Minimal Polynomial Synthesis of Finite Sequences

Kwan kyu Lee

Abstract

We develop two algorithms that nd a minimal polynomial of a nite sequence. One uses Euclid’s algorithm, and the other is in essence a minimal polynomial version of the Berlekamp-Massey algorithm. They are formulated naturally and proved algebraically using polynomial arithmetic. They connect up seamlessly with decoding procedure of alternate codes.

Key words: Minimal polynomial, nite sequences, alternate codes

1. Introduction

The problem of nding a shortest linear recurrence that generates a nite sequence occupies the central position in decoding alternate codes such as BCH, RS, and Goppa codes. Massey[1] claried this by showing that Berlekamp’s algorithm[2] that is central in decoding BCH codes can best be regarded as a solution of the problem. For this contribution, Berlekamp’s algorithm is now called the Berlekamp-Massey algorithm. On the other hand, Sugiyama et. al. used Euclid’s algorithm as substitute of the Berlekamp-Massey algorithm.

Following Massey’s spirit, we develop algorithms nding a shortest linear recurrence of a nite sequence, and apply these algorithms to decoding alternate codes. However, our algorithms synthesize a minimal polynomial of a nite sequence instead of a connection polynomial and linear complexity pair as the Berlekamp-Massey algorithm does. I believe that this is not a trivial point. Indeed the formulation of the problem in terms of minimal polynomials and characteristic polynomials allows us a natural and algebraic development of algorithms solving the problem, and further eliminates undue complications in the decoding procedure of alternate codes, as we will verify in the following sections.

In Section 3, we use Euclid’s algorithm to formulate an algorithm synthesizing a minimal polynomial of a nite sequence. Obviously the idea comes from the decoding procedure using Euclid’s algorithm by Sugiyama et al. In Section 4, we develop an iterative algorithm in somewhat generic form, doing the same task. Then in Section 5, we show that this iterative algorithm in an explicit form is in essence a minimal polynomial version of the Berlekamp-Massey algorithm. In Section 6, we compare our iterative algorithm with the algorithm using Euclid’s algorithm and nds a way to remove their discrepancy. In Section 7, we brie describe the decoding procedure of alternate codes, where any of minimal polynomial synthesis algorithms we developed is the central component of the procedure. In the nal section, we discuss related works.

2. Preliminaries

Let  be an arbitrary eld. Let  be a nite sequence of length  over . A monic polynomial  is said to be a characteristic polynomial of  if the linear recurrence

\[ s_{i+n}c_i + s_{i+n-1}c_{i+n-1} + \cdots + c_i = 0 \quad (1 \leq i \leq n) \]  

is satised. This means that the linear recurrence together with the initial  elements  satises the whole sequence . We also regard any polynomial of degree \( n \) is trivially a characteristic polynomial. Hence there exists a characteristic polynomial of  of the least degree. The least degree is called the linear complexity of  and any characteristic polynomial of the least
degree is called a minimal polynomial of $s$. In general, a minimal polynomial of a nil sequence is not unique.

Throughout this paper, we consider a nil sequence $s = s_1s_2\ldots s_N$ of length $N$ over $F$. Let $1 \leq s \leq N$. For the subsequence $s_1s_2\ldots s_N$ of $s$, we associate a polynomial

$$S_s(x) = s_1x^{r_1} + s_2x^{r_2} + \ldots + s_Nx^{r_N},$$

whose reciprocal polynomial is

$$S_s(x) = s_1x^{R-N} + s_2x^{R-N} + \ldots s_Nx^{R-N}.$$

For convenience, we identify the subsequence $s_1s_2\ldots s_N$ with the polynomial $2^{s_1}$. For example, we will say $C(x)$ is a characteristic polynomial of $S_s(x)$ for short. We will always work with $S_s(x)$ rather than $S_s(x)$ because of

**Lemma 1.** $C(x)$ is a characteristic polynomial of $S_s(x)$ if and only if

$$C(x)S_s(x) + (V(x))^{n+1} - R(x)$$

for some $V(x)$, where $R(x) \neq 0$. If $C(x)$ and $R(x)$ are respectively the quotient and the remainder of $(x)S_s(x)$ divided by $x^n$, and in particular uniquely determined by $(x)$ and $S_s(x)$.

**Proof.** Let $C(x) = x^n + c_1x^{n-1} + \ldots + c_n$. It suffices to note that for $0 \leq i \leq n-1$, the vanishing of the constant term of $x^n$ in the expression of $S_s(x)$ is equivalent to the relation (1). The last assertion is clear.

Lemma 1 is trivial but is fundamental for all the subsequent results. The remaining lemmas will be used in later sections.

**Lemma 2.** Suppose $C(x)$ of degree $\leq n$ is a characteristic polynomial of $S_s(x)$ so that $V(x)$ and $R(x)$ are as in the previous lemma. Then $C(x)$ is not a characteristic polynomial of $S_{s_p}(x)$ if and only if $\deg(xR(x) + s_{p-1}C(x)) = \deg C(x)$.

**Proof.** Note that

$$C(x)S_{s_p}(x) - (xR(x) + s_{p-1}C(x)) = x(xR(x) + s_{p-1}C(x)) - xR(x) - V(x)^\frac{n+1}{x} - s_{p-1}C(x).$$

Therefore $C(x)S_{s_p}(x) + (V(x))^{n+1} - xR(x) + s_{p-1}C(x)$. Here

$$\deg(xR(x) + s_{p-1}C(x)) \leq \deg \deg C(x) \leq n+1$$

because $\deg R(x) < \deg C(x)$. Therefore $xR(x) + s_{p-1}C(x)$ is the remainder of $C(x)S_{s_p}(x)$ divided by $x^{n+1}$. Now the conclusion follows by Lemma 1.

Let $C(x) = x^n + c_1x^{n-1} + \ldots + c_n$. If $C(x)$ is a characteristic polynomial of $S_n(x)$, then

$$\deg R(x) = \deg C(x).$$

with $\deg R(x) < \deg C(x)$. Let us determine explicitly the coefficient $a$ of $x^i$ in $xR(x) + s_{p-1}C(x)$. Let $R(x) - r_0x^{i-1} - \ldots - r_{n-1}$. Since $r_0 = \deg C(x)$, we have

$$r_0 = c_{p-1} + c_2 + \ldots + c_{p-1} = 0$$

A similar result holds for $C(x)$ a characteristic polynomial of $S_{s_n}(x)$.

