Shift-Register Synthesis (Modulo m) *

J. A. Reeds and N. J. A. Sloane
Mathematics and Statistics Research Center
Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

The Berlekamp-Massey algorithm takes a sequence of elements from a field and finds the shortest linear recurrence (or linear feedback shift register) that can generate the sequence. In this paper we extend the algorithm to the case when the elements of the sequence are integers modulo *m*, where *m* is an arbitrary integer with known prime decomposition.

Key words. Berlekamp-Massey algorithm, shift-register synthesis, linear recurrences.

---

* This appeared in ‘‘SIAM J. Comput.’’, vol. 14 (1985), pp. 505–513.

I. Introduction

The Berlekamp-Massey algorithm used in decoding BCH codes also solves the following problem: given a sequence $S_0, S_1, \ldots, S_{n-1}$ of elements from a field, find the shortest linear recurrence (or linear feedback shift register) that will generate the sequence ([1, Chapter 7], [21]). The algorithm can be used to decode other codes ([18]-[20],[23],[27],[32]), is related to the Euclidean algorithm and the computation of Padé approximations ([6],[7],[22],[34]), and has been extensively studied ([2],[8]-[10],[13],[26],[33],[35]). W. F. Lunnon, in an unpublished manuscript [17], has pointed out that a version of the quotient-difference algorithm ([12],[14],[15]) can be used to find the shortest linear recurrence which generates a given sequence of complex numbers. This is not as efficient as Berlekamp-Massey algorithm, although it has the advantage of being easier to remember, at least in its simplest form. Games and Chan [11] give a fast algorithm for finding the shortest linear recurrence in the special case of a binary sequence of period 2^k. Nevertheless, in spite of this extensive literature, it appears that until now no-one has extended the Berlekamp-Massey algorithm so as to find the shortest linear recurrence that will generate a given sequence of numbers modulo *m*, where *m* is an arbitrary (but known) integer. (For the case when *m* is unknown, see [24],[25].) In this note we describe such an algorithm. The original algorithm ([1],[21]) fails in this case because not all numbers have inverses modulo *m*, and other versions — such as those involving the Euclidean algorithm [32] — fail because certain polynomial rings are no longer principal ideal domains.

There are several obvious applications of the new algorithm, for example in certifying random number generators (see the discussion in [29]), or in decoding BCH codes defined over the integers modulo *m* (see [3],[4],[28],[30],[31]).

Section II establishes our notation for linear recurrences, and Section III uses the Chinese remainder theorem to reduce the problem to the case when *m* is a prime power. The algorithm itself is given in Section IV and its justification in Section V. The final section contains an example.

Gustavson [13] has analyzed the complexity of the Berlekamp-Massey algorithm, and the complexity of our algorithm is essentially the same. If the modulus *m* is fixed, $O(n^2)$ steps are required to synthesize a sequence of length *n*. Changing from a sequence modulo *p* to a sequence modulo p^e increases the number of steps by a factor of *e*.

II. Linear Recurrences and Linear Feedback Shift Registers

Our notation follows Massey's description [21] of the Berlekamp-Massey algorithm, and some familiarity with that paper would be helpful (although not essential) in reading this one. Let *R* be a commutative ring containing the unit element 1, and let R^* denote the set of all units (or invertible elements) of *R* ([16]). The sequence $S_0, S_1, \ldots, S_{n-1}$, where all $S_i \in R$, is said to obey a *linear recurrence of length* ℓ, or to be generated by a *linear feedback shift register of length* ℓ, if there are elements $a_0 = 1, a_1, \ldots, a_\ell \in R$ such that

$$\sum_{i=0}^{\ell} a_i S_{j-i} = 0, \quad \text{for} \quad j = \ell, \ldots, n-1 \ . \tag{1}$$

It is convenient to express equation (1) in terms of polynomials from $R[x]$. Let $a(x) = a_0 + a_1 x + \ldots + a_\ell x^\ell$, $S(x) = S_0 + S_1 x + \ldots + S_{n-1} x^{n-1}$. Then (1) is equivalent to

$$\begin{cases} S(x)\,a(x) \equiv b(x) \pmod{x^n}, \\ a(0) = 1 \,, \end{cases} \qquad (2)$$

for some polynomial $b(x) \in R[x]$ of degree $\leq \,,\, -1$. Thus the length of the recurrence or shift register is $, \geq \max\{\deg a(x),\, 1+\deg b(x)\}$, and in fact there is no loss of generality in assuming that $, = \max\{\deg a(x),\, 1+\deg b(x)\}$. We write $A = (a(x),\, b(x))$ and define $L(A) = \max\{\deg a(x),\, 1+\deg b(x)\}$. By convention $\deg\,(0) = -\infty$.

