6.0 Decoding BCH and RS Codes

6.1 Conventional Decoding

- based on roots of codewords;
- syndrome polynomials are computed;
- solutions lead to
  - error locator polynomial - roots are the **locations** of the errors;
  - error magnitude polynomial - solutions yield the **values** of the errors for nonbinary codes.
- "decoding algorithm" usually means the method for obtaining these polynomials.

- Substitute roots of $g(x)$ into $r(x) \Rightarrow 2t$ equations.
- Solve this “overspecified” system for a polynomial, roots of which are the *error locations.*
- Also solve for set of *error magnitudes*.
- Typically, these decoders decode correctly up to the design distance.

6.2 Basics of Decoding BCH and RS Codes

- Receive:

$$r(x) = c(x) + e(x) = \sum_{i=0}^{n-1} r_i \cdot x^i$$

- where

$$c(\alpha^j) = 0,\ j = 1, 2, \ldots, 2t.$$

- Compute *syndromes:*

$$S_j = r(\alpha^j) = e(\alpha^j) = e_0 + e_1\alpha^j + e_2\alpha^{2j} + \cdots + e_{n-1}\alpha^{(n-1)j},$$

- where $e_i \in \{0, 1\}$.

- Suppose errors occurred at locations $i_1, i_2, \ldots, i_\ell, \ldots, i_\nu,\ \nu \leq t$.
- *For now,* consider the binary case.

$$e_{i_\ell} = \begin{array}{l} 1,\ \ell = 1, 2, \ldots, \nu \leq t \\ 0,\ \text{otherwise.} \end{array}$$

- Then,

$$S_j = e(\alpha^j) = \alpha^{ji_1} + \alpha^{ji_2} + \cdots + \alpha^{ji_\nu},\ j = 1, 2, \ldots, 2\nu$$

- We call $i_1, i_2, \ldots, i_\nu$ the *error locators*.
- **Notation:** Let $X_\ell = \alpha^{i_\ell}$. Then,

$$S_j = \sum_{\ell=1}^{\nu} X_\ell^j,\ j = 1, 2, \ldots, 2t.$$

Expanding gives,

$$\begin{aligned} S_1 &= X_1 + X_2 + \cdots + X_\nu \\ S_2 &= X_1^2 + X_2^2 + \cdots + X_\nu^2 \\ &\vdots \\ S_{2t} &= X_1^{2t} + X_2^{2t} + \cdots + X_\nu^{2t} \end{aligned}$$

Definition 1 *These are called the* **power sum symmetric functions** *of the* $\{X_i\}$.

□

Note that ν is *unknown* to the decoder.

Let

$$\begin{aligned}\Lambda(x) &= (1 - X_1 x)(1 - X_2 x) \cdots (1 - X_\nu x) \\ &= \sum_{i=0}^{\nu} \Lambda_i x^i\end{aligned}$$

Definition 2 $\Lambda(x)$ *as defined above is called the* **error locator polynomial**.

□

Clearly

$$\Lambda(1/X_\ell) = 0,\ \ell = 1, 2, \ldots, \nu$$

and

$$\begin{aligned}
\Lambda_0 &= 1 \\
\Lambda_\nu &= X_1 X_2 \cdots X_\nu \\
\Lambda_1 &= X_1 + X_2 + \cdots + X_\nu \\
\Lambda_2 &= \sum_{i<j} X_i X_j \\
&\vdots
\end{aligned}$$

Definition 3 *These* $\{\Lambda_i\}$ *are called the* **elementary symmetric functions of the error locators.**

Newton's identities relate the elementary symmetric functions and the power sum symmetric functions:

$$\begin{aligned} S_1 + \Lambda_1 &= 0 \\ S_3 + \Lambda_1 S_2 + \Lambda_2 S_1 + \Lambda_3 &= 0 \\ S_5 + \Lambda_1 S_4 + \Lambda_2 S_3 + \Lambda_3 S_2 + \Lambda_4 S_1 + \Lambda_5 &= 0 \\ &\vdots \\ S_{2t-1} + \Lambda_1 S_{2t-2} + \Lambda_2 S_{2t-3} + \cdots + \Lambda_t S_{t-1} &= 0 \end{aligned}$$

Example: Suppose $\nu = 1$. Then

$$S_j = X_1^j, \ j = 1, 2.$$

from which we learn $S_1 = X_1$. The error locator polynomial becomes:

$$\Lambda(x) = (1 - X_1 x) = 1 - S_1 x.$$

Example Suppose $\nu = 2$.

