

Algebraic Decoding of Reed-Solomon and BCH Codes

Henry D. Pfister

July, 2008 (rev. 0)

November 8th, 2012 (rev. 1)

November 15th, 2013 (rev. 2)

1 Reed-Solomon and BCH Codes

1.1 Introduction

Consider the (n, k) cyclic code over \mathbb{F}_{q^m} with generator polynomial

$$g(x) = \prod_{j=1}^{n-k} (x - \alpha^{a+jb}),$$

where $\alpha \in \mathbb{F}_{q^m}$ is an element of order n and $\gcd(b, n) = 1$. This is a cyclic Reed-Solomon (RS) code with $d_{\min} = n - k + 1$. Adding roots at all the conjugates of $\{\alpha^{a+b}, \alpha^{a+2b}, \dots, \alpha^{a+(n-k)b}\}$ (w.r.t. the subfield $K = \mathbb{F}_q$) also allows one to define a length- n BCH subcode over \mathbb{F}_q with $d_{\min} \geq n - k + 1$. Also, any decoder that corrects all patterns of up to $t = \lfloor (n - k)/2 \rfloor$ errors for the original RS code can be used to correct the same set of errors for any subcode.

The RS code operates by encoding a message polynomial $m(x) = \sum_{j=0}^{k-1} m_j x^j$ of degree at most $k - 1$ into a codeword polynomial $c(x) = \sum_{j=0}^{n-1} c_j x^j$ using

$$c(x) = m(x)g(x).$$

During transmission, some additive errors are introduced and described by the error polynomial $e(x) = \sum_{j=0}^{n-1} e_j x^j$. Let t denote the number of errors so that $e(x) = \sum_{j=1}^t e_{\sigma(j)} x^{\sigma(j)}$. The received polynomial is $r(x) = \sum_{j=0}^{n-1} r_j x^j$ where

$$r(x) = c(x) + e(x).$$

In these notes, the RS decoding is introduced using the methods of Peterson-Gorenstein-Zierler (PGZ), the Berlekamp-Massey Algorithm (BMA) and the Euclidean method of Sugiyama. All of these are based on the idea of an error-locator polynomial, whose mathematical roots can be traced back to a method of fitting data to exponentials introduced by the engineer Baron Gaspard De Prony in 1796 [1]. The first decoder computes this polynomial using direct matrix inversion, the second is based on linear feedback shift registers (LFSRs), and the third uses the extended Euclidean algorithm (EEA) for polynomials.

1.2 Syndromes and Error Locators

Suppose there are ν errors. Then, the (frequency domain) *syndrome sequence* is defined, for $j = 1, 2, \dots, n - k$, by

$$S_j \triangleq r(\alpha^{a+jb}) = \underbrace{c(\alpha^{a+jb})}_0 + e(\alpha^{a+jb}) = \sum_{i=0}^{n-1} e_i \alpha^a \alpha^{ijb} = \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a (\beta^{\sigma(i)})^j,$$

where $\beta = \alpha^b$ has order n as well because $\gcd(b, n) = 1$. For a BCH code over \mathbb{F}_q , we observe that $r(x) \in \mathbb{F}_q$ implies that

$$[S_j]^q = [r(\alpha^{a+jb})]^q = \left[\sum_{i=0}^{n-1} r_i \alpha^{a+jb} \right]^q = \sum_{i=0}^{n-1} r_i^q \alpha^{qa+qjb} = \alpha^{(q-1)a} \sum_{i=0}^{n-1} r_i \alpha^{a+qjb} = \alpha^{(q-1)a} S_{qj}.$$

For example, this implies that $S_{2j} = S_j^2$ when $q = 2$. The *syndrome polynomial* is defined to be

$$S(x) \triangleq \sum_{j=1}^{2t} S_j x^{j-1}.$$

1.3 Examples

The following examples were taken from [2, pp. 184-185] and use the Galois field $\mathbb{F}_{16} = \mathbb{F}_2[\alpha]/\langle\alpha^4 + \alpha + 1\rangle$, where $\alpha^4 + \alpha + 1$ is a primitive polynomial. Using the power representation α^i for $i = 0, 1, \dots, 14$, multiplication is easy. To perform addition, we use the following addition table, which gives the value of $\alpha^i + \alpha^j$ for $i, j \in \{0, 1, \dots, 14\}$.

