J. of the Chosun Natural Science
Vol. 1, No. 2 (2008) pp. 149 – 159

Minimal Polynomial Synthesis of Finite Sequences

Kwan kyu Lee[†]

Abstract

We develop two algorithms that nd a minimal polynomial of a finite 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 connects up seamlessly with decoding procedure of alternant codes.

Key words : Minimal polynomial, nite sequences, alternant codes

1. Introduction

The problem of nding a shortest linear recurrence that generates a nite sequence occupies the central position in decoding alternant 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.[3] used Euclid's algorithm as a 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 alternant 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 alternant 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 briey describe the decoding procedure of alternant 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.

Department of Mathematics, Chosun University, Gwangju 501-759, Korea

[†]Corresponding author: kwankyu@chosun.ac.kr
(Received : August 14, Revised : August 29, 2008, Accepted : September 5, 2008)

2. Preliminaries

Let $\mathbb{F}$ be an arbitrary eld. Let $s = s_1,s_2,...,s_n$ be a nite sequence of length n over $\mathbb{F}$. A monic polynomial $C(x) = x^v+c_1x^{v-1}+ \cdots c_{v-1}x+c_{v-1}x+c_v$ over $\mathbb{F}$ is said to be a characteristic polynomial of s if the linear recurrence

$$s_{i+v}+c_1s_{i+v-1}+ \cdots +c_vs_i = 0 \ (1\le i\le n-v) \tag{1}$$

is satised. This means that the linear recurrence together with the initial v elements $s_1,s_2,...,s_v$ denes the whole sequence s. We also regard any polynomial of degree$\ge n$ is trivially a characteristic polynomial. Hence there exists a characteristic polynomial of s of the least degree. The least degree is called the linear complexity of s, and any characteristic polynomial of the least degree is called a minimal polynomial of s. In general, a minimal polynomial of a nite sequence is not unique.

Throughout this paper, we consider a nite sequence $s=s_1,s_2,\ldots,s_N$ of length N over $\mathbb{F}$. Let $1\le n\le N$. For the subsequence $s_1,s_2,\ldots,s_n$ of s, we associate a polynomial

$$S_n(x)=s_1+s_2x+\cdots+s_nx^{n-1},$$

whose reciprocal polynomial is

$$S_n^*(x)=s_1x^{n-1}+s_2x^{n-2}+\cdots s_n$$

For convenience, we identify the subsequence $s_1,s_2,\ldots,s_n$ of s with the polynomial $S_n^*(x)$. For example, we will say $C(x)$ is a characteristic polynomial of $S_n^*(x)$ for short. We will always work with $S_n^*(x)$ rather than $S_n(x)$ because of

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

$$C(x)S_n^*(x)+V(x)x^n=R(x)$$

for some $V(x)$, $R(x)$ with deg $R(x)<$ deg $C(x)$. *If deg $C(x)\le n$, then $V(x)$ and $R(x)$ are respectively the quotient and the remainder of $C(x)\,S_n^*(x)$. divided by x^n, and in particular uniquely determined by $C(x)$ and $S_n^*(x)$.*

Proof. Let $C(x)=x^v+c_1x^{v-1}+\cdots+c_{v-1}x+c_v$. It suffices to note that for $1\le i\le n-v$, the vanishing of the coecient of x^{n-i} in the expansion of $C(x)\,S_n^*(x)$ 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 $\le n$ is a characteristic polynomial of $S_n^*(x)$ so that $V(x)$ and $R(x)$ are as in the previous lemma. Then $C(x)$ is not a characteristic polynomial of $S_{n+1}^*(x)$ if and only if deg$(xR(x)+s_{n+1}C(x))=$* deg $C(x)$.

Proof. Note that

$$\begin{aligned}C(x)\,S_{n+1}^*(x)&=C(x)(\,xS_{n+1}^*(x)+s_{n+1})\\&=xC(x)\,S_n^*(x)+s_{n+1}C(x)\\&=x(R(x)-V(x)x^n)+s_{n+1}C(x)\\&=-V(x)x^{n+1}+xR(x)+s_{n+1}C(x).\end{aligned}$$

Therefore $C(x)\,S_{n+1}^*(x)+V(x)x^{n+1}=xR(x)+s_{n+1}C(x)$. Here deg$(xR(x)+s_{n+1}C(x))\le$deg $C(x)<n+1$ because deg $R(x)<$deg $C(x)\le n$. Therefore $xR(x)+s_{n+1}C(x)$ is the remainder of $C(x)\,S_{n+1}^*(x)$ divided by x^{n+1}. Now the conclusion follows by Lemma 1.

Let $C(x)=x^v+c_1x^{v-1}+\cdots c_{v-1}x+c_v$ with $v\le n$ so that $C(x)\,S_n^*(x)+V(x)x^n=R(x)$

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

$$r_0=c_1s_n+c_2s_{n-1}+\cdots c_vs_{n-v+1}$$

So $a=s_{n+1}c_1s_n+c_2s_{n-1}+\cdots c_vs_{n-v+1}$. Hence Lemma 2 means trivially that $C(x)$ is a characteristic polynomial of $S_{n+1}^*(x)$ if and only if

$$a=s_{n+1}+c_1s_n+c_2s_{n-1}+\cdots+c_vs_{n-v+1}=0$$

Lemma 3. Suppose *$C(x)$ is a characteristic polynomial of $S_n^*(x)$ but not of $S_{n+1}^*(x)$. If $D(x)$ is a characteristic polynomial of $S_{n+1}^*(x)$, then* deg $C(x)$+deg $D(x)\ge n+1$.

