Decoding BCH/RS Codes

Yunghsiang S. Han (韩永祥)

School of Electrical Engineering & Intelligentization
Dongguan University of Technology (东莞理工学院)
China
E-mail: yunghsiangh@gmail.com

Decoding Procedure

- The BCH/RS codes decoding has four steps:
  1. Syndrome computation
  2. Solving the key equation for the error-locator polynomial $\Lambda(x)$
  3. Searching error locations given the $\Lambda(x)$ polynomial by simply finding the inverse roots
  4. (Only nonbinary codes need this step) Determine the error magnitude at each error location by error-evaluator polynomial $\Omega(x)$
- The decoding procedure can be performed in time or frequency domains.
- This lecture only considers the decoding procedure in

time domain. The frequency domain decoding can be found in [1, 2].

Syndrome Computation

- Let $\alpha, \alpha^2, \ldots, \alpha^{2t}$ be the $2t$ consecutive roots of the generator polynomial for the BCH/RS code, where α is an element in finite field $GF(q^m)$ with order n.
- Let $y(x)$ be the received vector. Then define the syndrome S_j, $1 \leq j \leq 2t$, as follows:

$$\begin{aligned} S_j &= y(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) \\ &= \sum_{i=0}^{n-1} e_i (\alpha^j)^i \\ &= \sum_{k=1}^{v} e_{i_k} \alpha^{i_k j}, \end{aligned} \quad (1)$$

where n is the code length and it is assumed that v errors occurred in locations corresponding to time indexes $i_1, i_2, \ldots, i_v$.

- When n is large one can calculate syndromes by the minimum polynomial for α^j.
- Let $\phi_j(x)$ be the minimum polynomial for α^j. That is, $\phi_j(\alpha^j) = 0$. Let $y(x) = q(x)\phi_j(x) + r_j(x)$, where $r_j(x)$ is the remainder and the degree of $r_j(x)$ is less than the degree of $\phi_j(x)$, which is at most m.
- $S_j = y(\alpha^j) = q(\alpha^j)\phi_j(\alpha^j) + r_j(\alpha^j) = r_j(\alpha^j)$.
- For ease of notation we reformulate the syndromes as

$$S_j = \sum_{k=1}^{v} Y_k X_k^j, \text{ for } 1 \leq j \leq 2t,$$

where $Y_k = e_{i_k}$ and $X_k = \alpha^{i_k}$.

- The system of equations for syndromes is

$$\begin{aligned} S_1 &= Y_1 X_1 + Y_2 X_2 + \cdots + Y_v X_v \\ S_2 &= Y_1 X_1^2 + Y_2 X_2^2 + \cdots + Y_v X_v^2 \\ S_3 &= Y_1 X_1^3 + Y_2 X_2^3 + \cdots + Y_v X_v^3 \\ &\vdots \\ S_{2t} &= Y_1 X_1^{2t} + Y_2 X_2^{2t} + \cdots + Y_v X_v^{2t}. \end{aligned}$$

Key Equation

- Recall that the error-locator polynomial is

$$\Lambda(x) = (1 - xX_1)(1 - xX_2)\cdots(1 - xX_v) = \Lambda_0 + \sum_{i=1}^{v} \Lambda_i x^i,$$

where $\Lambda_0 = 1$.

- Define the infinite degree syndrome polynomial (though we only know the first $2t$ coefficients) as

$$\begin{aligned} S(x) &= \sum_{j=0}^{\infty} S_{j+1} x^j \\ &= \sum_{j=0}^{\infty} x^j \left(\sum_{k=1}^{v} Y_k X_k^{j+1} \right) \end{aligned}$$

$$= \sum_{k=1}^{v} \frac{Y_k X_k}{1 - x X_k}.$$

- Define the error-evaluator polynomial as

$$\begin{aligned} \Omega(x) &\stackrel{\triangle}{=} \Lambda(x) S(x) \\ &= \sum_{k=1}^{v} Y_k X_k \prod_{\substack{j=1 \\ j \neq k}}^{v} (1 - x X_j). \end{aligned}$$

- The degree of the error-evaluator polynomial is less than v.
- Actually we only know the first $2t$ terms of $S(x)$ such that we have

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

- Since the degree of $\Omega(x)$ is at most $v-1$ the terms of $\Lambda(x)S(x)$ from x^v through x^{2t-1} are all zeros.
- Then

$$\sum_{k=0}^{v} \Lambda_k S_{j-k} = 0, \text{ for } v+1 \leq j \leq 2t. \quad (3)$$

- The above system of equations is the same as the key equation given previously if we only consider those equations up to $j = 2v$ (remember that $v \leq t$).
- Thus, (2) is also known as *key equation*.
- Solving key equation to determine the coefficients of the

error-locator polynomial is a hard problem and it will be mentioned later.

- The key equation becomes

$$\Lambda(x)(1 + S(x)) \equiv \Omega(x) \bmod x^{2t+1} \tag{4}$$

if we define the infinite degree syndrome equation as

$$S(x) = \sum_{j=1}^{\infty} S_j x^j. \tag{5}$$

Chien Search

- The next important decoding step is to find the actual error locations $X_1 = \alpha^{i_1}, X_2 = \alpha^{i_2}, \ldots, X_v = \alpha^{i_v}$.
- Note that $\Lambda(x)$ has roots $X_1^{-1} = \alpha^{-i_1}, X_2^{-1} = \alpha^{-i_2}, \ldots, X_v^{-1} = \alpha^{-i_v}$.
- Observe that an error occurs in position i if and only if $\Lambda(\alpha^{-i}) = 0$ or

$$\sum_{k=0}^{v} \Lambda_k \alpha^{-ik} = 0.$$

- Then

$$\Lambda(\alpha^{-(i-1)}) = \sum_{k=0}^{v} \Lambda_k \alpha^{-ik+k} = \sum_{k=0}^{v} \left(\Lambda_k \alpha^{-ik}\right) \alpha^k.$$

- This suggests that the potential error locations are tested in succession starting with time index $n-1$.