**Lemma 3.** Suppose $C(x)$ is a characteristic polynomial of $S_s(x)$ but not of $S_{s_n}(x)$. If $D(x)$ is a characteristic polynomial of $S_{s_n}(x)$, then

$$\deg C(x) + \deg D(x) \geq n+1.$$

**Proof.** By Lemmas 1 and 2, $C(x)S_{s_n}(x) + (V(x))^{n+1} - R(x)$ with $\deg R(x) < \deg C(x)$, but

$$C(x)S_{s_n}(x) + (V(x))^{n+1} - xR(x) + s_{p-1}C(x)$$

with $\deg (xR(x) + s_{p-1}C(x)) = \deg C(x)$. Suppose $D(x)$ is a characteristic polynomial of $S_{s_n}(x)$ so that

$$D(x)x^{n+1} - xR(x) + s_{p-1}C(x)$$

with $\deg (xR(x) + s_{p-1}C(x)) = \deg D(x)$. Now multiply $D(x)$ on both sides of (2) and multiply $C(x)$ on both sides of (3) and subtract the resulting two equations. Then

$$D(x)(V(x) - C(x))x^{n+1} - D(x)(xR(x) + s_{p-1}C(x)) - C(x)R(x)$$

(4) is possible only when

$$n + 1 \leq \deg D(x)(xR(x) + s_{p-1}C(x)) - \deg D(x)C(x).$$

We have

$$C(x)S_{s_p}(x) + V(x)^{n+1} - xR(x) + s_{p-1}C(x).$$

with $\deg R(x) < \deg C(x)$. If $2 \deg C(x) \leq n$, and $C(x)$
and \(V(x)\) are relatively prime, then \(C(x)\) is the unique minimal polynomial of \(S^n_k(x)\).

**Proof.** Suppose \(D(x)\) is a minimal polynomial of \(S^n_k(x)\) so that
\[
D(x)S^n_k(x) + W(x)x^d = P(x)
\]
with \(\deg P(x) < \deg D(x)\). Then
\[
(D(x)V(x) - C(x)W(x))x^d - D(x)R(x) - C(x)P(x)
\]
Note that \(\deg D(x)V(x) - C(x)W(x)) < n\) because
\[
\deg D(x)V(x) - C(x)W(x)) \leq \deg C(x) + \deg D(x) < 2 \deg C(x) \leq n,
\]
\[
\deg C(x)P(x) < \deg C(x) + \deg D(x) \leq 2 \deg C(x) \leq n
\]
Therefore \(D(x)V(x) - C(x)W(x)) = 0\). Since \(D(x)\) and \(V(x)\) are relatively prime, \(C(x)\) divides \(D(x)\). This implies that \(C(x)\) is the unique minimal polynomial of \(S^n_k(x)\).

3. Minimal Polynomial Synthesis Using Euclid's Algorithm

Suppose \(S^n_k(x)\) is not zero. Lemma 1 immediately prompts us to consider Euclid's GCD algorithm partially applied for \(x^n\) and \(S^n_k(x)\), which proceeds as follows:

\[
\begin{align*}
x^n - Q(x)S^n_k(x) + R_0(x) \\
S^n_k(x) - Q(x)R_0(x) + R_1(x) \\
R_0(x) - Q(x)R_1(x) + R_2(x) \\
R_1(x) - Q(x)R_2(x) + R_3(x) \\
R_2(x) - Q(x)R_3(x) + R_4(x) \\
\vdots
\end{align*}
\]

where \(\deg R_0(x) < \deg Q(x)\) for \(1 \leq i \leq k\). It is convenient to let \(R_k(x) = x^n, R_{k-1}(x) = S^n_k(x)\) so that (5) begins with
\[
\begin{align*}
S^n_k(x) = 0 \cdot S^n_k(x) + R_k(x) \\
R_k(x) = 0 \cdot R_k(x) + R_{k-1}(x) \\
R_{k-1}(x) = 0 \cdot R_{k-1}(x) + R_{k-2}(x) \\
\vdots
\end{align*}
\]

Using matrices, we write
\[
\begin{bmatrix}
x^n \\ S^n_k(x)
\end{bmatrix} = 
\begin{bmatrix}
Q(x) & I & \ldots & I \\
I & 0 & \ldots & 0
\end{bmatrix}
\begin{bmatrix}
R_k(x) \\ R_{k-1}(x) \\ \vdots \\ R_1(x) \\ R_0(x)
\end{bmatrix}
\]

Let
\[
\begin{bmatrix}
G_0(x)G_{k-1}(x) \\
H_0(x)H_{k-1}(x)
\end{bmatrix} = 
\begin{bmatrix}
Q(x) & I & \ldots & I \\
I & 0 & \ldots & 0
\end{bmatrix}
\begin{bmatrix}
G_0(x) \\
G_{k-1}(x) \\
\vdots \\
G_1(x) \\
G_0(x)
\end{bmatrix}
\]

with \(G_i(x) = 1, H_i(x) = 0\). It is useful to note
\[
\deg G_0(x) = \sum_{i} \deg Q(x)
\]

\[
\deg G_0(x) + \deg R_0(x) < \deg G_0(x) + \deg R_{k-1}(x) - N.
\]

Now
\[
\begin{bmatrix}
R_{k-1}(x) \\ R_k(x)
\end{bmatrix} = 
\begin{bmatrix}
G_0(x)G_{k-1}(x) \\
H_0(x)H_{k-1}(x)
\end{bmatrix}
\begin{bmatrix}
S^n_k(x) \\ x^n
\end{bmatrix}
\]

Therefore
\[
G_0(x)S^n_k(x) - H_0(x)x^n = (-1)^{k+1} R_0(x)
\]

Hence \(G_0(x)\) is a characteristic polynomial of \(S^n_k(x)\) if the condition
\[
\deg G_0(x) = \sum_{i} \deg Q(x) + \deg R_0(x)
\]

is satisfied. Clearly this condition is eventually satisfied as Euclid's algorithm proceeds.

**Proposition 5.** Let \(k\) be the smallest such that (7) is satisfied. Then \(G_0(x)\), made monic, is a minimal polynomial of \(S^n_k(x)\).

