CS3236 Project: Reed-Solomon Codes

Pan Jing Bin and Yip Jung Hon

Contents

1 Introduction

1.1 Introduction to Reed-Solomon Codes

Reed-Solomon codes are a family of error-correcting codes that are simple in nature yet have extremely powerful error correction capabilities. They were first introduced by Irving S. Reed and Gustave Solomon in 1960 and have since been widely used in various consumer and data transmission technologies, such as Blu-ray discs and satellite communication systems. In this report, we will explore the theoretical foundations of Reed-Solomon codes.

The main tool for studying Reed-Solomon codes is commutative algebra, especially ring theory. However, we will not delve into the algebraic details and focus on the coding aspects instead. Therefore, we assume that the reader has a basic background in mathematics undergraduate algebra, including group theory and ring theory. Some familiarity with Galois theory would also be helpful. Our main references are Sections 1.2 and 5.1 in [WB99].

2 General Theory

2.1 Linear and Cyclic Codes

We first generalise the terminology and notation introduced in lecture to arbitrary codes (as opposed to only binary codes).

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_q\}$ be the set of symbols. A ***word*** of length n over $\mathcal{A}$ is a sequence $x_1x_2\ldots x_n$ with $x_i \in \mathcal{A}$ for all $1 \leq i \leq n$. A ***code*** of length n over $\mathcal{A}$ is a nonempty subset C of $\mathcal{A}^n$. An element of C is called a ***codeword*** of C. A code of length n and cardinality M is called an (n, M)-code.

Definition 2.1.1. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be words of length n over $\mathcal{A}$.

The ***Hamming distance*** between $\boldsymbol{x}$ and $\boldsymbol{y}$ is the number of positions in which $\boldsymbol{x}$ and $\boldsymbol{y}$ differ. It is denoted by $d_H(\boldsymbol{x}, \boldsymbol{y})$. The ***minimum distance*** of a code C is denoted $d_H(C)$ by abuse of notation, and is the quantity:

$$d_H(C) = \min_{\boldsymbol{x}\neq\boldsymbol{y}\in C} d_H(\boldsymbol{x}, \boldsymbol{y}).$$

If a (n, M)-code has distance d, we sometimes use (n, M, d) to denote the code.

Let p denote a prime number, and let $\mathbb{F}_q$ denote the finite field of q elements. Note that q is necessarily a prime power. An element $\mathbb{F}_q$ is called ***primitive*** if every nonzero element of $\mathbb{F}_q$ can be written as a power of α. In other words, $\mathbb{F}_q \setminus \{0\} = \{1, \alpha, \cdots, \alpha^{q-2}\}$. It is well-known that every finite field contains a primitive element.

Definition 2.1.2 (Linear Codes). A ***linear code*** over a finite field $\mathbb{F}_q$ is a code whereby the symbol set is $\mathcal{A} = \mathbb{F}_q$, and C is a linear subspace of $\mathbb{F}_q^n$. If $\dim(C) = k$, then C is called a $[n, k]$-linear code over $\mathbb{F}_q$. Moreover, if $d_H(C) = d$, we sometimes use $[n, k, d]$ to denote the code.

Remark 2.1.3. In this report, round brackets (n, M) are used to denote general codes, while square brackets $[n, k]$ are used to denote linear codes.

Remark 2.1.4. If C is a $[n, k]$-linear code, it is k-dimensional vector space over $\mathbb{F}_q$; thus it is a (n, q^k)-code.

Example 2.1.5. Fix $n = 8$ and $q = 2$. Let $\mathcal{A}$ be $\mathbb{F}_2 = \{0, 1\}$. Let C be the following subspace of $\mathcal{A}^n$:

$$C = \left\{x_1x_2\ldots x_8 \in \mathcal{A}^n \mid x_1 + x_2 + \cdots + x_8 = 0\right\}.$$

Then C is a $(8, 2^7)$-code. It is further a $[8, 7]$-linear code.

Definition 2.1.6 (Cyclic Codes). Let $\sigma : \mathcal{A}^n \to \mathcal{A}^n$ be the ***right-shift*** operator. In other words, if $\boldsymbol{c} \in C$ is a codeword such that $\boldsymbol{c} = c_0c_1c_2\ldots c_{n-1}$, then $\sigma(\boldsymbol{c}) = c_{n-1}c_0c_1\ldots c_{n-2}$. A code $C \subseteq \mathbb{F}_q^n$ is said to be ***cyclic*** over $\mathbb{F}_q$ if:

- It is a linear code over $\mathbb{F}_q$ and,
- For any $\boldsymbol{c} \in C$, $\sigma(\boldsymbol{c}) \in C$.

Let $\mathbb{F}_q[x]$ denote the ring of polynomials over the field $\mathbb{F}_q$. For any $n \in \mathbb{Z}_{\geq 1}$, let $\mathbb{F}_q[x; n] = \mathbb{F}_q[x]/(x^n - 1)$ (with normal polynomial addition and multiplication modulo $x^n - 1$) denote the quotient ring; each element in $\mathbb{F}_q[x; n]$ can

be represented uniquely by a polynomial of degree strictly less than n; we henceforth use this identification. Now, given a codeword $\boldsymbol{c} \in C$ such that $\boldsymbol{c} = c_0c_1c_2 \dots c_{n-1}$, it can be uniquely represented as a polynomial $c(x) \in \mathbb{F}_q[x;n]$:

$$c_0c_1c_2 \dots c_{n-1} \mapsto c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}.$$

The right-shift operator acting on $c_0c_1c_2 \dots c_{n-1}$ corresponds to a 'multiplication of x' action on $\mathbb{F}_q[x;n]$ in the following way: σ sends $c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ to $x(c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}) \pmod{x^n - 1}$. We illustrate this concept with the following example.

Example 2.1.7. Let $n = q = 3$, and let $\mathcal{A} = \mathbb{F}_q = \{0, 1, 2\}$. Suppose $112 \in C$. Then 112 has the polynomial representation $1 + x + 2x^2 \in \mathbb{F}_q[x;3]$. On one hand, the right-shift operator sends 112 to 211. One the other hand, by transforming the polynomial in the way described above, we obtain:

$$\begin{aligned} x(1 + x + 2x^2) \pmod{x^3 - 1} &= x + x^2 + 2x^3 \pmod{x^3 - 1} \\ &= 2 + x + x^2 \pmod{x^3 - 1}. \end{aligned}$$

From now on, we identify elements in C as elements in $\mathbb{F}_q[x;n]$ and vice versa. The reader should note the abuse of notation.

