The Berlekamp-Massey Algorithm revisited

Nadia Ben Atti (*), Gema M. Diaz–Toca (†) Henri Lombardi (‡)

Abstract

We propose a slight modification of the Berlekamp-Massey Algorithm for obtaining the minimal polynomial of a given linearly recurrent sequence. Such a modification enables to explain it in a simpler way and to adapt it to lazy evaluation.

MSC 2000: 68W30, 15A03

Key words: Berlekamp-Massey Algorithm. Linearly recurrent sequences.

1 Introduction: The usual Berlekamp-Massey algorithm

Let $\mathbb{K}$ be an arbitrary field. Given a linearly recurrent sequence, denoted by $\mathcal{S}(x) = \sum_{i=0}^{\infty} a_i x^i$, $a_i \in \mathbb{K}$, we wish to compute its minimal polynomial, denoted by $P(x)$. Recall that if $P(x)$ is given by $P(x) = \sum_{i=0}^{d} p_i x^i$ denotes such polynomial, then $P(x)$ is the polynomial of the smallest degree such that $\sum_{i=0}^{d} p_i a_{j+i} = 0$, for all j in $\mathbb{N}$.

Let suppose that the minimal polynomial of $\mathcal{S}(x)$ has degree bound n. Under such hypothesis, the Berlekamp-Massey Algorithm only requires the first $2n$ coefficients of $\mathcal{S}(x)$ in order to compute the minimal polynomial. Such coefficients define the polynomial $S = \sum_{i=0}^{2n-1} a_i\, x^i$.

A large literature can be consulted nowadays in relation to the Berlekamp's Algorithm. The (original) Berlekamp's Algorithm was created for decoding Bose-Chaudhuri-Hocquenghem (BCH) codes in 1968 (see [1]). One year later, the original version of this algorithm has been simplified by Massey (see [5]). The similarity of the algorithm to the extended Euclidean Algorithm can be found in several articles, for instance, in [2],[3], [6], [9] and [10]. Some more recent interpretations of the Berlekamp-Massey Algorithm in terms of Hankel Matrices and Padé approximations can be found in [4] and [7].

The usual interpretation of the Berlekamp-Massey Algorithm for obtaining $P(x)$ is expressed in pseudocode in Algorithm 1.

In practice, we must apply the simplification of the extended Euclidean Algorithm given in [3], to find exactly the Berlekamp-Massey Algorithm. Such simplification is based on the fact that initial R_0 is equal to x^{2n}.

Although Algorithm 1 is not complicated, it seems to be no easy to find a direct and transparent explanation for the determination of the degree of P. In the literature, we think there is a little confusion with the different definitions of minimal polynomial and with the different ways of defining

* Equipe de Mathématiques, CNRS UMR 6623, UFR des Sciences et Techniques, Université de Franche-Comté, 25 030 Besançon cedex, France. nadia.benatti@ensi.rnu.tn

† Dpto. de Matematicas Aplicada. Universidad de Murcia, Spain. gemadiaz@um.es, partially supported by the Galois Theory and Explicit Methods in Arithmetic Project HPRN-CT-2000-00114

‡ Equipe de Mathématiques, CNRS UMR 6623, UFR des Sciences et Techniques, Université de Franche-Comté, 25 030 Besançon cedex, France, lombardi@math.univ-fcomte.fr, partially supported by the European Union funded project RAAG CT-2001-00271

Algorithm 1 ***The Usual Berlekamp-Massey Algorithm***

Input: $n \in \mathbb{N}$. The first $2n$ coefficients of a linearly recurrent sequence defined over $\mathbb{K}$, given by the list $[a_0, a_1, \ldots, a_{2n-1}]$. The minimal polynomial has degree bound n.

Output : The minimal polynomial P of the sequence.

Start

Local variables : $R, R_0, R_1, V, V_0, V_1, Q$: polynomials in x

initialization

$R_0 := x^{2n}$; $R_1 := \sum_{i=0}^{2n-1} a_i\, x^i$; $V_0 = 0$; $V_1 = 1$;

loop

while $n \leq \deg(R_1)$ **do**

$(Q, R) :=$ quotient and remainder of R_0 divided by R_1 ;

$V := V_0 - Q\, V_1$;

$V_0 := V_1$; $V_1 := V$; $R_0 := R_1$; $R_1 := R$;

end while

exit

$d := \max(\deg(V_1), 1 + \deg(R_1))$; $P := x^d V_1(1/x)$; Return $P := P/\text{leadcoeff}(P)$.