**Proof.** We claim that if \(\deg G_0(x) \leq \deg C(x) + \deg G_{i-1}(x) + -1 \leq i \leq k-1\) and
\[
C(x)S^n_k(x) + V(x)x^n = P(x),
\]

then \(\deg P(x) = \deg C(x) + \deg G_{i-1}(x) + \deg R_i(x)\) if \(G_0(x)\) divides \(C(x)\) or \(\deg P(x) + \deg G_{i-1}(x) + \deg G_0(x) + \deg R_i(x)\) otherwise. Then in either case,
\[
\deg P(x) = \deg C(x) + \deg G_0(x) + \deg R_i(x) \geq \deg C(x)
\]

because \(\deg R_i(x) \geq \deg G_0(x)\). Hence \(C(x)\) is not a characteristic polynomial of \(S^n_k(x)\), by Lemma 1. This proves that \(G_0(x)\) is a minimal polynomial of \(S^n_k(x)\).

We prove the claim by induction. Let \(i = 1\). Since \(G_1(x) + 1\) divides \(C(x)\), we need to show \(\deg P(x) \geq \deg C(x) + \deg R_0(x)\). Note that (8) \(\deg (x) - 1\) gives
\[
V(x)x^n = P(x) - C(x)R_0(x)
\]

Since \(\deg C(x)R_0(x) < \deg G_0(x)R_0(x) - N\), this equation is possible only if \(\deg P(x) \geq \deg C(x)R_0(x) - \deg C(x) + \deg R_0(x)\).

Assume the claim is true for all \(j\) with \(1 \leq j < i\). To prove the claim for \(i\), let \(C(x) = Q(x)G_0(x) + C_1(x)\) with \(\deg C_1(x) < \deg C(x)\). If \(C(x) = -1\), then (8) \(\deg (x) - 6\) with \(k = 1\) gives
\[
(V(x) + Q(x)H_0(x))x^n = P(x) + (-1)^{k+1} R_0(x)
\]
Since $\deg Q(x)R(x) < \deg G_{s_j}(x)R(x) - N$, this equation implies that $\deg P(x) \geq \deg Q(x)R(x) - \deg C(x) - \deg G(x) + \deg R(x)$. If $D(x) \neq 0$, then (8) $Q(x) \cdot (6)$ with $k = i$ gives

$$D(x)S_j(x)+(V(x)\cdot Q(x)R(x)x^{N})P(x)+(-1)^{r}Q(x)R(x)$$

Applying the induction hypothesis to $D(x)$,

$$\deg(P(x)+(-1)^{r}Q(x)R(x)) \geq \deg R(x) + \deg G(x) + \deg R(x)$$

where $j$ is chosen so that $\deg G(x) \leq \deg D(x) - \deg G_{s_j}(x)$. Here note that

$$\text{deg}(Q(x)R(x)) = \text{deg}(C(x)) - \deg G(x) + \deg R(x) < \deg G_{s_j}(x) + \deg R(x) - \deg R_{s_i}(x)$$

Therefore (9) is possible only if $\deg P(x) \geq \deg R_{s_i}(x)$. This proves the claim.

Thus the algorithm given below is valid.

Algorithm E. Let $s$ be a sequence $s_1s_2...s_j$ of length $N$ over $\mathbb{F}$. This algorithm outputs a minimal polynomial of $s$.

E1 (Initialization). If $s$ is a zero sequence, then output 1, and the algorithm terminates.

Otherwise $D(x) \leftarrow x^{N}, D(x)S_j(x)$, and set $G(x) \leftarrow 1, G(x) \leftarrow 0$.

E2 (Division). Compute the quotient $Q(x)$ and the remainder $R(x)$ of $D(x)$ divided by $D(x)$. Set $G(x)G(x)$ and set

$$G(x) \leftarrow G(x)Q(x) + G(x), G(x) \leftarrow \tilde{G}(x)$$

and set $D(x) \leftarrow D(x), D(x) \leftarrow R(x)$.

E3 (Stop condition). If $\deg R(x) < \deg G(x)$, then output $G(x)$ multiplied by the inverse of the leading coefficient of $G(x)$. Otherwise return to E2.

Proposition 6. For any finite sequence $S_j(x)$, there is a unique minimal polynomial $C(x)$ satisfying

$$C(x)S_j(x) + V(x)x^{N} = R(x)$$

with $\deg R(x) < \deg C(x)$, and $\deg C(x) + \deg R(x) < N$.

Proof. Algorithm E finds such a minimal polynomial. To prove uniqueness, let $D(x)$ be a minimal polynomial of $S_j(x)$ such that

$$D(x)S_j(x) + W(x)x^N = P(x)$$

with $\deg P(x) < \deg D(x)$ and $\deg P(x) + \deg D(x) < N$. Then $D(x) \cdot (10) - C(x) \cdot (11)$ gives

$$(D(x)V(x) - C(x)R(x))x^N = D(x)R(x) - C(x)P(x).$$

Note that $\deg(D(x)R(x) - C(x)P(x)) < N$ because $\deg D(x)R(x) = \deg C(x) + \deg R(x) < N$.

Therefore $D(x)V(x) - C(x)R(x) = 0$. Note that $C(x)$ and $V(x)$ are relatively prime because otherwise dividing by the common factor in (10), we see $C(x)$ is no longer a minimal polynomial of $S_j(x)$. Then it follows that $C(x) = D(x)$.

4. Iterative Minimal Polynomial Synthesis

Let $1 \leq r < N$. Suppose $C(x)$ of degree $r$ is a characteristic polynomial of $S_j(x)$ so that

$$C(x)S_j(x) + V(x)x^{r} = R(x)$$

with $\deg R(x) < \deg C(x)$. Then as we have done earlier,

$$C(x)S_j(x) + V(x)x^{r-1} - xR(x) + C(x)R(x) = 0$$

Let $a$ be the coefficient of $x$ in $xR(x) + C(x)$. If $a \neq 0$, then $C(x)$ is also a characteristic polynomial of $S_j(x)$ by Lemma 2. Assume $a = 0$. Then $C(x)$ is not a characteristic polynomial of $S_j(x)$. An important idea is that we can overcome this difficulty by exploiting previous $R(x)$ that is a characteristic polynomial of $S_j(x)$ but not of $S_j(x)$ for some $m > r$. So we suppose

$$B(x)S_j(x) + U(x)x^{m} = Q(x)$$

with $\deg Q(x) < \deg B(x)$ but $\deg(U(x)x^{m} + C(x)) = \deg B(x)$. Now with $S_j(x) = x^{m}B(x) + S_j(x)$, we have

$$B(x)S_j(x) - B(x)x^{m}B(x) + S_j(x) + S(x)B(x)$$

Therefore

$$B(x)S_j(x) + U(x)x^{m} - x^{m-n}Q(x) + S(x)B(x)$$

Observe that if $b$ is the leading coefficient of $xQ(x) + x^{m-n}B(x)$, then $b$ is still the leading coefficient of $x^{m-n}Q(x) + S(x)B(x)$, and  $\deg(x^{m-n}Q(x) + S(x)B(x)) = \deg(B(x)) + m-n$.  