It is easy to verify that a code is cyclic if and only if its corresponding subset in $\mathbb{F}_q[x;n]$ is an ideal.

The following theorem will be useful later.

Theorem 2.1.8. Let $n, q \in \mathbb{Z}_{\geq 1}$ be given, where q is a prime power. Let $C \subseteq \mathbb{F}_q^n$ be cyclic. Again, we identify the set C with an ideal in $C \subseteq \mathbb{F}_q[x;n]$. Then,

(i) There exists a polynomial $g(x) \in \mathbb{F}_q[x;n]$ such that every $c(x) \in C \subseteq \mathbb{F}_q[x;n]$ can be written as

$$c(x) = g(x)f(x) \pmod{x^n - 1}$$

for some $f(x) \in \mathbb{F}_q[x;n]$. Moreover,

(ii) $g(x)$ divides $x^n - 1$ in $\mathbb{F}_q[x]$ and,

(iii) $\deg(g(x)) = n - \dim(C)$.

The polynomial $g(x)$ is called the ***generator polynomial*** of C.

Proof. If $C = \{0\}$ or $C = \mathbb{F}_q^n$, it is clear that $g(x) = x^n - 1$ and $g(x) = 1$ respectively satisfies the theorem. We henceforth assume $C \neq \{0\}$ and $C \neq \mathbb{F}_q^n$. Choose $g(x) \in C$ to be the monic polynomial of smallest degree. Let $c(x) \in C$ the polynomial representation of any codeword. By the division algorithm, we have

$$c(x) = f(x)g(x) + r(x) \quad \text{in } \mathbb{F}_q[x] \tag{1}$$

for some polynomials $f(x), r(x) \in \mathbb{F}_q[x]$, with either $r(x) = 0$ or $\deg(r(x)) < \deg(g(x))$. Recall that C is an ideal of $\mathbb{F}_q[x;n]$, so $c(x), g(x) \in C$ imply $r(x) \in C$. But if $r(x) \neq 0$, $r(x)$ is a nonzero polynomial in $\mathbb{F}_q[x;n]$ and has smaller degree than $g(x)$, a contradiction. Thus $r(x) = 0 \in \mathbb{F}_q[x]$. This shows (i). Let $\deg(g(x)) = m$. Note that another consequence of the preceding argument is that $\deg(f(x)) \leq n - m - 1$ since Equation (1) holds in $\mathbb{F}_q[x]$.

As C is an ideal, we apply the division algorithm to obtain $x^n - 1 = f(x)g(x) + r(x)$. As $x^n - 1, g(x) \in C$, we see that $r(x) \in C$. Yet $g(x)$ is chosen to be a non-zero polynomial of smallest degree in C, thus $r(x) = 0$. This shows (ii).

Lastly, consider the 'multiplication by $g(x)$' map:

$$\varphi : \mathbb{F}_q[x;n] \to C, \quad \text{given by,} \quad \varphi\left(f(x)\right) \to f(x)g(x).$$

The map is well-defined because C is an ideal and so every element of the form $f(x)g(x)$ is in C.

This map is injective on the subset $\left\{f(x) \in \mathbb{F}_q[x] \mid \deg(f(x)) \leq n - m - 1\right\} \subseteq \mathbb{F}[x;n]$ because the polynomials in its

image will have degree less than n. So we have

$$|C| \geq \left|\left\{f(x) \in \mathbb{F}_q[x] \mid \deg(f(x)) \leq n-m-1\right\}\right| = q^{n-m}.$$

For the other inequality, the argument in (i) shows that every $c(x) \in C$ is of the form $c(x) = f(x)g(x) \in \mathbb{F}_q[x]$ for some polynomial $f(x)$ satisfying $\deg(f(x)) \leq n-m-1$. There are q^{n-m} choices for f; thus $|C| \leq q^{n-m}$. This implies that $\dim(C) = n-m = n-\deg(g(x))$, as desired. This shows (iii). □

The concept of generator polynomials is very important in the theory of Reed-Solomon codes; readers may want to think of it as an analogue of the generator matrix taught in class.

2.2 The Singleton Bound

Let C be a code. We say that C is ***u-error-detecting*** if whenever a code is sent and $\leq u$ errors are incurred, the received word is not in C. Hence one can always detect the existence of $\leq u$ errors. We also say that C is ***v-error-correcting*** if whenever a code is sent and $\leq v$ errors are incurred, the original word can always be decoded by finding a $\boldsymbol{c} \in C$ with minimum distance to the erranous codeword. It requires some thinking to see that $d_H(C) = d$, then C is $(d-1)$-error-detecting and $\lfloor (d-1)/2 \rfloor$-error-correcting.

Intuitively, we should think of codewords this way – the larger the $d_H(C)$, the better it is at detecting and correcting errors. Thus, in choosing our code C, one should always find C such that $d_H(C)$ is maximized. However, as $d_H(C)$ increases, the cardinality of C decreases. The tradeoff of an increased error detecting and correcting capability is that the number of possible codewords we are allowed to send decreases. Thus, a natural question arises: for $q, n, d \in \mathbb{Z}_{\geq 1}$ fixed, what is the maximum M such that an (n, M, d) code exists?

Definition 2.2.1. Let $q, n, d \in \mathbb{Z}_{\geq 1}$ be fixed, with $1 \leq d \leq n$. Let $A_q(n, d)$ denote the largest possible integer M such that a (n, M, d) code exists over $\mathbb{F}_q$. A code C such that $|C| = A_q(n, d)$ is called an ***optimal code***.

Here are some examples:

- $A_q(n, d) \leq q^n$.
- $A_q(n, 1) = q^n$.
- $A_q(n, 2) = q^{n-1}$. (See Example 2.1.5).
- $A_q(n, n) = q$, by taking every $\boldsymbol{c} \in C$ to be a constant vector.