III. The Chinese Remainder Theorem

In our problem R is the ring $\mathbf{Z}_m = \mathbf{Z}/m\mathbf{Z}$ of integers modulo m, where $m \geq 2$ is a given integer whose factorization is known. We wish to find an algorithm which, when presented with a sequence $S_0,\ldots,S_{n-1}$, all $S_i \in \mathbf{Z}_m$, will find a linear recurrence $A = (a(x),\, b(x))$ that generates the sequence, i.e. satisfies (2), and has minimal length $, = L(A)$. By the Chinese remainder theorem [16] it is enough to solve the problem in the case when the modulus is a prime power. For suppose $m = \prod_i p_i^{e_i}$, $e_i \geq 1$, where the p_i are distinct primes, and assume that for each i we have found a minimal length recurrence $(a^{(i)}(x),\, b^{(i)}(x))$, of length $,_i$ say, that generates the sequence $S_0,\ldots,S_{n-1}$ modulo $p_i^{e_i}$. By the Chinese remainder theorem we can find a pair $(a(x),\, b(x))$ with

$$a(x) \equiv a^{(i)}(x),\; b(x) \equiv b^{(i)}(x) \pmod{p_i^{e_i}}$$

for all i, of length $, = \max\{,_i\}$, and it is straightforward to show that $(a(x),\, b(x))$ is a minimal length recurrence generating $S_0,\ldots,S_{n-1}$ modulo m.

IV. The Algorithm

Given $S_0, S_1, \ldots, S_{n-1} \in R$, where $R = \mathbf{Z}_{p^e}$, p = prime, $e \geq 1$, we wish to find a linear recurrence $A = (a(x),\ b(x))$ of minimal length $, = L(A)$ satisfying (2). The key idea is to consider not just (2) but the following more general problem. For all $\eta = 0, 1, \ldots, e-1$, find pairs $A_\eta = (a_\eta(x),\ b_\eta(x))$ such that

$$\begin{cases} S(x)\, a_\eta(x) \equiv b_\eta(x) \pmod{x^n}, \\ a_\eta(0) = p^\eta, \end{cases} \tag{3}$$

and $L(A_\eta) = ,_\eta$ is minimized.

Our algorithm is an iterative procedure that, for all $0 \leq k \leq n$, $0 \leq \eta < e$, calculates pairs

$$A_\eta^{(k)} = (a_\eta^{(k)}(x),\ b_\eta^{(k)}(x))$$

satisfying

$$\begin{cases} S(x)\, a_\eta^{(k)}(x) \equiv b_\eta^{(k)}(x) \qquad \pmod{x^k}\ , \\ a_\eta^{(k)}(0) = p^\eta \end{cases}$$

and minimizing $L(A_\eta^{(k)})$. Let $p^{u_{\eta k}}$ $(0 \leq u_{\eta k} \leq e)$ be the highest power of p dividing the coefficient of x^k in

$$S(x)\, a_\eta^{(k)}(x) - b_\eta^{(k)}(x)\ .$$

(If this coefficient is zero we take $u_{\eta k} = e$.) Then at the k-th step in the iteration, the following property holds for all $0 \leq r < k$:

(P_r) For all $0 \le g < e$, either

$$L(A_g^{(r+1)}) = L(A_g^{(r)}) \tag{4}$$

or else there exists an $h = f(g,r)$ with

$$g + u_{hr} < e \ , \tag{5}$$

$$L(A_g^{(r+1)}) = r+1 - L(A_h^{(r)}) \ , \tag{6}$$

$$L(A_g^{(r+1)} > L(A_g^{(r)}) \ . \tag{7}$$

(This property is the analog for our problem of the conditions that Berlekamp gives in [1, p. 183, Eq. (7.314)] and Massey gives in [21, p. 123, Eqs. (11)-(13)].) Given this data, our algorithm calculates $A_\eta^{(k+1)}$ and $f(\eta,k)$, $0 \le \eta < e$, such that P_k holds. The quantities $L(A_\eta^{(k)})$ also obey the inequalities

$$L(A_{\eta+1}^{(k)}) \le L(A_\eta^{(k)}) \le L(A_\eta^{(k+1)}) \ . \tag{8}$$

We can now state the algorithm. (A more compact version, suitable for computer implementation, is given at the end of this section.)

The algorithm (theorem-proving version)

Given $S_0, S_1, \ldots, S_{n-1}$, all $S_i \in R = \mathbf{Z}_{p^e}$, we wish to find a pair $A = (a(x),\ b(x))$ such that $S(x)\,a(x) \equiv b(x) \pmod{x^n}$, $a(0) = 1$, and the length , $= L(A) = \max\{\deg\ a(x),\ 1 + \deg\ b(x)\}$ is minimized.