Proof. By Lemmas 1 and 2, $C(x)\,S_n^*(x)+V(x)x^n=R(x)$ with deg $R(x)<$deg $C(x)$, but

$$C(x)\,S_{n+1}^*(x)+V(x)x^{n+1}=xR(x)+s_{n+1}C(x) \tag{2}$$

with deg$(xR(x)+s_{n+1}C(x))=$ deg $C(x)$. Suppose $D(x)$ is a characteristic polynomial of $S_{n+1}^*(x)$ so that

$$D(x)\,S_{n+1}^*(x)+W(x)x^{n+1}=P(x) \tag{3}$$

with deg $P(x)<$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

$$\begin{aligned}&(D(x)V(x)-C(x)W(x))x^{n+1}\\&=D(x)(xR(x)+s_{n+1}C(x))-C(x)P(x)\end{aligned} \tag{4}$$

Since

$$\begin{aligned}&\deg C(x)P(x)<\deg C(x)D(x)\\&=\deg D(x)(xR(x)+\ s^{n+1}C(x)),\end{aligned}$$

(4) is possible only when

$$n+1\le \deg D(x)(xR(x)+s_{n+1}C(x))=\deg D(x)C(x)$$

Lemma 4. *Suppose $C(x)$ is a characteristic polynomial of $S_n^*(x)$* so that

$$C(x)\,S_n^*(x)+V(x)x^n=R(x)$$

with deg $R(x)<$ deg $C(x)$. If 2 deg $C(x)\le n$, and $C(x)$

and $V(x)$ *are relatively prime, then* $C(x)$ *is the unique minimal polynomial of* $S_n^*(x)$.

Proof. Suppose $D(x)$ is a minimal polynomial of $S_n^*(x)$ so that

$$D(x)S_n^*(x) + W(x)x^n = P(x)$$

with deg $P(x)$<deg $D(x)$. Then

$$(D(x)V(x) - C(x)W(x))x^n = D(x)R(x) - C(x)P(x)$$

Note that $\deg(D(x)R(x)-C(x)P(x)) < n$ because

$$\deg D(x)R(x) \le \deg C(x) + \deg R(x) < 2 \deg C(x) \le n,$$
$$\deg C(x)P(x) < \deg C(x) + \deg D(x) \le 2 \deg C(x) \le n$$

Therefore $D(x)V(x)-C(x)W(x)=0$. Since $C(x)$ and $V(x)$ are relatively prime, $C(x)$ divides $D(x)$. This implies that $C(x)$ is the unique minimal polynomial of $S_n^*(x)$.

3. Minimal Polynomial Synthesis Using Euclid's Algorithm

Suppose $S_N^*(x)\neq 0$. Lemma 1 immediately prompts us to consider Euclid's GCD algorithm partially applied for x^N and $S_N^*(x)$, which proceeds as follows:

$$\begin{aligned} x^N &= Q_0(x)S_N^*(x)+R_0(x)\\ S_N^*(x) &= Q_1(x)R_0(x)+R_1(x)\\ R_0(x) &= Q_2(x)R_1(x)+R_2(x) \\ &\cdots \\ R_{k\text{-}2}(x) &= Qk(x)R_{k\text{-}1}(x)+Rk(x), \end{aligned} \tag{5}$$

where deg $R_i(x)$<deg $Q_i(x)$ for $1 \le i \le k$. It is convenient to let $R_{-2}(x)=x^N$, $R_{-1}(x)=S_N^*(x)$ so that (5) begins with

$$S_N^*(x)=0 \cdot R_{-2}(x)+R_{-1}(x)$$
$$R_{-2}(x)=Q_0(x)R_{-1}(x)+R_0(x)$$
$$\cdots$$

Using matrices, we write

$$\begin{bmatrix} x^N \\ S_N^*(x) \end{bmatrix} = \begin{bmatrix} Q_0(x) & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} Q_1(x) & 1 \\ 1 & 0 \end{bmatrix}\cdots\begin{bmatrix} Q_k & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} R_{k-1}(x) \\ R_k(x) \end{bmatrix}$$

Let

$$\begin{bmatrix} G_k(x) & G_{k-1}(x) \\ H_k(x) & H_{k-1}(x) \end{bmatrix} = \begin{bmatrix} Q_0(x) & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} Q_1(x) & 1 \\ 1 & 0 \end{bmatrix}\cdots\begin{bmatrix} Q_k(x) & 1 \\ 1 & 0 \end{bmatrix}$$

with $G_{-1}(x)=1$, $H_{-1}(x)=0$. It is useful to note

$$\deg \mathrm{Gk}(x) = \sum_{i=0}^{k} \deg Q_i(x)$$

$$\deg G_k(x) + \deg R_k(x) < \deg G_k(x) + \deg R_{k\text{-}1}(x) = N.$$

Now

$$\begin{bmatrix} R_{k-1}(x) \\ R_k(x) \end{bmatrix} = \begin{bmatrix} G_k(x) & G_{k-1}(x) \\ H_k(x) & H_{k-1}(x) \end{bmatrix}\begin{bmatrix} x^N \\ S_N^*(x) \end{bmatrix}$$

Therefore

$$Gk(x)S_N^*(x) - H_k(x)x^N = (-1)^{k+1}R_k(x) \tag{6}$$

Hence $G_k(x)$ is a characteristic polynomial of $S_N^*(x)$ if the condition

$$\deg G_k(x)=\sum_{i=0}^{k} \deg Q_i(x)>\deg R_k(x) \tag{7}$$

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

Proposition 5. *Let k be the smallest such that* (7) *is satised. Then* $G_k(x)$, *made monic, is a minimal polynomial of* $S_N^*(x)$.