In general, pinning down the value $A_q(n, d)$ is a very difficult task. For even small values of $q, n, d \in \mathbb{Z}_{\geq 1}$, only lower and upper bounds are known instead of exact values. See http://www.codetables.de/ for the best-known upper and lower bounds for $[n, k]$-linear codes. Nevertheless, we have some general bounds on $A_q(n, d)$, one of which is ***the Singleton bound***.

Theorem 2.2.2 (The Singleton Bound). Let $q, n, d \in \mathbb{Z}_{\geq 1}$ be fixed, with $1 \leq d \leq n$. Then

$$A_q(n, d) \leq q^{n-d+1}.$$

This bound is known as ***the Singleton bound***.

Proof. Let C be an optimal (n, M, d)-code so that $M = A_q(n, d)$. For every $c \in C$, delete the last $d-1$ digits of c. This gives us a new code C' in $\mathbb{F}_q^{n-d+1}$. As $d_H(C) = d$, any two different codewords in C will give rise to distinct codewords in C'. So $|C'| = |C|$. Yet $|C'| \leq q^{n-d+1}$, so $A_q(n, d) \leq q^{n-d+1}$. □

This means that if C is a linear code, the dimension of C is bounded by $n-d+1$ (the size of linear codes are always a power of q). Thus, $\dim(C) = k \leq n-d+1$. Rearranging, we have:

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

We give a special name for linear codes attaining this bound.

Definition 2.2.3. Let C be a $[n, k, d]$-linear code over $\mathbb{F}_q$. If $k + d = n + 1$, then C is called a ***maximum distance separable (MDS)*** code.

3 Original Approach

In this section, we will describe and analyse Reed and Solomon's original approach, as described in their 1960 seminal paper "Polynomial Codes over Certain Finite Fields" [RS60]. We will only outline the main idea of the algorithm. Note that this algorithm is different from the more modern generator polynomial approach, which will be described in Section 4.

3.1 Description of the Algorithm

Fix a finite field $\mathbb{F}_q$ of q elements and let α be a primitive element of $\mathbb{F}_q$.

Suppose that we have a tuple of n information symbols $\boldsymbol{m} = (m_0, m_1, \cdots, m_{n-1}) \in \mathbb{F}_q^n$, where we assume $n < q$. The necessity of the assumption will gradually become apparent as we describe the algorithm. We first use the symbols to construct a polynomial $f(x) \in \mathbb{F}_q[x]$ by setting

$$f(x) = m_0 + m_1 x + \cdots + m_{n-1} x^{n-1}.$$

A ***Reed-Solomon codeword*** $\boldsymbol{c}$ is then formed by evaluating $f(x)$ at each of the q elements in $\mathbb{F}_q$.

$$\boldsymbol{c} = (c_0, c_1, \cdots, c_{q-1}) = \big(f(0), f(\alpha), \cdots, f(\alpha^{q-1})\big). \tag{2}$$

If $\boldsymbol{m}' = (m'_0, m'_1, \cdots, m'_{n-1})$ is another n-tuple that is different from $\boldsymbol{m}$, then $\boldsymbol{m}'$ will define a polynomial of degree $n-1$ that is different from $\boldsymbol{m}$. As such, the two polynomials cannot agree at more than $n-1$ points. Since $q > n$, we see that $\boldsymbol{m}$ and $\boldsymbol{m}'$ must define different Reed-Solomon codewords. This implies that the Reed-Solomon code is nonsingular. A complete set of codewords is obtained by letting the n information symbols $(m_0, m_1, \cdots, m_{n-1})$ range over all possible values in $\mathbb{F}_q^n$. Since there are q^n elements in $\mathbb{F}_q^n$, there are q^n codewords in this Reed-Solomon code.

Note that since the sum of two polynomials of degree less than or equal to $n-1$ is another polynomial of degree less than or equal to $n-1$, the Reed-Solomon code is a linear code. Now, to recover $\boldsymbol{m}$ from the Reed-Solomon codeword $\boldsymbol{c}$, first observe that $\boldsymbol{c}$ can be related by Equation (2) to a system of q linear equations in n variables, as shown below

$$\begin{aligned}
f(0) &= m_0 \\
f(\alpha) &= m_0 + m_1\alpha + m_2\alpha^2 + \cdots + m_{n-1}\alpha^{n-1} \\
f(\alpha^2) &= m_0 + m_1\alpha^2 + m_2\alpha^4 + \cdots + m_{n-1}\alpha^{(n-1)2} \\
&\vdots \\
f(\alpha^{q-1}) &= m_0 + m_1\alpha^{q-1} + m_2\alpha^{2(q-1)} + \cdots + m_{n-1}\alpha^{(n-1)(q-1)}.
\end{aligned} \tag{3}$$

Since $q > n$, this is an overdetermined system of linear equations and any n of these expressions can be used to construct a system of linear equations in n variables. For instance, if the $i_0+1, i_1+1, \cdots, i_{n-1}+1$ rows were chosen, for $0 \le i_j \le q-1$ (we assume for convenience of notation that the first row is not chosen), then we obtain

$$\begin{pmatrix}
1 & \alpha^{i_0} & \alpha^{2i_0} & \cdots & \alpha^{(n-1)i_0} \\
1 & \alpha^{i_1} & \alpha^{2i_1} & \cdots & \alpha^{(n-1)i_1} \\
1 & \alpha^{i_2} & \alpha^{2i_2} & \cdots & \alpha^{(n-1)i_2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha^{i_{n-1}} & \alpha^{2i_{n-1}} & \cdots & \alpha^{(n-1)i_{n-1}}
\end{pmatrix}
\begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{n-1} \end{pmatrix}
=
\begin{pmatrix} f(\alpha^{i_0}) \\ f(\alpha^{i_1}) \\ f(\alpha^{i_2}) \\ \vdots \\ f(\alpha^{i_{n-1}}) \end{pmatrix}. \tag{4}$$