Step 0. We start the algorithm with $k = 0$, and for each $\eta = 0,1,\ldots,e-1$ define

$$a_\eta^{(0)}(x) = p^\eta, \; b_\eta^{(0)}(x) = 0,$$
$$a_\eta^{(1)}(x) = p^\eta, \; b_\eta^{(1)}(x) = p^\eta S_0,$$

and $A_\eta^{(i)} = (a_\eta^{(i)}(x), \; b_\eta^{(i)}(x))$, for $i = 0,1$. Let $S_0 = \delta p^\varepsilon$ for $\delta \in R^*$, $0 \le \varepsilon \le e$ (if $S_0 = 0$ set $\delta = 1$ and $\varepsilon = e$). Then $L(A_\eta^{(0)}) = 0$, and $L(A_\eta^{(1)}) = 1$ if $\eta + \varepsilon < e$ or $= 0$ if $\eta + \varepsilon \ge e$. We also define

$$\theta_{\eta 0} = \delta, \; u_{\eta 0} = \eta + \varepsilon, \text{ if } \eta + \varepsilon < e,$$
$$\theta_{\eta 0} = 1, \; u_{\eta 0} = e, \quad \text{ if } \eta + \varepsilon \ge e$$

(these values are consistent with Eq. (9) below). Finally we set $f(\eta, 0) = 0$ for all η.

The following step is carried out for each $k = 1, 2, \ldots, n-1$.

Step k. This produces $A_\eta^{(k+1)}$. For each $\eta = 0, 1, \ldots, e-1$ we perform the following calculations. Define $\theta_{\eta k} \in R^*$ and $0 \le u_{\eta k} \le e$ by

$$S(x) a_\eta^{(k)}(x) \equiv b_\eta^{(k)}(x) + \theta_{\eta k} p^{u_{\eta k}} x^k \quad (\mathrm{mod}\ x^{k+1}) \; . \tag{9}$$

($\theta_{\eta k} p^{u_{\eta k}}$ is the *current discrepancy* in the notation of [21].)

Case I. If $u_{\eta k} = e$ set $A_\eta^{(k+1)} = A_\eta^{(k)}$.

Case II. If $u_{\eta k} < e$ define

$$g = e - 1 - u_{\eta k} \; , \tag{10}$$

so that $0 \le g < e$, and put

$$f(\eta, k) = g \tag{11}$$

There are now two subcases.

Case IIa. If $L(A_g^{(k)}) = 0$ we set

$$A_\eta^{(k+1)} = A_\eta^{(k)} + (0,\ \theta_{\eta k} p^{u_{\eta k}} x^k) \tag{12}$$

Case IIb. If $L(A_g^{(k)}) > 0$ then for some $0 \le r < k$ we have

$$L(A_g^{(r)}) < L(A_g^{(r+1)}) = L(A_g^{(k)}) \ . \tag{13}$$

r is the time of the most recent length change in the sequence $L(A_g^{(0)}), L(A_g^{(1)}), \ldots$. From (5), (6) and (13) it follows that

$$L(A_g^{(k)}) = L(A_g^{(r+1)}) = r+1-L(A_h^{(r)}) \tag{14}$$

where $h = f(g,r)$ and

$$g + u_{hr} < e \ . \tag{15}$$

From (10) and (15), $u_{hr} \le u_{\eta k}$. Thus the power of p from the past can be used to annihilate the power of p in the current discrepancy, and we define

$$a_\eta^{(k+1)}(x) = a_\eta^{(k)}(x) - \theta_{\eta k}\theta_{hr}^{-1} p^{u_{\eta k}-u_{hr}} x^{k-r} a_h^{(r)}(x) \ , \tag{16}$$

$$b_\eta^{(k+1)}(x) = b_\eta^{(k)}(x) - \theta_{\eta k}\theta_{hr}^{-1} p^{u_{\eta k}-u_{hr}} x^{k-r} b_h^{(r)}(x) \tag{17}$$

and $A_\eta^{(k+1)} = (a_\eta^{(k+1)}(x),\ b_\eta^{(k+1)}(x))$. Then

$$\begin{cases} S(x)\, a_\eta^{(k+1)}(x) \equiv b_\eta^{(k+1)}(x) \qquad (\text{mod } x^{k+1}) \ , \\ a_\eta^{(k+1)}(0) = p^\eta \ . \end{cases}$$

This concludes Step k.