Proof. We claim that if deg $G_i(x)$≤deg $C(x)$<deg $G_{i+1}(x)$ with $-1 \le i \le k-1$ and

$$C(x)S_N^*(x) + V(x)x^N = P(x), \tag{8}$$

then deg $P(x) \ge$ deg $C(x)$−deg $G_i(x)$+deg $R_i(x)$ if $G_i(x)$ divides $C(x)$ or deg $P(x)$≥deg $G_{i+1}(x)$ deg $G_i(x)$+deg $R_i(x)$ otherwise. Then in either case,

$$\deg P(x) \ge \deg C(x) - \deg G_i(x) + \deg R_i(x) \ge \deg C(x)$$

because deg $R_i(x)$≥deg $G_i(x)$. Hence $C(x)$ is not a characteristic polynomial of $S_N^*(x)$. by Lemma 1. This proves that $G_k(x)$ is a minimal polynomial of $S_N^*(x)$.

We prove the claim by induction. Let $i=-1$. Since $G_{-1}(x)=1$ divides $C(x)$, we need to show deg $P(x)$≥deg $C(x)$ +deg $R_{-1}(x)$. Note that (8) $C(x)\cdot$(6) with $k=-1$ gives

$$V(x)x_N = P(x)-C(x)R_{-1}(x)$$

Since deg $C(x)R_{-1}(x)$<deg $G_0(x)R_{-1}(x)=N$, this equation is possible only if deg $P(x)$≥deg $C(x)R_{-1}(x)$=deg $C(x)$+deg $R_{-1}(x)$.

Assume the claim is true for all j with $-1 \le j <$ i. To prove the claim for i, let $C(x) = Q(x)G_i(x)+D(x)$ with deg $D(x)$<deg $G_i(x)$. If $D(x)=0$, then (8) $-Q(x)\cdot$(6) with $k=i$ gives

$$(V(x)+Q(x)H_i(x))x^N = P(x)+(-1)^iQ(x)R_i(x)$$

Since deg $Q(x)R_i(x) <$ deg $G_{i+1}(x)R_i(x) = N$, this equation implies that deg $P(x) \geq$ deg $Q(x)R_i(x)$ = deg $C(x)-$ deg $G_i(x)$ + deg $R_i(x)$. If $D(x) \neq 0$, then (8) – $Q(x)$ · (6) with $k = i$ gives

$$D(x)\,S_N^*(x)+(V(x)+Q(x)H_i(x))x^N=P(x)+(-1)^iQ(x)R_i(x)$$

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

$$\deg(P(x) + (-1)^iQ(x)R_i(x)) \geq \deg R_j(x) \geq \deg R_{i\text{-}1}(x), \quad (9)$$

where j is chosen so that deg $G_j(x)$≤deg $D(x)$<deg $G_{j+1}(x)$. Here note that

$$\begin{aligned}\deg Q(x)R_i(x) &= \deg C(x) - \deg G_i(x) + \deg R_i(x)\\ &<\deg G_{i+1}(x) - \deg G_i(x) + \deg R_i(x)\\ &= \deg Q_{i+1}(x) + \deg Ri(x) = \deg R_{i\text{-}1}(x)\end{aligned}$$

Therefore (9) is possible only if deg $P(x)$≥deg $R_{i\text{-}1}(x)$. This proves the claim.

Thus the algorithm given below is valid.

Algorithm E. Let s be a sequence $s_1,s_2,...,s_N$ 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 $D(x)\leftarrow x^N ,D(x)\,S_N^*(x)$, and set $G(x)\leftarrow 1,G(x)\leftarrow 0$,

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

$$G(x)\leftarrow G(x)Q(x)+\tilde{G}(x), \tilde{G}(x)\leftarrow \tilde{G}'(x)\,,$$
and set $\tilde{D}(x)\leftarrow D(x), Dx\leftarrow R(x)$

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

Proposition 6. *For any nite sequence* $S_N^*(x)$, there is *a unique minimal polynomial* $C(x)$ *satisfying*

$$C(x)\,S_N^*(x) + V(x)x^N = R(x) \qquad (10)$$

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 $S_N^*(x)$ such that

$$D(x)\,S_N^*(x) + W(x)x^N = P(x) \qquad (11)$$

with deg $P(x)$<deg $D(x)$ and deg $P(x)$+deg $D(x)$<N. Then $D(x)$ · (10)–$C(x)$ · (11) gives

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

Note that deg$(D(x)R(x) - C(x)P(x)) < N$ because

$$\begin{aligned}&\deg D(x)R(x) = \deg C(x) + \deg R(x) < N,\\ &\deg C(x)P(x) = \deg D(x) + \deg P(x) < N\end{aligned}$$

Therefore $D(x)V(x) - C(x)W(x)$=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 $S_N^*(x)$. Then it follows that $C(X)$=$D(x)$.