조선자연과학논문집 제3권 제2호, 2008
To summarize, let $v = \deg C(x)$ and $w = \deg B(x)$ assuming $w \leq v$: Let $K(x) = xR(x) + s_{p,0}(x)$ and $J(x) = x^{m-1}(Q(x) + S(x)B(x))$. Then (12) and (13) are again

\begin{align*}
C(x)S_{p,0}(x) + V(x)x^{m-1} &= K(x) \\ B(x)S_{p,1}(x) + U(x)x^{p-1} &= J(x)
\end{align*}

with $\deg K(x) = v$ and $\deg J(x) = w + n - m$. Let $a$ and $b$ be the leading coefficients of $K(x)$ and $J(x)$, respectively. Let $c = a/b$.

If $v \leq u + n - m$, we choose a monic polynomial $N(x)$ of degree $(w + u + n - m) - v$ so that $\deg(N(x)K(x) - cJ(x)) = w + u + n - m$. Then $N(x) \cdot (14) - c \cdot (15)$ gives

$$(M(x)C(x) - cB(x))S_{p,0}(x) + (M(x)V(x) - cU(x))x^{m-1} = K(x)$$(14)

and $N(x)$ is a minimal polynomial of $S_{p,0}(x)$.

As $\deg(M(x)C(x) - cB(x)) = w + u + n - m$, we find $M(x)C(x) - cB(x)$ is a characteristic polynomial of $S_{p,0}(x)$.

If $v > u + n - m$, we choose a monic polynomial $M(x)$ of degree $v - (w + u + n - m)$ so that $\deg(K(x) - cN(x)J(x)) < v$. This time $(14) - c \cdot (15)$ gives

$$(C(x) - cN(x)B(x))S_{p,0}(x) + (V(x) - cN(x)U(x))x^{m-1} = K(x) - cN(x)J(x)$$(15)

As $\deg(C(x) - cN(x)B(x)) = v$, we find $C(x) - cN(x)B(x)$ is a characteristic polynomial of $S_{p,0}(x)$.

From this discussion, there emerges an iterative algorithm that constructs a characteristic polynomial $C(x)$ of $S_{p,0}(x)$ successively from $n - 1$ to $n - N$, while maintaining the last $B(x)$ that is a characteristic polynomial of $S_{p,0}(x)$ but not of $S_{p,1}(x)$. Below we will show that these characteristic polynomials are in fact minimal polynomials. But first we need to discuss the initialization of the algorithm.

If $s$ is a zero sequence, then clearly 1 is the unique minimal polynomial of $s$. Otherwise let $1 \leq s \leq N$ be the smallest such that $s$ is not zero. Then $B_{s-1}$ is the unique minimal polynomial of the zero sequence $s_{s-1}s_{s-2} \ldots s_1$ so that

$$B_{s-1}(s_{s-1}S_{s-1}(x) - U_{s-1}(x)) = Q_{s-1}(x),$$

where $S_{s-1}(x) = 0$, $U_{s-1}(x) = 0$, and $Q_{s-1}(x) = 0$. As required, $B_{s-1}(x)$ is not a characteristic polynomial of $S_{s-1}(x)$ because

\begin{align*}
\deg(Q_{s-1}(x) + s_{s-1}B_{s-1}(x)) &= 0 - 0 = \deg B_{s-1}(x).
\end{align*}

On the other hand, any monic polynomial of degree 1 is a minimal polynomial of the sequence $s_1s_2 \ldots s_p$. Indeed for any monic polynomial $C_{s,1}(x)$ of degree 1,

$$C_{s,1}(x)S_{s,1}(x) + V_{s,1}(x)x = R_{s,1}(x),$$

where $S_{s,1}(x) = s_1$, $V_{s,1}(x) = -s_1$, and $R_{s,1}(x) = s_1C_{s,1}(x) = s_1x$, we now are ready to introduce an iterative minimal polynomial synthesis algorithm.

**Algorithm 1.** Let $s$ be a sequence $s_1s_2 \ldots s_p$ of length $N$ over $\mathbb{F}$. This algorithm outputs a minimal polynomial of $s$.

1. (Initialization). If $s$ is a zero sequence, then output 1, and the algorithm terminates. Otherwise set $m \leftarrow n - 1$, $m \leftarrow r$, $b \leftarrow s_p$, $B_{s,0}(x) \leftarrow 1$, $Q_{s,0}(x) \leftarrow 0$, $C_{s,0}(x) \leftarrow C_{s,0}(x)$, $R_{s,0}(x) \leftarrow R_{s,0}(x)$.

2. (Beginning of iteration). Set $K_{s,0}(x) \leftarrow xB_{s,0}(x) + s_pC_{s,0}(x)$.

3. (No adjustment). If $m = 0$, then set $R_{s,0}(x) \leftarrow K_{s,0}(x)$, and go to step 6. Otherwise set $J_{s,0}(x) \leftarrow x\deg(Q_{s,0}(x))B_{s,0}(x)$ where $x^{\deg(J_{s,0}(x))} = 1$.

4. (Jump). Let $u = \deg B_{s,0}(x)$, $v = \deg C_{s,0}(x)$. Let $c = a/b$. If $v > u + n - m$, then choose a monic polynomial $N(x)$ of degree $v - (u + n - m)$.

5. (Adjustment). If $v > u + n - m$, then choose a monic polynomial $N(x)$ of degree $v - (u + n - m)$.

6. (End of iteration). Set $m \leftarrow m - 1$. If $m < N$, then return to step 1. Otherwise output $C_{s,0}(x)$, and the algorithm terminates. Notice that $B_{s,0}(x)$ is updated only in step 4. Note also that we did not specify the initial $C_{s,0}(x)$ in step 1 and how we choose $N(x)$ in steps 4 and 6. The algorithm behaves differently depending on how we specify these. However, regardless of these specifications, Algorithm 1 works correctly. We now make an important observation.  

