\mathbf{m}}(X) = \sum_{i=0}^{k-1} m_i X^i.$$

With known $\boldsymbol{\alpha}, q, k$, we can encode $\mathbf{m}$ from evaluation of polynomial at all α_i's. With enough evaluation points, one can reconstruct the original polynomial. The reconstruction is similar to draw a polynomial function graph by knowing the position of several points. Fig 1 and Fig 2 are two examples where n points uniquely defines polynomial with degree $n+1$.

Note that the codewords of Reed-Solomon code $RS(n, k)$ is of the form $(f(a_1), f(a_2), ..., f(a_n))$ where $(a_1, a_2, ..., a_n)$ is a collection of values of independent variables and f is a polynomial of degree less than k. By mapping from a_j to α^j where α is a primitive element, the codewords form a cyclic code, which we will discuss in more detail in the next subsection.

C. Cyclic codes

We use the definition of cyclic codes from [9] to see how it works.

Definition II.3. A subset S of $\mathbb{F}_q^n$ is *cyclic* if $(a_{n-1}, a_0, a_1, ..., a_{n-2}) \in S$ whenever $(a_0, a_1, ..., a_{n-1}) \in S$. A linear code C is called a *cyclic code* if C is a cyclic set.

Given a word $(a_0, a_1, ..., a_{n-1}) \in \mathbb{F}_q^n$, we can obtain the extension of the word $(a_{n-r}, ..., a_{n-1}, a_0, a_1, ..., a_{n-r-1})$ by cyclically shifting *r* positions. Note that cyclic sets should be in linear spaces, or they are not cyclic codes.

For example, $\{000, 110, 101, 011\}$ is a linear code over $\mathbb{F}_2^3$.

In order to convert the combinatorial structure of cyclic codes into algebraic one, then matching polynomial forms discussed before, we consider the following mapping:

$$\begin{aligned} &\pi : \mathbb{F}_q^n \to \mathbb{F}_q[x]/(x^n - 1), \\ &(a_0, a_1, ..., a_{n-1}) \mapsto a_0 + a_1 x + ... + a_{n-1}x^{n-1}. \end{aligned} \tag{5}$$

Then π is an $\mathbb{F}_q$-linear transformation of vector spaces over $\mathbb{F}_q$. $x^n - 1$ here represents the upper bound of polynomials mapped from binary strings. Recall how we use polynomials to represent binary strings, we have $\pi(C) = \{0, 1 + x, 1 + x^2, x + x^2\} \subset \mathbb{F}_2[x]/(x^3 - 1)$ based on the above example code $C = \{000, 110, 101, 011\}$. We introduce the definition *ideal* to further explain how cyclic codes are used in Theorem 1.

Definition II.4. Let R be a ring. A nonempty subset I of R is called an *ideal* if
(i) both $a + b$ and $a - b$ belong to I, for all $a, b \in I$;
(ii) $r \cdot a \in I$, for all $r \in R$ and $a \in I$.

For example, in the ring $\mathbb{F}_2[x]/(x^3 - 1)$, the subset

$$I := \{0, 1 + x, x + x^2, 1 + x^2\}$$

is an *ideal*.

We find the following result useful for subsequent derivation.

Theorem 2. ([9, Ch. 7]) Let π be the linear map defined in 5. Then a nonempty subset C of $\mathbb{F}_q^n/(x^n - 1)$ is a cyclic code if and only if $\pi(C)$ is an ideal of $\mathbb{F}_q[x]/(x^n - 1)$

Let $\mathbf{c} = (c_0, c_1, ..., c_{n-1})$ be a codeword of C. Then we can map the $\mathbf{c}$ to its polynomial form:

$$\pi(\mathbf{c}) = c_0 + c_1 x + ... + c_{n-2}x^{n-2} + c_{n-1}x^{n-1}$$

Since $\mathbf{c}$ is a codeword of C, we have the above polynomial as an element of $\pi(C)$, the mapping of subset C. From Theorem 2, since $\pi(C)$ is an ideal of $\mathbb{F}_q[x]/(x^n - 1)$, we have

$$\begin{aligned} x\pi(C) &= c_0x + c_1x^2 + ... + c_{n-2}x^{n-1} + c_{n-1}x^n \\ &= c_{n-1} + c_0x + c_1x^2 + ... + c_{n-2}x^{n-1} \\ &(\text{used } x^n - 1 \ = \ 0 \text{ in } \mathbb{F}_q[x]/(x^n-1)) \end{aligned}$$

is an element of $\pi(C)$, which you can observe the cyclic property in the coefficients of polynomials.