- Then the odd syndromes are:

$$\begin{aligned} S_1 &= X_1 + X_2 \\ S_3 &= X_1^3 + X_3^3 \end{aligned}$$

- and the error locator polynomial is:

$$\begin{aligned} \Lambda(x) &= (1 - X_1 x)(1 - X_2 x) \\ &= 1 + (X_1 + X_2)x + X_1 X_2 x^2 \end{aligned}$$

- Clearly, $\Lambda_0 = 1,\ \Lambda_1 = S_1$.
- Cubing S_1 and solving simultaneously with S_3 gives

$$\Lambda_2 = \frac{S_3 + S_1^3}{S_1}.$$

6.3 Peterson's Algorithm

6.3.1 Binary Codes

Newton's Identities in matrix form are:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ S_2 & S_1 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \cdots & S_{t-1} \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \vdots \\ \Lambda_t \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ \vdots \\ -S_{2t-1} \end{bmatrix},$$

which we can also write as

$$\mathbf{A} \cdot \mathbf{\Lambda} = -\mathbf{S} \qquad (1)$$

Properties:

- $\mathbf{A}$ (known) must be non-singular in order to solve for $\mathbf{\Lambda}$.
- $\mathbf{A}$ is non-singular if t or $t-1$ errors have occurred.
- More generally,

  Theorem (Berlekamp) *If* $\mathbf{A}$ *is* $t \times t$*, then the dimension of the null space of the row space of* $\mathbf{A}$ *is*

$$\left| \left\lfloor \frac{t - \ \deg \ \Lambda(x)}{2} \right\rfloor \right|$$

- Notice that if $\mathbf{A}$ is of full rank, the foregoing evaluates to 0.

Peterson's algorithm:

1. Write down Newton's Identities (N.I.) as above.
2. If $\det[A] = 0$, remove 2 rightmost columns and 2 bottom rows.
3. Test and repeat until $\det[A] \neq 0$
4. Invert and solve for the $\{\Lambda_i\}$.
5. Find roots of $\Lambda(x)$.
   - If roots are not distinct or $\Lambda(x)$ does not have roots in the desired field, go to 9
6. Complement bit positions in received vector that correspond to roots of $\Lambda(x)$.
7. If the corrected word does not satisfy all syndromes, go to 9
8. Output corrected word. STOP
9. Declare decoder failure. STOP

Some Decoding Examples

Direct Decoding

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ S_2 & S_1 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5} & \cdots & S_{t-1} \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \vdots \\ \Lambda_t \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ \vdots \\ -S_{2t-1} \end{bmatrix}, \quad (3)$$

or,

$$\mathbf{A} \cdot \mathbf{\Lambda} = -\mathbf{S}$$

For simple cases, we solve (3) directly:

Single error correction $(t = 1)$

$$\Lambda_1 = S_1$$

Double error correction: $(t = 2)$

$$\begin{bmatrix} 1 & 0 \\ S_2 & S_1 \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \end{bmatrix},$$

$$\begin{aligned} \Lambda_1 &= S_1 \\ \Lambda_2 &= \frac{S_3 + S_1^3}{S_1} \end{aligned}$$

Triple error correction: $(t = 3)$

$$\begin{bmatrix} 1 & 0 & 0 \\ S_2 & S_1 & 1 \\ S_4 & S_3 & S_2 \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \end{bmatrix},$$

$$\Lambda_1 = S_1$$

$$\Lambda_2 = \frac{S_1^2 S_3 + S_5}{S_1^3 + S_3}$$

$$\Lambda_3 = (S_1^3 + S_3) + S_1 \Lambda_2$$

Quadruple error correction: $(t = 4)$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ S_2 & S_1 & 1 & 0 \\ S_4 & S_3 & S_2 & S_1 \\ S_6 & S_5 & S_4 & S_3 \end{bmatrix} \cdot \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \end{bmatrix} = \begin{bmatrix} -S_1 \\ -S_3 \\ -S_5 \\ -S_7 \end{bmatrix},$$

$$\Lambda_1 = S_1$$

$$\Lambda_2 = \frac{S_1(S_7 + S_1^7) + S_3(S_1^5 + S_5)}{S_3(S_1^3 + S_3) + S_1(S_1^5 + S_5)}$$