	1	α	α^2	α^3	α^4	α^5	α^6	α^7	α^8	α^9	α^{10}	α^{11}	α^{12}	α^{13}	α^{14}
1	0	α^4	α^8	α^{14}	α	α^{10}	α^{13}	α^9	α^2	α^7	α^5	α^{12}	α^{11}	α^6	α^3
α	α^4	0	α^5	α^9	1	α^2	α^{11}	α^{14}	α^{10}	α^3	α^8	α^6	α^{13}	α^{12}	α^7
α^2	α^8	α^5	0	α^6	α^{10}	α	α^3	α^{12}	1	α^{11}	α^4	α^9	α^7	α^{14}	α^{13}
α^3	α^{14}	α^9	α^6	0	α^7	α^{11}	α^2	α^4	α^{13}	α	α^{12}	α^5	α^{10}	α^8	1
α^4	α	1	α^{10}	α^7	0	α^8	α^{12}	α^3	α^5	α^{14}	α^2	α^{13}	α^6	α^{11}	α^9
α^5	α^{10}	α^2	α	α^{11}	α^8	0	α^9	α^{13}	α^4	α^6	1	α^3	α^{14}	α^7	α^{12}
α^6	α^{13}	α^{11}	α^3	α^2	α^{12}	α^9	0	α^{10}	α^{14}	α^5	α^7	α	α^4	1	α^8
α^7	α^9	α^{14}	α^{12}	α^4	α^3	α^{13}	α^{10}	0	α^{11}	1	α^6	α^8	α^2	α^5	α
α^8	α^2	α^{10}	1	α^{13}	α^5	α^4	α^{14}	α^{11}	0	α^{12}	α	α^7	α^9	α^3	α^6
α^9	α^7	α^3	α^{11}	α	α^{14}	α^6	α^5	1	α^{12}	0	α^{13}	α^2	α^8	α^{10}	α^4
α^{10}	α^5	α^8	α^4	α^{12}	α^2	1	α^7	α^6	α	α^{13}	0	α^{14}	α^3	α^9	α^{11}
α^{11}	α^{12}	α^6	α^9	α^5	α^{13}	α^3	α	α^8	α^7	α^2	α^{14}	0	1	α^4	α^{10}
α^{12}	α^{11}	α^{13}	α^7	α^{10}	α^6	α^{14}	α^4	α^2	α^9	α^8	α^3	1	0	α	α^5
α^{13}	α^6	α^{12}	α^{14}	α^8	α^{11}	α^7	1	α^5	α^3	α^{10}	α^9	α^4	α	0	α^2
α^{14}	α^3	α^7	α^{13}	1	α^9	α^{12}	α^8	α	α^6	α^4	α^{11}	α^{10}	α^5	α^2	0

Example 1. Consider the $(15, 7, 5)$ binary primitive narrow-sense BCH code with generator polynomial $g(x) = x^8 + x^7 + x^6 + x^4 + 1$, which has zeros at α^i for $i \in \{1, 2, 3, 4, 6, 8, 9, 12\}$. If the received sequence is $r(x) = x^{10} + x^9 + x^6 + x^5 + x + 1$, then the syndromes are given by

$$\begin{aligned} S_1 &= r(\alpha) = \alpha^{10} + \alpha^9 + \alpha^6 + \alpha^5 + \alpha + 1 = \alpha^2 \\ S_2 &= r(\alpha^2) = S_1^2 = \alpha^4 \\ S_3 &= r(\alpha^3) = 1 + \alpha^{12} + \alpha^3 + 1 + \alpha^3 + 1 = \alpha^{11} \\ S_4 &= r(\alpha^4) = S_2^2 = \alpha^8. \end{aligned}$$

Therefore, the syndrome polynomial is given by

$$S(x) = \alpha^8 x^3 + \alpha^{11} x^2 + \alpha^4 x + \alpha^2.$$

Example 2. Consider the $(15, 9, 7)$ primitive narrow-sense RS code over \mathbb{F}_{16} with generator polynomial $g(x) = x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6$, which has zeros at α^i for $i \in \{1, 2, 3, 4, 5, 6\}$. If the received sequence is

$$r(x) = \alpha^7 x^{11} + \alpha^4 x^7 + \alpha^4 x^6 + \alpha^5 x^5 + \alpha^2 x^4 + x^3 + \alpha^{10} x^2 + \alpha^7,$$

then the syndromes are given by

$$\begin{aligned} S_1 &= r(\alpha) = \alpha^7 \alpha^{11} + \alpha^4 \alpha^7 + \alpha^4 \alpha^6 + \alpha^5 \alpha^5 + \alpha^2 \alpha^4 + \alpha^3 + \alpha^{10} \alpha^2 + \alpha^7 = \alpha^5 \\ S_2 &= r(\alpha^2) = \alpha^7 \alpha^7 + \alpha^4 \alpha^{14} + \alpha^4 \alpha^{12} + \alpha^5 \alpha^{10} + \alpha^2 \alpha^8 + \alpha^6 + \alpha^{10} \alpha^4 + \alpha^7 = \alpha^7 \\ S_3 &= r(\alpha^3) = \alpha^7 \alpha^3 + \alpha^4 \alpha^6 + \alpha^4 \alpha^3 + \alpha^5 + \alpha^2 \alpha^{12} + \alpha^9 + \alpha^{10} \alpha^6 + \alpha^7 = \alpha^{10} \\ S_4 &= r(\alpha^4) = \alpha^7 \alpha^{14} + \alpha^4 \alpha^{13} + \alpha^4 \alpha^9 + \alpha^5 \alpha^5 + \alpha^2 \alpha + \alpha^{12} + \alpha^{10} \alpha^8 + \alpha^7 = \alpha^5 \\ S_5 &= r(\alpha^5) = \alpha^7 \alpha^{10} + \alpha^4 \alpha^5 + \alpha^4 + \alpha^5 \alpha^{10} + \alpha^2 \alpha^5 + \alpha^3 + \alpha^{10} \alpha^{10} + \alpha^7 = \alpha^7 \\ S_6 &= r(\alpha^6) = \alpha^7 \alpha^6 + \alpha^4 \alpha^{12} + \alpha^4 \alpha^6 + \alpha^5 + \alpha^2 \alpha^9 + \alpha^3 + \alpha^{10} \alpha^{12} + \alpha^7 = \alpha^3. \end{aligned}$$