Note that the matrix on the left has columns that are linearly independent over $\mathbb{F}_q$ and so admits a unique solution that can be explicitly computed. Indeed, if $\lambda_0, \cdots, \lambda_{n-1} \in \mathbb{F}_q$ are scalars such that

$$\lambda_0 \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} + \lambda_1 \begin{pmatrix} \alpha^{i_0} \\ \alpha^{i_1} \\ \alpha^{i_2} \\ \vdots \\ \alpha^{i_{n-1}} \end{pmatrix} + \lambda_2 \begin{pmatrix} \alpha^{2i_0} \\ \alpha^{2i_1} \\ \alpha^{2i_2} \\ \vdots \\ \alpha^{2i_{n-1}} \end{pmatrix} + \cdots + \lambda_{n-1} \begin{pmatrix} \alpha^{(n-1)i_0} \\ \alpha^{(n-1)i_1} \\ \alpha^{(n-1)i_2} \\ \vdots \\ \alpha^{(n-1)i_{n-1}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

then the polynomial $g(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \cdots + \lambda_{n-1} x^{n-1}$ is of degree $n-1$ and satisfy

$$g(\alpha^{i_j}) = 0 \quad \text{for all } 0 \leq j \leq n-1.$$

Since the number of roots of g is larger than its degree, we must have $g \equiv 0$ and so $\lambda_j = 0$ for each $0 \leq j \leq n-1$. Given $\boldsymbol{c}$, we can solve these equations to recover $\boldsymbol{m}$; this gives a decoding algorithm.

3.2 Error Detecting and Error Correcting Capabilities

We now evaluate the error detecting and error correcting capabilities of the Reed-Solomon code. Assume that t of the codeword coordinates (i.e. the $f(\alpha^i)$'s) are corrupted by noise during transmission and are received incorrectly. This means that the rows in (3) corresponding to the corrupted coordinates are incorrect and would lead to an incorrect solution if one or more of them were used in the system of equations in (4). Further suppose that we do not know where the errors are. A naive approach to rectify the issue is to simply construct all possible distinct systems of n linear equations from the set (3) and then take the majority opinion among all the solutions. To analyse this method, we first need the following proposition.

Proposition 3.2.1. Let $\boldsymbol{m}' = (m'_0, m'_1, \cdots, m'_{n-1}) \in \mathbb{F}_q^n$ be such that $\boldsymbol{m} \neq \boldsymbol{m}'$. Then there are at most $\binom{t+n-1}{n}$ systems of n linear equations that has $\boldsymbol{m}'$ as the unique solution.

Proof. By viewing each linear equation as a hyperplane in $\mathbb{F}_q^n$, we see that a system of n linear equations yield $\boldsymbol{m}'$ as the solution if and only if the intersection of the n hyperplanes is precisely $\{\boldsymbol{m}'\}$. This happens only if each hyperplane contains the point $\boldsymbol{m}'$. Next, we claim that at most $t+n-1$ hyperplanes can contain the point $\boldsymbol{m}'$. This is because if there exists a family $\mathcal{F}$ of at least $t+n$ hyperplanes that contains $\boldsymbol{m}'$, then $\mathcal{F}$ necessarily contains a subfamily $\widetilde{\mathcal{F}}$ of n uncorrupted hyperplanes. Since $\widetilde{F}$ is a subfamily of $\mathcal{F}$, the intersection of these n hyperplanes will be precisely $\{\boldsymbol{m}'\}$ (recall that any system of n linear equations have a unique solution). But the n uncorrupted hyperplanes also correspond to a uncorrupted system of n linear equations, hence must also contain $\boldsymbol{m}$. This is a contradiction as $\boldsymbol{m}'$ is no longer the unique solution. □

If all of the n linear equations in the system (4) are uncorrupted, then the correct solution $\boldsymbol{m}$ will be obtained. This occurs for precisely $\binom{q-t}{n}$ equations. Recall that taking the majority opinion among all solutions yields the correct solution if and only if the correct solution occurs with higher frequency than any other solution. From Proposition 3.2.1, we deduce that the transmitted codeword will be decoded correctly when $\binom{q-t}{n} > \binom{t+n-1}{n}$. This condition holds if and only if $t+n-1 < q-t$. Rearranging, this is equivalent to $2t < q-n+1$. To summarise, a Reed-Solomon code of length q and dimension n can correct up to t errors, where

$$t = \left\lfloor \frac{q-n+1}{2} \right\rfloor. \tag{5}$$

Next, we will analyse the robustness of the Reed-Solomon code under erasures. When a part of the signal $(f(0), f(\alpha), \cdots, f(\alpha^{q-1}))$, say $f(\alpha^j)$, is erased, the $(j+1)$-th row in (3) will be removed from consideration. Hence if there are v erasures, there will be only $q-v$ equations left in (3). Together with the discussion above, we conclude that the Reed-Solomon code can correct t errors and v erasures as long as

$$2t + v < q - n + 1.$$

Perhaps unsurprisingly, the Reed-Solomon code can correct more erasures than errors. One way in which engineers leverage on this observation when designing real world systems is as follows. In many digital communication systems, a demodulator is used to make a rough estimate as to whether a given symbol at the output of the detector is reliable. For example, the detector may consist of a simple hard limiter: signals below a certain threshold are translated into zeros and signals above the threshold are translated into ones. However, this means that if a particular signal is very close to the threshold, then it has a significant probability of being incorrect. As such, the demodulator may erase the signal to prevent it from being assigned a binary value that has a high chance of being incorrect.

4 Generator Polynomial Approach

4.1 Description of the Code

Let $n, q \in \mathbb{Z}_{\geq 1}$ and let $\mathbb{F} = \mathbb{F}_q$ be a finite field.

The Reed-Solomon code can also be defined via a more modern, generator polynomial approach. Recall from Theorem 2.1.8 that a cyclic code can be constructed by specifying a generator polynomial, $g(x)$. The cyclic code can then be constructed as follows taking all elements of the form $\{f(x)g(x) \pmod{x^n - 1} \mid f(x) \in \mathbb{F}_q[x; n]\}$. We hence proceed by specifying the generator polynomial of a Reed-Solomon code.

Definition 4.1.1 (Reed-Solomon Code). Let $q, d, \theta \in \mathbb{Z}_{\geq 1}$ be fixed and let $\mathbb{F}_q$ be a finite field. Let α be a primitive element of $\mathbb{F}_q$. A ***Reed-Solomon code*** C is a length $q-1$ code constructed by the generator polynomial

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

Note that $g(x)$ has degree $d-1$. Theorem 2.1.8 then implies that $\dim(C) = q - d$. Thus C is a $[q-1, q-d]$-cyclic code. We denote the Reed-Solomon code with these parameters as $\mathrm{RS}(q-1, q-d)$.