1. Summing all terms of $\Lambda(\alpha^{-i})$ at index i tests to see whether $\Lambda(\alpha^{-i})=0$
2. Then to test at index $i-1$ only requires multiplying the kth term of $\Lambda(\alpha^{-i})$ by α^k for all k and summing all terms again
3. This procedure is repeated until index 0 is reached
4. The initial value for kth term is $\Lambda_k\alpha^{-nk}=\Lambda_k$

Forney's Formula

- For nonbinary BCH or RS codes one still needs to determine the error magnitude for each error location.
- These values, $Y_1, Y_2, \ldots, Y_v$, can be obtained by utilizing the error-evaluator polynomial. This step is known as *Forney's formula*.
- By substituting $X_k^{-1} = \alpha^{-i_k}$ into the error-evaluator polynomial we have

$$\Omega(X_k^{-1}) = Y_k X_k \prod_{\substack{j=1 \\ j \neq k}}^{v} (1 - X_k^{-1} X_j).$$

- By taking the formal derivative of $\Lambda(x)$ and also evaluating it at $x = X_k^{-1}$ we have

$$\begin{aligned}\Lambda'(X_k^{-1}) &= -X_k \prod_{\substack{j=1\\ j\neq k}}^{v} (1 - X_k^{-1} X_j) \\ &= \frac{-1}{Y_k}\Omega(X_k^{-1}).\end{aligned}$$

- Thus the error magnitude Y_k is given by

$$Y_k = -\frac{\Omega(X_k^{-1})}{\Lambda'(X_k^{-1})} = -\frac{\Omega(\alpha^{-i_k})}{\Lambda'(\alpha^{-i_k})}. \qquad (6)$$

- Clearly, the above formula can be determined by a search procedure similar to Chien Search.
- Usually, $\Omega(x)$ can be obtained by solving the key

equation.

The Euclidean Algorithm [1]

- Euclidean algorithm is a recursive technology to find the greatest common divisor (GCD) of two numbers or two polynomials.
- The Euclidean algorithm is as follows. Let $a(x)$ and $b(x)$ represent the two polynomials, which $deg\,[a(x)] \geq deg\,[b(x)]$. Divide $a(x)$ by $b(x)$. If the remainder, $r(x)$, is zero, then GCD $d(x) = b(x)$. If the remainder is not zero, then replace $a(x)$ with $b(x)$, replace $b(x)$ with $r(x)$, and repeat.
- Considering a simple example, where $a(x) = x^5 + 1$ and $b(x) = x^3 + 1$. Then

$$\begin{aligned} x^5+1 &= x^2(x^3+1)+(x^2+1) \\ x^3+1 &= x(x^2+1)+(x+1) \\ x^2+1 &= (x+1)(x+1)+0 \end{aligned}$$

- Since $d(x)$ divides x^5+1 and x^3+1 it must also divide x^2+1. Since it divides x^3+1 and x^2+1 it must also divide $x+1$. Consequently, $x+1=d(x)$.
- The useful aspect of this process is that, at each iteration, a set of polynomials $f_i(x)$, $g_i(x)$, and $r_i(x)$ are generated such that

$$f_i(x)a(x)+g_i(x)b(x)=r_i(x). \qquad (7)$$

- A way to obtain $f_i(x)$ and $g_i(x)$ is as follows.

- Define $q_i(x)$ to be the quotient polynomial that is produced by dividing $r_{i-2}(x)$ by $r_{i-1}(x)$. Then, for $i \geq 1$,

$$\begin{aligned} r_i(x) &= r_{i-2}(x) - q_i(x)r_{i-1}(x) \\ f_i(x) &= f_{i-2}(x) - q_i(x)f_{i-1}(x) \\ g_i(x) &= g_{i-2}(x) - q_i(x)g_{i-1}(x), \end{aligned}$$

where the initial values are

$$\begin{aligned} f_{-1}(x) &= g_0(x) = 1 \\ f_0(x) &= g_{-1}(x) = 0 \\ r_{-1}(x) &= a(x) \\ r_0(x) &= b(x). \end{aligned} \tag{8}$$

- There are two useful properties of the algorithm:

1. $deg\,[r_i(x)] < deg\,[r_{i-1}(x)]$;
2. $deg\,[g_i(x)] + deg\,[r_{i-1}(x)] = deg\,[a(x)]$.

The Sugiyama Algorithm for Solving Key Equation [1]

- The Sugiyama algorithm utilizes Euclidean algorithm to solve the key equation. Hence, the Sugiyama algorithm is also called Euclidean algorithm.
- (7) can be written as

$$g_i(x)b(x) \equiv r_i(x) \bmod a(x).$$

- Comparing (2) with the above equation, they are equivalent when

$$a(x) = x^{2t},\ b(x) = S(x)$$
$$g_i(x) = \Lambda_i(x),\ r_i(x) = \Omega_i(x).$$

- The Euclidean algorithm produces a sequence of solutions to the key equation.

- When $v \le t$ one needs to know which solutions produced is the desired solution. It can be determined as follows.
- By the property of Euclidean algorithm, we have

$$deg\left[g_i(x)\right] + deg\left[r_{i-1}(x)\right] = 2t$$

and

$$deg\left[g_i(x)\right] + deg\left[r_i(x)\right] < 2t.$$

- If $v \le t$, then $\deg\left[\Omega(x)\right] < \deg\left[\Lambda(x)\right] \le t$ $(\deg\left[r_\ell(x)\right] < \deg\left[g_\ell(x)\right] \le t)$.
- There exists only one polynomial $\Lambda(x)$ with degree no greater than t which satisfies the key equation.
- If $\deg\left[r_{\ell-1}(x)\right] \ge t$, then $\deg\left[g_\ell(x)\right] \le t$. Since $\deg\left[r_\ell(x)\right] < t$, $\deg\left[g_{\ell+1}(x)\right] > t$.

- The results at the ℓth step provide the only solution to the key equation that is of interest.

Summary of the Sugiyama Decoding algorithm

1. Apply Euclidean algorithm to $a(x) = x^{2t}$ and $b(x) = S(x)$.
2. Use the initial conditions of (8).
3. Stop when $deg\,[r_\ell(x)] < t$.
4. Set $\Lambda(x) = g_\ell(x)$ and $\Omega(x) = r_\ell(x)$.

- Note that the algorithm will give an error-locator polynomial no matter whether $v \leq t$ or not. Thus, a circuit to check for valid error-locator polynomial must be performed during Chien search.
- One can check whether the number of roots found by

Chien search is the same as the degree of the error-locator polynomial or not. If they are agreed, the valid error-locator polynomial is assumed. Otherwise, too-many-error alert is reported.

Example

Consider the triple-error-correcting BCH code where generator polynomial has $\alpha, \alpha^2, \ldots, \alpha^6$ as roots and α is a primitive element of $GF(2^4)$ with $\alpha^4 = \alpha + 1$. Let the received vector $y(x) = x^7 + x^2$. We now want to find the error locations of the received vector.