$$\Lambda_3 = (S_1^3 + S_3) + S_1\Lambda_2$$

$$\lambda_4 = \frac{(S_5 + S_1^2 S_3) + (S_1^3 + S_3)\Lambda_2}{S_1}$$

Quintuple error correction: $(t = 5)$

$$
\begin{aligned}
\Lambda_1 &= S_1 \\
\Lambda_2 &= \frac{(S_1^3 + S_3)[(S_1^9 + S_9) + S_1^4(S_5 + S_1^2 S_3) + S_3^2(S_1^3 + S_3)]}{(S_1^3 + S_3)[(S_7 + S_1^7) + S_1 S_3(S_1^3 + S_3)] + (S_5 + S_1^2 S_3)(S_1^5 + S_5)} \\
&+ \frac{[(S_1^5 + S_5)(S_7 + S_1^7) + S_1(S_3^2 + S_1 S_5)]}{(S_1^3 + S_3)[(S_7 + S_1^7) + S_1 S_3(S_1^3 + S_3)] + (S_5 + S_1^2 S_3)(S_1^5 + S_5)} \\
\Lambda_3 &= (S_1^3 + S_3) + S_1 \Lambda_2 \\
\Lambda_4 &= \frac{(S_1^9 + S_9) + S_3^2(S_1^3 + S_3) + S_1^4(S_5 + S_1^2 S_3)}{(S_1^5 + S_5)} \\
&\quad + \frac{[(S_7 + S_1^7) + S_1 S_3(S_1^3 + S^3)]\Lambda_2}{(S_1^5 + S_5)} \\
\Lambda_5 &= (S_5 + S_1^2 S_3) + S_1 \Lambda_4 + (S_1^3 + S_3)\Lambda_2
\end{aligned}
$$

Suggested study problem

- Design a BCH code with $n = 7$ and $t = 4$.
- Select some code word $\mathbf{c}$ from your code.
- For $t = 0$ to $t = 2$ do
  - Select an error vector $\mathbf{e}$ of weight t.
  - Form the received vector $\mathbf{r} = \mathbf{c} + \mathbf{e}$
  - Decode $\mathbf{r}$ using the direct method above.

Double Error Correction using Peterson's Algorithm

For $n = 31$ let

$$g(x) = 1 + x^3 + x^5 + x^6 + x^8 + x^9 + x^{10}$$

the roots of which include $\{\alpha, \alpha^2, \alpha^3, \alpha^4\}$.

Let the received vector $\mathbf{r}$ be

$$\mathbf{r} = (0010000110011000000000000000000)$$

or

$$r(x) = x^2 + x^7 + x^8 + x^{11} + x^{12}$$

You should verify that

$$\begin{aligned} S_1 &= r(\alpha) = \alpha^7 \\ S_2 &= r(\alpha^2) = \alpha^{14} \\ S_3 &= r(\alpha^3) = \alpha^8 \\ S_4 &= r(\alpha^4) = \alpha^{28} \end{aligned}$$

Since $t = 2$, we use the foregoing to get

$$\begin{aligned}\Lambda_1 &= S_1 = \alpha^7 \\ \Lambda_2 &= \frac{S_3 + S_1^3}{S_1} = \alpha^{15}\end{aligned}$$

Then, the **error locator polynomial** is

$$\begin{aligned}\Lambda(x) &= 1 + \alpha^7 x + \alpha^{15} x^2 \\ &= (1 + \alpha^5 x)(1 + \alpha^{10} x)\end{aligned}$$

which indicates that the errors are at the 5^{th} and 10^{th} places of $\mathbf{r}$, and that the transmitted codeword most likely was

$$\mathbf{c} = 0010010110111000000000000000000)$$

and

$$\begin{aligned}c(x) &= x^2 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} \\ &= x^2 g(x)\end{aligned}$$

Another example

$$g(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15}$$

where, again $n = 31$ but now, $t = 3$.

Suppose

$$r(x) = x^{10}.$$

What was the most likely transmitted word?

...

The **all-zero word!**..