Therefore, the syndrome polynomial is given by

$$S(x) = \alpha^3 x^5 + \alpha^7 x^4 + \alpha^5 x^3 + \alpha^{10} x^2 + \alpha^7 x + \alpha^5.$$

2 Algebraic Decoding

2.1 Peterson-Gorenstein-Zierler Decoding

2.1.1 The Error-Locator Polynomial

If there are ν errors, then the syndrome relationships (3) for $j = 1, 2, \dots, n - k$ provide $n - k$ equations involving the 2ν unknowns $\sigma(i)$ and $e_{\sigma(i)}$ for $i = 1, 2, \dots, \nu$. Since these equations are nonlinear, solving for these unknowns requires a clever trick. For $\nu \leq t \leq \lfloor \frac{n-k}{2} \rfloor$, let $\Lambda(x) \triangleq \sum_{j=0}^t \Lambda_j x^j$ be any degree- t polynomial that satisfies $\Lambda(0) = 1$ and $\Lambda(\beta^{-\sigma(i)}) = 0$ for $i = 1, \dots, \nu$. Then, the coefficients of $\Lambda(x)$ have a linear relationship with the syndromes. This can be seen by summing the equation

$$0 = \Lambda(\beta^{-\sigma(i)}) = \Lambda_t (\beta^{-\sigma(i)})^t + \Lambda_{t-1} (\beta^{-\sigma(i)})^{t-1} + \dots + \Lambda_1 (\beta^{-\sigma(i)}) + \Lambda_0$$

for $i = 1, \dots, \nu$ with the coefficients $e_{\sigma(i)} \alpha^a (\beta^{\sigma(i)})^k$. For $k = t + 1, t + 2, \dots, 2t$, this gives

$$\begin{aligned} 0 &= \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a (\beta^{\sigma(i)})^k \Lambda(\beta^{-\sigma(i)}) \\ &= \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a (\beta^{\sigma(i)})^k \sum_{j=0}^t \Lambda_j (\beta^{-\sigma(i)})^j \\ &= \sum_{j=0}^t \Lambda_j \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a (\beta^{\sigma(i)})^{k-j} \\ &= \sum_{j=0}^t \Lambda_j S_{k-j}. \end{aligned} \tag{1}$$

The derivation implies that any polynomial $\Lambda(x)$ with constant term 1 and roots at $\beta^{-\sigma(i)}$ (i.e., the inverse of α to the error location) for $i = 1, \dots, \nu$ must satisfy this equation. The minimal-degree polynomial $\Lambda(x)$ that satisfies these conditions is called the *error-locator polynomial*. It is easy to see that it must have one root at each location and is, therefore, the degree- ν polynomial defined by

$$\Lambda(x) \triangleq \prod_{i=1}^{\nu} (1 - x\beta^{\sigma(i)}).$$

This polynomial allows the error position to be revealed by factoring $\Lambda(x)$.

Since $\Lambda_0 = 1$, (1) defines the linear system

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_t & S_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t} \end{bmatrix}, \tag{2}$$

where the i -th row is given by the equation for $k = t + i$. If $t = \nu$, then this matrix will be invertible because there is a unique solution. If it is not invertible, one can sequentially reduce t by 1 until the matrix becomes invertible. After solving for the error-locator polynomial, one can evaluate it at all points in \mathbb{F}_q^* to determine the error locations. An efficient method of doing this is called a *Chien search*.

For binary BCH codes, the error magnitudes must be 1. After correcting the “errors”, one must also check that the resulting vector is a codeword by reducing it modulo $g(x)$. If it is not a codeword, then the decoder should declare a detected error. For non-binary codes, one can solve for the error

magnitudes using the error locations and the implied frequency-domain parity-check matrix. For fixed error locations, one can write the first ν syndromes as linear functions of the error magnitudes using

$$\begin{bmatrix} \alpha^a (\beta^{\sigma(1)})^1 & \alpha^a (\beta^{\sigma(2)})^1 & \dots & \alpha^a (\beta^{\sigma(\nu)})^1 \\ \alpha^a (\beta^{\sigma(1)})^2 & \alpha^a (\beta^{\sigma(2)})^2 & \dots & \alpha^a (\beta^{\sigma(\nu)})^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^a (\beta^{\sigma(1)})^\nu & \alpha^a (\beta^{\sigma(2)})^\nu & \dots & \alpha^a (\beta^{\sigma(\nu)})^\nu \end{bmatrix} \begin{bmatrix} e_{\sigma(1)} \\ e_{\sigma(2)} \\ \vdots \\ e_{\sigma(\nu)} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_\nu \end{bmatrix}.$$

This matrix is invertible because it is Vandermonde.