First we need to calculate the syndrome coefficients. By (1), we have

$$S(x) = x^4 + \alpha^3 x^3 + \alpha^9 x + \alpha^{12}.$$

Next we perform Sugiyama algorithm as follows:

i	$\Lambda_i(x)(g_i(x))$	$\Omega_i(x)(r_i(x))$	$q_i(x)$
-1	0	x^6	–
0	1	$S(x)$	–
1	$x^2+\alpha^3x+\alpha^6$	$\alpha^{11}x+\alpha^3$	$x^2+\alpha^3x+\alpha^6$

Thus, $\Lambda(x) = x^2 + \alpha^3 x + \alpha^6$. By performing Chien search we can find the roots of $\Lambda(x)$ are α^{-7} and α^{-2} and consequently, $e(x) = x^7 + x^2$.

The Berlekamp-Massey Algorithm for Solving Key Equation [3]

- For simplicity, we only consider binary BCH codes.
- The Berlekamp-Massey (BM) algorithm builds the error-locator polynomial by requiring that its coefficients satisfy a set of equations called the Newton's identities rather than (3). The Newton's identities are:

$$\begin{aligned}
S_1 + \Lambda_1 &= 0, \\
S_2 + \Lambda_1 S_1 + 2\Lambda_2 &= 0, \\
S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + 3\Lambda_3 &= 0, \\
&\vdots \\
S_v + \Lambda_1 S_{v-1} + \cdots + \Lambda_{v-1} S_1 + v\Lambda_v &= 0,
\end{aligned}$$

and for $j > v$:

$$S_j + \Lambda_1 S_{j-1} + \cdots + \Lambda_{v-1} S_{j-v+1} + \Lambda_v S_{j-v} = 0.$$

- It turns out that we only need to look at the first, third, fifth,...of these equations. For notation ease, we number these Newton identities as (noting that $i\Lambda_i = \Lambda_i$ when i is odd):

$$
\begin{array}{ll}
1) & S_1 + \Lambda_1 = 0, \\
2) & S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + \Lambda_3 = 0, \\
3) & S_5 + \Lambda_1 S_4 + \Lambda_2 S_3 + \Lambda_3 S_2 + \Lambda_4 S_1 + \Lambda_5 = 0, \\
\vdots & \\
\mu) & S_{2\mu-1} + \Lambda_1 S_{2u-2} + \Lambda_2 S_{2\mu-3} + \cdots + \Lambda_{2\mu-2} S_1 + \Lambda_{2\mu-1} = 0 \\
\vdots &
\end{array} \qquad (9)
$$

- Define a sequence of polynomials $\Lambda^{(\mu)}(x)$ of degree d_μ indexed by μ as follows:

$$\Lambda^{(\mu)}(x) = 1 + \Lambda_1^{(\mu)} x + \Lambda_2^{(\mu)} x^2 + \cdots + \Lambda_{d_\mu}^{(\mu)} x^{d\mu}.$$

- The polynomial $\Lambda^{(\mu)}(x)$ is calculated to be the minimum degree polynomial whose coefficients satisfy all of the first μ numbered equations of (9).
- For each polynomial, its *discrepancy* Δ_μ, which measures how far $\Lambda^{(\mu)}(x)$ is from satisfying the $\mu + 1$st identity, is defined as

$$\Delta_\mu = S_{2\mu+1} + \Lambda_1 S_{2u} + \Lambda_2 S_{2\mu-1} + \cdots + \Lambda_{d_\mu} S_{2\mu+1-d_\mu}. \quad (10)$$

- One starts with two initial polynomials, $\Lambda^{(-1/2)}(x) = 1$ and $\Lambda^{(0)}(x) = 1$, and then generate $\Lambda^{(\mu)}$ iteratively in a manner that depends on the discrepancy.
- The discrepancy $\Delta_{-1/2} = 1$ by convention and the remaining discrepancies are calculated.

- The Berlekamp-Massey algorithm is as follows:

1. $\Lambda^{(-1/2)}(x) = 1$, $\Lambda^{(0)}(x) = 1$, and $\Delta_{-1/2} = 1$.
2. Start from $\mu = 1$ and repeat the next two steps until $\mu = t$.
3. Calculate Δ_μ according to (10). If $\Delta_\mu = 0$, then
$$\Lambda^{(\mu+1)}(x) = \Lambda^{(\mu)}(x).$$
4. If $\Delta_\mu \neq 0$, find a value $-(1/2) \leq \rho < \mu$ such that $\Delta_\rho \neq 0$ and $2\rho - d_\rho$ is as large as possible. Then
$$\Lambda^{(\mu+1)}(x) = \Lambda^{(\mu)}(x) + \Delta_\mu \Delta_\rho^{-1} x^{2(\mu-\rho)} \Lambda^{(\rho)}(x).$$

- The error-locator polynomial is $\Lambda(x) = \Lambda^{(t)}(x)$.
- If this polynomial had degree greater than t, more than t errors have been made, and uncorrectable alert should be declared.

Example

Consider the same BCH code and received vector as in the previous example. Then

$$S(x) = x^4 + \alpha^3 x^3 + \alpha^9 x + \alpha^{12}.$$

Next we perform Berlekamp-Massey algorithm as follows:

μ	$\Lambda^{(\mu)}(x)$	Δ_μ	d_μ	$2\mu - d_\mu$	
-1/2	1	1	0	-1	
0	1	α^{12}	0	0	
1	$1+\alpha^{12}x$	α^6	1	1	(take $\rho = -1/2$)
2	$1+\alpha^{12}x+\alpha^9x^2$	0	2	2	(take $\rho = 0$)
3	$1+\alpha^{12}x+\alpha^9x^2$	-	-	-	

$1+\alpha^{12}x+\alpha^9x^2$ has the same roots as $\alpha^6+\alpha^3x+x^2$ which was found by the Sugiyama algorithm.

LFSR Interpretation of Berlekamp-Massey Algorithm[4]

- Key equations:

$$S_j = -\sum_{i=1}^{v} \Lambda_i S_{j-i}, \quad j = v+1, v+2, \ldots, 2t.$$

- The formula describes the output of a linear feedback shift register (LFSR) with coefficients $\Lambda_1, \Lambda_2, \ldots, \Lambda_v$.
- The problem to find the error locator polynomial is then equivalent to find the smallest number of coefficients of an LFSR such that it can produce $S_1, S_2, \ldots, S_{2t}$, i.e., to find a shortest such LFSR.
- In the Berlekamp-Massey algorithm, one builds the LFSR that produces the entire sequence of syndromes by

successively modifying an existing LFSR. This procedure starts with an LFSR that could produce S_1 and end at an LFSR that produces the entire sequence of syndromes.