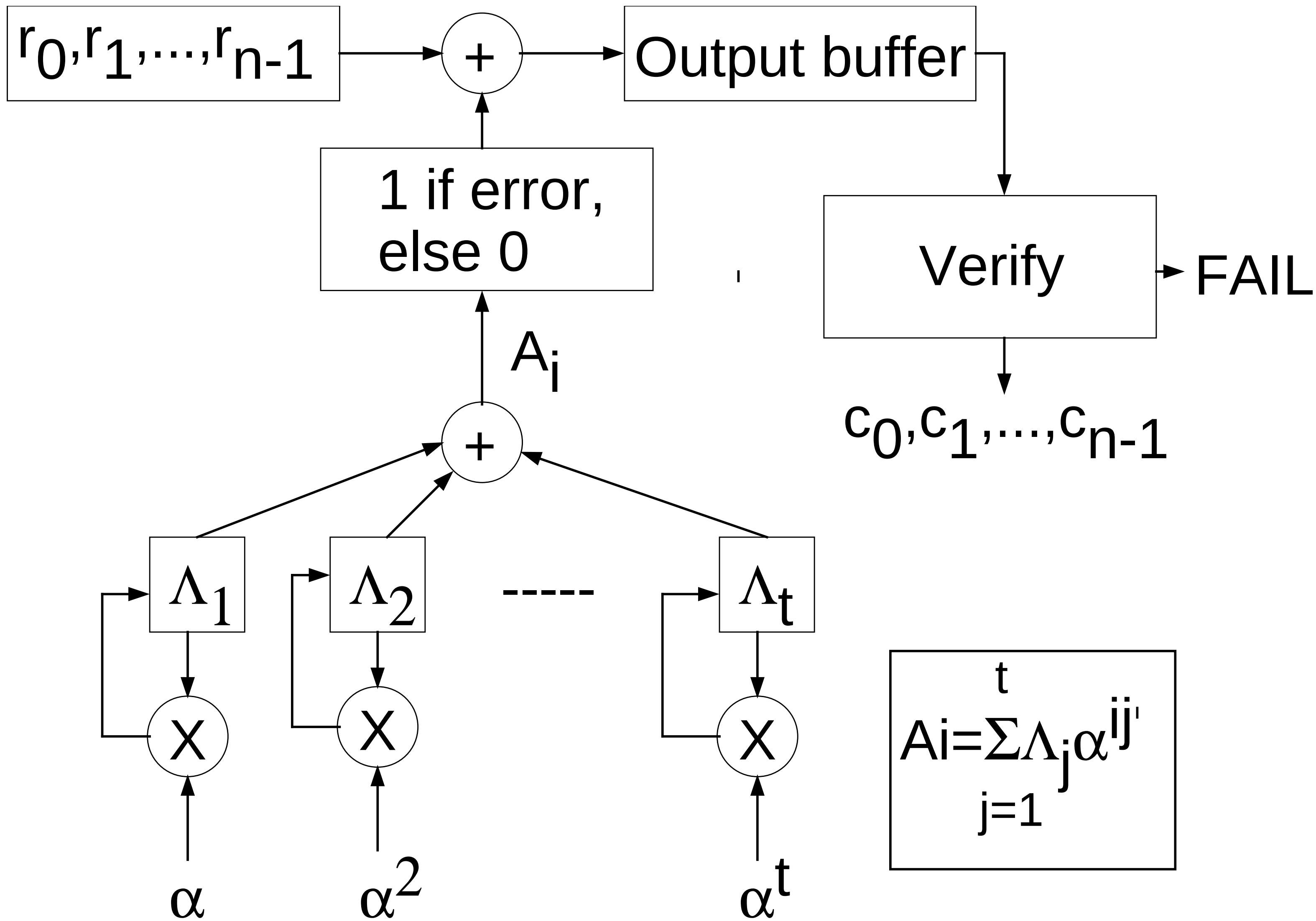
Why?

The Chien Search: Solving the error locator polynomial

1. Repeatedly multiply each Λ_i by α^i.
2. Sum each set of products to get $A_i = \Lambda(\alpha^i) - 1 = \sum_{j=1}^{t} \Lambda_j \alpha^{ij}$.
3. If $\Lambda(\alpha^j) = 0$ then
   - $A_i = 1$ and an error occurred at the coordinate associated with $\alpha^{-j} = \alpha^{n-1}$.
   - So, add 1 to received bit r_{n-j}.
4. Otherwise do nothing.

Verification:

- Use similar circuit with Λ_i replaced by c_i (decoder output) and include $c_0 = 1$.
- This tests for whether the powers of α are roots of $c(x)$.



r0,r1,...,rn-1
+
Output buffer
1 if error,
else 0
Verify
FAIL
Ai
c0,c1,...,cn-1
+
Λ1
Λ2

Λt
X
X
X
α
α2
αt
t
Ai=ΣΛjαij
j=1

6.3.2 Peterson-Gorenstein-Zierler Algorithm for Non-binary Codes

- As before, write syndromes:

$$S_j = e_0 + e_1\alpha^j + e_2\alpha^{2j} + \cdots + e_{n-1}\alpha^{(n-1)j},\ j = 1, \ldots, 2t.$$

- Expand in matrix form:

$$\begin{aligned} S_1 &= e_{i_1}X_1 + e_{i_2}X_2 + \cdots + e_{i_\nu}X_\nu \\ S_2 &= e_{i_1}X_1^2 + e_{i_2}X_2^2 + \cdots + e_{i_\nu}X_\nu^2 \\ S_3 &= e_{i_1}X_1^3 + e_{i_2}X_2^3 + \cdots + e_{i_\nu}X_\nu^3 \\ &\vdots \\ S_{2t} &= e_{i_1}X_1^{2t} + e_{i_2}X_2^{2t} + \cdots + e_{i_\nu}X_\nu^{2t} \end{aligned} \tag{4}$$

- Decoder must compute:
  - Error locators

$$\{X_\ell,\ \ell = 1, 2, \ldots, \nu\}$$

  - Error magnitudes

$$\{e_{i_\ell},\ \ell = 1, 2, \ldots, \nu\}$$

- (Recall that the $\{e_{i_\ell}\}$ are known in the binary case.)
- **But:** The syndromes are no longer *power-sum symmetric functions.*
- Use different method to get sets of linear functions in the unknown locators and magnitudes.