Note that each codeword of cyclic codes corresponds to the coefficients of a polynomial. The weight of the codeword $wt(\mathbf{c})$ is the number of nonzero elements in it.

III. Results

We discuss how Theorem 1, we analyze how it works step by step.

A. The existence of primitive elements

The premise of Theorem 1 is the existence of primitive element α. Let us recall the definition from [9]:

Definition III.1. An element α in finite field $\mathbb{F}_q$ is called a *primitive element* (or *generator*) of $\mathbb{F}_q$ if $\mathbb{F}_q = \{0, \alpha, \alpha^2, ..., \alpha^{q-1}\}$.

From our assumption of $\mathbb{F}_q$, we have q is an odd prime. So 2 can always be a primitive element since $\gcd(2, q) = 1$. If we have an integer a such that $\gcd(a, q) > 1$ where $a = 1$ *or* $a = q$, $a^j \mod q$ will be the same which cannot generate all elements in $\mathbb{F}_q$. The odd prime q determines the number of elements in the finite field and also allows the existence of primitive element α which satisfies $\gcd(\alpha, q) = 1$.

B. Distance and error correction

Before discussing the details of the theorem, let's clarify a property of codes. By the following definition and theorem from [9], we see how *distance* is defined and how it connects to the limit weight of codewords.

Definition III.2. Let $\mathbf{x}$ and $\mathbf{y}$ be words of length n over an alphabet A. The (*Hamming*) *distance* from $\mathbf{x}$ to $\mathbf{y}$, denoted by $d(\mathbf{x}, \mathbf{y})$, is defined to be the number of places at which $\mathbf{x}$ and $\mathbf{y}$ differ. If $\mathbf{x} = x_1...x_n$ and $\mathbf{y} = y_1...y_n$, then

$$d(\mathbf{x}, \mathbf{y}) = d(x_1, y_1) + ... + d(x_n, y_n),$$

where x_i and y_i are regarded as words of length 1, and

$$d(x_i, y_i) = \begin{cases} 1 \text{ if } x_i \neq y_i \\ 0 \text{ if } x_i = y_i. \end{cases}$$

The above definition defines $d(\mathbf{x}, \mathbf{y})$ which helps obtain the distance of codes.

Definition III.3. For a code C containing at least two words, the (*minimum*) *distance* of C, denoted by $d(C)$, is

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

Distance measures the difference between two binary strings. The distance of C is the minimum distance between every two words. For example, let $C = \{00000, 11010, 11111\}$, we have

$$\begin{aligned} d(00000, 11010) &= 3, \\ d(00000, 11111) &= 5, \\ d(11111, 11010) &= 2. \end{aligned}$$

Therefore, we have $d(C) = 2$.

Theorem 3. A code C is v-error-correcting if and only if $d(C) \geq 2v + 1$; i.e., a code with distance d is a $\lfloor (d-1)/2 \rfloor$-error-correcting code. Here, $\lfloor x \rfloor$ is the greatest integer less than or equal to x.

The theorem clarifies the relationship between the minimum weight of code C, denoted by $wt(C)$ and the distance $d(C)$. Given codeword e representing the set of coefficients of $E(x)$, we have the inequalities

$$d(C) \geq 2wt(e) + 1$$

and

$$wt(e) \leq \frac{d(C) - 1}{2}.$$

In Theorem 1, we have $wt(e) = t$. Recalling that the weight represents nonzero elements in e, that is the number of nonzero coefficients of $E(x)$. So we need $d(C) \geq 2t + 1$ degree. Since we know the values $E(\alpha^t)$, $E(\alpha^{t-1})$, ..., $E(\alpha^1), E(\alpha^0), E(\alpha^{-1}), ..., E(\alpha^{-t+1}), E(\alpha^{-t})$ and the requirement of minimum distance, and the $2t + 1$ values can then determine $E(x)$.

To see this, we need the following definition of syndrome from [9]

Definition III.4. A *linear code* C of length n over $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$.