4. Iterative Minimal Polynomial Synthesis

Let 1≤n<N. Suppose $C(x)$ of degree v is a characteristic polynomial of $S_N^*(x)$ so that

$$C(x)\,S_N^*(x) + V(x)x^n = R(x)$$

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

$$C(x)\,S_{n+1}^*(x) + V(x)x^{n+1} = xR(x)+s_{n+1}C(x) \qquad (12)$$

Let a be the coecient of x^v in $xR(x)$+$s_{n+1}C(x)$. If a=0, then $C(x)$ is also a characteristic polynomial of $S_{n+1}^*(x)$ by Lemma 2. Assume a≠0. Then $C(x)$ is not a characteristic polynomial of $S_{n+1}^*(x)$. An important idea is that we can overcome this diculty by exploiting previous $B(x)$ that is a characteristic polynomial of $S_m^*(x)$ but not of $S_{m+1}^*(x)$ for some m<n. So we suppose

$$B(x)\,S_m^*(x) + U(x)x^{\text{m}} = Q(x)$$

with deg $Q(x)$<deg $B(x)$ but deg$(xQ(x)$+$s_{m+1}B(x))$=deg $B(x)$. Now with $S(x)$=$s_{m+1}x^{nm}$+$s_{m+2}x^{n\text{-}m\text{-}1}$+ ··· s_{n+1}, we have

$$\begin{aligned}B(x)\,S_{n+1}^*(x) &= B(x)(x^{n+1\text{-}m}\,S_m^*(x) + S(x))\\ &= x^{n+1\text{-}m}B(x) + S_m^*(x) + S(x)B(x)\\ &= x^{n+1\text{-}m}(Q(x) - U(x)x^m) + S(x)B(x)\\ &= -U(x)x^{n+1}x^{n+1\text{-}m}Q(x) + S(x)B(x)\end{aligned}$$

Therefore

$$B(x)\,S_{n+1}^*(x) + U(x)x^{n+1} = x^{n+1\text{-}m}Q(x) + S(x)B(x) \qquad (13)$$

Observe that if b is the leading coeffcient of $xQ(x)$ +$s^{m+1}B(x)$, then b is still the leading coecient of $x^{n+1\text{-}m}Q(x)$+$S(x)B(x)$, and deg$(x^{n+1\text{-}m}Q(x)$+$S(x)B(x))$= deg$(B(x))$ +n–m.

To summarize, let v=deg $C(x)$ and u=deg $B(x)$ assuming $u \le v$. Let $K(x)=xR(x)+s_{n+1}C(s)$ and $J(x)= x^{n+1-m}Q(x)+S(x)B(x)$. Then (12) and (13) are again

$$C(x)\,S^*_{n+1}(x) + V(x)x^{n+1} = K(x) \quad (14)$$
$$B(x)\,S^*_{n+1}(x) + U(x)x^{n+1} = J(x) \quad (15)$$

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

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

$$(N(x)C(x) - cB(x))\,S^*_{n+1}(x) + (N(x)V(x) - cU(x))x^{n+1}$$

As $\deg(N(x)C(x)-cB(x))=u+n-m$, we find $N(x)C(x)-cB(x)$ is a characteristic polynomial of $S^*_{n+1}(x)$.

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

$$(C(x)-cN(x)B(x))\,S^*_{n+1}(x)+ (V(x)-cN(x)U(x))x^{n+1}= K(x)-cN(x)J(x).$$

As $\deg(C(x)-cN(x)B(x))=v$, we find $C(x)-cN(x)B(x)$ is a characteristic polynomial of $S^*_{n+1}(x)$.

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

If s is a zero sequence, then clearly 1 is the unique mminimal polynomial of s. Otherwise let $1 \le r \le N$ be the smallest such that s_r is not zero. Then $B_0(x)=1$ is the unique minimal polynomial of the zero sequence $s_1,s_2,\ldots,s_{r-1}$ so that

$$B_0(x)S_{r-1}(x)+U_0(x) \cdot x^{r-1}=Q_0(x),$$

where $S^*_{r-1}(x)=0$, $U_0(x)=0$, and $Q_0(x)=0$. As required, $B_0(x)$ is not a characteristic polynomial of $S^*_r(x)$ because $\deg(xQ_0(x)+s_rB_0(x))=\deg(s_r)=0=\deg B_0(x)$.

On the other hand, any monic polynomial of degree r is a minimal polynomial of the sequence $s_1,s_2,\ldots,s_{r-1},s_r$. Indeed for any monic polynomial $C_0(x)$ of degree r,

$$C_0(x)S^*_r(x)+V_0(x)x^r=R_0(x),$$

where $S^*_r(x) = s_r$, $V_0(x) = -s_r$, and $R_0(x) = s_rC_0(x)-s_rx^r$

Now we are ready to introduce an iterative minimal polynomial synthesis algorithm.

Algorithm I. Let s be a sequence $s_1,s_2,\ldots,s_N$ of length N over $\mathbb{F}$. This algorithm outputs a minimal polynomial of s.

I1 (Initialization). If s is a zero sequence, then output 1, and the algorithm terminates. Otherwise set $m \leftarrow r-1$, $n \leftarrow r$, $b \leftarrow s_r$, $B(x) \leftarrow 1$,$Q(x) \leftarrow 0$,$C(x) \leftarrow C_0(x)$,$R(x) \leftarrow R_0(x)$. If $n=N$, then output $C(x)$, and the algorithm terminates.

I2 (Beginning of iteration). Set $K(x) \leftarrow xR(x)+s_{n+1}C(x)$. Set $a \leftarrow$the coecient of x^v in $K(x)$ where v=deg $C(x)$.

I3 (No adjustment). If a=0, then set $R(x) \leftarrow K(x)$, and go to **I6**. Otherwise set $J(x) \leftarrow x^{n+1-m}Q(x)+S(x)B(x)$ where $S(x)=s_{m+1}x^{n-m}+s_{m+2}x^{n-m-1}+s_{n+1}$.