Example 3. Continuing Example 1, we use the PGZ method to compute the error-locator polynomial. We assume there are 2 errors¹ and write the matrix equation

$$\begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} \implies \begin{bmatrix} \alpha^2 & \alpha^4 \\ \alpha^4 & \alpha^{11} \end{bmatrix} \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} \alpha^{11} \\ \alpha^8 \end{bmatrix}.$$

Since the matrix is invertible, the error-locator polynomial of degree-2 is uniquely determined and one can find the solution $\Lambda_2 = \alpha^{14}$ and $\Lambda_1 = \alpha^2$. Therefore, the error-locator polynomial $\Lambda(x) = \alpha^{14}x^2 + \alpha^2x + 1$. Evaluating this polynomial at α^i for all $i \in \{0, 1, \dots, 14\}$ shows that

$$\begin{aligned} \Lambda(\alpha^5) &= \alpha^{14}\alpha^{10} + \alpha^2\alpha^5 + 1 = \alpha^9 + \alpha^7 + 1 = 0 \\ \Lambda(\alpha^{11}) &= \alpha^{14}\alpha^7 + \alpha^2\alpha^{11} + 1 = \alpha^6 + \alpha^{13} + 1 = 0. \end{aligned}$$

Since $\alpha^{-5} = \alpha^{10}$ and $\alpha^{-11} = \alpha^4$, this is consistent with errors in bit 4 and bit 10. To check this, we let $\hat{e}(x) = x^4 + x^{10}$ and compute

$$r(x) - e(x) \bmod g(x) = x^9 + x^6 + x^5 + x^4 + x + 1 \bmod g(x) = 0$$

because $(x+1)g(x) = x^9 + x^6 + x^5 + x^4 + x + 1$.

Example 4. Continuing Example 2, we use the PGZ method to compute the error-locator polynomial. We assume there are 3 errors and write the matrix equation

$$\begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} \begin{bmatrix} \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_4 \\ -S_5 \\ -S_6 \end{bmatrix} \implies \begin{bmatrix} \alpha^5 & \alpha^7 & \alpha^{10} \\ \alpha^7 & \alpha^{10} & \alpha^5 \\ \alpha^{10} & \alpha^5 & \alpha^7 \end{bmatrix} \begin{bmatrix} \Lambda_3 \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} \alpha^5 \\ \alpha^7 \\ \alpha^3 \end{bmatrix}.$$

Since the matrix is invertible, the error-locator polynomial of degree-3 is uniquely determined and one can find the solution $\Lambda_3 = \alpha^4$, $\Lambda_2 = \alpha^6$, and $\Lambda_1 = \alpha^5$. Therefore, the error-locator polynomial $\Lambda(x) = \alpha^4x^3 + \alpha^6x^2 + \alpha^5x + 1$. Evaluating this polynomial at α^i for all $i \in \{0, 1, \dots, 14\}$ shows that

$$\begin{aligned} \Lambda(\alpha^4) &= \alpha^4\alpha^{12} + \alpha^6\alpha^8 + \alpha^5\alpha^4 + 1 = \alpha + \alpha^{14} + \alpha^9 + 1 = 0 \\ \Lambda(\alpha^9) &= \alpha^4\alpha^{12} + \alpha^6\alpha^3 + \alpha^5\alpha^9 + 1 = \alpha + \alpha^9 + \alpha^{14} + 1 = 0 \\ \Lambda(\alpha^{13}) &= \alpha^4\alpha^9 + \alpha^6\alpha^{11} + \alpha^5\alpha^{13} + 1 = \alpha^{13} + \alpha^2 + \alpha^3 + 1 = 0. \end{aligned}$$

Since $\alpha^{-4} = \alpha^{11}$, $\alpha^{-9} = \alpha^6$, and $\alpha^{-13} = \alpha^2$, this is consistent errors in symbol positions 2, 6, and 11. To calculate the magnitudes, we write the matrix equation

$$\begin{bmatrix} \alpha^{\sigma(1)} & \alpha^{\sigma(2)} & \alpha^{\sigma(3)} \\ \alpha^{2\sigma(1)} & \alpha^{2\sigma(2)} & \alpha^{2\sigma(3)} \\ \alpha^{3\sigma(1)} & \alpha^{3\sigma(2)} & \alpha^{3\sigma(3)} \end{bmatrix} \begin{bmatrix} e_{\sigma(1)} \\ e_{\sigma(2)} \\ e_{\sigma(3)} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \implies \begin{bmatrix} \alpha^2 & \alpha^6 & \alpha^{11} \\ \alpha^4 & \alpha^{12} & \alpha^7 \\ \alpha^6 & \alpha^3 & \alpha^3 \end{bmatrix} \begin{bmatrix} e_2 \\ e_6 \\ e_{11} \end{bmatrix} = \begin{bmatrix} \alpha^5 \\ \alpha^7 \\ \alpha^{10} \end{bmatrix}.$$

Solving this gives $e_2 = 1$, $e_6 = \alpha^3$, and $e_{11} = \alpha^7$. To check this, we let $\hat{e}(x) = x^2 + \alpha^3x^6 + \alpha^7x^{11}$ observe that

$$r(x) - \hat{e}(x) = \alpha^4x^7 + \alpha^7x^6 + \alpha^5x^5 + \alpha^2x^4 + x^3 + \alpha^5x^2 + \alpha^7 = (\alpha + \alpha^4x)g(x).$$

¹If this leads to a valid codeword, then there is no codeword at distance 1. If not, one should repeat this process assuming there is one error.