Recall:

$$\Lambda(x) = \prod_{\ell=1}^{\nu}(1 - X_\ell x).$$

Therefore, for some error locator X_ℓ:

$$\Lambda(X_\ell^{-1}) = \Lambda_\nu X_\ell^{-\nu} + \Lambda_{\nu-1} X_\ell^{-(\nu-1)} + \cdots + \Lambda_0 = 0$$

Then form

$$\sum_{\ell=1}^{\nu} e_{i_\ell} X_\ell^j \Lambda(X_\ell^{-1}),$$

and substitute

$$S_j = e_{i_1} X_1^j + e_{i_2} X_2^j + \cdots + e_{i_\nu} X_\nu^j.$$

This gives

$$\Lambda_\nu S_{j-\nu} + \Lambda_{\nu-1} S_{j-\nu+1} + \cdots + \Lambda_1 . S_{j-1} = -S_j.$$

Also, recall $\Lambda_0 = 1$.

Let $\nu = t$ and expand in matrix form:

$$\mathbf{A}'\Lambda = \begin{bmatrix} S_1 & S_2 & \cdots & S_t \\ S_2 & S_3 & \cdots & S_{t+1} \\ \vdots & & & \\ S_{t-1} & S_t & \cdots & S_{2t-2} \\ S_t & s_{t+1} & \cdots & S_{2t-1} \end{bmatrix} \begin{bmatrix} \Lambda_t \\ \Lambda_{t-1} \\ \vdots \\ \Lambda_2 \\ \Lambda_1 \end{bmatrix} \begin{bmatrix} -S_{t+1} \\ -S_{t+2} \\ \vdots \\ -S_{2t-1} \\ -S_{2t} \end{bmatrix}$$

One can show:

- $\mathbf{A}'$ is nonsingular if exactly t errors occurred.
- $\mathbf{A}'$ is singular if $\nu < t$ errors occurred.
- As before, removal of appropriate numbers of rows and columns gives nonsingular matrix and reveals actual number of errors.

Outline of PGZ Algorithm

1. From the $\{S_j\}$, compute $\mathbf{A}'$.
   (a) If $|\mathbf{A}'| = 0$, delete rightmost column and entire bottom row.
   (b) Repeat until nonsingular.
2. Solve for $\mathbf{\Lambda}$; construct $\Lambda(x)$.
3. If roots of $\Lambda(x)$ are not in the desired field or are not distinct, declare decoding failure. STOP
4. Substitute $\{X_\ell\}$ into the $\{S_j\}$. Reduce to matrix form:

$$\mathbf{B}e = \begin{bmatrix} X_1 & X_2 & \cdots & X_\nu \\ X_1^2 & X_2^2 & \cdots & X_\nu^2 \\ \vdots & & & \\ X_1^\nu & X_2^\nu & \cdots & X_\nu^\nu \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_\nu} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_\nu \end{bmatrix}$$

5. Solve for $\{e_{i_\ell}\}$ Output corrected word. STOP

6.4 Berlekamp's Algorithm for Binary (BCH) Codes

- Peterson's alg: # of GF multiplications $\sim \nu^2$
- Cumbersome for $\nu >\sim 6$
- Complexity of *Berlekamp algorithm* $\sim$ linear with ν.
- Introduce Berlekamp's for binary codes
- Study Massey's formulation of Berlekamp's for non-binary codes.