Definition III.5. (i) A *generator matrix* for a linear code C is a matrix G whose rows form a basis of C.
(ii) A *parity-check matrix* H for a linear code C is a generator matrix for the dual code $C^{\perp}$.

The above definition of linear code and parity-check matrix help connect parity-check matrix of a linear code with length n, dimension k, and distance d, to syndrome.

Definition III.6. Let C be an $[n, k, d]$-linear code over $\mathbb{F}_q$ and let H be a parity-check matrix for C. For any $\mathbf{w} \in \mathbb{F}_q^n$, the *syndrome* of $\mathbf{w}$ is the word $S(\mathbf{w}) = \mathbf{w}H^T \in \mathbb{F}_q^{n-k}$.

First, let's start with analyzing $E(\alpha^j)$ for $j \in \{-t, ..., 0, 1, ..., t\}$. We have each $E(\alpha^j) = \sum_{i=0}^{n-1} e_i(\alpha^j)^i$ where e_i represent coefficients, forming a codeword $e = (e_0, e_1, ..., e_{n-1})$, and there are n terms in the polynomial. The polynomial of $E(\alpha^j)$ can be transformed into multiplication of two matrices:

$$\begin{pmatrix} e_0 & \cdots & e_{n-1} \end{pmatrix} \begin{pmatrix} (\alpha^j)^0 \\ \vdots \\ (\alpha^j)^{n-1} \end{pmatrix}$$

Given $j \in \{-t, ..., 0, 1, ..., t\}$, we combine the matrices into one matrix:

$$\mathbf{M}_\alpha = \begin{pmatrix} (\alpha^{-t})^0 & (\alpha^{-t+1})^0 & \cdots & (\alpha^t)^0 \\ (\alpha^{-t})^1 & (\alpha^{-t+1})^1 & \cdots & (\alpha^t)^1 \\ \vdots & \vdots & \ddots & \vdots \\ (\alpha^{-t})^{n-1} & (\alpha^{-t+1})^{n-1} & \cdots & (\alpha^t)^{n-1} \end{pmatrix}$$

and we have the multiplication:

$$\begin{pmatrix} e_0 & \cdots & e_{n-1} \end{pmatrix} \begin{pmatrix} (\alpha^{-t})^0 & (\alpha^{-t+1})^0 & \cdots & (\alpha^t)^0 \\ (\alpha^{-t})^1 & (\alpha^{-t+1})^1 & \cdots & (\alpha^t)^1 \\ \vdots & \vdots & \ddots & \vdots \\ (\alpha^{-t})^{n-1} & (\alpha^{-t+1})^{n-1} & \cdots & (\alpha^t)^{n-1} \end{pmatrix}$$

Then we apply transposition to the two matrices e and M_α and generate:

$$\begin{pmatrix} (\alpha^{-t})^0 & (\alpha^{-t})^1 & \cdots & (\alpha^{-t})^{n-1} \\ (\alpha^{-t+1})^0 & (\alpha^{-t+1})^1 & \cdots & (\alpha^{-t+1})^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ (\alpha^t)^0 & (\alpha^t)^1 & \cdots & (\alpha^t)^{n-1} \end{pmatrix} \begin{pmatrix} e_0 \\ \vdots \\ e_{n-1} \end{pmatrix}$$

Then we successfully transform $\mathbf{M}_\alpha$ into the parity-check matrix and e into e^T. The result of multiplication is a $(2t+1) \times 1$ matrix, which is just the *syndrome* of $E(x)$.

Definition III.7. Let C be a linear code of length n over $\mathbb{F}_q$, and let $\mathbf{u} \in \mathbb{F}_q^n$ be any vector of length n; we define the *coset* of C determined by u to be the set

$$C + \mathbf{u} = \{\mathbf{v} + \mathbf{u} : \mathbf{v} \in C\} (= \mathbf{u} + C)$$

The above definition defines coset and help analyze the property of syndrome.

Note that all words in a given coset yield the same syndrome, so a syndrome of a coset can be the syndrome of any word in it. Since there is one-to-one correspondence between e and its syndrome if $2wt(e) + 1 \leq d(C)$, we can uniquely determine e given *syndrome* and thus recover e and its polynomial-form $E(x)$.