- Let L_k denote the length of the LFSR produced at stage k of the algorithm.
- Let

$$\Lambda^{[k]}(x) = 1 + \Lambda_1^{[k]} x + \cdots + \Lambda_{L_k}^{[k]} x^{L_k}$$

be the connection polynomial at stage k, indicating the connections for the LFSR capable of producing the output sequence $\{S_1,\ S_2,\ \ldots,\ S_k\}$. That is

$$S_j = -\sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{j-i}, \quad j = L_k + 1, L_k + 2, \ldots, k.$$

- Assume that we have a connection polynomial $\Lambda^{[k-1]}(x)$ of length L_{k-1} that produces $\{S_1,\ S_2,\ \ldots,\ S_{k-1}\}$ for some $k-1<2t$.
- Then $\hat{S}_k = -\sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}$.
- If $\hat{S}_k$ is equal to S_k, then there is no need to update the LFSR, so $\Lambda^{[k]}(x)=\Lambda^{[k-1]}(x)$ and $L_k=L_{k-1}$.
- Otherwise, there is some nonzero *discrepancy* associated with $\Lambda^{[k-1]}(x)$,

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

In this case, we update the connection polynomial using the formula

$$\Lambda^{[k]}(x) = \Lambda^{[k-1]}(x) + Ax^{\ell}\Lambda^{[m-1]}(x), \qquad (11)$$

where A is some element in the finite field, ℓ is an integer, and $\Lambda^{[m-1]}(x)$ is one of the prior connection polynomials produced by our processes associated with nonzero discrepancy d_m.

- The new discrepancy is then

$$d'_k = \sum_{i=0}^{L_k} \Lambda_i^{[k]} S_{k-i} = \sum_{i=0}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i} + A \sum_{i=0}^{L_{m-1}} \Lambda_i^{[m-1]} S_{k-i-\ell}.$$

- We can find an A and an ℓ to make the new discrepancy zero as follows. Let

$$\ell = k - m.$$

Then the second summation gives

$$A \sum_{i=0}^{L_{m-1}} \Lambda_i^{[m-1]} S_{m-i} = A d_m.$$

If we choose

$$A = -d_m^{-1} d_k,$$

then

$$d'_k = d_k - d_m^{-1} d_k d_m = 0.$$

- We still need to prove that such selection indeed makes a shortest LSFR.

Characterization of LFSR Length

- Suppose that an LFSR with connection polynomial $\Lambda^{[k-1]}(x)$ of length L_{k-1} produces the sequence $\{S_1,\ S_2,\ \ldots,\ S_{k-1}\}$, but not $\{S_1,\ S_2,\ \ldots,\ S_k\}$. Then any connection polynomial that produces the latter sequence must have a length L_k satisfying $L_k \geq k - L_{k-1}$.
- This can be proved as follows. We assume that $L_{k-1} < k-1$; otherwise, it is trivial. We then prove it by contradiction with assuming that $L_k \leq k - 1 - L_{k-1}$. We can observe that

$$-\sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{j-i} \begin{cases} = S_j & j = L_{k-1}+1, L_{k-1}+2, \ldots, k-1 \\ \neq S_k & j = k \end{cases}$$

and

$$-\sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{j-i} = S_j \quad j = L_k + 1, L_k + 2, \ldots, k.$$

In particular, we have

$$S_k = -\sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{k-i}.$$

Since $k - L_k \geq L_{k-1} + 1$, all values of S_j involved in the above summation can be substituted by $-\sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{j-i}$. Hence,

$$S_k = -\sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{k-i} = \sum_{i=1}^{L_k} \Lambda_i^{[k]} \sum_{j=1}^{L_{k-1}} \Lambda_j^{[k-1]} S_{k-i-j}.$$

Interchanging the order of summation we have

$$S_k = \sum_{j=1}^{L_{k-1}} \Lambda_j^{[k-1]} \sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{k-i-j}.$$

However, we have

$$S_k \neq -\sum_{i=1}^{L_{k-1}} \Lambda_i^{[k-1]} S_{k-i}.$$

By the assumption, $L_k + 1 \leq k - L_{k-1}$,

$$S_k \neq \sum_{j=1}^{L_{k-1}} \Lambda_j^{[k-1]} \sum_{i=1}^{L_k} \Lambda_i^{[k]} S_{k-i-j},$$

which contradicts to what we just derived.

- Since the shortest LFSR that produces the sequence

$\{S_1,\ S_2,\ \ldots,\ S_k\}$ must also produce the first part of that sequence, we must have $L_k \geq L_{k-1}$. Thus, we have

$$L_k \geq \max(L_{k-1}, k - L_{k-1}).$$

- In the update procedure, if $\Lambda^{[k]}(x) \neq \Lambda^{[k-1]}(x)$, then a new LFSR can be found whose length satisfies $L_k = \max(L_{k-1}, k - L_{k-1})$.
- It can be proved by induction on k. When $k = 1$ we take $L_0 = 0$ and $\Lambda^{[0]}(x) = 1$. We find that $d_1 = S_1$. If $S_1 = 0$, then no update is necessary. If $S_1 \neq 0$, then we take $\Lambda^{[m]}(x) = \Lambda^{[0]}(x) = 1$, so that $\ell = 1 - 0 = 1$. Also take $d_m = 1$. The updated polynomial is

$$\Lambda^{[1]}(x) = 1 + S_1 x,$$

which has degree $L_1 = \max(L_0, 1 - L_0) = 1$.

Now let $\Lambda^{[m-1]}(x),\ m < k-1$, denote the *last* connection polynomial before $\Lambda^{[k-1]}(x)$ with $L_{m-1} < L_{k-1}$ that can produce the sequence $\{S_1,\ S_2,\ \ldots,\ S_{m-1}\}$ but not the sequence $\{S_1,\ S_2,\ \ldots,\ S_m\}$. Then $L_m = L_{k-1}$. By the inductive hypothesis,

$$L_m = m - L_{m-1} = L_{k-1},\ \text{or}\ -m + L_{m-1} = -L_{k-1}.$$

Since $\ell = k - m$, we have

$$\boxed{L_k = \max(L_{k-1}, k - m + L_{m-1}) = \max(L_{k-1}, k - L_{k-1}).}$$

- In the update step if $2L_{k-1} \geq k$, the connection polynomial is updated, but there is no change in length.

Welch-Berlekamp Key Equation

- Welch-Berlekamp (WB) key equation was invented in 1983.
- It is no need to calculate syndromes.
- It uses coefficients of a remainder polynomial to represent errors (syndromes).
- There are several methods to solve WB key equation such as Welch-Berlekamp algorithm, Lagrange-Euclidean algorithm, and Modular approach.