2.2 The Berlekamp-Massey Decoding Algorithm

While the PGZ algorithm is conceptually simple, it can require the inversion of an $i \times i$ matrix for $i = 1, 2, \dots, t$ in the worst case. Since each inversion has a complexity of roughly $i^3/2$ operations, this approach leads to a worst case complexity of roughly $t^4/6$ operations. The Berlekamp-Massey algorithm starts with the observation that (1) can be rewritten, for $j = t + 1, t + 2, \dots, n - k$, as

$$S_j = - \sum_{i=1}^t \Lambda_i S_{j-i}.$$

This implies that the syndrome sequence S_1, S_2, \dots can be generated by a linear feedback shift register (LFSR) with coefficients $\Lambda_1, \Lambda_2, \dots, \Lambda_t$. The shortest LFSR that generates the syndrome sequence is unique and corresponds to $t = \nu$ and gives the error-locator polynomial.

The trick is to solve recursively for a sequence of LFSRs that generate the initial part of the syndrome sequence. Let the connection polynomial² of a length- L_k minimal length LFSR that generates the first k elements of the syndrome be

$$\Lambda^{[k]}(x) = \sum_{i=0}^{L_k} \Lambda_i^{[k]} x^i.$$

To be precise, we say that $\Lambda^{[k]}(x)$ generates the first k elements if

$$S_j = - \sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{j-i},$$

for $j = L_k + 1, \dots, k$. In particular, the shift register is initialized to contain the first L_k elements, S_1, S_2, \dots, S_{L_k} . Then, the j -th clock outputs S_j and computes S_{j+L_k} from S_1, S_2, \dots, S_{L_k} . This also introduces a subtle distinction between L_k and the degree of $\Lambda^{[k]}(x)$. If $L_k > \deg(\Lambda^{[k]}(x))$, then there is an LFSR of length $\deg(\Lambda^{[k]}(x))$ that generates S_j, S_{j+1}, \dots, S_k for $j = L_k - \deg(\Lambda^{[k]}(x)) + 1$ but it cannot generate S_1, \dots, S_k .

Starting from $\Lambda^{[-1]}(x) = \Lambda^{[0]}(x) = 1$, $\Delta_{-1} = \Delta_0 = 1$, and $L_{-1} = L_0 = 0$, we write

$$\hat{S}_k = - \sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}.$$

If $\hat{S}_k = S_k$, then the current LFSR generates the next syndrome and we can choose $\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x)$ with $L_k = L_{k-1}$. If this does not occur, then there is a non-zero discrepancy

$$\Delta_k = S_k - \hat{S}_k = S_k + \sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i} = \sum_{i=0}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}.$$

Let $m = \min \{i \in \{0, 1, \dots, k-1\} \mid L_i = L_{k-1}\}$ be the iteration index before the last length change. Then, we can update the connection polynomial to be

$$\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x) - \Delta_k \Delta_m^{-1} x^{k-m} \Lambda^{[m-1]}(x),$$

where $L_k = \max(L_{k-1}, L_{m-1} + k - m)$ is not necessarily equal to $\deg(\Lambda^{[k]}(x))$ for the reason mentioned earlier. This implies that

$$\begin{aligned} \sum_{i=0}^{L_k} \Lambda_i^{[k]} S_{k-i} &= \sum_{i=0}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i} - \Delta_k \Delta_m^{-1} \sum_{i=k-m}^{L_{m-1}+k-m} \Lambda_{i-(k-m)}^{[m-1]} S_{k-i} \\ &= \underbrace{\sum_{i=0}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}}_{\Delta_k} - \Delta_k \Delta_m^{-1} \underbrace{\sum_{j=0}^{L_{m-1}} \Lambda_j^{[m-1]} S_{k-j}}_{\Delta_m} = 0. \end{aligned}$$

Therefore, the updated polynomial correctly predicts the k -th syndrome. The proof that this update produces the shortest LFSR is too lengthy for these notes [3, Sec. 6.4.3] [4, Ch. 6].

²For $k < 2\nu$, this may not be unique.

Algorithm 1 The Berlekamp-Massey Algorithm

```

1:  $\Lambda^{[-1]}(x) \leftarrow 1, \Lambda^{[0]}(x) \leftarrow 1$ 
2:  $L_{-1} \leftarrow 0, L_0 \leftarrow 0$ 
3:  $\Delta_1 = 1$ 
4: for  $k \leftarrow 1$  to  $2t$  do
5:    $\Delta_k \leftarrow \sum_{i=0}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}$ 
6:    $m \leftarrow \min \{i \in \{0, 1, \dots, k-1\} \mid L_i = L_{k-1}\}$ 
7:   if  $\Delta_k = 0$  then
8:      $\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x)$ 
9:      $L_{k-1} = L_k$ 
10:  else
11:     $\Lambda^{[k]}(x) \leftarrow \Lambda^{[k-1]}(x) - \Delta_k \Delta_m^{-1} x^{k-m} \Lambda^{[m-1]}(x)$ 
12:     $L_k \leftarrow \max(L_{k-1}, L_{m-1} + k - m)$ 
13:  end if
14: end for
15:  $\Lambda(x) \leftarrow \Lambda^{[2t]}(x)$ 