6.4.1 Introduction and General Approach

Decoding steps common to most BCH/RS algorithms:

1. Calculate syndromes: $S_1, S_2, \ldots, S_{2t}$
2. Calculate $\Lambda_1, \Lambda_2, \ldots, \Lambda_t$ from the $\{S_j\}$.
3. Calculate error locations $\{X_\ell\}$ from the $\{\Lambda_i\}$
4. Calculate error values $\{Y_\ell\}$ from the $\{X_\ell\}$, $\{S_j\}$. (Non-binary case)

6.4.2 Berlekamp's iterative method for binary codes (offered without proof). Define a **syndrome polynomial** to be

$$S(x) = S_1 x + S_2 x^2 + \cdots + S_{2t+1} x^{2t+1} + \cdots$$

of arbitrarily large degree. Now let

$$\begin{aligned}
\Omega(x) &\triangleq [1 + S(x)]\Lambda(x) \\
&= (1 + S_1 x + S_2 x^2 + \cdots + S_{2t+1} x^{2t+1} + \cdots) \\
&\quad \cdot (1 + \Lambda_1 x + \Lambda_2 x^2 + \cdots) \\
&= 1 + (S_1 + \Lambda_1)x + (S_2 + S_1\Lambda_1 + \Lambda_2)x^2 \\
&+ (S_3 + S_2\Lambda_1 + S_1\Lambda_2 + \Lambda_3)x^3 + \cdots \\
&= 1 + \Omega_1 x + \Omega_2 x^2 + \Omega_3 x^3 + \cdots
\end{aligned}$$

Notes:

1. Comparison of the coefficients with Newton's identities shows that the coefficients of the **odd** powers of x are identically zero.

2. Although the polynomials have arbitrary degree, only the first $2t$ of the $\{S_i\}$ are known.

Therefore, we write

$$\begin{aligned} \Omega(x) &= [1 + S(x)]\Lambda(x) \mod x^{2t+1} \\ &= 1 + \Omega_2 x^2 + \Omega_4 x^4 + \cdots \mod x^{2t+1} \end{aligned}$$

Berlekamp's iterative algorithm solves for $\Lambda(x)$ iteratively, by breaking the problem down into a set of steps,

$$[1 + S(x)]\Lambda^{(2k)}(x) = 1 + \Omega_2 x^2 + \Omega_4 x^4 + \cdots \mod x^{2t+1}$$

for k from 1 to t.

1. **Initialize** $k = 0,\ \Lambda^{(0)}(x) = 1,\ T^{(0)} = 1.$
2. **Let** $\Delta^{(2k)}$ be the coefficient of x^{2k+1} in $\Lambda^{(2k)}[1 + S(x)]$.
3. **Compute**

$$\Lambda^{2k+2}(x) = \Lambda^{(2k)}(x) + \Delta^{(2k)}[x \cdot T^{(2k)}(x)]$$

4. (a) **if** $\Delta^{(2k)} = 0$ *or* $\deg[\Lambda^{(2k)}(x)] > k$

$$T^{(2k+2)}(x) = x^2 T^{(2k)}(x)$$

(b) **else if** $\Delta^{(2k)} \neq 0$ *and* $\deg[\Lambda^{(2k)}(x)] \leq k$

$$T^{(2k+2)}(x) = \frac{T^{(2k)}(x)}{\Delta^{(2k)}}$$

(c) **Set** $k = k + 1$. If $k < t$ go to step 2.

(d) **Apply** Chien search, test the roots, output status, STOP.

6.4.3 Examples:

1. $(15, 5)$, 3-error correcting binary BCH code (6.6, p 215, L&C)
   - **Receive** $r(x) = x^3 + x^5 + x^{12}$. Then

$$S_1 = S_2 = S_4 = 1 \qquad S_3 = \alpha^{10}$$
$$S_5 = \alpha^{10} \qquad S_6 = \alpha_5$$

   - **Initialize** $k = 0, \quad \Lambda^{(0)} = 1, \quad T^{(0)}(x) = 1.$
   - $\Delta^{(0)}$ is the coefficient of x in

$$\Lambda^{(0)}(x)[1 + S_1 x + \cdots]$$

   So, $\Delta^{(0)} = S_1 = 1$

k	$\Lambda^{(2k)}$	$\Delta^{(2k)}$	$T^{(2k)}$
0	1	$S_1 = 1$	1

$$\begin{aligned}\Lambda^{(2)}(x) &= \Lambda^{(0)}(x) + \Delta^{(0)}[x \cdot T^{(0)}(x)] \\ &= 1 + S_1 \cdot x \cdot 1 \\ &= 1 + S_1 x\end{aligned}$$

- $k = k + 1 = 1$. $\Delta^{(2)} =$ coefficient of x^3 in

$$\Lambda^{(2)}(x)[1 + S_1 x + S_2 x^2 + S_3 x^3 + \cdots].$$

Or $\Delta^{(2)} = S_1 S_2 + S_3 = S_1^3 + S_3 = \alpha^5$. And

$$T^{(2)}(x) = \frac{x \cdot 1}{S_1} = x$$

So, now we have...