At the end of Step $n-1$ the algorithm terminates and the desired pair $A = (a(x),\ b(x))$ is given by $A_0^{(n)} = (a_0^{(n)}(x),\ b_0^{(n)}(x))$.

The initial values of $A_\eta^{(k)}$ are as follows. Let $S_i = S_i^* p^{\in_i}$, where $S_i^* \in R^*$,

$0 \le \in_i \le e, i = 0$ or 1. Then

$$A_\eta^{(0)} = (p^\eta , 0) , \quad L(A_\eta^{(0)}) = 0 , \tag{18}$$

$$A_\eta^{(1)} = (p^\eta , p^\eta S_0) , \tag{19}$$

$$L(A_\eta^{(1)}) = 1 \quad \text{for} \quad \eta \le e - \in_0 - 1 , = 0 \quad \text{for} \quad \eta \ge e - \in_0 ,$$

and

$$\begin{aligned} A_\eta^{(2)} &= (p^\eta , p^\eta S_0) \quad \text{for} \quad e - \in_1 \le \eta , \\ A_\eta^{(2)} &= (p^\eta , p^\eta (S_0 + S_1 x)) \quad \text{for} \quad 0 \le \eta \le \in_0 - \in_1 - 1 , \\ A_\eta^{(2)} &= (p^\eta - p^{\eta - \in_0 + \in_1} S_0^{*-1} S_1^* x , S_0) , \quad \text{for} \quad \in_0 - \in_1 \le \eta \le e - \in_1 - 1 . \end{aligned} \tag{20}$$

The algorithm as presented above calculates and saves various intermediate quantities not needed in subsequent steps. The following is a more streamlined version that needs to remember only $O(e)$ intermediate quantities, some of which are polynomials.

The algorithm (computer-implementation version).

Given $S_0, S_1, \ldots, S_{n-1}$, all $S_i \in R = \mathbf{Z}_{p^e}$, this algorithm produces a pair $A = (a(x), b(x))$ such that $S(x) a(x) \equiv b(x) \pmod{x^n}$, $a(0) = 1$, and the length , $= L(A) = \max\{\deg a(x), 1 + \deg b(x)\}$ is minimized.

Step 0. For each $\eta = 0, 1, \ldots, e-1$ set

$$a_\eta(x) = p^\eta , \quad b_\eta(x) = 0 ,$$

$$a_\eta^{\text{new}}(x) = p^\eta , \quad b_\eta^{\text{new}}(x) = p^\eta S_0 ,$$

and find θ_η and u_η, $\theta_\eta \in R^*$, $0 \le u_\eta \le e$, such that

$$S(x) a_\eta(x) - b_\eta(x) \equiv \theta_\eta p^{u_\eta} \pmod{x} .$$

The following step is carried out for each $k = 1,2,\ldots,n-1$.

Step k. There are three parts.

First, for each $g = 0,1,\ldots,e-1$, if $L(a_g^{\text{new}}(x),\ b_g^{\text{new}}(x)) > L(a_g(x),\ b_g(x))$, set

$$a_g^{\text{old}}(x) = a_h(x)\ ,$$

$$b_g^{\text{old}}(x) = b_h(x)\ ,$$

$$\theta_g^{\text{old}} = \theta_h\ ,$$

$$u_g^{\text{old}} = u_h\ ,$$

$$r_g = k-1\ ,$$

where $h = e-1-u_g$.

Second, for each $\eta = 0,1,\ldots,e-1$, set $a_\eta(x) = a_\eta^{\text{new}}(x)$ and $b_\eta(x) = b_\eta^{\text{new}}(x)$.

Third, for each $\eta = 0,1,\ldots,e-1$, find θ_η and u_η, $\theta_\eta \in R^*$, $0 \le u_\eta \le e$, such that

$$S(x)a_\eta(x) - b_\eta(x) \equiv \theta_\eta p^{u_\eta} x^k \pmod{x^{k+1}}\ .$$

Set $g = e-1-u_\eta$. Then (I) if $u_\eta = e$ set

$$a_\eta^{\text{new}}(x) = a_\eta(x), \quad b_\eta^{\text{new}}(x) = b_\eta(x)\ ;$$

or (IIa) if $u_\eta \ne e$ and $L(a_g(x),\ b_g(x)) = 0$, set

$$a_\eta^{\text{new}}(x) = a_\eta(x),\ b_\eta^{\text{new}}(x) = b_\eta(x) + \theta_\eta p^{u_\eta} x^k\ ;$$

or (IIb) if $u_\eta \ne e$ and $L(a_g(x),\ b_g(x)) \ne 0$, set

$$a_\eta^{\text{new}}(x) = a_\eta(x) - \theta_\eta(\theta_g^{\text{old}})^{-1} p^{u_\eta - u_g^{\text{old}}} x^{k-r_g} a_g^{\text{old}}(x)\ ,$$

$$b_\eta^{\text{new}}(x) = b_\eta(x) - \theta_\eta(\theta_g^{\text{old}})^{-1} p^{u_\eta - u_g^{\text{old}}} x^{k-r_g} b_g^{\text{old}}(x)\ .$$

At the end of Step $n-1$ the algorithm terminates and the desired pair $(a(x),\ b(x))$ is given by $(a_0^{\text{new}}(x),\ b_0^{\text{new}}(x))$.