Now, given a codeword $\boldsymbol{c} \in C$, the ***weight*** of the codeword is the number of nonzero letters of the code. The ***weight*** of the code, denoted $w(C)$, is the minimum of all the weights of every $\mathbf{0} \neq \boldsymbol{c} \in C$. What is the distance of a $[q-1, q-d]$ Reed-Solomon code? We know from lecture that its distance is $w(C)$. So let's compute $w(C)$.

Theorem 4.1.2. If C is a Reed-Solomon $[q-1, q-d]$-code, then $w(C) = d$. Thus, Reed-Solomon codes are MDS.

Proof. Let $w = w(C)$ and suppose that $w < d$. We do the case where $w = d-1$ as the other cases are similar. Let $\boldsymbol{c} = c_0 c_1 c_2 \dots c_{q-2} \in C$ be the nonzero vector witnessing this minimum (note that $n = q-1$). As $c(x) = g(x)f(x) \in \mathbb{F}[x]$ for some $f(x) \in \mathbb{F}[x; q-1]$ and $g(x)$ has $\alpha^{\theta}, \alpha^{\theta+1}, \dots, \alpha^{\theta+d-2}$ as its roots, we see that $\alpha^{\theta}, \alpha^{\theta+1}, \dots, \alpha^{\theta+d-2}$ are also roots of $c(x)$. Suppose the nonzero coordinates of $\boldsymbol{c}$ are $0 \leq j_1 < j_2 < \dots < j_w \leq q-2$. Then one has a system of $d-1$ equations:

$$\begin{aligned}
c_{j_1} \cdot (\alpha^{\theta})^{j_1} + \dots + c_{j_w} \cdot (\alpha^{\theta})^{j_w} &= c(\alpha^{\theta}) = 0. \\
c_{j_1} \cdot (\alpha^{\theta+1})^{j_1} + \dots + c_{j_w} \cdot (\alpha^{\theta+1})^{j_w} &= c(\alpha^{\theta+1}) = 0. \\
&\vdots \\
c_{j_1} \cdot (\alpha^{\theta+d-2})^{j_1} + \dots + c_{j_w} \cdot (\alpha^{\theta+d-2})^{j_w} &= c(\alpha^{\theta+d-2}) = 0.
\end{aligned}$$

We can write this more concisely in matrix form:

$$\begin{pmatrix}
(\alpha^{\theta})^{j_1} & (\alpha^{\theta})^{j_2} & \dots & (\alpha^{\theta})^{j_w} \\
(\alpha^{\theta+1})^{j_1} & (\alpha^{\theta+1})^{j_2} & \dots & (\alpha^{\theta+1})^{j_w} \\
\vdots & \vdots & \ddots & \vdots \\
(\alpha^{\theta+d-2})^{j_1} & (\alpha^{\theta+d-2})^{j_2} & \dots & (\alpha^{\theta+d-2})^{j_w}
\end{pmatrix}
\begin{pmatrix} c_{j_1} \\ c_{j_2} \\ \vdots \\ c_{j_w} \end{pmatrix}
= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{6}$$

Since $w = d-1$, this is a square matrix. We have that

$$\det \begin{pmatrix}
(\alpha^{\theta})^{j_1} & (\alpha^{\theta})^{j_2} & \dots & (\alpha^{\theta})^{j_w} \\
(\alpha^{\theta+1})^{j_1} & (\alpha^{\theta+1})^{j_2} & \dots & (\alpha^{\theta+1})^{j_w} \\
\vdots & \vdots & \ddots & \vdots \\
(\alpha^{\theta+d-2})^{j_1} & (\alpha^{\theta+d-2})^{j_2} & \dots & (\alpha^{\theta+d-2})^{j_w}
\end{pmatrix}
= \alpha^{\theta(j_1+\dots+j_w)} \det \begin{pmatrix}
1 & 1 & \dots & 1 \\
\alpha^{j_1} & \alpha^{j_2} & \dots & \alpha^{j_w} \\
\vdots & \vdots & \ddots & \vdots \\
(\alpha^{d-2})^{j_1} & (\alpha^{d-2})^{j_2} & \dots & (\alpha^{d-2})^{j_w}
\end{pmatrix}.$$

As α was chosen to be a primitive element, $\alpha^i \neq \alpha^j$ for all $0 \leq i \neq j \leq d-2$. Thus the determinant of the Vandemonde matrix in (6) is nonzero, forcing $c(x)$ to be a zero polynomial, a contradiction.

The general case where $w < d$ can be handled by choosing a submatrix in (6).

Hence, we have shown that $w \geq d$. The Singleton Bound (Theorem 2.2.2) gives $q - d + w \leq q$. So $w \leq d$. $w = d$, as desired. □

Remark 4.1.3. In fact, the class of Reed-Solomon codes is a special case of a more general class of codes known collectively as the ***Bose-Chaudhuri-Hocquenghem (BCH)*** codes. The following is true: if C is a non-trivial BCH code, then $d(C) = w(C) \geq d$. Among the Reed-Solomon codes, RS$(255, 223)$ is the most popular. It has distance 33, and so can correct up to 16 errors.

Reed and Solomon's original approach to constructing their codes fell out of favor following the discovery of the generator polynomial approach. This is because the latter approach admits much faster decoding algorithms. However, drawing comparisons between the original approach and the generator polynomial approach is outside the scope of this report.

5 Encoding Algorithms

5.1 Encoding Algorithms for Cyclic Codes

Reed-Solomon codes (and more generally cyclic codes) have a relatively simple encoding scheme. Let C be a RS$(q-1, q-d)$ Reed-Solomon code over a finite field $\mathbb{F}_q$ with generator polynomial $g(x)$ (necessarily of degree $d-1$). Let its dimension be $k = q-d$. A message $\boldsymbol{u} = (u_0, u_1, \ldots, u_{q-k-1}) \in \mathbb{F}_q^{q-k}$ can be regarded as a polynomial $u(x) = u_0 + u_1 x + \cdots + u_{q-k-1}x^{q-k-1}$. We want to assign $u(x)$ with a polynomial in C. First, divide $x^{k-1}u(x)$ by $g(x)$ to get the remainder polynomial $r(x) = r_0 + r_1 x + \cdots + r_{d-2}x^{d-2}$. Thus, we get $x^{k-1}u(x) = q(x)g(x) + r(x)$ for some polynomial $q(x)$.