I4 (Jump). Let u=deg $B(x)$, v=deg $C(x)$. Let c=a/b. If $v \le u+n-m$, then choose a monic polynomial $N(x)$ of degree $u+n-m-v$. Set $C'(x) \leftarrow C(x)$, and $R'(x) \leftarrow R(x)$, and set

$$C(x) \leftarrow N(x)C(x) - cB(x),$$
$$R(x) \leftarrow N(x)K(x) - cJ(x),$$

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

I5 (Adjustment). If $v>u+n-m$, then choose a monic polynomial $N(x)$ of degree $v-(u+n-m)$. Set

$$C(x) \leftarrow C(x)-cN(x)B(x),$$
$$R(x) \leftarrow K(x)-cN(x)J(x)$$

I6 (End of iteration). Set $n \leftarrow n+1$. If $n<N$, then return to **I2**. Otherwise output $C(x)$, and the algorithm terminates. Notice that $B(x)$ is updated only in **I4**. Note also that we did not specify the initial $C_0(x)$ in **I1** and how we choose $N(x)$ in **I4** and **I5**. The algorithm behaves dierently depending on how we specify these. However, regardless of these specications, Algorithm I works correctly. We rst make an important observation.

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

Proof. At the beginning of the rst iteration, that is, just after the initialization, we have $m=r-1$, $n=r$, u=0, $v=r$. So $u+v=r=m+1$. 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 **I3** or **I5**, then there occurs no change to u, v, and m, and hence the equality holds. If the algorithm gets into **I4**, then variables change as $u=u'$, $v=u'+n'+-m'$, $m=n'$, where prime indicates the old value of each variable. So $u+v=v'+u'+n'-m'=m'+1+n'-m'=n'+1=m+1$. 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 $s_1, s_2,..., s_n$ of s.*

Proof. First note that the linear complexity of $S_n^*(x)$ is monotone increasing function of n.

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

If the algorithm follows **I3**, then $C(x)$ is still a characteristic polynomial of $S_{n+1}^*(x)$. Since $C(x)$ is a minimal polynomial of $S_n^*(x)$, it is clear that $C(x)$ is also a minimal polynomial of $S_{n+1}^*(x)$

If the algorithm follows **I4**, then $N(x)C(x)-cB(x)$ is a characteristic polynomial of $S_{n+1}^*(x)$ .Since $\deg(N(x)C(x) -cB(x))=u+n-m$, we see the linear complexity l of $S_{n+1}^*(x)$ satifises $l\le u+n-m$. By Lemma 3, it also holds that $l+v\ge n+1$. Now $u+n-m\ge l\ge n+1-v=n+1-(m+1-u)=u+n-m$. Therefore $l=u+n-m$. Hence $N(x)C(x) -cB(x)$ is a minimal polynomial of $S_{n+1}^*(x)$

If the algorithm follows **I5**, then $C(x)-cN(x)B(x)$ is a characteristic polynomial of $S_{n+1}^*(x)$. Since $\deg(C(x)-cN(x)B(x))= v =\deg C(x)$ and $C(x)$ is a minimal polynomial of $S_n^*(x)$, it follows that $C(x)-cN(x)B(x)$ is a minimal polynomial of $S_{n+1}^*(x)$.

As we said earlier, the initial $C_0(x)$ in *I1* can be arbitrary monic polynomial of degree r. But the simplest choice may be $C_0(x)=x^r$. In **I4**, the simplest choice of a monic polynomial $N(x)$ may be $x^{u+n-m-v}$. Similarly in **I5**, the simplest choice of $N(x)$ may be $x^{v-(u+n-m)}$. Moreover with these choices, we do not need to compute and store $J(x)$ and $K(x)$ because only their leading coecients are really necessary to determine the course of the algorithm. Then also $R(x)$ 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 $s_1,s_2,...,s_N$ of length N over $\mathbb{F}$. This algorithm outputs a minimal polynomial of s.

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

J2 (Beginning of iteration). Let $C(x)= x^v+c_1x^{v-1}+ \cdots + c_v$. Set $a\leftarrow s_{n+1}+c_1s_n+ \cdots +c_vs_{n+1-v}$.

J3 (No adjustment). If $a = 0$, then go to **J6**

J4 (Jump). Let $u = \deg B(x)$, $v = \deg C(x)$. Let $c = a/b$. If $v\le u+n-m$, then set $C'(x)\leftarrow C(x)$ and set

$$C(x)\leftarrow x^{u+n-m-v}\, C(x) - cB(x),$$

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

J5 (Adjustment). If $v > u+n-m$, then set

$$C(x)\leftarrow C(x) - cx^{v-(u+n-m)}B(x).$$

J6 (End of iteration). Set $n\leftarrow n+1$. If $n<N$, 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 eectively the algorithm automatically initializes itself with

$$m\leftarrow r-1,\ n\leftarrow r,\ b\leftarrow s_r,\ B(x)\leftarrow 1,\ C(x)\leftarrow x^r-s_r$$

when it reaches the rst 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.

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 rst stage, we make the algorithm **J** with **J1′** work with the pair of the connection polynomial $C^*(x)$ and the degree v instead of the characteristic polynomial $C(x)$ of degree v:

$$C(x) = x^v+c_1x^{v-1}+c \leftrightarrow v\{C^*(x)=1+c_1x+\cdots+c_vx^v,\ v\}$$

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

Algorithm BM′. Let s be a sequence $s_1,s_2,...,s_N$ of length N over $\mathbb{F}$. This algorithm outputs a minimal connection polynomial and the linear complexity of s.