The variables in this version of the algorithm are related to those in the original version as follows. At the conclusion of Step k we have, for each η,

$$(a_\eta(x),\ b_\eta(x)) = (a_\eta^{(k)}(x),\ b_\eta^{(k)}(x)) \ ,$$
$$(a_\eta^{\text{new}}(x),\ b_\eta^{\text{new}}(x)) = (a_\eta^{(k+1)}(x),\ b_\eta^{(k+1)}(x)) \ ,$$
$$\theta_\eta = \theta_{\eta k}, \ \text{ and } \ u_\eta = u_{\eta k} \ .$$

If

$$g = g_\eta = e-1-u_\eta = e-1-u_{\eta k}$$

then

$$r_\eta = \max\{r\colon\ r < k \ \text{ and } \ L(a_g^{(r)}(x),\ b_g^{(r)}(x)) < L(a_g^{(r+1)}(x),\ b_g^{(r+1)}(x))\} \ ,$$
$$\theta_g^{\text{old}} = \theta_{hr_\eta} \ ,$$
$$u_g^{\text{old}} = u_{hr_\eta} \ ,$$

where $h = f(g, r_\eta)$.

V. Proof of Correctness

It is convenient to denote the set of all pairs $(a(x),\ b(x))$ satisfying

$$\begin{cases} S(x)\,a(x) \equiv b(x) \qquad (\bmod\ x^k) \\ a(0) = p^\eta \end{cases} \tag{21}$$

by $\%_\eta^{(k)}$, and to let

$$@_\eta^{(k)} = \{ \ (a(x),\ b(x)):\ \ S(x)\,a(x) \equiv b(x) + \theta p^\eta x^k \pmod{x^{k+1}} \quad \text{for some} \quad \theta \in R^* \}$$

for $0 \le \eta \le e$. Note that if $(a(x),\ b(x)) \in \%_\eta^{(k)}$ then $(pa(x),\ pb(x)) \in \%_{\eta+1}^{(k)}$, and $(a(x),\ b(x))$ is in $@_u^{(k)}$ for some u, $0 \le u \le e$; while if $u = e$ then $(a(x),\ b(x)) \in \%_\eta^{(k+1)}$. Furthermore, from (9),

$$A_\eta^{(k)} \in \%_\eta^{(k)} \cap @_{u_{\eta k}}^{(k)} \ . \tag{22}$$

The proof that the algorithm works is based on two lemmas. The first is a generalization of [21, Lemma 1] and part of [1, Theorem 7.42].

Lemma 1

If $(a(x),\ b(x)) \in \%_\eta^{(k)}$ *and* $(c(x),\ d(x)) \in @_u^{(k-1)}$, *where* $\eta + u < e$, *then*

$$L(a(x),\ b(x)) \ + \ L(c(x),\ d(x)) \ge k \ . \tag{23}$$

Proof. Working modulo x^k we have

$$S(x)\,a(x) \equiv b(x) \ , \qquad S(x)\,c(x) \equiv d(x) \ + \ \theta p^u x^{k-1} \ ,$$

for $\theta \in R^*$, so

$$\begin{aligned} b(x)\,c(x) - a(x)\,d(x) &\equiv \theta p^u x^{k-1} a(x) \\ &\equiv \theta p^u x^{k-1} a(0) \ = \ \theta p^{\eta+u} x^{k-1} \ , \end{aligned} \tag{24}$$

which does not vanish. Therefore the degree of the left-hand side of (24) is at least $k-1$. But

$$\deg(b(x)c(x) - a(x)d(x)) \le \max\{\deg(b(x)c(x)\ ,\ \deg\ a(x)d(x)\}$$

$$\le L(a(x),b(x)) + L(c(x),d(x)) - 1\ , \qquad (25)$$

as required.

A pair $(a(x),\ b(x))$ is said to have *minimal length* in $\%_\eta^{(k)}$ if $(a(x),\ b(x)) \in \%_\eta^{(k)}$ and if $L(a(x),\ b(x)) \le L(a'(x),\ b'(x))$ holds for all $(a'(x),\ b'(x)) \in \%_\eta^{(k)}$. The second lemma shows how Lemma 1 can be used to verify that a particular pair has minimal length.