**Lemma 7.** As the algorithm iterates from $m = 1$ up to $m = N$, the equality $u + v + m + 1$ always holds at the beginning of each iteration.

**Proof.** At the beginning of the $r$th iteration, that is, just after the initialization, we have $m = n - 1$, $m_r = r$, $u = 0$, $v = r$. So $u + v + m + 1$. Now assuming $u + v + m + 1$ at the beginning of the current iteration, we show that the equality still holds after finishing the current iteration. If the algorithm gets into step 13 or 15, then there occurs no change.
to \(u, v, \) and \(m,\) and hence the equality holds. If the algorithm gets into 14, then variables change as \(u \rightarrow u', v \rightarrow u'n - m', m \rightarrow n',\) where prime indicates the old value of each variable. So \(u'v + u'n + m'n - m + 1 = u'n - m'n + 1 + m' + 1.\) Thus the equality still holds.

**Proposition 8.** As the algorithm iterates from \(n - 1\) up to \(n - N,\) \(C(x)\) always holds a minimal polynomial of the subsequence \(s_1, s_2, \ldots, s_n\) of \(s.

**Proof.** First note that the linear complexity of \(S^*_2(s)\) is monotone increasing function of \(n.\)

It is clear that \(C(x)\) is a minimal polynomial of \(S_2(s)\) just after the initialization. Now we show that assuming that \(C(x)\) is a minimal polynomial of \(S_2^*(s)\) at the beginning of an iteration, the new \(C(x)\) at the end of the iteration is a minimal polynomial of \(S_{n-1}(s).\)

If the algorithm follows 13, then \(C(x)\) is still a characteristic polynomial of \(S_{n-1}(s).\) Since \(C(x)\) is a minimal polynomial of \(S_{n-1}^*(s),\) it is clear that \(C(x)\) is also a minimal polynomial of \(S_{n-1}^*(s).\)

If the algorithm follows 14, then \(N(x) \cdot C(x) \cdot B(x)\) is a characteristic polynomial of \(S_{n-1}^*(s).\) Since \(\deg(N(x) \cdot C(x) \cdot B(x)) = \deg(N) + \deg(C) + \deg(B) + \deg(A) - \deg(A) = \deg(A) + \deg(B) + 1,\) we see the linear complexity \(l\) of \(S_{n-1}^*(s)\) satisfies \(Es(n-1, m)\). By Lemma 3, it also holds that \(f \leq n+1.\)

Now \(u \cdot n \geq 2u+1 \cdot n-r-1 = n-1 - u \cdot n - m,\) Therefore \(f \leq n-1.\) Hence \(M(x) \cdot C(x) \cdot B(x)\) is a minimal polynomial of \(S_{n-1}^*(s).\)

If the algorithm follows 15, then \(C(x) \cdot \bar{S}(x) / N(x) \cdot B(x)\) is a characteristic polynomial of \(S_{n-1}^*(s).\) Since \(\deg(C(x) \cdot \bar{S}(x) / N(x) \cdot B(x)) = \deg(C) + \deg(S) + \deg(B) + \deg(A) - \deg(A) = \deg(A) + \deg(B) + 1,\) it is a minimal polynomial of \(S_{n-1}^*(s).\) If \(C(x) \cdot \bar{S}(x) / N(x) \cdot B(x)\) is a minimal polynomial of \(S_{n-1}^*(s),\) it follows that \(C(x) \cdot \bar{S}(x) / N(x) \cdot B(x)\) is a minimal polynomial of \(S_{n-1}^*(s).\)

As we said earlier, the initial \(C(x)\) in \(H\) can be arbitrary monic polynomial of degree \(r.\) But the simplest choice may be \(C(x) = x^r.\) In 14, the simplest choice of a monic polynomial \(M(x)\) may be \(x^{u'[n-r-1]}\). Similarly in 15, the simplest choice of \(M(x)\) may be \(x^{u'[n-r]}\). Moreover with these choices, we do not need to compute and store \(J(x)\) and \(K(x)\) because only their leading coefficients are really necessary to determine the course of the algorithm. Then also \(B(x)\) and \(Q(x)\) become redundant. With these considerations, we may formulate a concrete simple form of the iterative algorithm.

**Algorithm J.** Let \(s\) be a sequence \(s_1, s_2, \ldots, s_N\) of length \(N\) over \(F.\) This algorithm outputs a minimal polynomial of \(s.\)

**J1** (Initialization). Set \(r \leftarrow 1.\) Repeat \(r \leftarrow r + 1\) until \(s_r = 0.\) If \(r > N\) then output 1, and the algorithm terminates. Otherwise set \(m \leftarrow r, b \leftarrow s_r, B(x) \leftarrow 1,\) and \(C(x) \leftarrow x.\)

**J2** (Beginning of iteration). Let \(C(x) = x^{u'[n-r-1]} + \ldots + c_0.\) Set \(a = s_{r-1} + c_1 s_{r+1} + \ldots + c_{r-2} s_{2r-1}.\)

**J3** (No adjustment). If \(a \neq 0,\) then go to J6.

**J4** (Jump). Let \(v - \deg(B(x)), v' - \deg(C(x)).\) Let \(c = ab.\) If \(v \leq s_r - m,\) then set \(C(x) = C(x) + c \cdot B(x)\) and set \(C(x) = x^{u'[n-r-m]} C(x) - c B(x),\) and set \(m \leftarrow n, b \leftarrow c, B(x) \leftarrow C(x).\)

**J5** (Adjustment). If \(v > s_r - m,\) then set \(C(x) = C(x) - c x^{u'[n-r-m]} B(x).\)

**J6** (End of iteration). Set \(m \leftarrow r + 1.\) If \(r < N,\) then return to J2. Otherwise output \(C(x),\) made monic and the algorithm terminates.

For J2, see the remark just below Lemma 2.