BM′1 (Initialization). Set $m\leftarrow 1$, $n\leftarrow 0$, $b\leftarrow 1$, $u\leftarrow 0$, $v\leftarrow 0$, $B^*(x)\leftarrow 1$, $C(x)\leftarrow 1$.

BM′2 (Beginning of iteration). Let $C^*(x)=1+c_1x+\cdots+c_vx^v$. Set $a\leftarrow s_{n+1}+c_1s_n+\cdots c_vs_{n+1-v}$.

BM′3 (No adjustment). If $a = 0$, then go to **BM′6**.

BM′4 (Jump). Let $c = a/b$. If $v\leftarrow u+n-m$, then set $T(x)\leftarrow C(x)$, $w\leftarrow v$ and set

$$C^*(x)\leftarrow C(x)-cx^{n-m}B^*(x),\ v\leftarrow u+n-m$$

and set $m\leftarrow n$, $b\leftarrow a$, $u\leftarrow w$, $B^*(x)\leftarrow T(x)$.

BM′5 (Adjustment). If $v > u+n-m$, then set

$$C^*(x)\leftarrow C(x)-cx^{n-m}B^*(x).$$

BM′6 (End of iteration). Set $n\leftarrow n+1$. If $n<N$, then return to **BM′2**. Otherwise output the pair $\{C^*(x),v\}$, and the algorithm terminates.

In the second stage, we use the fact that the equality $u+v=m+1$ always holds at the beginning of each iteration, as we showed in Lemma 7. It is clear that this is still true in the algorithm **BM′**. We also make variable changes $r=n-m$ and ñ=n+1, which make variables u and m redundant.

Algorithm BM. Let s be a sequence $s_1,s_2,...,s_N$ of length N over $\mathbb{F}$. This algorithm outputs a minimal connection polynomial and the linear complexity of s.

BM1 (Initialization). Set $r\leftarrow 1$, $\tilde{n}\leftarrow 1$, $b\leftarrow 1$, $v\leftarrow 0$, $B^*(x)\leftarrow 1$, $C^*(x)_{\tilde{n}}1$.

BM2 (Beginning of iteration). Let $C^*(x)=1+c_1x+\cdots+c_vx$. Set $a\leftarrow s_{\tilde{n}}+c_1s_{\tilde{n}-1}+\cdots c_vs_{\tilde{n}-v}$.

BM3 (No adjustment). If $a=0$, then go to **BM6**.

BM4 (Jump). Let $c=a/b$. If $2v\leftarrow\tilde{n}$, then set $T(x)\leftarrow C(x)$, and set

$$C^*(x)\leftarrow C^*(x)-cx^r B^*(x),\ v\leftarrow\tilde{n}-v$$

and set $r\leftarrow 0$, $b\leftarrow a$, $B^*(x)\leftarrow T(x)$.

BM5 (Adjustment). If $2_v>\tilde{n}$, then set

$$C^*(x)\leftarrow C(x)-cx^r B^*(x).$$

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

This is exactly the venerable 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 dierent minimal polynomials of the same nite sequence. Of course, this does not prove that one of the algorithms works incorrectly because there can be many minimal polynomials of a nite sequence in general. Recall that the minimal polynomial Algorithm **E** outputs is characterized in Proposition 6.

We can relieve this discrepancy by specializing Algorithm **I** in a dierent way. We choose the same $C_0(x)=x^r$ as in Algorithm **J**. But this time, in **I4** we choose $N(x)$ to be the quotient of $cJ(x)$ divided by $K(x)$ and in **I5** we choose $N(x)$ to be the quotient of $K(x)$ divided by $cJ(x)$. Of course, this amounts to employ the expensive Euclid's algorithm in choosing $N(x)$ so that the resulting algorithm becomes much less ecient 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 $s_1,s_2,...,s_N$ 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 $s_r\neq 0$. If $r > N$ then output 1, and the algorithm terminates. Otherwise set $m\leftarrow r-1$, $n\leftarrow r$, $b\leftarrow s_r$, $B(x)\leftarrow 1$, and $C(x)\leftarrow x$. If $n=N$ then output $C(x)$, and the algorithm terminates.

K2 (Beginning of iteration). Set $K(x)\leftarrow xR(x)+s_{n+1}C(x)$.

Set $a \leftarrow$ the coecient of x^{v} in K(x) where v = deg $C(x)$.

K3 (No adjustment). If $a = 0$, then set $R(x) \leftarrow K(x)$, and go to **K6**. Otherwise set $J(x) \leftarrow x^{n+1-m}Q(x)+S(x)B(x)$ where $S(x)=s_{m+1}x^{n-m}+s_{m+2}x+s_{n+1}$.

K4 (Jump). Let u=deg $B(x)$, v=deg $C(x)$, and $c=a/b$. If $v \le u+n-m$, then compute the quotient $N(x)$ of $cJ(x)$ divided by $K(x)$. Set $C'(x) \leftarrow C(x)$, and $R'(x) \leftarrow R(x)$, and set

$$C(x) \leftarrow N(x)C(x) - cB(x),$$
$$R(x) \leftarrow N(x)K(x) - cJ(x),$$

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

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

$$C(x) \leftarrow C(x) - cN(x)B(x),$$
$$R(x) \leftarrow K(x) - cN(x)J(x).$$

K6 (End of iteration). Set $n \leftarrow n+1$. If $n<N$, 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, $C(x)=x^{r}$,$R(x)=0$, $n=r$. Hence deg $C(x)$+deg $R(x) = -\infty <$ n.