```

Example 5. Continuing Example 1, we use the Berlekamp-Massey algorithm to compute the error-locator polynomial and observe that it matches the result of the PGZ method.

k	S_k	\hat{S}_k	Δ_k	$\Lambda^{[k]}(x)$	L_k	m
0			1	1	0	
1	α^2	0	α^2	$1 + \alpha^2(1)x$	1	0
2	α^4	$\alpha^2(\alpha^2) = \alpha^4$	0	$1 + \alpha^2x$	1	1
3	α^{11}	$\alpha^2(\alpha^4) = \alpha^6$	α	$(1 + \alpha^2x) + \alpha(\alpha^{-2})x^2$	2	1
4	α^8	$\alpha^2(\alpha^{11}) + \alpha^{14}(\alpha^4) = \alpha^8$	0	$1 + \alpha^2x + \alpha^{14}x^2$	2	3

Example 6. Continuing Example 2, we use the Berlekamp-Massey algorithm to compute the error-locator polynomial and observe that it matches the result of the PGZ method.

k	S_k	\hat{S}_k	Δ_k	$\Lambda^{[k]}(x)$	L_k	m
0			1	1	0	
1	α^5		α^5	$1 + \alpha^5(1)x$	1	0
2	α^7	$\alpha^5(\alpha^5) = \alpha^{10}$	α^6	$(1 + \alpha^5x) + \alpha^6(\alpha^{-5})x = 1 + \alpha^2x$	1	1
3	α^{10}	$\alpha^2(\alpha^7) = \alpha^9$	α^{13}	$(1 + \alpha^2x) + \alpha^{13}(\alpha^{-5})x^2 = 1 + \alpha^2x + \alpha^8x^2$	2	1
4	α^5	$\alpha^2(\alpha^{10}) + \alpha^8(\alpha^7) = \alpha^{14}$	α^3	$1 + \alpha^2x + \alpha^8x^2 + \alpha^3(\alpha^{-13})x(1 + \alpha^2x) =$ $1 + \alpha x + \alpha^{11}x^2$	2	3
5	α^7	$\alpha(\alpha^5) - \alpha^{11}(\alpha^{10}) = 0$	α^7	$1 + \alpha x + \alpha^{11}x^2 + \alpha^7(\alpha^{-13})x^2(1 + \alpha^2x) =$ $1 + \alpha x + \alpha^2x^2 + \alpha^{11}x^3$	3	3
6	α^3	$\alpha(\alpha^7) - \alpha^2(\alpha^5) - \alpha^{11}(\alpha^{10}) = \alpha^2$	α^9	$1 + \alpha x + \alpha^2x^2 + \alpha^{11}x^3 + \alpha^9(\alpha^{-7})x(1 + \alpha x + \alpha^{11}x^2) =$ $1 + \alpha^5x + \alpha^6x^2 + \alpha^4x^3$	3	5

2.3 Sugiyama's Euclidean Decoding Algorithm

2.3.1 The Key Equation and the Error-Evaluator Polynomial

An alternative approach is to use the Euclidean algorithm to find the error-locator polynomial. To describe this, the syndrome is first extended to a semi-infinite sequence (S_1, S_2, \dots) by defining

$$S_j = \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a \left(\beta^{\sigma(i)} \right)^j, \quad (3)$$

for $j \in \{1, 2, \dots\}$ and noting that the two definitions coincide for $j = 1, 2, \dots, n - k$. The *extended syndrome function* $\tilde{S}(x)$ is defined to be

$$\begin{aligned}\tilde{S}(x) &\triangleq \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a \frac{\beta^{\sigma(i)}}{1 - x\beta^{\sigma(i)}} \\ &= \sum_{j=1}^{\infty} \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a x^{j-1} \left(\beta^{\sigma(i)}\right)^j \\ &= \sum_{j=1}^{\infty} S_j x^{j-1}.\end{aligned}$$

It is important here to comment on the meaning of infinite sums over finite fields. Unlike the continuous case, no two distinct points can be considered close to one another. Therefore, convergence in the limit is the same as eventual equality. Thus, the second and third equalities do not hold for evaluations but instead imply that the (infinite) power series expansions of the two expressions match term by term.

The *error-evaluator polynomial* $\Omega(x) \triangleq \sum_{j=0}^{\nu} \Omega_j x^j$ is given by

$$\begin{aligned}\Omega(x) &\triangleq \Lambda(x) \tilde{S}(x) \\ &= \prod_{j=1}^{\nu} \left(1 - x\beta^{\sigma(j)}\right) \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a \frac{\beta^{\sigma(i)}}{1 - x\beta^{\sigma(i)}} \\ &= \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a \beta^{\sigma(i)} \prod_{j \neq i} \left(1 - x\beta^{\sigma(j)}\right).\end{aligned}$$