k	$\Lambda^{(2k)}$	$\Delta^{(2k)}$	$T^{(2k)}$
0	1	$S_1 = 1$	1
1	$1 + x$	α^5	x

$$\begin{aligned}\Lambda^{(4)}(x) &= \Lambda^{(2)}(x) + \Delta^{(2)}[x \cdot T^{(2)}(x)] \\ &= 1 + x + \alpha^5 \cdot x \cdot x \\ &= 1 + x + \alpha^5 x^2\end{aligned}$$

- $k = k + 1 = 2$ and $\Delta^{(4)} =$ the coefficient of x^5 in

$$\Lambda^{(4)}(x)[1 + S_1 x + S_2 x^2 + S_3 x^3 + S_4 x^4 + S_5 x^5 + \cdots]$$

Or $\Delta^{(4)} = S_5 + S_4 + \alpha^5 \cdot S_3 = \alpha^{10}$.

$$\begin{aligned}T^{(4)}(x) &= \frac{x \cdot \Lambda^{(2)}(x)}{\Delta^{(2)}} \\ &= \alpha^{10} x + \alpha^{10} x^2\end{aligned}$$

k	$\Lambda^{(2k)}$	$\Delta^{(2k)}$	$T^{(2k)}$
0	1	$S_1 = 1$	1
1	$1 + x$	α^5	x
2	$1 + x + \alpha^5 x^2$	α^{10}	$\alpha^{10} x + \alpha^{10} x^2$

- $k = k + 1 = 3$

$$\begin{aligned} \Lambda^{(6)}(x) &= \Lambda^{(4)}(x) + \Delta^{(4)}(x)[x \cdot T^{(4)}(x)] \\ &= 1 + x + \alpha^5 x^2 + \alpha^{10} \cdot x(\alpha^{10} x + \alpha^{10} x^2) \\ &= 1 + \alpha + \alpha^5 x^3 \end{aligned}$$

k	$\Lambda^{(2k)}$	$\Delta^{(2k)}$	$T^{(2k)}$
0	1	$S_1 = 1$	1
1	$1 + x$	α^5	x
2	$1 + x + \alpha^5 x^2$	α^{10}	$\alpha^{10} x + \alpha^{10} x^2$
3	$1 + \alpha + \alpha^5 x^3$	$---$	$---$

2. $(31, 16)$, 3-error correcting binary BCH code.

- The 3-error correcting $(31, 16)$ binary BCH code;
- Consecutive roots are $\alpha, \alpha^2, \ldots, \alpha^6$ where α is primitive in $GF(32)$.

$$r(x) = 1 + x^9 + x^{11} + x^{14}$$

Using $m_\alpha(x) = 1 + x^2 + x^5$ we get

$$\begin{aligned}
S_1 &= r(\alpha) = 1 + \alpha^9 + \alpha^{11} + \alpha^{14} = 1 \\
S_2 &= r(\alpha^2) = 1 \\
S_3 &= r(\alpha^3) = 1 + \alpha^3 \\
S_4 &= 1 \\
S_5 &= \alpha^2 + \alpha^3 \\
S_6 &= 1 + \alpha + \alpha^3
\end{aligned}$$

and

$$\begin{aligned} S(x) &= x + x^2 + (1+\alpha^3)x^3 + x^4 + (\alpha^2+\alpha^3)x^5 + (1+\alpha+\alpha^3)x^6 \\ &= x + x^2 + \alpha^{29}x^3 + x^4 + \alpha^{23}x^5 + \alpha^{27}x^6 \end{aligned}$$

Exercise (optional): Using Berlekamp's iterative method, try to derive the error locator polynomial,

$$\Lambda(x) = 1 + x + \alpha^{16}x^2 + \alpha^{17}x^3.$$

If more than t errors occur...

1. Alg. can terminate with $\Lambda(x)$ of correct degree and roots (RARE).
2. $\Lambda(x)$ can decode to (incorrect but) closest code word.
3. $\Lambda(x)$ will have degree $\nu \le t$ but fewer than ν *distinct* roots, making it an *illegitimate* error locator polynomial.

6.4.4 The Berlekamp-Massey Algorithm for nonbinary codes

- For binary codes, we used Berlekamp's formulation of his decoder.
- For non-binary codes, we will examine *Massey's explanation of Berlekamp's iterative algorithm*.
- Begin with the recursion derived for the PGZ Algorithm:

$$\Lambda_\nu S_{j-\nu} + \Lambda_{\nu-1} S_{j-\nu+1} + \cdots + \Lambda_1 S_{j-1} = -S_j$$

- This describes the operation of a linear feedback shift register (LFSR).