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 $n>$deg $R(x)$+deg $C(x)$=2 deg $C(x)-1$. After setting $R(x) \leftarrow K(x)$, we have deg $R(x)$+deg $C(x)<2$ deg $C(x)<n+1$. If s_{n+1} was zero, then $R(x)$ is set to be $xR'(x)$, prime indicating the old value. Hence deg $R(x)$+deg $C(x)$=1+deg $R'(x)$+deg $C(x)<1+n$. In either case, deg $C(x)$+deg $R(x)<n+1$.

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

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

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

7. Decoding Alternant Codes

Recall that a minimal polynomial $C(x)$ of a nite sequence $S_N^*(x)$ is a monic polynomial of the least possible degree such that $C(x)S_N^*(x)+V(x)=R(x)$ 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 make these algorithms 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 $s_1,s_2,...,s_N$ of length N over $\mathbb{F}$. This algorithm outputs $C(x),V(x)$ so that $\mathrm{C(x)}S_N^*(x)+V(x)x^N=R(x)$, 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

$$\tilde{D}(x) \leftarrow x^N, D(x) = S_N^*(x)\,,$$
$$G(x) \leftarrow 1, \tilde{G}(x) \leftarrow 0, H(x) \leftarrow 0, \tilde{H}(x) \leftarrow 1$$

F2 (Division). Compute the quotient $Q(x)$ and the remainder $R(x)$ of $\tilde{D}(x)$ divided by $D(x)$. Set $G'(x) \leftarrow G(x)$ and $H'(x) \leftarrow H(x)$ and set

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

and set $\tilde{D}(x) \leftarrow D(x), D(x) \leftarrow R(x)$

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 coecient 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 ecient as possible in implementation. So we rst modify Algorithm **I** to output $V(x)$ as well as $C(x)$, and then specialize the modied algorithm as we did to get Algorithm **J**, but now using **J1′** as the initialization step. Then we make several minor adjustments for the sake of eciency. We use the fact $u+v=m+1$ to remove variables u and m. We make a variable change $\tilde{n}=n+1$. We use variable v to track the degree of $C(x)$. We modify **L2**, **L4** and **L5** to remove expensive division operations. (This modication certainly involves non-monic characteristic and minimal polynomials. But this does not aect any of the results above.) Finally we modify **L6** to make the output $C(x)$ monic. The resulting algorithm is

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

L1 (Initialization). Set $\tilde{n}\leftarrow 1$, $b\leftarrow 1$, $v\leftarrow 0$, $B(x)\leftarrow 1$, $C(x)\leftarrow 1$, $U(x)\leftarrow 1$, $V(x)\leftarrow 0$.

L2 (Beginning of iteration). Let $C(x)=c_0x^v+c_1x^{v-1}+\cdots+c_v$. Set $a\leftarrow c_0s_{\tilde{n}}+c_1s_{\tilde{n}-1}+\cdots+c_vs_{\tilde{n}-v}$.

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

L4 (Jump). If $2v\leq\tilde{n}$, then set $C'(x)\leftarrow C(x)$ and $V(x)\leftarrow V(x)$, and set

$$C(x)\leftarrow bx^{\tilde{n}-2v}C(x)-aB(x),$$
$$V(x)\leftarrow bx^{\tilde{n}-2v}V(x)-aU(x)$$

and set $b\leftarrow a$, $v\leftarrow\tilde{n}-v$, $B(x)\leftarrow C'(x)$, $U(x)\leftarrow V'(x)$.

L5 (Adjustment). If $2v>\tilde{n}$, then set

$$C(x)\leftarrow bC(x)-ax^{2v-\tilde{n}}B(x),$$
$$V(x)\leftarrow bV(x)-ax^{2v-\tilde{n}}U(x)$$

L6 (End of iteration). Set $\tilde{n}\leftarrow\tilde{n}+1$. then return to **L2**. Otherwise set $C(x)\leftarrow cC(x)$ and $V(x)\leftarrow cV(x)$ where c is the inverse of the leading coecient of $C(x)$, and output $C(x)$ and $V(x)$, and the algorithm terminates.

Now we review the alternant codes. Let $\mathbb{F}$/K be a eld extension of degree m. Let $\alpha_1, \alpha_2,..., \alpha_n$ be distinct elements of $\mathbb{F}$. Let $y_1, y_2,..., y_n$ be nonzero elements of $\mathbb{F}$. Let r be a positive integer. The alternant code $A(\alpha1,\alpha2,...,\alpha n;\ y_1,y_2,...,y_n;\ r)$ is the linear code over K having parity check matrix

$$H=\begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ y_1\alpha_1 & y_2\alpha_2 & \cdots & y_n\alpha_n \\ \cdots & \cdots & \cdots & \cdots \\ y_1\alpha_1^{r-1} & y_2\alpha_2^{r-1} & \cdots & y_n\alpha_n^{r-1} \end{bmatrix}$$

The alternant code has parameters $[n,k,d]$ with $n-mr\leq k\leq n-r$ and d$\leq r+1$. For more details, see[4].