It is easy to see that $\Omega(x)$ has degree at most $\nu - 1$. The polynomial $\Omega(x)$ is called the error-evaluator polynomial because

$$\begin{aligned}\Omega\left(\beta^{-\sigma(k)}\right) &= \sum_{i=1}^{\nu} e_{\sigma(i)} \alpha^a \beta^{\sigma(i)} \prod_{j \neq i} \left(1 - \beta^{-\sigma(k)} \beta^{\sigma(j)}\right) \\ &= e_{\sigma(k)} \alpha^a \left[\beta^{\sigma(k)} \prod_{j \neq k} \left(1 - \beta^{\sigma(j)}\right) \right]_{x=\beta^{-\sigma(k)}} \\ &= -e_{\sigma(k)} \alpha^a \Lambda' \left(\beta^{-\sigma(k)}\right),\end{aligned}$$

where, using the product rule, the formal derivative of $\Lambda(x)$ is given by

$$\Lambda'(x) = - \sum_{i=1}^{\nu} \beta^{\sigma(i)} \prod_{j \neq i} \left(1 - x\beta^{\sigma(j)}\right).$$

Using $\Omega(x)$, one can compute the error magnitudes using

$$e_{\sigma(k)} = - \frac{\Omega\left(\beta^{-\sigma(k)}\right)}{\alpha^a \Lambda' \left(\beta^{-\sigma(k)}\right)}. \quad (4)$$

This approach is known as Forney's method.

The decoder is required to compute both the error-locator and error-evaluator polynomials from the finite syndrome polynomial $S(x) \triangleq \sum_{j=1}^{2t} S_j x^{j-1}$. Since $\tilde{S}(x) = S(x) + x^{2t} w(x)$, for some $w(x)$, we find that

$$\Omega(x) = \Lambda(x) \tilde{S}(x) = \Lambda(x) S(x) + \Lambda(x) x^{2t} w(x)$$

and $\deg(\Omega(x)) < \nu$. Thus, we arrive at the *key equation* for RS decoding which is given by

$$\Omega(x) \equiv \Lambda(x) S(x) \pmod{x^{2t}}.$$

In fact, if $\nu \leq t$, then any degree- ν polynomial $\Theta(x)$ that satisfies $\deg(\Theta(x) S(x) \pmod{x^{2t}}) < \nu$ must also satisfy $\Theta(x) = c\Lambda(x)$ [4, Prop. 6.1]. Therefore, this equation can also be used to find error-locator and error-evaluator polynomials.

2.3.2 The Extended Euclidean Algorithm

The extended Euclidean algorithm (EEA) computes the greatest common divisor of two elements a_1, a_2 from a Euclidean domain E (e.g., a ring of polynomials over a field) and coefficients $u, v \in E$ such that $a_0u + a_1v = \gcd(a_1, a_2)$. The algorithm proceeds by dividing a_j by a_{j+1} so that $a_j = a_{j+1}q_{j+1} + a_{j+2}$ with quotient q_{j+1} and remainder a_{j+2} . Each step of the Euclidean algorithm works because the division implies that $\gcd(a_j, a_{j+1}) = \gcd(a_{j+1}, a_{j+2})$. For polynomials, the Euclidean algorithm terminates when $a_j = 0$. This always occurs because $\deg(a_2) < \deg(a_1)$ holds by assumption and $\deg(a_{j+2}) < \deg(a_{j+1})$ holds by induction.

The extended algorithm also computes u_j, v_j recursively so that $a_j = u_j a_1 + v_j a_2$. Starting from $a_3 = a_1 - q_2 a_2$ (i.e., $u_3 = 1$ and $v_3 = -q_2$), we have the recursion

$$a_{j+2} = a_j - q_{j+1} a_{j+1} = (u_j a_1 + v_j a_2) - q_{j+1} (u_{j+1} a_1 + v_{j+1} a_2).$$

This gives the recursions $u_{j+2} = u_j - q_{j+1} u_{j+1}$ and $v_{j+2} = v_j - q_{j+1} v_{j+1}$ starting from $u_3 = 1$ and $v_3 = -q_2$.

The decoding the RS codes is accomplished using a partial application of the EEA algorithm to compute $\gcd(x^{2t}, S(x))$. The extended part of the algorithm generates a sequence of relationships of the form

$$u_j(x)x^{2t} + v_j(x)S(x) = a_j(x),$$

where the degree of $a_j(x)$ is decreasing with j . Let j^* be the first step where $\deg(a_j(x)) < t$ and stop the algorithm at this point. Viewing the above relationship as a congruence modulo x^{2t} gives

$$v_j(x)S(x) \equiv a_j(x) \pmod{x^{2t}},$$

and we see that $v_{j^*}(x)$ and $a_{j^*}(x)$ satisfy the key equation with $\deg(a_{j^*}(x)) < t$. In this case, the polynomials $v_{j^*}(x), a_{j^*}(x)$ must also satisfy

$$\begin{aligned} v_{j^*}(x) &= c\Lambda(x) \\ a_{j^*}(x) &= c\Omega(x), \end{aligned}$$

for some constant c . This means that we can run the EEA until the remainder term has degree less than t . After that, we can solve for c using $c = v_j(0)$ and compute $\Lambda(x), \Omega(x)$. After the error-locator and error-evaluator polynomials are known, decoding proceeds by factoring $\Lambda(x)$ to find the error locations and then using (4) to compute the error magnitudes.

A careful reader might observe that this decoding algorithm is independent of the $u_j(x)$ polynomial. Hence, the Euclidean decoding algorithm can be summarized as follows.