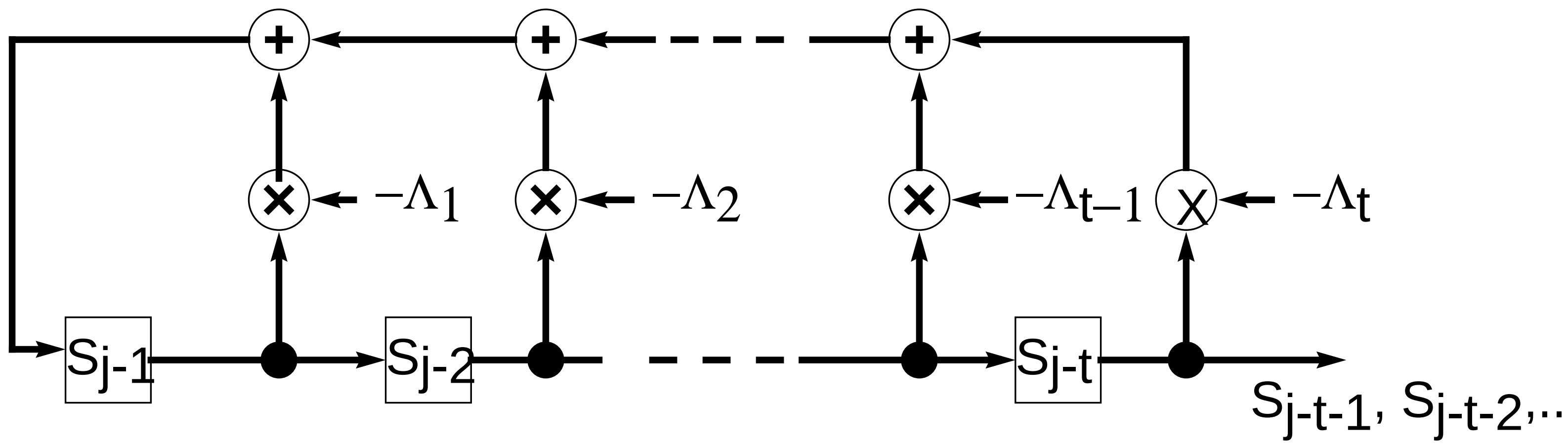


Figure 1: LFSR to genetrate sequence of syndromes.

- Massey showed that *determining coefficients of E.L.P. from syndromes (Berlekamp) is equivalent to synthesizing the minimum length FSR that generates the syndrome sequence.*

- BMA algorithm is used throughout Computer Science to design a minimum length FSR to generate any given sequence.

- This minimum length is often known as the *complexity* of the sequence.

Preliminaries

1. Terminology:

- $\Lambda(x)$ called the *connection polynomial* of the LFSR.
- $T(x)$ is the *correction polynomial*.
- $\Delta^{(2k)}$ is the *discrepancy*.
- L is the length of the LFSR.
- The process is indexed by k.

2. Objective: *Find the* $\Lambda(x)$ *for a LFSR that generates* $S_{t+1}, S_{t+2}, \ldots$ *when initialized with* $S_1, S_2, \ldots, S_t$.

3. Outline of Algorithm:

   (a) Postulate the shortest possible LFSR.

   (b) Try to generate the entire syndrome sequence.

   (c) Compare LFSR output with correct syndromes.

   (d) When discrepancy is observed

      i. modify LFSR according to prescribed rule;

      ii. re-start LFSR

   (e) Continue to the next discrepancy or to the end.

Details of BMA

1. Compute syndromes $S_1, \cdots, S_{2t}$.
2. Initialize:

$$\begin{aligned} k &= 0 \\ \Lambda^{(0)}(x) &= 1 \\ L &= 0 \\ T(x) &= x \end{aligned}$$

3. $k = k = 1$; Compute discrepancy.

$$\Delta^{(k)} = S_k - \sum_{i=1}^{L} \Lambda_i^{(k-1)} S_{k-1}.$$

4. If $\Delta^{(k)} = 0$, GOTO 8. ELSE: continue.

5. Modify connection polynomial.

$$\Lambda^{(k)}(x) = \Lambda^{(k-1)}(x) - \Delta^{(k)} T(x)$$

6. If $2L \geq k$, GOTO 8. ELSE: continue.

7. Change register length; update correction term.

$$\begin{aligned} L &= k - L \\ T(x) &= \Lambda^{(k-1)}(x)/\Delta^{(k)} \\ T(x) &= x \cdot T(x) \end{aligned}$$

8. If $k < 2t$ GOTO 3. ELSE: continue.

9. Solve $\Lambda(x)$.

(a) If roots are distinct and in correct field
  - find error magnitudes;
  - correct corresponding locations in $r(x)$;
  - END

(b) Otherwise
  - Declare decoding FAILURE.
  - STOP