Notations

- The generator polynomial for an (n, k) RS code can be written as

$$g(x) = \prod_{i=1}^{2t} (x - \alpha^i).$$

- Let $L_c = \{0, 1, \ldots, 2t-1\}$ be the index set of the check locations. Let $L_{\alpha^c} = \{\alpha^k, 0 \leq k \leq 2t-1\}$.
- Let $L_m = \{2t, 2t+1, \ldots, n-1\}$ be the index set of the message locations. Let $L_{\alpha^m} = \{\alpha^k, 2t \leq k \leq n-1\}$.
- Define *remainder polynomial* as

$$r(x) = y(x) \bmod g(x)$$

and

$$r(x) = \sum_{i=0}^{2t-1} r_i x^i.$$

- Let $E(x)$ be the error pattern. It can be proved that

$$r(x) \equiv E(x) \bmod g(x)$$

and

$$r(\alpha^k) = E(\alpha^k) \text{ for } k \in \{1, 2, \ldots, 2t\}.$$

Errors in Message Location

- Assume that $e \in L_m$ with error value Y.
- $r(\alpha^k) = E(\alpha^k) = Y(\alpha^k)^e = YX^k,\ k \in \{1, 2, \ldots, 2t\}$, where $X = \alpha^e$ is the error locator.
- Define $u(x) = r(x) - Xr(\alpha^{-1}x)$ which has degree less than $2t$.
- $u(\alpha^k) = r(\alpha^k) - Xr(\alpha^{-1}\alpha^k) = YX^k - XYX^{k-1} = 0$ for $k \in \{2, 3, \ldots, 2t\}$.
- $u(x)$ has roots at $\alpha^2, \alpha^3, \ldots, \alpha^{2t}$, so that $u(x)$ is divisible by

$$p(x) = \prod_{k=2}^{2t}(x - \alpha^k) = \sum_{i=0}^{2t-1} p_i x^i.$$

- Thus, $u(x) = ap(x)$, where $a \in GF(q^m)$.
- Equating coefficients between $u(x)$ and $p(x)$ we have

$$r_i(1 - X\alpha^{-i}) = ap_i, \ i = 0, 1, \ldots, 2t - 1.$$

That is,

$$r_i(\alpha^i - X) = a\alpha^i p_i, \ i = 0, 1, \ldots, 2t - 1.$$

- Define the error locator polynomial as $W_m(x) = x - X = x - \alpha^e$.
- Since $r(\alpha) = E(\alpha) = YX$,

$$Y = X^{-1} r(\alpha) = X^{-1} \sum_{i=0}^{2t-1} r_i \alpha^i$$

$$= \quad X^{-1}\sum_{i=0}^{2t-1}\frac{a\alpha^i p_i}{W_m(\alpha^i)}\alpha^i = aX^{-1}\sum_{i=0}^{2t-1}\frac{\alpha^{2i}p_i}{(\alpha^i - X)}.$$

- Define $f(x) = x^{-1}\sum_{i=0}^{2t-1}\frac{\alpha^{2i}p_i}{(\alpha^i - x)}$ for $x \in L_{\alpha^m}$. $f(x)$ can be pre-computed for all values of $x \in L_{\alpha^m}$.
- $Y = af(X)$ and

$$r_i = \frac{Y\alpha^i p_i}{f(X)W_m(\alpha^i)}.$$

- Assume that there are $v \geq 1$ errors, with error locators X_i and corresponding error values Y_i for $i = 1, 2, \ldots, v$.

- By linearity we have

$$r_k = p_k \alpha^k \sum_{i=1}^{v} \frac{Y_i}{f(X_i)(\alpha^k - X_i)}, \ \ k = 0, 1, \ldots, 2t-1.$$

- Define

$$F(x) = \sum_{i=1}^{v} \frac{Y_i}{f(X_i)(x - X_i)}$$

having poles at the error locations.

- Let

$$F(x) = \sum_{i=1}^{v} \frac{Y_i}{f(X_i)(x - X_i)} = \frac{N_m(x)}{W_m(x)},$$

where $W_m(x) = \prod_{i=1}^{v}(x - X_i)$ is the error locator polynomial for the errors among the message locations. Note that the error locator polynomial defined here is different from previously defined by Peterson.

- It is clear that $\deg(N_m(x)) < \deg(W_m(x))$.
- We have

$$N_m(\alpha^k) = \frac{r_k}{p_k \alpha^k} W_m(\alpha^k), \ k \in L_c = \{0, 1, \ldots, 2t-1\}.$$

- $N_m(x)$ and $W_m(x)$ have the degree constraints $\deg(N_m(x)) < \deg(W_m(x))$ and $\deg(W_m(x)) \leq t$.

Errors in Check Locations

- For a single error occurring in a check location $e \in L_c$, $r(x) = E(x)$.
- $u(x) = r(x) - Xr(\alpha^{-1}x) = 0$.
- We have

$$r_k = \begin{cases} Y & k = e \\ 0 & \text{otherwise.} \end{cases}$$

WB Key Equation

- Let $E_m = \{i_1, i_2, \ldots, i_{v_l}\} \subset L_m$ denote the error locations among the message locations.
- Let $E_c = \{i_{v_l+1}, i_{v_l+2}, \ldots, i_v\} \subset L_c$ denote the error locations among the check locations.
- The (error location, error value) pairs are (X_i, Y_i), $i = 1, 2, \ldots, v$.
- By linearity,

$$\begin{aligned} r_k &= p_k \alpha^k \sum_{j=1}^{v_l} \frac{Y_{i_j}}{f(X_{i_j})(\alpha^k - X_{i_j})} \\ &\quad + \begin{cases} Y_j & \text{if error locator } X_j \text{ is in check location } k \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

- We have

$$N_m(\alpha^k) = \frac{r_k}{p_k \alpha^k} W_m(\alpha^k), \ k \in L_c \setminus E_c.$$

- Let $W_c(x) = \prod_{i \in E_c}(x - \alpha^i)$ be the error locator polynomial for errors in check locations.
- Let $N(x) = N_m(x) W_c(x)$ and $W(x) = W_m(x) Wc(x)$.
- Since $N(\alpha^k) = W(\alpha^k) = 0$ for $k \in E_c$, we have

$$N(\alpha^k) = \frac{r_k}{p_k \alpha^k} W(\alpha^k), \ k \in L_c = \{0, 1, \ldots, 2t-1\}. \quad (12)$$

- (12) is the Welch-Berlekamp (WB) key equation subject to the conditions

$$\deg(N(x)) < \deg(W(x)) \text{ and } \deg(W(x)) \leq t.$$

- We write (12) as

$$N(x_i) = W(x_i)y_i, \ i = 1, 2, \ldots, 2t \tag{13}$$

for "points" $(x_i, y_i) = (\alpha^{i-1}, r_{i-1}/(p_{i-1}\alpha^{i-1}))$, $i = 1, 2, \ldots, 2t$.