End.

the sequence. Here, we introduce a slight modification of the algorithm which makes it more comprehensible and natural. We did not find in the literature such a modification before the first submission of this article (May 2004). However, we would like to add that you can also find it in [8], published in 2005, without any reference.

2 Some good reasons to modify the usual algorithm

By the one hand, as it can be observed at the end of Algorithm 1, we have to compute the (nearly) reverse polynomial of V_1, in order to obtain the right polynomial. The following example helps us to understand what happens:

$$
\begin{aligned}
& n = d = 3, \\
& S = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 = 1 + 2x + 7x^2 - 9x^3 + 2x^4 + 7x^5, \\
& \text{Algorithm } 1(3, [1, 2, 7, -9, 2, 7]) \Rightarrow P = x + x^2 + x^3, \\
& \text{with } V_1 = v_0 + v_1 x + v_2 x^2 = 49/67(1 + x + x^2), \\
& \text{and } R \text{ such that } S\, V_1 = R \bmod x^6,\ \deg(R) = 2 \\
& \text{which implies that}
\end{aligned}
$$

$$
\begin{array}{lllllll}
\text{coeff}(S\, V_1, x, 3) & = & a_1 v_2 + a_2 v_1 + a_3 v_0 & = & 2v_2 + 7v_1 - 9v_0 & = & 0, \\
\text{coeff}(S\, V_1, x, 4) & = & a_2 v_2 + a_3 v_1 + a_4 v_0 & = & 7v_2 - 9v_1 + 2v_0 & = & 0, \\
\text{coeff}(S\, V_1, x, 5) & = & a_3 v_2 + a_4 v_1 + a_5 v_0 & = & -9v_2 + 2v_1 + 7v_0 & = & 0.
\end{array}
$$

Hence, the right degree of P is given by the degree of the last R_1 plus one because x divides P. Observe that $a_0 v_2 + a_1 v_1 + a_2 v_0 = 490/67 \neq 0$. We would like to obtain directly the desired polynomial from V_1.

Moreover, by the other hand, in Algorithm 1 all the first $2\,n$ coefficients are required to start the usual algorithm, where n only provides a degree bound for the minimal polynomial. Consequently, it may be possible that the true degree of P is much smaller that n and so, less coefficients of the sequence are required to obtain the wanted polynomial.

So, we suggest a more natural, efficient and direct way to obtain P. Our idea is to consider the polynomial $\widehat{S} = \sum_{i=0}^{2n-1} a_i\, x^{2n-1-i}$ as the initial R_1. Observe that in this case, using the same notation

as in Algorithm 1, the same example shows that it is not necessary to reverse the polynomial V_1 at the end of the algorithm.

$$
\begin{aligned}
& n = d = 3, \\
& \widehat{S} = a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 = x^5 + 2\,x^4 + 7\,x^3 - 9\,x^2 + 2\,x + 7, \\
& \text{Algorithm}\,2\,(3, [1, 2, 7, -9, 2, 7]) \Rightarrow P = x + x^2 + x^3, \\
& \text{with } V_1 = v_0 + v_1x + v_2x^2 + v_3x^3 = -9/670(x + x^2 + x^3), \\
& \text{and } R \text{ such that } \widehat{S}\,V_1 = R \bmod x^6, \deg(R) = 2 \\
& \text{which implies that}
\end{aligned}
$$

$$
\begin{array}{lllllllll}
\text{coeff}(\widehat{S}\,V_1, x, 3) & = & a_2v_0 + a_3v_1 + a_4v_2 + a_5v_3 & = & -9v_1 + 2v_2 + 7v_3 & = & 0, \\
\text{coeff}(\widehat{S}\,V_1, x, 4) & = & a_1v_0 + a_2v_1 + a_3v_2 + a_4v_3 & = & 7v_1 - 9v_2 + 2v_3 & = & 0, \\
\text{coeff}(\widehat{S}\,V_1, x, 5) & = & a_0v_0 + a_1v_1 + a_2v_2 + a_3v_3 & = & 2v_1 + 7v_2 - 9v_3 & = & 0.
\end{array}
$$

Furthermore, when $n \gg \deg(P)$, the algorithm can admit a lazy evaluation. In other words, the algorithm can be initiated with less coefficients than $2n$ and if the outcome does not provide the wanted polynomial, we increase the number of coefficients but remark that it is not necessary to initiate again the algorithm because we can take advantages of the computations done before. We will explain this application of the algorithm in Section 3.