5. Toward the Berlekamp-Massey Algorithm

We can further simplify Algorithm J by replacing the initialization step J1 with

**J1′** Set \(m \leftarrow 1, n \leftarrow 0, b \leftarrow 1, B(x) \leftarrow 1, C(x) \leftarrow 1.\)

If we trace the initial behavior of the algorithm J with J1′, it turns out that collectively the algorithm automatically initializes itself with

\[m \leftarrow r + 1, n \leftarrow r, b \leftarrow s_r, B(x) \leftarrow 1, C(x) \leftarrow x - s_r,\]

when it reaches the rth nonzero \(s_r.\) This is a valid initialization because any monic polynomial of degree \(r\) can be the initial \(C(x).\)

Now we almost perceive the famous Berlekamp-Massey algorithm from the algorithm J with J1′. Indeed it is now quite an easy matter to convert this algorithm to the Berlekamp-Massey algorithm. The main ingredient of this conversion is to use a connection polynomial instead of a characteristic polynomial. Recall that a connection polynomial is the reciprocal polynomial of a characteristic polynomial. Not to lose any information that a characteristic polynomial carries, a connection polynomial should be paired with the degree of the corresponding characteristic polynomial.

조선자연과학논문집 제1권 제1호, 2008
We will call a connection polynomial corresponding to a minimal polynomial a minimal connection polynomial. We carry out the conversion in two stages. On the first stage, we make the algorithm J with JH' work with the pair of the connection polynomial \( C(x) \) and the degree \( v \) instead of the characteristic polynomial \( C(x) \) of degree \( v \):

\[
C(x) = x^n + c_1 x^{n-1} + \cdots + c_v x^v
\]

This is our first version of the Berlekamp-Massey algorithm.

Algorithm BM1: Let \( s \) be a sequence \( s_0, s_1, \ldots, s_N \) of length \( N \) over \( \mathbb{F} \). This algorithm outputs a minimal connection polynomial and the linear complexity of \( s \).

- BM1 (Initialization). Set \( m \leftarrow 1, \ b \leftarrow 0, \ v \leftarrow 0, \ B^*(x) \leftarrow 1, \ C(x) \leftarrow 1 \).
- BM2 (Beginning of iteration). Let \( C^*(x) = 1 + c_1 x + \cdots + c_v x^v \). Set \( a \leftarrow s_0, b \leftarrow c_1, c_1, \ldots, c_v, 0 \).
- BM3 (No adjustment). Set \( a = 0 \). Then return to BM6.
- BM4 (Jump). Let \( c = a b \). If \( v = u + n - m \), then set \( T(x) = C(x), \ v \leftarrow v \) and set

\[
C^*(x) = C(x) - c B^*(x), \ v \leftarrow u + n - m
\]

and set

\[
C^*(x) = C^*(x) - c_1 B^*(x), \ v \leftarrow n - v
\]

and set \( r = 0, b = a, B^*(x) = T(x) \).
- BM5 (Adjustment). If \( 2 \nmid \bar{n} \), then set

\[
C^*(x) = C(x) - c B^*(x).
\]

BM6 (End of iteration). Set \( \bar{n} \leftarrow \bar{n} + 1, \ n \leftarrow r + 1 \). If \( n \leq N \), then return to BM2. Otherwise output the pair \( \{ C(x), v \} \), and the algorithm terminates.

This is exactly the Berlekamp-Massey algorithm.

6. Iterative Minimal Polynomial Synthesis with Euclid’s Algorithm

Algorithms E and J do the same thing—they nd a minimal polynomial of the given finite sequence. Then it is quite frustrating to nd that the minimal polynomials they output are not always identical. That is, often they nd different minimal polynomials of the same finite sequence. Of course, this does not prove that one of the algorithms works incorrectly because there can be many minimal polynomials of a finite sequence in general. Recall that the minimal polynomial Algorithm E outputs is characterized in Proposition 6.

We can resolve this discrepancy by specializing Algorithm I in a different way. We choose the same \( C_{\bar{x}}(x) = x^\bar{n} \) as in Algorithm J. But this time, in I4 we choose \( N(x) \) to be the quotient of \( c_{\bar{x}}(x) \) divided by \( K(x) \) and in I5 we choose \( N(x) \) to be the quotient of \( K(x) \) divided by \( c_{\bar{x}}(x) \). Of course, this amounts to employing the expensive Euclid’s algorithm in choosing \( N(x) \) so that the resulting algorithm becomes much less efficient than Algorithm J. But still the result is interesting. Now we introduce Algorithm K that always outputs the same minimal polynomials with Algorithm E.

Algorithm K. Let \( s \) be a sequence \( s_0, s_1, \ldots, s_N \) of length \( N \) over \( \mathbb{F} \). This algorithm outputs a minimal polynomial of \( s \).

- K1 (Initialization). Set \( r \leftarrow 1 \). Repeat \( r \leftarrow r + 1 \) until \( s \neq 0 \). If \( r > N \) then output 1, and the algorithm terminates. Otherwise set \( m \leftarrow r - 1, \ n \leftarrow r, \ b = s_0, \ B(x) \leftarrow 1, \) and \( C(x) \leftarrow x \). If \( r = N \) then output \( C(x) \), and the algorithm terminates.
- K2 (Beginning of iteration). Set \( K(x) = x R(x) + s_{r-1} C(x) \).
Set \( a \leftarrow \) the coefficient of \( x^i \) in \( K(x) \) where \( i = \deg C(x) \).

**K3** (No adjustment). If \( a = 0 \), then set \( R(x) \leftarrow K(x) \), and go to **K6**. Otherwise set \( J(x) \leftarrow x^{n-m} R(x) + S(x) R(x) \) where \( S(x) = x^{e_n} \cdot x^{e_{n-1}} \cdot \ldots \cdot x^{e_2} \cdot x^{e_1} \).

**K4** (Jump). Let \( t \leftarrow \deg R(x) \), \( v = \deg C(x) \), and \( c = a/b \). If \( v \geq n-m \), then compute the quotient \( M(x) \) of \( cJ(x) \) divided by \( K(x) \). Set \( C(x) \leftarrow C(x) \), and \( R(x) \leftarrow R(x) \), and set:

\[
C(x) = N(x) C(x) - cR(x),
\]

\[
R(x) = N(x) K(x) - cI(x),
\]

and set \( m \leftarrow n \), \( b \leftarrow a \), \( B(x) \leftarrow C(x) \), \( Q(x) \leftarrow R(x) \).

**K5** (Adjustment). If \( v + n - m \), then compute the quotient \( N(x) \) of \( K(x) \) divided by \( cJ(x) \). Set

\[
C(x) = C(x) - c\cdot M(x) R(x),
\]

\[
R(x) = K(x) - c\cdot N(x) J(x),
\]

and set \( m \leftarrow m+1 \), \( r \leftarrow r+1 \). If \( m \geq n \), then return to **K2**. Otherwise output \( C(x) \), and the algorithm terminates.

