Linear Algebra and Applications, 439(9):2515-2526, November 2013.

A Fraction Free Matrix Berlekamp/Massey Algorithm*

Erich Kaltofen and George Yuhasz
Dept. of Mathematics, North Carolina State University,
Raleigh, North Carolina 27695-8205 USA
kaltofen@math.ncsu.edu; gyuhasz@math.ncsu.edu; http://www.kaltofen.us

Abstract

We describe a fraction free version of the Matrix Berlekamp/Massey algorithm. The algorithm computes a minimal matrix generator of linearly generated square matrix sequences over an integral domain. The algorithm performs all operations in the integral domain, so all divisions performed are exact. For scalar sequences, the matrix algorithm specializes to a more efficient algorithm than the algorithm currently in the literature. The proof of integrality of the matrix algorithm gives a new proof of integrality for the scalar specialization.

1 Introduction

We present a fraction free Matrix Berlekamp/Massey algorithm to compute a integral minimal matrix generator of a linearly generated square matrix sequence over an integral domain. The algorithm performs exact divisions so that all intermediate values remain in the integral domain. The exact divisions control the coefficient growth of the intermediate values. The scalar specialization of the matrix algorithm (square matrix sequences of dimension 1) is a different scalar algorithm than other algorithms in previous literature. This scalar algorithm uses a more efficient implementation polynomial pseudo-division. As a result, the algorithm maintains smaller intermediate values.

There has been much work done on translating algorithms designed with field arithmetic into fraction free algorithms that work over integral domains. A well known algorithm is fraction free Gaussian elimination [Bareiss 1968]. In [Nakos et al. 1997; Corless et al. 2006; Zhou and Jeffrey 2008], the authors describe extensions of the fraction free Gaussian elimination into other common linear algebra algorithms such LU decomposition, diagonalization, etc. Fraction free forms of the Euclidean algorithm for polynomials has been explored by [Collins 1967; Brown 1971; Brown and Traub 1971; Brown 1978]. An important result in [Brown and Traub 1971] is the Fundamental Theorem of Subresultants.

The Berlekamp/Massey algorithm ([Berlekamp 1968; Massey 1969]) computes a minimal generator of a linearly generated scalar sequence. The algorithm is equivalent to the extended Euclidean algorithm [Dornstetter 1987]. In [Dickinson et al. 1974; Coppersmith 1994;

*This research was supported in part by the National Science Foundation of the USA under Grants CCR-0305314 and CCF-0514585.

Kaltofen and Yuhasz 2013] the authors describe generalizations of the Berlekamp/Massey algorithm to compute minimal matrix generators of linearly generated matrix sequences. A fraction free variant of the Berlekamp/Massey algorithm is given in [Giesbrecht et al. 2002]. The method is an application of the Fundamental Theorem of Subresultants to the Dornstetter transformation of the algorithm.

A fraction free version of the Matrix Berlekamp/Massey algorithm seems to be missing from the literature. We describe a fraction free version of of the Matrix Berlekamp/Massey algorithm for square matrix sequences. The algorithm requires that the matrix sequence be a *normalizable remainder sequence*. These normalizable remainder sequences are similar to the sequences in [Kaltofen and Villard 2004]. Normalizable here requires the non-singularity of the matrix leading coefficients in the remainders (discrepancies), not that the degrees of all quotient polynomials are 1 in the "normal polynomial remainder sequences" in the literature. All scalar sequences are normalizable in our sense. Our normality requirement is also related to the *normal points* used in to compute fraction free matrix padé systems in [Beckermann et al. 1997]. A fraction free matrix Berlekamp/Massey algorithm for arbitrary sequences has not been found. When the matrix algorithm is specialized to the scalar case, the resulting algorithm is different than that of [Giesbrecht et al. 2002]. The latter algorithm relies on pseudo-division as prescribed by its use of subresultants. Our algorithm builds the pseudo-division step by step in the manner of [Hearn 1972] instead of using a large power multiplication. Thus the intermediate values of the algorithm are smaller or equal to those of the previous algorithm.

Section 2 describes the fraction free Matrix Berlekamp/Massey algorithm and the issues surrounding the algorithm. Section 3 gives a proof of integrality and a discussion of the minimality of the algorithm. Section 5 has two example scalar sequences and a theorem describing the unique minimal generators of scalar sequences.