Algorithm 2 Sugiyama's Euclidean Decoding Algorithm

- 1: $a_1(x) \leftarrow x^{2t}, a_2(x) \leftarrow S(x)$
 - 2: $v_1(x) \leftarrow 0, v_2(x) \leftarrow 1$
 - 3: $j \leftarrow 2$
 - 4: **while** $\deg(a_j(x)) \geq t$ **do**
 - 5: $j \leftarrow j + 1$
 - 6: $[q_j(x), a_j(x)] = a_{j-2}(x) \text{ div } a_{j-1}(x)$
 - 7: $v_j(x) = v_{j-2}(x) - v_{j-1}(x)q_{j-1}(x)$
 - 8: **end while**
 - 9: $\Lambda(x) \leftarrow v_j(x)/v_j(0)$
 - 10: $\Omega(x) \leftarrow a_j(x)/v_j(0)$
-

Example 7. Continuing Example 1, we use the Sugiyama algorithm to compute the error-locator polynomial. The results are shown in the table below and the algorithm terminates with $j^* = 4$ because $a_4(x)$ has degree less than $t = 2$. Finally, we observe that it matches the result of the PGZ method.

j	$a_j(x)$	$q_j(x)$	$u_j(x)$	$v_j(x)$
1	x^4		1	0
2	$\alpha^8 x^3 + \alpha^{11} x^2 + \alpha^4 x + \alpha^2$	$\alpha^7 x + \alpha^{10}$	0	1
3	$\alpha x^2 + \alpha^4 x + \alpha^{12}$	$\alpha^7 x$	1	$\alpha^7 x + \alpha^{10}$
4	$\Omega(x) = \alpha^2$		$\alpha^7 x$	$\Lambda(x) = \alpha^{14} x^2 + \alpha^2 x + 1$

Example 8. Continuing Example 2, we use the Sugiyama algorithm to compute the error-locator polynomial. The results are shown in the table below and the algorithm terminates with $j^* = 5$ because $a_5(x)$ has degree less than $t = 3$. Dividing by $v_5(0) = \alpha^3$ gives the error-locator $\Lambda(x) = \alpha^4x^3 + \alpha^6x^2 + \alpha^5x + 1$ (which matches the PGZ result and error-evaluator $\Omega(x) = \alpha^5x^2 + \alpha^6x + \alpha^5$).

j	$a_j(x)$	$q_j(x)$	$u_j(x)$	$v_j(x)$
1	x^6		1	0
2	$\alpha^3x^5 + \alpha^7x^4 + \alpha^5x^3 + \alpha^{10}x^2 + \alpha^7x + \alpha^5$	$\alpha^{12}x + \alpha$	0	1
3	$x^4 + \alpha^{10}x^3 + \alpha^{13}x^2 + x + \alpha^6$	$\alpha^3x + \alpha^5$	1	$\alpha^{12}x + \alpha$
4	$\alpha^8x^3 + \alpha^{10}x^2 + \alpha^{10}x + \alpha^3$	$\alpha^7x + \alpha^{11}$	$\alpha^3x + \alpha^5$	$x^2 + \alpha^{10}x + \alpha^{13}$
5	$\alpha^3\Omega(x) = \alpha^8x^2 + \alpha^9x + \alpha^8$	$x + \alpha^5$	$\alpha^{10}x^2 + \alpha^5x + \alpha^4$	$\alpha^3\Lambda(x) = \alpha^7x^3 + \alpha^9x^2 + \alpha^8x + \alpha^3$

The error magnitudes can be found by evaluating the error-evaluator polynomial and are given by

$$e_{\sigma(i)} = \Psi\left(\alpha^{-\sigma(i)}\right) = -\frac{\Omega\left(\alpha^{-\sigma(i)}\right)}{\Lambda'\left(\alpha^{-\sigma(i)}\right)} = \frac{\alpha^8x^2 + \alpha^9x + \alpha^8}{\alpha^7x^2 + \alpha^8} \Bigg|_{x=\alpha^{-\sigma(i)}}.$$

We verify that this matches the PGZ result by computing

$$\begin{aligned} e_{\sigma(1)} &= \Psi\left(\alpha^{13}\right) = \frac{\alpha^8\alpha^{11} + \alpha^9\alpha^{13} + \alpha^8}{\alpha^7\alpha^{11} + \alpha^8} = 1 \\ e_{\sigma(2)} &= \Psi\left(\alpha^9\right) = \frac{\alpha^8\alpha^3 + \alpha^9\alpha^9 + \alpha^8}{\alpha^7\alpha^3 + \alpha^8} = \alpha^3 \\ e_{\sigma(3)} &= \Psi\left(\alpha^4\right) = \frac{\alpha^8\alpha^8 + \alpha^9\alpha^4 + \alpha^8}{\alpha^7\alpha^8 + \alpha^8} = \alpha^7. \end{aligned}$$

References

- [1] J. K. Wolf, "Decoding of Bose-Chaudhuri-Hocquenghem codes and Prony's method for curve fitting," *IEEE Trans. Inform. Theory*, vol. 13, no. 4, p. 608, Oct. 1967.
- [2] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.
- [3] T. K. Moon, *Error correction coding: mathematical methods and algorithms*. Wiley-Interscience, 2005.
- [4] R. Roth, *Introduction to coding theory*. Cambridge University Press, 2006.