Next, we introduce our modified Berlekamp-Massey Algorithm in pseudocode (Algorithm 2):

Algorithm 2 ***Modified Berlekamp-Massey Algorithm***

Input: $n \in \mathbb{N}$. The first $2n$ coefficients of a linearly recurrent sequence defined over $\mathbb{K}$, given by the list $[a_0, a_1, \ldots, a_{2n-1}]$. The minimal polynomial has degree bound n.
Output : The minimal polynomial P of the sequence.
Start
Local variables : $R, R_0, R_1, V, V_0, V_1, Q$: polynomials in x ; $m = 2n - 1$: integer.
    # initialization
  $m := 2n - 1$; $R_0 := x^{2n}$; $R_1 := \sum_{i=0}^{m} a_{m-i}\,x^i$; $V_0 = 0$; $V_1 = 1$;
    # loop
  while $n \leq \deg(R_1)$ **do**
    $(Q, R) :=$ quotient and remainder of R_0 divided by R_1;
    $V := V_0 - Q\,V_1$;
    $V_0 := V_1$; $V_1 := V$; $R_0 := R_1$; $R_1 := R$;
  end while
    # exit
  Return $P := V_1/\text{lc}(V_1)$;
End.

Now we prove our result. Let $\underline{a} = (a_n)_{n\in\mathbb{N}}$ be an arbitrary list and i, r, $p \in \mathbb{N}$. Let $\mathrm{H}^{\underline{a}}_{i,r,p}$ denote the following Hankel matrix of order $r \times p$,

$$
\mathrm{H}^{\underline{a}}_{i,r,p} = \begin{bmatrix}
a_i & a_{i+1} & a_{i+2} & \ldots & a_{i+p-1} \\
a_{i+1} & a_{i+2} & & & a_{i+p} \\
a_{i+2} & & & & \\
\vdots & & & & \vdots \\
a_{i+r-1} & a_{i+r} & \ldots & \ldots & a_{i+r+p-2}
\end{bmatrix}
$$

and let $\mathrm{P}^{\underline{a}}(x)$ be the minimal polynomial of $\underline{a}$.

The next proposition shows the well known relation between the rank of Hankel matrix and the sequence.

Proposition 1 *Let $\underline{a}$ be a linearly recurrent sequence . If $\underline{a}$ has a generating polynomial of degree $\leq n$, then the degree d of its minimal polynomial $\mathrm{P}^{\underline{a}}$ is equal to the rank of the Hankel matrix*

$$\mathrm{H}^{\underline{a}}_{0,n,n} = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & & \iddots & a_{n-1} & a_n \\ a_2 & & \iddots & \iddots & \vdots & \vdots \\ \vdots & \iddots & \iddots & & \vdots & \vdots \\ a_{n-2} & a_{n-1} & \cdots & \cdots & a_{2n-2} & a_{2n-1} \\ a_{n-1} & a_n & \cdots & \cdots & a_{2n-1} & a_{2n-2} \end{bmatrix}.$$

The coefficients of $\mathrm{P}^{\underline{a}}(x) = x^d - \sum_{i=0}^{d-1} g_i x^i \in \mathbb{K}[x]$ are provided by the unique solution of the linear system

$$\mathrm{H}^{\underline{a}}_{0,d,d}\, G = \mathrm{H}^{\underline{a}}_{d,d,1},$$

that is,

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{d-1} \\ a_1 & a_2 & & \iddots & a_d \\ a_2 & & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & & \vdots \\ a_{d-1} & a_d & \cdots & \cdots & a_{2d-2} \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{d-1} \end{bmatrix} = \begin{bmatrix} a_d \\ a_{d+1} \\ a_{d+2} \\ \vdots \\ a_{2d-1} \end{bmatrix}. \tag{1}$$

As an immediate corollary of Proposition 1, we have the following result.

Corollary 2 *Using the notation of Proposition 1, a vector $Y = (p_0, \ldots, p_n)$ is solution of*

$$\mathrm{H}^{\underline{a}}_{0,n,n+1}\, Y = 0,$$

that is,

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} & a_n \\ a_1 & a_2 & & \iddots & a_n & a_{n+1} \\ a_2 & & \iddots & \iddots & \vdots & \vdots \\ \vdots & \iddots & \iddots & & \vdots & \vdots \\ a_{n-1} & a_n & \cdots & \cdots & a_{2n-2} & a_{2n-1} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix} = 0 \tag{2}$$

if and only if the polynomial $P(x) = \sum_{i=0}^{n} p_i x^i \in \mathbb{K}[x]$ is multiple of $\mathrm{P}^{\underline{a}}(x)$.