2 Fraction Free Matrix Berlekamp/Massey Algorithm

We begin by describing the fraction free Matrix Berlekamp/Massey algorithm. The algorithm is given as a matrix algorithm but the scalar equivalents are given when they are simplified by the considering the case $N = 1$. A discussion of normalizable remainder sequences and their importance to Algorithm 2.1 follows.

2.1 Algorithm

Input $M(z) \in D^{N\times N}[[z]]$ the power series defined by the matrix sequence where $M(z) = \sum_{i\geq 0} M_i z^i$ and $M_i \in D^{N\times N}$ with D an integral domain.
δ an upper bound on the degree of the matrix generator.
Note that the algorithm will process at most $2\cdot\delta$ elements of the sequence.

Output $F(z) \in D^{N\times N}[z]$ a minimal matrix generator with $\deg(F(z)) \leq \delta$. The generator is valid for the given δ. The generator $F(z)$ is the minimal matrix generator of a sequence with the same sequence prefix as that of $M(z)$. If the generator is not the minimal generator of the given sequence, then the minimal generator of $M(z)$ has degree greater than δ or the sequence is not a normalizable remainder sequence.

In the case that the initial prefix of the input sequence is not a normalizable remainder sequence, then the algorithm returns "singular sequence".

Variables

t an index for the current sequence element being processed.

$\Lambda_t(z) \in D^{N\times N}[z]$ the reversal current generator.

$B_t(z) \in D^{N\times N}[z]$ the auxiliary polynomial used to eliminate discrepancies.

L_t the nominal degree of $\Lambda_t(z)$. We have throughout the algorithm that $\deg(\Lambda_t) \leq L_t$.

Δ_t, the coefficient of z^t in the polynomial $M(z)\cdot\Lambda_{t-1}(z)$, often referred to in literature as the current discrepancy.

$\rho \in D$ the diagonal entry of the evaluation of $B_{t-1}(z)$. ρ is initialized to 1.

ϵ a counter of the number of times the current ρ has been used.

γ the difference in the new and previous degree when L_t is increased.

$g \in D$ and $h \in D$ scalar divisors used to control coefficient growth.

FFMBM1 $t \leftarrow 0$
$\Lambda_{-1}(z) \leftarrow I_N$
$B_{-1}(z) \leftarrow 0^{N\times N}$
$L_{-1} \leftarrow 0$
$\rho \leftarrow 1$
$g \leftarrow h \leftarrow 1$.

FFMBM2 while $t+1-L_{t-1} \leq \delta$ do Steps FFMBM3 through FFMBM12

FFMBM3 $\Delta_t \leftarrow \mathit{Coeff}\,(t; M(z)\cdot\Lambda_{t-1}(z))$

FFMBM4 if $\Delta_t \neq 0^{N\times N}$ and $2L_{t-1} < t+1$ do step FFMBM5

FFMBM5 if $\det(\Delta_t) = 0$ then return "singular sequence"

We must compute $\mathrm{adj}(\Delta_t)$ and $\det(\Delta_t)$. Both can be computed using the fraction free diagonalization algorithm in [Nakos et al. 1997].

$$\Lambda_t(z) \leftarrow \rho\cdot\Lambda_{t-1}(z) - B_{t-1}(z)\cdot\Delta_t$$

The multiplication on the left is scalar multiplication and the multiplication on the right is matrix multiplication. If $N = 1$ then both multiplications are scalar multiplications.

$$B_t(z) \leftarrow z\cdot\Lambda_{t-1}(z)\cdot\mathrm{adj}(\Delta_t)$$

Note that if $N = 1$, then $\mathrm{adj}(\Delta_t) = 1$.

$$\rho \leftarrow \det(\Delta_t)$$
$$L_t \leftarrow t+1-L_{t-1}$$
$$\gamma \leftarrow t+1-2L_{t-1} = L_t - L_{t-1}$$

$\epsilon \leftarrow 0$

FFMBM6 if $\Delta \neq 0^{N\times N}$ and $2L_{t-1} \geq t+1$ do step FFMBM7

FFMBM7
$$\Lambda_t(z) \leftarrow \rho \cdot \Lambda_{t-1}(z) - B_{t-1}(z) \cdot \Delta_t$$
$$B_t(z) \leftarrow z \cdot B_{t-1}(z)$$
$$L_t \leftarrow L_{t-1}$$
$$\epsilon \leftarrow \epsilon + 1$$

The use of ϵ is an implementation of the pseudo-division described in [Hearn 1972]. The algorithm skips over zeros in the pseudo-division and recovers the necessary multiplications in step FFMBM11. Brown [1978] describes a similar improvement of the PRS algorithm.