Finding the Error Values

- Denote the error values corresponding to an error locator X_i as $Y[X_i]$.
- By definition,

$$\sum_{i=1}^{v_l} \frac{Y[X_i]}{f(X_i)(x - X_i)} = \frac{N_m(x)W_c(x)}{W_m(x)W_c(x)} = \frac{N(x)}{\prod_{i \in E_{cm}}(x - X_i)},$$

  where $E_{cm} = E_c \cup E_m$.
- Suppose we want determine $Y[X_k]$ at message location. Multiplying both sides of the above equation by $W(x) = \prod_{i \in E_{cm}}(x - X_i)$ and evaluate at $x = X_k$, we have

$$\frac{Y[X_k] \prod_{\substack{i \neq k \\ i \in E_{cm}}} (X_k - X_i)}{f(X_k)} = N(X_k).$$

- Taking the formal derivative, we obtain

$$W'(x) = \sum_{j \in E_{cm}} \prod_{i \neq j} (x - X_i)$$

and

$$W'(X_k) = \prod_{\substack{i \neq k \\ i \in E_{cm}}} (X_k - X_i).$$

- Thus,

$$\boxed{Y[X_k] = f(X_k) \frac{N(X_k)}{W'(X_k)}.}$$

- When the error is in a check location, $X_j = \alpha^k$ for $k \in E_c$, we have

$$r_k = Y[X_j] + p_k \alpha^k \sum_{i=1}^{v_l} \frac{Y[X_i]}{f(X_i)(\alpha^k - X_i)} = Y[X_j] + p_k X_j \frac{N_m(X_j)}{W_m(X_j)}.$$

Thus,

$$Y[X_j] = r_k - p_k X_j \frac{N_m(X_j)}{W_m(X_j)}.$$

- Both $N(X_j) = N_m(X_j)W_c(X_j)$ and $W(X_j) = W_m(X_j)W_c(X_j)$ (Since $W_c(X_j) = 0$) are 0 so a "L'Hopitial's rule" must be used. Since

$$N'(X_j) = N_m(X_j)W_c'(X_j) + N_m'(X_j)W_c(X_j) = N_m(X_j)W_c'(X_j)$$

and

$$W'(X_j) = W_m(X_j)W_c'(X_j) + W_m'(X_j)W_c(X_j) = W_m(X_j)W_c'(X_j),$$

so

$$\frac{N'(X_j)}{W'(X_j)} = \frac{N_m(X_j)}{W_m(X_j)} \neq 0.$$

- Then

$$Y[X_j] = r_k - p_k X_j \frac{N'(X_j)}{W'(X_j)}.$$

Rational Interpolation Problem

- Given a set of points (x_i, y_i), $i = 1, 2, \ldots, m$ over some field $\mathbb{F}$, find polynomials $N(x)$ and $W(x)$ with $\deg(N(x)) < \deg(W(x))$ satisfying

$$N(x_i) = W(x_i)y_i, \; i = 1, 2, \ldots, m. \tag{14}$$

- A solution to the rational interpolation problem provides a pair $[N(x), W(x)]$ satisfying (14).

Welch-Berlekamp Algorithm

- We are interested in a solution satisfying $\deg(N(x)) < \deg(W(x))$ and $\deg(W(x)) \leq m/2$.
- The rank of a solution $[N(x), W(x)]$ is defined as

$$\text{rank}[N(x), W(x)] = \max\{2\deg(W(x)), 1 + 2\deg(N(x))\}.$$

- WB algorithm constructs a solution to the rational interpolation problem of rank$\leq m$ and show that it is unique.
- Since the solution is unique, by the definition of the rank, the degee of $N(x)$ is less than the degree of $W(x)$.
- Let $P(x)$ be an interpolation polynomial such that $P(x_i) = y_i$, $i = 1, 2, \ldots, m$.

- The equation $N(x_i) = W(x_i)y_i$ is equivalent to
$$N(x) = W(x)P(x) \pmod{(x - x_i)}.$$
- By Chinese remainder theorem we have
$$N(x) = W(x)P(x) \pmod{\Pi(x)}, \tag{15}$$
where $\Pi(x) = \prod_{i=1}^{m}(x - x_i)$.
- Suppose $[N(x), W(x)]$ is a solution to (14) and that $N(x)$ and $W(x)$ shares a common factor $f(x)$, such that $N(x) = n(x)f(x)$ and $W(x) = w(x)f(x)$. If $[n(x), w(x)]$ is also a solution to (14), the solution $[N(x), W(x)]$ is said to be reducible. Otherwise, it is irreducible.
- There exists at least one irreducible solution to (15) with rank$\leq m$.

- **Proof:** Let $S = \{[N(x), W(x)] | \operatorname{rank}(N(x), W(x)) \leq m\}$ be the set of polynomial meeting the rank specification. For $[N(x), W(x)] \in S$ and $[M(x), V(x)] \in S$ and f a scalar value, define

$$\begin{aligned} [N(x), W(x)] + [M(x), V(x)] &= [N(x) + M(x), W(x) + V(x)] \\ f[N(x), W(x)] &= [fN(x), fW(x)]. \end{aligned}$$

Then S is a module over $\mathbb{F}[x]$.

- A basis for the $N(x)$ component is

$$\{1, x, \ldots, x^{\lfloor (m-1)/2 \rfloor}\} \ (1 + \lfloor (m-1)/2 \rfloor \text{ dimensions}).$$

- A basis for the $W(x)$ component is

$$\{1, x, \ldots, x^{\lfloor m/2 \rfloor}\} \ (1 + \lfloor m/2 \rfloor \text{ dimensions}).$$

- So the dimension of the Cartesian product is $1 + \lfloor (m-1)/2 \rfloor + 1 + \lfloor m/2 \rfloor = m + 1$.

- Let
$$N(x) - W(x)P(x) = Q(x)\Pi(x) + R(x).$$
- Define the mapping
$$E : S \longrightarrow \{h \in \mathbb{F}[x] | \deg(h(x)) < m\} \qquad (16)$$
by $E([N(x), W(x)]) = R(x)$.
- The dimension of the range of E is m.
- E is a linear mapping from a space of dimension $m+1$ to a space of dimension m, so the dimension of its kernel is > 0. ■
- We say that $[N(x), W(x)]$ satisfy the interpolation(k) problem if
$$N(x_i) = W(x_i)y_i, \ \ i = 1, 2, \ldots k.$$
- We also express the interpolation(k) problem as
$$N(x) = W(x)P_k(x) \pmod{\Pi_k(x)},$$

where $\Pi_k(x) = \prod_{i=1}^{k}(x - x_i)$ and $P_k(x)$ is a polynomial that interpolations the first k points, $P_k(x_i) = y_i,\ i = 1, 2, \dots, k$.