Proof.
By Proposition 1 the dimension of $\mathrm{Ker}(\mathrm{H}^{\underline{a}}_{0,n,n+1})$ is $n-d$. For $0 \leq j \leq n-1$, let C_j denote the jth column of $\mathrm{H}^{\underline{a}}_{0,n,n+1}$, that is $C_j = \mathrm{H}^{\underline{a}}_{j,n,1} = [a_j, a_{j+1}, \ldots, a_{n+j-1}]^t$. Since $\mathrm{P}^{\underline{a}}(x)$ is a generating polynomial of $\underline{a}$, for $d \leq j \leq n-1$, we obtain that

$$C_j - \sum\nolimits_{i=j-d}^{j-1} g_{i-j+d}\, C_i = 0.$$

Thus the linear independent columns $[-g_0, \ldots, -g_{d-1}, 1, 0, \ldots, 0]^t, \ldots, [0, \ldots, 0, -g_0, \ldots, -g_{d-1}, 1]^t$ define a basis of $\mathrm{Ker}(\mathrm{H}^{\underline{a}}_{0,n,n+1})$. Therefore, $Y = (p_0, \ldots, p_n)$ verifies $\mathrm{H}^{\underline{a}}_{0,n,n+1}\, Y = 0$ if and only if the polynomial $P(x) = \sum_{i=0}^{n} p_i x^i$ is a multiple of $\mathrm{P}^{\underline{a}}(x)$. □

If we consider $m = 2n-1$ and $\widehat{S} = \sum_{i=0}^{m} a_{m-i} x^i$, by applying Equation (2) we obtain:

$$\exists\, R, U \in \mathbb{K}[x] \text{ such that } \deg(R) < n,\ \deg(P) \leq n \text{ and } P(x)S(x) + U(x)x^{2n} = R(x). \tag{3}$$

Hence, it turns out that finding the minimal polynomial of $\underline{a}$ is equivalent to solving (3) for the minimum degree of P. Moreover, it's well known that

- the extended Euclidean Algorithm, with x^{2n} and $\widehat{S}$, provides an equality as (3) when the first remainder of degree smaller than $< n$ is reached. Let denote such remainder by R_k,
- if we consider other polynomials $P'(x)$, $U'(x)$ and $R'(x)$ such that $P'(x)\widehat{S}(x)+U'(x)x^{2n} = R'(x)$ and $\deg(R') < \deg(R_{k-1})$, then $\deg(P') \geq \deg(P)$ and $\deg(U') \geq \deg(U)$.

That proves that our modification of Berlekamp-Massey Algorithm is right.

3 Lazy Evaluation

Our modified Berlekamp-Massey Algorithm admits a lazy evaluation, which may be very useful in solving the following problem.

Let $f(x) \in \mathbb{K}[x]$ be a squarefree polynomial of degree n. Let B be the universal decomposition algebra of $f(x)$, let A be a quotient algebra of B and $a \in A$. Thus, A is a zero–dimensional algebra given by

$$A \simeq \mathbb{K}[X_1, \ldots, X_n]/\langle f_1, \ldots, f_n \rangle ,$$

where $f_1, \ldots, f_n$ define a Gröbner basis. Our aim is to compute the minimal polynomial of a, or at least, one of its factors. However, the dimension of A, denoted by m, over $\mathbb{K}$ as vector space is normally too big to manipulate matrices of order m. Therefore, we apply the idea of Wiedemann's Algorithm, by computing the coefficients of a linearly recurrent sequence, $a_t = \phi(x^t)$, where ϕ is a linear form over A. Moreover, since the computation of x^t is usually very expensive and the minimal polynomial is likely to have degree smaller than the dimension, we are interested in computing the smallest possible number of coefficients in order to get the wanted polynomial.

Hence, we first choose $l < m$. We start Algorithm 2 with l and $[\phi(x^0), \ldots, \phi(x^{2l-1})]$ as input, obtaining a polynomial as a result. Now, we test if such a polynomial is the minimal one. If this is not the case, we choose again another l', $l < l' \leq m$, and we repeat the process with $2l'$ coefficients. However, in this next step, it is possible to take advantages of all the quotients computed before (with the exception of the last one), such that Euclidean Algorithm starts at $R_0 = U_0 x^{2l'} + V_0 \sum_{i=0}^{2l'-1} (\phi(x^{2l'-1-i})x^i)$ and $R_1 = U_1 x^{2l'} + V_1 \sum_{i=0}^{2l'-1} (\phi(x^{2l'-1-i})x^i)$, where U_0, V_0, U_1 and V_1 are Bezout coefficients computed in the previous step. Manifestly, repeating this argument again and again, we obtain the minimal polynomial.

The following pseudocode tries to facilitate the understanding of our lazy version of Berlekamp-Massey Algorithm.