**Proposition 9.** Algorithms **K** and **E** output the same minimal polynomial of a finite sequence.

**Proof.** We will show that after the initialization and after finishing each iteration, \( \deg C(x) = \deg R(x) = n \).

Then \( C(x) \) is the unique minimal polynomial characterized in Proposition 6. Therefore Algorithms **E** and **K** necessarily output the same minimal polynomial.

After the initialization, \( C(x) = x^0 \cdot R(x) = 0 \), \( r = r \). Hence \( \deg C(x) = \deg R(x) = -\infty \).

Suppose the algorithm follows **K3**. If \( s_{n-1} \) was not zero, then \( a = 0 \) is possible only if \( \deg R(x) = \deg C(x) = 1 \). So \( n \geq \deg R(x) + \deg C(x) + 2 \). After setting \( R(x) \leftarrow K(x) \), we have \( \deg R(x) = \deg C(x) + 2 \). If \( s_{n-1} \) was zero, then \( R(x) \) is set to be \( x^0 \). Hence \( \deg R(x) = \deg C(x) + 1 \). In either case, \( \deg C(x) = \deg R(x) = n \).

If the algorithm follows **K4**, then \( \deg C(x) = u \), \( \deg R(x) = v \). Therefore \( \deg C(x) = \deg R(x) = u + v - m + 1 \) because \( u + v - m + 1 \).

If the algorithm follows **K5**, then \( \deg C(x) = v \), \( \deg R(x) = u - n - m \). Therefore \( \deg C(x) = \deg R(x) = u - n - m + 1 \) because \( u + v - m + 1 \).

In any case, after updating \( m \leftarrow m + 1 \), we have \( \deg C(x) = \deg R(x) = n \).

7. Decoding Alternant Codes

Recall that a minimal polynomial \( C(x) \) of a finite sequence \( S(x) \) is a monic polynomial of the least possible degree such that \( C(x) S(x) + V(x) R(x) \) for some uniquely determined \( V(x) \) and \( R(x) \) with \( \deg R(x) < \deg C(x) \). We have hitherto developed algorithms that output such a \( C(x) \). But it is natural and straightforward to modify this algorithm to output \( V(x) \) and/or \( R(x) \) together with a minimal polynomial \( C(x) \). Indeed, by adapting Algorithm **E** appropriately, we obtain:

**Algorithm F.** Let \( s \) be a sequence \( s_1, s_2, \ldots, s_N \) of length \( N \) over \( \mathbb{F} \). This algorithm outputs \( (C(x), V(x)) \) so that \( C(x) S(x) + V(x) R(x) \) where \( C(x) \) is a minimal polynomial of \( s \) and \( \deg R(x) < \deg C(x) \).

**F1** (Initialization). If \( s \) is a zero sequence, then output 1, and the algorithm terminates. Otherwise set

\[
\tilde{D}(x) = x^n D(x) - S(x)^n, 
\]

\[
G(x) = H(x) = 0, I(x) = 1
\]

**F2** (Division). Compute the quotient \( Q(x) \) and the remainder \( R(x) \) of \( \tilde{D}(x) \) divided by \( D(x) \). Set

\[
G(x) = G(x) + \tilde{G}(x), \quad H(x) = H(x) + \tilde{H}(x)
\]

and set \( \tilde{D}(x) = D(x) \).

**F3** (Stop condition). If \( \deg R(x) < \deg G(x) \), then output \( G(x) \) and \( H(x) \), both multiplied by the inverse of the leading coefficient of \( G(x) \). Otherwise return to **F2**.

It is equally straightforward to modify the iterative algorithm I to output both \( C(x) \) and \( V(x) \). But this time we try to formulate a practical algorithm as efficient as possible in implementation. So we first modify Algorithm I to output \( V(x) \) as well as \( C(x) \), and then specialize the modified algorithm as we did to get Algorithm J, but now using \( J' \) as the initialization step. Then we make several minor adjustments for the sake of efficiency. We use the fact \( \alpha = \mu = 1 \) to remove variables \( \alpha \) and \( m \). We make a variable change \( \beta = m + 1 \). We use variable \( v \) to track the degree of \( C(x) \). We modify L2, L4 and L5 to remove expensive division operations. (This modification certainly involves non-monic characteristic and minimal polynomials. But this does not affect any of the results above.) Finally we modify L6 to make the out-
put $C(x)$ monic. The resulting algorithm is

Algorithm L. Let $s$ be a sequence $s_1, s_2, \ldots, s_N$ of length $N$ over $\mathbb{F}$. This algorithm outputs $C(x)$ and $V(x)$ so that $C(x)S'(x) + V(x)x^s = H(x)$, where $C(x)$ is a minimal polynomial of $s$ and $\deg R(x) = \deg C(x)$.

L1 (Initialization). Set $b \leftarrow 1, b' \leftarrow 1, v \leftarrow 0, B(x) \leftarrow 1$, $C(x) \leftarrow 1$, $U(x) \leftarrow 1 - V(x), V(x) \leftarrow 0$.

L2 (Beginning of iteration). Let $C(x) = c_0 x^N + \ldots + c_{N-1}$, set $\alpha = c_0 x^{N-1} + \ldots + c_{N-1}$, and set $b = c_0 x^s + \ldots + c_{N-1}$.

L3 (No adjustment). If $v = 0$, then go to L6.

L4 (Jump). If $2v > b$, then set $C(x) \leftarrow C(x) - V(x)$ and $V(x) \leftarrow V(x) - V(x)$, and set $C(x) \leftarrow C(x) - V(x)$ and $V(x) \leftarrow V(x) - V(x)$.

L5 (Adjustment). If $2v \geq b$, then set $C(x) \leftarrow C(x) - V(x)$ and $V(x) \leftarrow V(x) - V(x)$.

L6 (End of iteration). Set $b \leftarrow b - 1$, then return to L2. Otherwise set $C(x) \leftarrow C(x) - V(x)$ and $V(x) \leftarrow V(x)$ where $c$ is the inverse of the leading coefficient of $C(x)$, and output $C(x)$ and $V(x)$, and the algorithm terminates.