- The WB- algorithm finds a sequence of solution $[N(x), W(x)]$ of minimum rank satisfying the interpolation(k) problem, for $k = 1, 2, \dots, m$.
- If $[N(x), W(x)]$ is an irreducible solution to the interpolation(k) problem and $[M(x), V(x)]$ is another solution such that $\text{rank}[N(x), W(x)] + \text{rank}[M(x), V(x)] \leq 2k$, then $[M(x), V(x)]$ can be reduced to $[N(x), W(x)]$.
- **Proof:** By assumption, there exist two polynomials $Q_1(x)$ and $Q_2(x)$ such that

$$\begin{aligned} N(x) - W(x)P_k(x) &= Q_1(x)\Pi_k(x) \\ M(x) - V(x)P_k(x) &= Q_2(x)\Pi_k(x). \end{aligned} \tag{17}$$

Recall that $N(x_i) = y_i W(x_i)$ and $M(x_i) = y_i V(x_i)$ for

$i = 1, \ldots, k$. Hence

$$N(x_i)V(x_i) = M(x_i)W(x_i),\ i = 1, \ldots, k$$

which implies

$$\Pi_k(x)|(N(x)V(x) - M(x)W(x)). \tag{18}$$

- From the definition of the rank we have

$$\begin{aligned} & \deg(N(x)V(x)) = \deg(N(x)) + \deg(V(x)) \\ \leq \quad & \frac{\text{rank}[N(x), W(x)] - 1}{2} + \frac{\text{rank}[M(x), V(x)]}{2} < k \end{aligned}$$

and

$$\begin{aligned} & \deg(M(x)W(x)) = \deg(M(x)) + \deg(W(x)) \\ \leq \quad & \frac{\text{rank}[M(x), V(x)] - 1}{2} + \frac{\text{rank}[N(x), W(x)]}{2} < k. \end{aligned}$$

- Then $\deg(N(x)V(x) - M(x)W(x)) < k$. From (18), we have

$$N(x)V(x) - M(x)W(x) = 0. \tag{19}$$

- Let $d(x) = GCD(W(x), V(x))$. Then there exist two polynomials which are relatively prime such that

$$W(x) = d(x)w(x),\ V(x) = d(x)v(x). \tag{20}$$

- Substituting (20) into (19), we have

$$N(x)d(x)v(x) = M(x)d(x)w(x)$$

and

$$w(x)|N(x),\ v(x)|M(x).$$

- Let $\frac{N(x)}{w(x)} = \frac{M(x)}{v(x)} = h(x)$, so

$$N(x) = h(x)w(x) \text{ and } M(x) = h(x)v(x). \tag{21}$$

- Substituting (20) and (21) into (17), we have

$$h(x)w(x) - d(x)w(x)P_k(x) = Q_1(x)\Pi_k(x)$$

and

$$h(x)v(x) - d(x)v(x)P_k(x) = Q_2(x)\Pi_k(x).$$

- Since GCD$(w(x), v(x)) = 1$, there exists two polynomials $s(x), t(x)$ such that $s(x)w(x) + t(x)v(x) = 1$.
- Thus, we obtain

$$h(x) - d(x)P_k(x) = (s(x)Q_1(x) + t(x)Q_2(x))\Pi_k(x).$$

The above equation shows that $[h(x), d(x)]$ is also a solution. From (20) and (21), both $[N(x), W(x)]$ and $[M(x), V(x)]$ can be reduced to $[h(x), d(x)]$. Since $[N(x), W(x)]$ is irreducible, we have $\deg(w(x)) = 0$. ■

- If $[N(x), W(x)]$ and $[M(x), V(x)]$ are two solutions of interpolation(k) such that

$$\text{rank}[N(x), W(x)] + \text{rank}[M(x), V(x)] = 2k + 1,$$

then both of them are irreducible solutions, and $N(x)V(x) - M(x)W(x) = f\Pi_k(x)$ for some scalar f.

- **Proof:** Assume that the first conclusion is not correct. Then there exist two irreducible solutions, $[n(x), w(x)]$ and $[m(x), v(x)]$, such that

$$N(x) = f(x)n(x), \; W(x) = f(x)w(x),$$

$$M(x) = g(x)m(x), \; V(x) = g(x)v(x),$$

and $\deg(f(x)) + \deg(g(x)) > 0$. Then

$$\begin{aligned} & \text{rank}[n(x), w(x)] + \text{rank}[m(x), v(x)] \\ = \; & 2k + 1 - 2(\deg(f(x)) + \deg(g(x))) < 2k. \end{aligned}$$

By the previous result, $[n(x), w(x)]$ and $[m(x), v(x)]$ at most differ by a constant common factor. Hence, $\text{rank}[n(x), w(x)] + \text{rank}[m(x), v(x)]$ is even. Contradiction.

- Next we prove the second conclusion. It is easy to see that one of $\text{rank}[N(x), W(x)]$ and $\text{rank}[M(x), V(x)]$ is even and the other is odd. There are two cases:

Case 1: rank$[N(x), W(x)]$ **is odd.** We have

$$\begin{aligned} 2k+1 &= \text{rank}[N(x), W(x)] + \text{rank}[M(x), V(x)] \\ &= (1 + 2\deg(N(x)) + 2\deg(V(x)) \\ &> 2\deg(W(x)) + (1 + 2\deg(M(x))). \end{aligned}$$

Thus, $\deg(N(x)V(x)) = k$ and $\deg(W(x)M(x)) < k$.

Case 2: rank$[N(x), W(x)]$ **is even.** We have

$$\begin{aligned} 2k+1 &= \text{rank}[N(x), W(x)] + \text{rank}[M(x), V(x)] \\ &= 2\deg(W(x)) + (1 + 2\deg(M(x))) \end{aligned}$$

$$> \quad (1 + 2\deg(N(x))) + 2\deg(V(x)).$$

Thus, $\deg(N(x)V(x)) < k$ and $\deg(W(x)M(x)) = k$.