We can then encode $u(x)$ to the polynomial $x^{k-1}u(x) - r(x)$, which will be a polynomial in C. The vector form of the codeword is

$$(-r_0, -r_1, \ldots, -r_{d-2}, u_0, u_1, u_2, \ldots, u_{q-k-1}),$$

where the message is part of the codeword. It is clear that given such a codeword, we can easily recover the message. Thus, the problem becomes: if $\boldsymbol{c} \in C$ is sent and $\boldsymbol{y} \notin C$ is received, how can one recover the $\boldsymbol{c} \in C$ that was sent? This is where decoding algorithms come in.

6 Decoding Algorithms

Decoding algorithms play a crucial role in the analysis of Reed-Solomon codes. Let $C = \text{RS}(q-1, q-d)$ denote a Reed-Solomon code. Theoretically, any error of weight $\ell \leq \lfloor \frac{d-1}{2} \rfloor$ can be corrected by using minimum distance decoding, but this requires computing the Hamming distance between q^{q-d} errors, which is computationally expensive. Therefore, we seek more efficient methods. In this section, we present a basic decoding algorithm for Reed-Solomon codes. This algorithm is not the most optimal one – there are further enhancements that make the generator polynomial approach more preferable.

6.1 Syndrome Based Decoding

Let $C = \text{RS}(q-1, q-d)$ be a Reed-Solomon code, with all the notation (including θ) from Section 5. By Theorem 4.1.2, $d_H(C) = d$. Let $m = \lfloor \frac{d-1}{2} \rfloor$; then C is m-error-correcting. Suppose that a codeword $c = c_0c_1c_2 \ldots c_{q-2} \in C$ is sent and a codeword $y = y_0y_1y_2 \ldots y_{q-2}$ is received. Assume that there are no more than m errors incurred during the transmission. Write $y = c + e$, where $e = e_0e_1e_2 \ldots e_{q-2}$ are the error-bits incurred during the transmission (so that $w(e) \leq m$). Let $w(e) = \ell \leq m$ and let α be the primitive element used in the Reed-Solomon code. For each $\alpha^{\theta+i} \in \{\alpha^\theta, \alpha^{\theta+1}, \ldots, \alpha^{\theta+d-2}\}$, one has

$$y(\alpha^{\theta+i}) = c(\alpha^{\theta+i}) + e(\alpha^{\theta+i})$$

We know from Section 5 that $\alpha^\theta, \alpha^{\theta+1}, \ldots, \alpha^{\theta+d-2}$ are roots of $c(x)$. Thus we are left with:

$$y(\alpha^{\theta+i}) = e(\alpha^{\theta+i}).$$

We define the ***syndromes*** S_j, for $0 \leq j \leq d-2$, to be:

$$S_j := y(\alpha^{\theta+j}).$$

Suppose that we know the syndromes, and say the nonzero coordinates are in positions $0 \le i_1 < i_2 < \cdots < i_\ell \le q-2$. Then the error polynomial can be written as

$$e(x) = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \cdots + e_{i_\ell} x^{i_\ell},$$

where e_{i_1} is the value of the first error, e_{i_2} the value of the second, and so on. We thus have, for each $0 \le j \le d-2$,

$$S_j = y(\alpha^{\theta+j}) = e(\alpha^{\theta+j}) = e_{i_1}\alpha^{i_1\cdot(\theta+j)} + e_{i_2}\alpha^{i_2\cdot(\theta+j)} + \cdots + e_{i_\ell}\alpha^{i_\ell\cdot(\theta+j)}. \tag{7}$$

For $1 \le k \le \ell$, define $X_k = \alpha^{i_k}$ and $Y_k = e_{i_k}$. The X_k's and Y_k's are called the ***error location numbers*** and the ***error location values*** respectively.

Notation. For notational clarity, the index 'j' will run from 0 to $d-2$ and they will be used to index the $d-1$ syndromes. The index 'k' will run from 1 to ℓ and will index the ℓ many error location numbers and values. This will ***always*** be the case – we will not use j to index the error location numbers, and vice versa. Hence, (7) reduces to:

$$\begin{aligned} S_0 &= Y_1 X_1^{\theta} + Y_2 X_2^{\theta} + \cdots + Y_\ell X_\ell^{\theta} \\ S_1 &= Y_1 X_1^{\theta+1} + Y_2 X_2^{\theta+1} + \cdots + Y_\ell X_\ell^{\theta+1} \\ &\vdots \\ S_{d-2} &= Y_1 X_1^{\theta+d-2} + Y_2 X_2^{\theta+d-2} + \cdots + Y_\ell X_\ell^{\theta+d-2}. \end{aligned} \tag{8}$$

We know what S_j and θ are; but we do not know the X_k's and Y_k's. Hence our next goal is to solve for the X_k's and Y_k's. The issue is that this problem is ***non-linear***, so linear algebra tools cannot be directly applied to solve the above system of equations.

Define the ***error locator polynomial*** as

$$\Lambda(x) = \prod_{k=1}^{\ell} (1 - xX_k) = 1 + \Lambda_1 x + \cdots + \Lambda_\ell x^\ell \in \mathbb{F}_q[x]. \tag{9}$$

The roots of $\Lambda(x)$ are X_k^{-1}. If we are able to find the coefficients of $\Lambda(x)$, then we can evaluate $\Lambda(x)$ at each element of our finite field to find its roots and hence X_k. Such a method is feasible because polynomial evaluation is fast and our field is finite. There are also more computationally efficient algorithms such as Forney's algorithm which utilizes the extended Euclidean algorithm in $\mathbb{F}_q[x]$ to find these roots.