Lemma 2

Suppose in addition to the hypotheses of Lemma 1 that equality holds in (23). Then $(a(x),\ b(x))$ *has minimal length in* $\%_\eta^{(k)}$.

This is an immediate consequence of Lemma 1. We can now justify the correctness of the algorithm.

Theorem

For all $k = 0,1,\ldots,n$ *and* $\eta = 0,1,\ldots,e-1$, $A_\eta^{(k)}$ *has minimal length in* $\%_\eta^{(k)}$.

Proof. The proof is by inductin on *k*. The induction hypothesis is that, when beginning Step *k*,

properties $P_0, P_1, \ldots, P_{k-1}$ hold (see (4)-(7)), and

$A_g^{(r)}$ has minimal length in $\%_g^{(r)}$ for $0 \le r \le k$, $0 \le g < e$.

In Step *k* we compute $u_{\eta k}$ etc. from (9) and form $A_\eta^{(k+1)}$. To establish the induction we must show that, at the end of Step *k*, property P_k holds, i.e.

(P_k) For all $0 \le \eta < e$, either

$$L(A_\eta^{(k+1)} = L(A_\eta^{(k)}) \tag{26}$$

or else

$$\eta + u_{gk} < e \ , \tag{27}$$

$$L(A_\eta^{(k+1)}) = k+1-L(A_g^{(k)}) \ , \tag{28}$$

$$L(A_\eta^{(k+1)}) > L(A_\eta^{(k)}) \ ; \tag{29}$$

and that

$$A_\eta^{(k+1)} \text{ has minimal length in } \mathcal{A}_\eta^{(k+1)} \text{ for } 0 \le \eta < e \ .$$

The initialization, proving P_0 and the minimality of $A_\eta^{(0)}$ and $A_\eta^{(1)}$, is straightforward and we omit the details.

Suppose we are in Step k, and Case I obtains. Then (26) holds, and $A_\eta^{(k+1)}$ has minimal length by induction.

Suppose Case IIa obtains. We first establish P_k. We may assume (26) does not hold. Then

$$L(A_g^{(k)}) = 0 \ , \qquad A_g^{(k)} = (p^g \ , 0) \ ,$$

and, from (9),

$$S(x) p^g \equiv \theta_{gk} p^{u_{gk}} x^k \pmod{x^{k+1}} \ .$$

This implies that $p^{e-g} = p^{1+u_{\eta k}}$ divides each of $S_0, \ldots, S_{k-1}$ and $S_k = \theta p^{u_{gk}-g}$ for some $\theta \in R^*$. Let $S_i = p^{1+u_{\eta k}} S_i^*$ for $i < k$. Using (9) again, and remembering that $a_\eta^{(k)}(0) = p^\eta$, we have

$$\{p^{1+u_{\eta k}}(S_0^* + \ldots + S_{k-1}^* x^{k-1}) + \theta p^{u_{gk}-g} x^k\} \cdot \{p^\eta + \ldots\}$$
$$\equiv b_\eta^{(k)}(x) + \theta_{\eta k} p^{u_{\eta k}} x^k \pmod{x^{k+1}} . \tag{30}$$

Since $L(A_\eta^{(k+1)}) = k+1 > L(A_\eta^{(k)})$,

$$\deg\ b_\eta^{(k)}(x) \leq k-1 .$$

Equating coefficients of x^k in (30), and using (10), we obtain

$$\alpha p^{1+u_{\eta k}} + \theta p^{u_{gk}+\eta-e+1+u_{\eta k}} = \theta_{\eta k} p^{u_{\eta k}}$$

for some $\alpha \in R$. Since $\theta_{\eta k}$ is a unit, it follows that p does not divide

$$p^{u_{gk}+\eta-e+1} ,$$

i.e. $\eta + u_{gk} < e$, which is (27).

Next we show that (28) follows from (27). In fact we shall show that (27) implies

$$L(A_\eta^{(k+1)}) = \max\ \{L(A_\eta^{(k)})\ ,\ \ k+1-L(A_g^{(k)})\} . \tag{31}$$

From (16), (17), (14) we have

$$L(A_\eta^{(k+1)}) \leq \max\{L(A_\eta^{(k)}),\ k-r+L(A_h^{(r)})\}$$
$$= \max\{L(A_\eta^{(k)}),\ k+1-L(A_g^{(k)})\} .$$

But the reverse inequality follows from Lemma 1, using (22), and establishes (31). The minimality of $A_\eta^{(k+1)}$ now follows from (27), (28) and Lemma 2.