Now we review the alternating codes. Let $\mathbb{F}_K$ be a field extension of degree $m$. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be distinct elements of $\mathbb{F}$. Let $y_1, y_2, \ldots, y_n$ be nonzero elements of $\mathbb{F}$. Let $r$ be a positive integer. The alternating code $A(\alpha_1, \alpha_2, \ldots, \alpha_n, y_1, y_2, \ldots, y_n, r)$ is the linear code over $K$ having parity check matrix

$$H = \begin{bmatrix}
y_1 & y_2 & \cdots & y_n \\
y_1 & y_2 & \cdots & y_n \\
y_1 & y_2 & \cdots & y_n \\
y_1 & y_2 & \cdots & y_n \\
\end{bmatrix}$$

The alternating code has parameters $[n, k, d]$ with $n - mr < k < n - m(r + 1)$. For more details, see [4].

Suppose that a codeword $c = [c_1, c_2, \ldots, c_n]$ was sent through the channel, and $\nu$ errors occurred during transmission. The received word is thus

$$d = (d_1, d_2, \ldots, d_n) = (c_1 \pm e_1, c_2 \pm e_2, \ldots, c_n \pm e_n),$$

where $e = (e_1, e_2, \ldots, e_n)$ is the error word. We assume $2\nu < 2r$. The decoder can recover the codeword $c$ by performing the decoding procedure below.

The syndromes $S_0, S_1, \ldots, S_{n-1}$ are computed from the received word $d$ by

$$[S_0, S_1, \ldots, S_{n-1}] = [d_1, d_2, \ldots, d_n]$$

The syndromes are actually only dependent on the error word $e$,

$$[S_0, S_1, \ldots, S_{n-1}] = [e_1, e_2, \ldots, e_n]$$

as $d \equiv c \pm e$ and $c$ is a codeword so that $cH^T = 0$. That is,

$$S_j = \sum_{i=0}^{n-1} c_i j_i j_{i+1} \ldots j_{i+r-1} (0 \leq i \leq r-1).$$

Now let $j_1, j_2, \ldots, j_r$ be the indices for which $e_j$ is nonzero. Let

$$X_1 = \alpha j_1, X_2 = \alpha j_2, \ldots, X_r = \alpha j_r,$$

$$Y_1 = e_1 j_1, Y_2 = e_2 j_2, \ldots, Y_r = e_r j_r.$$

Then for $0 \leq i \leq r-1$,

$$S_j = \sum_{k=1}^{n} Y_k X_k^i.$$

Let us denote

$$L(x) = (x - X_1)(x - X_2) \ldots (x - X_r),$$

and

$$S^q(x) = S_0 x^q + S_1 x^{2q} + \ldots + S_{n-1} x^{(n-1)q}$$

Now observe that

$$L(x)S^q(x) = \prod_{i=1}^{r} (x - X_i)^{\sum_{i=0}^{n-1} Y_k X_k^i} X_k^{\sum_{i=0}^{n-1} Y_k X_k^i}$$

$$= \prod_{k=1}^{n} \sum_{i=0}^{n-1} X_k^i \prod_{k=1}^{n} \sum_{i=0}^{n-1} Y_k X_k^i$$

$$= \sum_{k=1}^{n} \sum_{i=0}^{n-1} X_k^i \prod_{k=1}^{n} \sum_{i=0}^{n-1} Y_k X_k^i$$

$$= \sum_{k=1}^{n} \sum_{i=0}^{n-1} Y_k X_k^i \prod_{k=1}^{n} (x - X_k) \prod_{k=1}^{n} (x - X_k)$$

Let

$$E(x) = -\sum_{k=1}^{n} Y_k \prod_{k=1}^{n} (x - X_k),$$

$$R(x) = -\sum_{k=1}^{n} Y_k X_k^i \prod_{k=1}^{n} (x - X_k)$$

Then we get the key equation

$$L(x)S^v(x) = E(x)x^v - R(x)$$

with $\deg R(x) = \deg L(x) = v$. So we see $L(x)$ is a characteristic polynomial of $S^v(x)$. Moreover 2 $\deg L(x) = 2v - sr$ and $L(x)$ and $E(x)$ are relatively prime.

So Lemma 4 implies that $L(x)$ is the unique minimal polynomial of $S^v(x)$.

Any of algorithms $F$ for $L$ applied to the finite sequence $S_0, S_1, \ldots, S_v$, will find $L(x)$ and $E(x)$. Then the number $v$ of errors can be computed by the degree of $L(x)$. The locations $L(x)$ of errors can be found by finding roots of $L(x)$. The error values $e_j$ are then computed by Forney’s formula

$$e_j = \frac{E(X_j)}{y_j L'(X_j)}$$

which comes from

$$E(X_j) = -\prod_{i=1}^v (X_j - X_i) = -y_j L'(X_j)$$

where $L(x)$ is the formal derivative of $L(x)$.

8. Discussion

The Berlekamp-Massey algorithm was invented almost thirty years ago. And there have been arduous efforts to better understand this important algorithm and improve its performance.

After Sugiyma et al. discovered that Euclid’s algorithm can be used to decode Goppa codes, replacing the Berlekamp-Massey algorithm, much effort was put to prove that both algorithms are internally equivalent as well as externally solving the same problem. See [5-8, 11-13]. Clearly Algorithm $E$ corresponds to Sugiyma et al.’s idea, and Algorithm $I$ corresponds to the Berlekamp-Massey algorithm. Explicit forms of Algorithm $I$ such as $J$ and $L$ does not necessarily nd the unique minimal polynomial characterized in Proposition 6 as Algorithm $E$ necessarily does. As they look different even externally, my opinion is that the two algorithms is hardy equivalent.

There are numerous algorithms based on Euclid’s algorithm. One of them is the closest to our work seems [9]. In it, McEliece and Shearer showed that Euclid’s algorithm can be used to nd the Padé approximants of a power series, interpreting Sugiyma et al.’s work. See also [10].

Many algorithms similar to and variants of the Berlekamp-Massey algorithm have been proposed. Some of them are [8, 9, 11-13]. In particular, Norton’s work seems to have much in common with the present work. His Algorithm $MR$ also output a minimal polynomial of a finite sequence and seems to be derived from essentially the same principle as our iterative algorithm $I$, though his exposition is much complicated.

Note that that the polynomial $L(x)$ we derived in (16) to locate errors is the reciprocal polynomial of the conventional error locator polynomial. It is now quite clear that using $L(x)$ leads to simpler decoding procedure at least in its description, if used with minimal polynomial synthesis algorithms such as $F$ and $L$. I nd that some authors already adopted $L(x)$ as the error locator polynomial. See [10, 12] for example.

References