Therefore, it suffices to focus our attention on finding out the values of $\Lambda_1, \ldots, \Lambda_\ell$. From Equation (9), we have, for each $1 \le k \le \ell$,

$$0 = \Lambda(X_k^{-1}) = 1 + \Lambda_1 X_k^{-1} + \cdots + \Lambda_\ell X_k^{-\ell}.$$

For a fixed k, multiplying both sides of the equation by $Y_k X_k^{j+\ell+\theta}$ for $0 \le j \le d-2$ yields

$$0 = Y_k X_k^{j+\ell+\theta} + \Lambda_1 Y_k X_k^{j+\ell+\theta-1} + \cdots + \Lambda_\ell Y_k X_k^{j+\theta}.$$

Such an equation holds for all j and k. Summing these equations for $k = 1, \ldots, \ell$ gives:

$$0 = \sum_{k=1}^{\ell} Y_k X_k^{j+\ell+\theta} + \Lambda_1 \sum_{k=1}^{\ell} Y_k X_k^{j+\ell+\theta-1} + \cdots + \Lambda_\ell \sum_{k=1}^{\ell} Y_k X_k^{j+\theta}.$$

From Equation (7), as long as $0 \le j, j+1, \ldots, j+\ell \le d-2$, one has,

$$-S_{j+\ell} = \Lambda_1 S_{j+\ell-1} + \Lambda_2 S_{j+\ell-2} + \cdots + \Lambda_\ell S_j.$$

As $2\ell \leq d-1$, this yields a system of ℓ equations. We write them out explicitly:

$$\begin{aligned}
\Lambda_1 S_{\ell-1} + \Lambda_2 S_{\ell-2} + \cdots + \Lambda_\ell S_0 &= -S_\ell \\
\Lambda_1 S_\ell + \Lambda_2 S_{\ell-1} + \cdots + \Lambda_\ell S_1 &= -S_{\ell+1} \\
&\vdots \\
\Lambda_1 S_{2\ell-2} + \Lambda_2 S_{2\ell-3} + \cdots + \Lambda_\ell S_{\ell-1} &= -S_{2\ell-1}.
\end{aligned}$$

More concisely, we can write the equations in matrix form:

$$\begin{pmatrix} S_0 & S_1 & \cdots & S_{\ell-1} \\ S_1 & S_2 & \cdots & S_\ell \\ \vdots & \vdots & \ddots & \vdots \\ S_{\ell-1} & S_\ell & \cdots & S_{2\ell-2} \end{pmatrix} \begin{pmatrix} \Lambda_\ell \\ \Lambda_{\ell-1} \\ \vdots \\ \Lambda_1 \end{pmatrix} = \begin{pmatrix} -S_\ell \\ -S_{\ell+1} \\ \vdots \\ -S_{2\ell-1} \end{pmatrix}. \tag{10}$$

Let $\boldsymbol{S_\ell}$ denote the matrix:

$$\boldsymbol{S_\ell} := \begin{pmatrix} S_0 & S_1 & \cdots & S_{\ell-1} \\ S_1 & S_2 & \cdots & S_\ell \\ \vdots & \vdots & \ddots & \vdots \\ S_{\ell-1} & S_\ell & \cdots & S_{2\ell-2} \end{pmatrix}. \tag{11}$$

A naive approach to solve the Λ_k's is to simply find the inverse of S. A natural question arises: is S invertible? We also have another problem – we do not know the value of ℓ, the number of errors incurred.

To proceed, we will adopt the following algorithm. Set ℓ' initially to be m, the error-correcting capability of our Reed-Solomon code. Construct the matrix $\boldsymbol{S_{\ell'}}$ as above. As it turns out, the matrix $\boldsymbol{S_{\ell'}}$ is invertible if and only if $\ell \geq \ell'$. If $\boldsymbol{S_{\ell'}}$ is not invertible, decrement ℓ' by 1 and try again; otherwise find its inverse and solve for the Λ_k's. The correctness of this algorithm is due to the following theorem.

Theorem 6.1.1. The matrix $\boldsymbol{S_{\ell'}}$ is invertible if and only if $\ell \geq \ell'$, where ℓ is the number of errors incurred.

Proof. Firstly, suppose that $\ell \geq \ell'$. To show that $\boldsymbol{S_{\ell'}}$ is invertible, we show that it has linearly independent columns. Let $\lambda_1, \lambda_2, \cdots, \lambda_{\ell'}$ be scalars satisfying

$$\lambda_1 \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{\ell'-1} \end{pmatrix} + \lambda_2 \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_{\ell'} \end{pmatrix} + \cdots + \lambda_{\ell'} \begin{pmatrix} S_{\ell'-1} \\ S_{\ell'} \\ \vdots \\ S_{2\ell'-2} \end{pmatrix} = 0. \tag{12}$$

We aim to show that $\lambda_1 = \lambda_2 = \cdots = \lambda_{\ell'} = 0$. Let's focus on the first equation.

$$\lambda_1 S_0 + \lambda_2 S_1 + \cdots + \lambda_{\ell'} S_{\ell'-1} = 0.$$

The above equation implies that

$$\lambda_1 \sum_{k=1}^{\ell} Y_k X_k^\theta + \lambda_2 \sum_{k=1}^{\ell} Y_k X_k^{\theta+1} + \cdots + \lambda_\ell \sum_{k=1}^{\ell} Y_k X_k^{\theta+\ell'-1} = 0,$$

where the summation runs from 1 to ℓ since ℓ is the true number of errors. Equivalently,

$$\sum_{j=0}^{\ell'-1} \sum_{k=1}^{\ell} \lambda_{j+1} Y_k X_k^{\theta+j} = 0.$$

Switching the order of the summation, we obtain:

$$\sum_{k=1}^{\ell}\sum_{j=0}^{\ell'-1}\lambda_{j+1}Y_kX_k^{\theta+j}=0.$$

Let

$$p(x)=\sum_{j=0}^{\ell'-1}\lambda_{j+1}x^{\theta+j} \tag{13}$$

so that $p(X_k)=\sum_{j=0}^{\ell'-1}\lambda_{j+1}X_k^{\theta+j}$. Then we have

$$\sum_{k=1}^{\ell}Y_kp(X_k)=0.$$

Using the same technique for the other equations, we end up with a system of ℓ equations:

$$\sum_{k=1}^{\ell}X_k^jY_kp(X_k)=0.$$

Writing it more compactly in matrix form,

$$\begin{pmatrix}1 & 1 & \cdots & 1\\ X_1 & X_2 & \cdots & X_\ell\\ \vdots & \vdots & \ddots & \vdots\\ X_1^{\ell-1} & X_2^{\ell-1} & \cdots & X_\ell^{\ell-1}\end{pmatrix}\begin{pmatrix}Y_1p(X_1)\\ Y_2p(X_2)\\ \vdots\\ Y_\ell p(X_\ell)\end{pmatrix}=\begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix}. \tag{14}$$