Finally, suppose Case IIb obtains. To establish P_k we may assume (26) does not hold, and so, from (16), (17),

$$k - r + L(A_h^{(r)}) > L(A_\eta^{(k)}) ,$$

i.e.

$$k + 1 > L(A_\eta^{(k)}) + L(A_g^{(k)}) , \tag{32}$$

using (14). Consider the polynomial

$$q(x) = a_\eta^{(k)}(x)\{S(x)a_g^{(k)}(x) - b_g^{(k)}(x)\}$$

$$- a_g^{(k)}(x)\{S(x)a_\eta^{(k)}(x) - b_\eta^{(k)}(x)\} \tag{33}$$

$$= a_g^{(k)}(x)b_\eta^{(k)}(x) - a_\eta^{(k)}(x)b_g^{(k)}(x) . \tag{34}$$

Then, just as in (25),

$$\deg\, q(x) \le L(A_\eta^{(k)}) + L(A_g^{(k)}) - 1$$

$$< k , \quad \text{by} \quad (32) .$$

On the other hand, from (33),

$$q(x) = (p^\eta + \ldots)(\theta_{gk}p^{u_{gk}}x^k + \ldots)$$

$$- (p^g + \ldots)(\theta_{\eta k}p^{u_{\eta k}}x^k + \ldots) , \tag{35}$$

containing only terms of degree $\ge k$. Therefore $q(x)$ is identically zero. but the coefficient of x^k in (35) is

$$\theta_{gk}p^{\eta + u_{gk}} - \theta_{\eta k}p^{g + u_{\eta k}}$$

and so

$$\eta + u_{gk} = g + u_{\eta k} . \tag{36}$$

(27) now follows from (10). The remainder of the proof is the same as in Case IIa.

VI. Example

As an example we find the shortest linear recurrence that can produce the sequence $S_0 = 6$, $S_1 = 3$, $S_2 = 1$, $S_3 = 5$, $S_4 = 6$ modulo 9. The computation is displayed in a pair of tables indexed by (k, η), $k = 0, 1, \ldots, 5$ and $\eta = 0, 1$. The first table shows the pairs $(a_\eta^{(k)}(x), b_\eta^{(k)}(x))$. The second table shows the quintuples (, $, u_{\eta k}, \theta_{\eta k}, f(\eta, k)$, Case), where , $= L(a_\eta^{(k)}(x), b_\eta^{(k)}(x))$, and Case is one of I, IIa, or IIb, indicating which case obtains in the computation of $(a_\eta^{(k+1)}(x), b_\eta^{(k+1)}(x))$. From the last line of Table 1, the shortest recurrence satisfied by the sequence is

$$S_n + 4S_{n-1} + 7S_{n-2} + S_{n-3} = 0, \quad n \geq 3 .$$

Acknowledgement

We thank the referee for some very helpful comments.

Table 1: $(a_{\eta}^{(k)}(x),\ b_{\eta}^{(k)}(x))$

	$\eta = 0$	$\eta = 1$
$k = 0$	(1,0)	(3,0)
1	(1,6)	(3,0)
2	$(1+4x,6)$	(3,0)
3	$(1+4x,6+4x^2)$	$(3+4x^2,0)$
4	$(1+4x,6+4x^2)$	$(3+4x^2,0)$
5	$(1+4x+7x^2+x^3,6+x^2)$,	$(3+3x^2+5x^3,3x^2)$

Table 2: (, , $u_{\eta k}$, $\theta_{\eta k}$, $f(\eta, k)$, Case)
* means "does not apply".

	$\eta = 0$	$\eta = 1$
$k = 0$	(0,1,2,0,*)	(0,2,1,0,*)
1	(1,1,1,0,IIb)	(0,2,1,*,I)
2	(1,0,4,1,IIa)	(0,1,1,0,IIb)
3	(3,2,1,*,I)	(2,2,1,*,I)
4	(3,0,8,1,IIb)	(2,0,4,1,IIb)
5	(3,*,*,*,*)	(3,*,*,*,*)

References

1. E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, NY, 1968.

2. E. R. Berlekamp, E. M. Fredricksen and R. C. Proto, *Minimum conditions for uniquely determining the generator of a linear sequence,* Utilitas Math., **5** (1974), 305-315.

3. I. F. Blake, *Codes over certain rings,* Information and Control, **20** (1972), 396-404.

4. I. F. Blake, *Codes over integer residue rings,* Information and Control, **29** (1975), 295-300.

5. W. A. Blankinship, *Solution of simultaneous linear diophantine equations,* Algorithm 288 in *Collected Algorithms from ACM,* Assoc. Computing Mach., New York, 2 Vols. 1978.