Obviously, the choice of l is not unique. Here we have started at $l = m/4$, adding two coefficients in every further step. In practice, the particular characteristics of the given problem could help to choose a proper l and the method of increasing it through the algorithm. Of course, the simplification of the Euclidean Algorithm in [3] must be considered to optimize the procedure.

Algorithm 3 ***The lazy Berlekamp-Massey Algorithm (in some particular context)***

Input: $m \in \mathbb{N}$, $C \in \mathbb{K}^n$, G: Gröbner basis, $a \in A$. The minimal polynomial has degree bound m.
Output : The minimal polynomial P of a

Start
Local variables : l, i: integers, $R, R_{-1}, R_0, R_1, V, V_{-1}, V_0, V_1, U, U_{-1}, U_0, U_1, S_0, S_1, Q$: polynomials in x, L, W:lists, validez;

initialization
$l = \lfloor m/4 \rfloor$;
$L := [1, a]$; $W := [1, \mathrm{Value}(a, C)]$;
$S_0 := x^{2l}$; $S_1 = W[1]\, x^{2l-1} + W[2]\, x^{2l-2}$;
# loop
for i **from** 3 **to** $2l$ **do**
$L[i] := \mathrm{normalf}(L[i-1]a, G)$; $V[i] := \mathrm{Value}(L[i], C)$; $S_1 = S_1 + V[i]x^{2l-i}$;
end for
$R_0 := S_0$; $R_1 := S_1$; $V_0 = 0$; $V_1 = 1$; $U_0 = 1$; $V_1 = 0$;
# loop
while $l \leq \deg(R_1)$ **do**
$(Q, R) :=$ quotient and remainder of R_0 divided by R_1 ;
$V := V_0 - QV_1$; $U := U_0 - QU_1$; $U_{-1} := U_0$; $V_{-1} := V_0$;
$V_0 := V_1$; $V_1 := V$;$U_0 := U_1$; $U_1 := U$; $R_0 := R_1$; $R_1 := R$;
end while
validez:=Subs$(x = a, V_1)$;
# loop
while validez $\neq 0$ **do**
$l := l + 1$;
# loop
for i **from** $2l - 1$ **to** $2l$ **do**
$L[i] := \mathrm{normalf}(L[i-1]a, G)$;
$W[i] := \mathrm{Value}(L[i], C)$;
end for
$S_0 = x^2 S_0$; $S_1 = x^2 S_1 + W[2l-1]x + W[2l]$;
$R_0 := U_{-1}S_0 + V_{-1}S_1$; $R_1 := U_0 S_0 + V_0 S_1$;
$U_1 := U_0$; $V_1 := V_0$; $U_0 := U_{-1}$; $V_0 := V_{-1}$;
# loop
while $l \leq \deg(R_1)$ **do**
$(Q, R) :=$ quotient and remainder of R_0 divided by R_1 ;
$V := V_0 - QV_1$; $U := U_0 - QU_1$; $U_{-1} := U_0$; $V_{-1} := V_0$;
$V_0 := V_1$; $V_1 := V$;$U_0 := U_1$; $U_1 := U$; $R_0 := R_1$; $R_1 := R$;
end while
validez:=Subs$(x = a, V_1)$
end while # exit
Return $P := V_1/\mathrm{leadcoeff}(P)$.
End.

References

[1] E.R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, ch. 7 (1968).

[2] U. Cheng, *On the continued fraction and Berlekamp's algorithm*, IEEE Trans. Inform. Theory, vol. IT-30, 541–44 (1984).

[3] J.L. Dornstetter, *On the equivalence Between Berlekamp's and Euclid's Algorithm*, IEEE Trans. Inform. Theory, vol. IT-33, no 3,428–431 (1987).

[4] E. Jonckheere and C. Ma, *A simple Hankel Interpretation of the Berlekamp–Massey Algorith*, Linear Algebra and its Applications 125, 65–76 (1989).

[5] J.L. Massey, *Shift register synthesis and BCH decoding*, IEEE Trans. Inform. Theory, vol. IT-15, 122–127 (1969).

[6] W.H. Mills, *Continued Fractions and Linear Recurrences*, Math. Comput. 29, 173–180 (1975).

[7] V. Pan, *New Techniques for the Computation of linear recurrence coefficients*, Finite Fields and Their Applications 6, 93–118 (2000).

[8] V. Shoup, *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press (2005).

[9] Y. Sugiyama et al. *A method for solving key equation for decoding Goppa codes*, Infor. Contr. vol 27, 87–99 (1975).

[10] L.R. Welch and R.A. Scholtx, *Continued fractions and Berlekamp's algorithm,* IEEE Trans. Inform. Theory, vol. IT-25, 18–27 (1979).