Now, the X_j's are all distinct (since α was chosen as a primitive element), so the Vandemonde matrix is invertible. The unique solution is hence $Y_1p(X_1)=Y_2p(X_2)=\cdots=Y_\ell p(X_\ell)=0$. As the Y_i's are the error values, $Y_1,\cdots,Y_\ell$ are nonzero, so $p(X_1)=p(X_2)=\cdots=p(X_\ell)=0$. If p is not the zero polynomial, then p is a polynomial of degree at most $\theta+\ell'-1$ with 0 a root of multiplicity θ. Hence it can have at most $\ell'-1$ distinct nonzero roots. Yet $p(X_1)=p(X_2)=\cdots=p(X_\ell)=0$. This is a contradiction since $\ell\geq\ell'>\ell'-1$. Thus p must be identically zero. Therefore, $\lambda_1=\lambda_2=\cdots=\lambda_{\ell'}=0$, as desired.

Conversely, suppose that $\ell'>\ell$. To show that $S_{\ell'}$ is not invertible, we proceed by going backwards. We want to show the existence of a nonzero polynomial of degree at most $\theta+\ell'-1$ satisfying (14) and hence (12). Let $p(x)=x^\theta(x-X_1)(x-X_2)\cdots(x-X_\ell)$, which is a nonzero polynomial since not all elements of $\mathbb{F}_q$ are roots of it. Then $\deg(p)=\ell+\theta\leq\ell'+\theta-1$. Such a choice of p solves the linear system (14) and hence (12). Writing p in the form as in Equation (13), we see that $\lambda_1,\cdots,\lambda_\ell$ satisfies (12) and so the first $\ell+1$ rows of $\boldsymbol{S_{\ell'}}$ are linearly dependent as desired. □

7 Putting it Together: An Example

The previous section is a little abstract; let's bring it down to earth by giving an example.

Suppose $\mathbb{F}=\mathbb{F}_{13}$, $\alpha=2$ (which is primitive), $g(x)=\prod_{i=1}^{5}(x-\alpha^i)$ (so $\theta=1$) with $\deg(g(x))=5$, $d=6,k=7$ (the distance of the code is 6 and the dimension of the code is 7) and $m=2$ (the code is 2-error correcting). Consider the Reed-Solomon code $\mathrm{RS}(q-1,q-d)=\mathrm{RS}(12,6)$, which has dimension k. According to Section 5, our message length should be $q-k=6$. Suppose we want to send the message $[3,1,4,1,5,9]$. Using the notation from Section 5,

$$\begin{aligned}u(x)&=9x^5+5x^4+x^3+4x^2+x+3.\\ r(x)&=11x^4+2x^3+4x+1.\\ c(x)&=9x^{11}+5x^{10}+x^9+4x^8+x^7+3x^6+2x^4+11x^3+9x+12.\end{aligned}$$

Thus, the message will get encoded to the code: $[12, 9, 0, 11, 2, 0, 3, 1, 4, 1, 5, 9] \in \mathbb{F}_{13}^{12}$.

Now, let's consider the scenario where one error is incurred during the transmission. Say the error pattern is $e = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5]$ (so $\ell = 1$). We can compute $y(x) = c(x) + e(x)$, which will be

$$y(x) = x^{11} + 5x^{10} + x^9 + 4x^8 + x^7 + 3x^6 + 2x^4 + 11x^3 + 9x + 12.$$

Now, supposing we don't know ℓ, the error locations and error values. With the algorithm in Section 6, we compute the syndromes, which are:

$$\{S_0, S_1, \ldots, S_{d-2}\} = [9, 11, 12, 6, 3].$$

We then compute $S_{\ell'}$ for $\ell' = 2, 1$, and see that for $\ell' = 2$,

$$S_{\ell'} = \begin{pmatrix} 9 & 11 \\ 11 & 12 \end{pmatrix},$$

which is invertible. For $\ell' = 1$,

$$S_{\ell'} = \begin{pmatrix} 9 \end{pmatrix}.$$

Thus we deduce that the number of errors $= \ell = 1$.

Using S_ℓ, we solve Equation (10) to get that $\lambda_1 = 6$ and the error location polynomial, $\Lambda(x) = 6x + 1$. Normally, we would check every element in $\mathbb{F}_{13}$ for a root, but in this case it is clear that 2 is the only root of $\Lambda(x)$. This means that $X_1 = 2^{-1} = 7$.

From X_1, we can find the error locations as $X_k = \alpha^{i_k}$, where i_k is the location of the k-error. As $2^{11} = 7 \pmod{13}$, the first (and only) error occurs at index 11.

Now, we can plug in the values of X_1 into Equation (7) to solve for Y_1. In general, there are more equations than variables, so we only need to pick ℓ equations. In this case, we choose the first equation, meaning that we have to solve $7Y_1 = 5 \pmod{13}$. Thus, $Y_1 = 5$.

Thus, the errors are at index 11 with error value 5, as desired.

8 Code

We implemented the decoding and encoding algorithm in Sections and 5 and 6 in SageMath 9.3, which can be found here. It only handles the simple case of $\theta = 1$. The parameters are set up as in the example presented in Section 7. Do note that the variable `alpha` depends on `p` in the code, so if one changes `p`, one should change `alpha` as well.

9 Contribution Report

Both of us did the report together. The work for each section was uniformly distributed among the two of us.

10 Conclusion

Reed-Solomon codes are a powerful and widely used tool for error correction in digital communication systems. They are also efficient in the sense that a relatively small amount of redundant information needs to be added to achieve a given level of error correction, as they achieve the Singleton Bound. The decoding algorithm given in Section 6 may seem inefficient as we are computing the determinant of many matrices. However, the ideas in Section 6 are merely a start; in practice there are more algorithms using the extended Euclidean algorithm to work out the error-location polynomial, which is much more efficient than the methods presented.

11 References

Reed, I. S., & Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, *8*(2), 300–304.

Wicker, S. B., & Bhargava, V. K. (1999). *Reed-solomon codes and their applications*. John Wiley & Sons.