6. R. P. Brent, F. G. Gustavson and D. Y. Yun, *Fast solution of Toeplitz systems of equations and computation of Pade´ approximants,* J. Algorithms, **1** (1980), 259-295.

7. A. Bultheel, *Recursive algorithms for nonnormal Pade´ tables,* SIAM J. Appl. Math., **39** (1980), 106-118.

8. P. H. Chen, *Multisequence linear shift register synthesis and its application to BCH decoding,* IEEE Trans. Commun., **COM-24** (1976), 438-440.

9. B. W. Dickinson, M. Morf and T. Kailath, *A minimal realization algorithm for matrix sequences,* IEEE Trans. Auto. Control, **AC-19** (1974), 31-38.

10. J. F. Dillon and R. A. Morris, *On a paper of Berlekamp, Fredricksen and Proto,* Utilitas Math., **5** (1974), 317-321.

11. R. A. Games and A. H. Chan, *A fast algorithm for determining the complexity of a binary sequence with period* 2^n, IEEE Trans. Information Theory, **IT-29** (1983), 144-146.

12. W. B. Gragg, *The Padé table and its relation to certain algorithms of numerical analysis,* SIAM Review, **14** (1972), 1-62.

13. F. G. Gustavson, *Analysis of the Berlekamp-Massey feedback shift-register synthesis algorithm,* IBM J. Res. Dev., **20** (1976), 204-212.

14. P. Henrici, *Quotient-difference algorithms,* in *Mathematical Methods for Digital Computers II,* edited by A. Ralston and H. Wilf, Wiley, NY, 1967, pp. 35-62.

15. W. B. Jones and W. J. Thron, *Continued Fractions: Analytic Theory and Applications,* Addison-Wesley, Reading, MA, 1980.

16. S. Lang, *Algebra,* Addison-Wesley, Reading, MA, 1971.

17. W. F. Lunnon, *Linear recurring sequences over the complex numbers,* unpublished manuscript, 1970.

18. R. J. McEliece, *The Theory of Information and Coding*, Addison-Wesley, Reading, MA, 1977.

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

20. D. M. Mandelbaum, *A method for decoding of generalized Goppa codes,* IEEE Trans. Information Theory, **IT-23** (1977), 137-140.

21. J. L. Massey, *Shift-register synthesis and BCH decoding,* IEEE Trans. Information Theory, **IT-15** (1969), 122-127.

22. W. M. Mills, *Continued fractions and linear recurrences,* Math. Comp., **29** (1975), 173-180.

23. N. J. Patterson, *The algebraic decoding of Goppa codes,* IEEE Trans. Information Theory, **IT-21** (1975), 203-207.

24. J. B. Plumstead, *Inferring a sequence generated by a linear congruence,* in *23rd Annual Symposium on Foundations of Computer Science,* IEEE Press, NY, 1982, pp. 153-159.

25. J. B. Plumstead, *Inferring sequences produced by pseudo-random number generators,* Ph.D. dissertation, Computer Science Dept., Univ. of Calif., Berkeley, 1983.

26. M. K. Sain, *Minimal torsion spaces and the partial input/output problem,* Info. Control, **29** (1975), 103-124.

27. D. V. Sarwate, *On the complexity of decoding Goppa codes,* IEEE Trans. Information Theory, **IT-23** (1977), 515-516.

28. P. Shankar, *On BCH codes over arbitrary integer rings,* IEEE Trans. Information Theory, **IT-25** (1979), 480-483.

29. N. J. A. Sloane, *Encrypting by random rotations*, pp. 71-128 of *Cryptography* (Proc. Workshop on Cryptography, Burg Feuerstein, Germany, March 29-April 2, 1982), Lecture Notes in Computer Science **149**, Springer-Verlag NY, 1983.

30. E. Spiegel, *Codes over* Z_m, Information and Control, **35** (1977), 48-51.

31. E. Spiegel, *Codes over* Z_m, *revisited*, Information and Control, **37** (1978), 100-104.

32. Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa, *A method for solving key equation for decoding Goppa codes*, Information Control, **27** (1975), 87-99.

33. K. K. Tzeng and G. L. Feng, *Shift-register synthesis for t sequences*, preprint.

34. L. R. Welch and R. A. Scholtz, *Continued fractions and Berlekamp's algorithm*, IEEE Trans. Information Theory, **IT-25** (1979), 19-27.

35. N. Zierler, *Linear recurring sequences and error-correcting codes*, in *Error Correcting Codes*, edited H. B. Mann, Wiley, NY, 1969, pp. 47-59.