Suppose that a codeword $c=(c_1,c_2,...,c_n)$ was sent through the channel, and v errors occurred during transmission. The received word is thus

$$d=(d_1,d_2,...,d_n)=(c_1,c_2,...,c_n)+(e_1,e_2,...,e_n),$$

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

The syndromes $S_0, S_1,..., S_{r-1}$ are computed from the received word d by

$$[S_0\ S_1\ \ldots\ S_{r-1}]=[d_1\ d_2\ \ldots\ d_n]\ H^T$$

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

$$[S_0\ S_1\ \ldots\ S_{r-1}]=[e_1\ e_2\ \ldots\ e_n]\ H^T$$

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

$$S_i=\sum_{j=1}^{n}e_jy_j\alpha_j^i\quad(0\leq i\leq r-1)$$

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

$$X_1=\alpha j_1,\ X_2=\alpha j_2,...,\ X_v=\alpha j_v,$$
$$Y_1=e_{j1}y_{j1},Y_2=e_{j2}y_{j2},...,Y_v=e_{jv}y_{jv}$$

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

$$S_i=\sum_{k=1}^{v}Y_kX_k^i$$

Let us dene

$$L(x)=(x-X_1)(x-X_2)\ \ldots\ (x-X_v), \tag{16}$$

and

$$S^*(x)=S_0x^{r-1}+S_1x^{r-2}+\cdots S_{r-1}$$

Now observe that

$$\begin{aligned} L(x)S^*(x) &= \prod_{l=1}^{v}(x-X_l)\sum_{i=0}^{r-1}\Big(\sum_{k=1}^{v}Y_kX_k^i\Big)x^{r-1-i} \\ &= \sum_{k=1}^{v}Y_k\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l)(x-X_k)\sum_{i=0}^{r-1}X_k^ix^{r-1-i} \\ &= \sum_{k=1}^{v}Y_k\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l)(x^r-X_k^r) \\ &= x^r\sum_{k=1}^{v}Y_k\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l)-\sum_{k=1}^{v}Y_kX_k^r\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l) \end{aligned}$$

Let

$$E(x)=-\sum_{k=1}^{v}Y_k\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l),$$

$$R(x)=-\sum_{k=1}^{v}Y_kX_k^r\prod_{\substack{l=1\\l\neq k}}^{v}(x-X_l)$$

Then we get the key equation

$$L(x)S^*(x)+E(x)x^r=R(x)$$

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

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

Any of algorithms **F** or **L** applied to the nite sequence $S_0, S_1,\ldots, S_{r-1}$ will find $L(x)$ and $E(x)$. Then the number v of errors can be computed by the degree of $L(x)$. The locations X_k of errors can be found by finding roots of $L(x)$. The error values e_{jk} are then computed by Forney's formula

$$e_{jk}=-\frac{E(X_k)}{y_{jk}L'(X_k)},$$

which comes from

$$E(X_k)=-Y_k\prod_{\substack{l=1\\l\neq k}}^{v}(X_k-X_l)=-Y_kL'(X_k)$$
$$=-e_{jk}\,y_{jk}\,L'(X_k),$$

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

8. Discussion

The Berlekamp-Massey algorithm was invented almost thirty years ago[1, 2]. And there have been arduous eort to better understand this important algorithm and improve its performance.

After Sugiyama et al.[3] discovered that Euclid's algorithm can be used to decode Goppa codes, replacing the Berlekamp-Massey algorithm, much effort were put to prove that both algorithms are internally equivalent as well as externally solving the same problem. See[5-8]. Clearly Algorithm **E** corresponds to Sugiyama 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 dierent even externally, my opinion is that the two algorithms is hardly equivalent.

There are numerous algorithms based on Euclid's algorithm. One of them that comes 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 Sugiyam 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 [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 nite 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 dened 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[12, 14] for example.

References

[1] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122-127, 1969.

[2] E. R. Berlekamp, *Algebraic Coding Theory*. McGraw-Hill, 1968.

[3] Y. Sugiyama, M. Kasahara, H. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inform. Contr.*, vol. 27, pp. 87-99, 1975.

[4] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[5] A. E. Heydtmann and J. M. Jensen, "On the equivalence of the Berlekamp-Massey and the Euclidean algorithms for decoding," *IEEE Trans. Inform. Theory*, vol. 46, no. 7, pp. 2614-2624, 2000.

[6] J. L. Dornstetter, "On the equivalence between Berlekamp's and Euclid's algorithms," *IEEE Trans. Inform. Theory*, vol. 33, no. 3, pp. 428-431, 1987.

[7] U. Cheng, "On the continued fraction and Berlekamp's algorithm," *IEEE Trans. Inform. Theory*, vol. 30, no. 3, pp. 541-544, 1984.

[8] L. R. Welch and R. A. Scholtz, "Continued fractions and Berlekamp's algorithm," *IEEE Trans. Inform. Theory*, vol. 25, no. 1, pp. 19-27, 1979.

[9] R. J. McEliece and J. B. Shearer, "A property of Euclid's algorithm and an application to Padé approximation," *Siam J. Appl. Math.*, vol. 34, no. 4, pp. 611-615, 1978.

[10] Y. Sugiyama, "An algorithm for solving discrete-time Wiener-Hopf equations based upon Euclid's algorithm," *IEEE Trans. Inform. Theory*, vol. 32, no. 3, pp. 394-409, 1986.
[11] P.Fitzpatrick, "On the key equation," *IEEE Trans. Inform. Theory*, vol. 41, no. 5, pp. 1290-1302, 1995.
[12] G. Norton, "On the minimal realizations of a nite sequence," *J. Symbolic Computation*, vol. 20, pp. 93-115, 1995.
[13] G.-L. Feng and K. K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Trans. Inform. Theory*, vol. 37, no. 5, pp. 1274-1287, 1991.
[14] D. R. Hankerson *et al.*, *Coding Theory and Cryptography*, 2nd ed. Marcel Dekker, 2000.