Recall equation (1) we proved before

$$P_\mathbf{s}(x, y) P_\mathbf{s}(\frac{1}{x}, \frac{1}{y}) = (n+1) + S_\mathbf{s}(x, y) + S_\mathbf{s}(\frac{1}{x}, \frac{1}{y}),$$

$E(x)$ can then be used for further data reconstruction based on equation (4).

To summarize, the relationship between syndrome and the weight of e is bi-direction. So the number of nonzero coefficients is limited by the amount of given evaluation of E(x) on consecutive powers $\alpha^i, i = -t, \ldots, t$. The given $2t+1$ values $E(\alpha^i)$ for $i \in \{-t, ..., 0, 1, ..., t\}$ only allows $E(x)$ to have t nonzero coefficients and we cannot reconstruct $t+1$ or more nonzero coefficients polynomial.

C. The time complexity $\mathcal{O}(n^2)$

Decoding time complexity of cyclic Reed-Solomon codes is well studied in the literature. Assuming the ratio t/n is bounded away from zero, there are decoding algorithms with running time $\mathcal{O}(n^2)$, including the Peterson-Gorenstein-Zierler algorithm [10], the Berlekamp-Massey algorithm [11], and the Euclidean decoding algorithm [12].

IV. Conclusion

In this report, we show how polynomial can express composition multisets of string $\mathbf{s}$ and introduce knowledge based on Reed-Solomon codes and cyclic codes to prove Theorem 1 step by step. The first step is to prove the existence of primitive element $\alpha \in \mathbb{F}_q$ where q is an odd prime. The next step is to show how distance determines the upper bound of $wt(e)$ for codeword e and how $E(\alpha^i)$ for $i \in \{-t, ..., 0, 1, ..., t\}$ are combined into parity-check matrix for obtaining the only syndrome of a coset and of any word in it. The last step is to clarify the decoding time complexity $\mathcal{O}(n^2)$ of cyclic Reed-Solomon codes based on algorithms in existence.

V. Future Work

According to Theorem 1, $E(x)$ with t nonzero coefficients can be uniquely determined. Then we can reconstruct a binary string $S_\mathbf{s}(x, y)$ even if there are t errors. Recall that efficient reconstruction in the presence of a constant number of t errors requires redundancy bits $\mathcal{O}(t^2 \log n)$. The future goal is to reduce the number of redundancy bits of reconstructing t errors.

Acknowledgment

The whole project cannot be done without the help of other two members: Professor Zitan Chen being responsible for instruction and Yaoyu Yang conducting understanding of basic algorithm.

References

[1] V. Zhirov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," Nature materials, vol. 15, no. 4, p. 366, 2016.

[2] A. Al Ouahabi, J.-A. Amalian, L. Charles, and J.-F. Lutz, "Mass spectrometry sequencing of long digital polymers facilitated by programmed inter-byte fragmentation," Nature communications, vol. 8, no. 1, p. 967, 2017.

[3] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized dna," Nature, vol. 494, no. 7435, p. 77, 2013.

[4] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," Angewandte Chemie International Edition, vol. 54, no. 8, pp. 2552–2555, 2015.

[5] J. Acharya, H. Das, O. Milenkovic,A. Orlitsky, and S. Pan,"String reconstruction from substring compositions," SIAM Journal on Discrete Mathematics, vol. 29, no. 3, pp. 1340–1371, 2015.

[6] S. Pattabiraman, R. Gabrys and O. Milenkovic, "Coding for Polymer-Based Data Storage," in IEEE Transactions on Information Theory, vol. 69, no. 8, pp. 4812-4836, Aug. 2023.

[7] R. Roth, Introduction to coding theory. Cambridge University Press, 2006.

[8] V. Guruswami, A. Rudra, M. Sudan, "Essential Coding Theory," unpublished.

[9] S. Ling, C. Xing, Coding Theory: A First Course. Cambridge University Press, 2004.

[10] D. Gorenstein and N. Zierler, "A Class Of Error-Correcting Codes In p^m Symbols," in Journal of the Society for Industrial and Applied Mathematics, vol. 9, no. 2, pp. 207-214, 1961.

[11] J. Massey, "Shift-register synthesis and BCH decoding," in IEEE Transactions on Information Theory, vol. 15, no. 1, pp. 122-127, January 1969, doi: 10.1109/TIT.1969.1054260.

[12] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, ”A method for solving key equation for decoding Goppa codes,” in Information and Control, 27 (1975), 87–99.