In either case,

$$\deg(N(x)V(x) - M(x)W(x)) = k.$$

We have proved that $\Pi_k(x)|N(x)V(x) - M(x)W(x)$ and then, $N(x)V(x) - M(x)W(x) = f\Pi_k(x)$. ■

- Let $[N(x), W(x)]$ and $[M(x), V(x)]$ be two solutions of interpolation(k) such that

$$\text{rank}[N(x), W(x)] + \text{rank}[M(x), V(x)] = 2k + 1$$

and $N(x)V(x) - M(x)W(x) = f\Pi_k(x)$ for some scalar f. Then $[N(x), W(x)]$ and $[M(x), V(x)]$ are complementary.

- If $[N(x), W(x)]$ is an irreducible solution to the interpolation(k) problem and $[M(x), V(x)]$ is one of its complements, then for any $a, b \in \mathbb{F}$ with $b \neq 0$, $[bM(x) - aN(x), bV(x) - aW(x)]$ is also one of its complements.

- **Proof:** It is easy to show that $[bM(x) - aN(x), bV(x) - aW(x)]$ is also a solution. Since $[M(x), V(x)]$ cannot be reduced to $[N(x), W(x)]$, $[bM(x) - aN(x), bV(x) - aW(x)]$ is also cannot be reduced to $[N(x), W(x)]$. Hence,

$$\text{rank}[N(x), W(x)] + \text{rank}[bM(x) - aN(x), bV(x) - aW(x)] = 2k+1,$$

and $[bM(x) - aN(x), bV(x) - aW(x)]$ is a complement of $[N(x), W(x)]$. ■

- Suppose that $[N(x), W(x)]$ and $[M(x), V(x)]$ are two complementary solutions of interpolation(k) problem. Suppose also that $[N(x), W(x)]$ is the solution of lower rank. Let $b = N(x_{k+1}) - y_{k+1}W(x_{k+1})$ and $a = M(x_{k+1}) - y_{k+1}V(x_{k+1})$.

If $b = 0$, then $[N(x), W(x)]$ and $[(x - x_{k+1})M(x), (x - x_{k+1})V(x)]$ are two complementary solutions of the interpolation$(k+1)$ problem and $[N(x), W(x)]$ is the solution with lower rank. If $b \neq 0$, then

$$[(x - x_{k+1})N(x), (x - x_{k+1})W(x)]$$

and

$$[bM(x) - aN(x), bV(x) - aW(x)]$$

are two complementary solutions. The solution with lower rank is the solution to the interpolation$(k+1)$ problem.

- **Proof:** If $b = 0$, it is clear that $[N(x), W(x)]$ is a solution to the interpolation$(k+1)$ problem. Also $M(x) \equiv V(x)P_k(x) \pmod{\Pi_k(x)}$ such that we have

$$(x - x_{k+1})M(x) \equiv (x - x_{k+1})V(x)P_{k+1}(x) \pmod{\Pi_{k+1}(x)}.$$

Since

$\text{rank}[(x - x_{k+1})M(x), (x - x_{k+1})V(x)] = \text{rank}[M(x), V(x)] + 2$ we have

$$\begin{aligned} & \text{rank}[N(x), W(x)] + \text{rank}[(x - x_{k+1})M(x), (x - x_{k+1})V(x)] \\ = \quad & 2k + 1 + 2 = 2(k+1) + 1. \end{aligned}$$

Now consider $b \neq 0$. Since $[N(x), W(x)]$ satisfies

$$N(x) \equiv W(x)P_{k+1}(x) \pmod{\Pi_k(x)}$$

it follows that

$$(x - x_{k+1})N(x) \equiv (x - x_{k+1})W(x)P_{k+1}(x) \pmod{\Pi_{k+1}(x)}.$$

Thus, $[(x - x_{k+1})N(x), (x - x_{k+1})W(x)]$ is a solution to the interpolation$(k+1)$ problem.

- From previous result, $[bM(x) - aN(x), bV(x) - aW(x)]$ is a complementary solution of $[N(x), W(x)]$ to interpolation(k) problem. To show that $[bM(x) - aN(x), bV(x) - aW(x)]$ is also a solution at the point (x_{k+1}, y_{k+1}), substituting a and b into the following to show that equality holds:

$$bM(x_{k+1}) - aN(x_{k+1}) = (bV(x_{k+1}) - aW(x_{k+1}))\, y_{k+1}.$$

It is clear that

$$\begin{aligned} & \text{rank}[(x - x_{k+1})N(x), (x - x_{k+1})W(x)] \\ + \quad & \text{rank}[bM(x) - aN(x), bV(x) - aW(x)] = 2(k+1) + 1. \end{aligned}$$

■

- The initial condition for WB algorithm is

$$N(x) = V(x) = 0, W(x) = M(x) = 1.$$

Algorithm 1 Welch-Belekamp Algorithm

Input: $(x_i, y_i),\ i = 1, \ldots, m$
$N^{(0)}(x) := V^{(0)}(x) := 0;\ M^{(0)}(x) := W^{(0)}(x) := 1;$
for $k = 0, 1, 2, \ldots, m-1$ **do**
  $b_k := N^{(k)}(x_{k+1}) - y_{k+1} W^{(k)}(x_{k+1});$
  $a_k := M^{(k)}(x_{k+1}) - y_{k+1} V^{(k)}(x_{k+1});$
  if $b_k = 0$ **then**
    $N^{(k+1)}(x) := N^{(k)}(x);\ W^{(k+1)}(x) := W^{(k)}(x);$
    $M^{(k+1)}(x) := (x - x_{k+1}) M^{(k)}(x);$
    $V^{(k+1)}(x) := (x - x_{k+1}) V^{(k)}(x);$
  else
    $M^{(k+1)}(x) := (x - x_{k+1}) N^{(k)}(x);$
    $V^{(k+1)}(x) := (x - x_{k+1}) W^{(k)}(x);$
    $N^{(k+1)}(x) := b_k M^{(k)}(x) - a_k N^{(k)}(x);$
    $W^{(k+1)}(x) := b_k V^{(k)}(x) - a_k W^{(k)}(x);$
    if $\mathrm{rank}[N^{(k+1)}(x), W^{(k+1)}(x)] > \mathrm{rank}[M^{(k+1)}(x), V^{(k+1)}(x)]$ **then**
      $\mathrm{swap}[N^{(k+1)}(x), W^{(k+1)}(x)] \leftrightarrow [M^{(k+1)}(x), V^{(k+1)}(x)]$
    end if
  end if
end for

References

[1] G. C. Clark, Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications*, New York, NY: Plenum Press, 1981.

[2] R. E. Blahut, *Theory and Practice of Error Control Codes*, Reading, MA: Addison-Wesley Publishing Co., 1983.

[3] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Cmodes*, Cabridge University Press, 2003.

[4] T.K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Hoboken, NJ: John Wiley &

Sons, Inc., 2005.