FFMBM8 if $\Delta_t = 0^{N\times N}$ do step FFMBM9

FFMBM9
$$\Lambda_t(z) \leftarrow \Lambda_{t-1}(z)$$
$$B_t(z) \leftarrow z \cdot B_{t-1}(z)$$
$$L_t \leftarrow L_{t-1}$$

FFMBM10 if $2L_t = t+1$ do step FFMBM11

FFMBM11
$$\Lambda_t(z) \leftarrow \frac{1}{g \cdot h^{\gamma \cdot N}} \cdot (\rho^{\gamma-\epsilon} \cdot \Lambda_t(z)) \quad \text{(the divisions are exact)}$$
$$g \leftarrow \rho$$
$$h \leftarrow \frac{g^{\gamma}}{h^{\gamma \cdot N - 1}} \quad \text{(the division is exact)}$$

FFMBM12 $t \leftarrow t+1$

FFMBM13 end while

FFMBM14 return $F(z) = z^{L_{t-1}} \cdot \Lambda_{t-1}(z^{-1})$

2.2 Normalizable Remainder Sequence

The fraction free Matrix Berlekamp/Massey algorithm in section 2.1 will always return a candidate generator for scalar sequences. However, when the input sequence has a higher dimension, then the input sequences must be a normalizable remainder sequence. Our concept of a normalizable remainder sequence is related to the results of Kaltofen and Villard [2004]. There the authors use a half-gcd algorithm to compute a minimal matrix generator. The half-gcd algorithm requires that the leading term of each remainder must be nonsingular so the matrix polynomial division is defined. Further, they require that the degree difference between each successive remainder polynomial is 1, so there is no gap that can exist in a gcd calculation.

Algorithm 2.1 enforces a similar nonsingularity requirement but removes the gap restriction. During the execution of the Algorithm 2.1, the degree of the candidate minimal generator is updated if step FFMBM5 is performed. Whenever a nonzero discrepancy requires

a degree increase, the discrepancy must be nonsingular. Such degree increases correspond to the division step of the half-gcd algorithm of Kaltofen and Villard [2004]. The determininant of the discrepancy is calculated using the diagonalization algorithm of Nakos et al. [1997]. If the determinant is zero, then the algorithm returns an error statement "singular sequence". Since all scalar sequences are normalizable remainder sequences, then the scalar algorithm will not return "singular sequence" for any input. The proofs of integrality given in section 3.2 require that the discrepancy have nonzero determinant. To relax the requirement of normalizable remainder sequences, a new algorithm with different proofs is needed and currently unknown. The use of the gap variable γ that represents the degree gap between successive generator degrees, allows us to remove the gap condition present in Kaltofen and Villard [2004].

Such nonsingularity requirements are not uncommon in polynomial algebra. In Beckermann et al. [1997], the authors compute matrix padé systems in a fraction free manner. Their algorithm proceeds on a diagonal path from *normal* point to *normal* point. These normal points are points in the padé table that obey a nonsingular requirement. These normal points are very similar to our normalizable remainder sequences, since both ideas imply that a well defined block matrix is nonsingular. In their case, a normal point is a point where a block Sylvester matrix is nonsingular. In our situation, a normalizable remainder sequence implies that the degree of each column of the generator is equal. Further as we shall see in Section 3, this implies that the $L_t \times L_t$, block Hankel matrix defined by the sequence is nonsingular for all t such that $L_t > 0$. In Beckermann and Labahn [2000], the authors generalize the definition of a normal point. They then are able to compute around singularities. Unfortunately we have not been able to apply those results and make a more general fraction free Matrix Berlekamp/Massey algorithm similar to the algorithm in Kaltofen and Yuhasz [2013].

3 Proof of correctness

The proof of correctness for Algorithm 2.1 is given in two parts. First we show that the algorithm computes a minimal matrix generator if the bound δ is correct. If the input δ is two small, then as is the case in Kaltofen and Yuhasz [2013], the candidate generator may be incorrect. Second we will show that all operations performed by Algorithm 2.1 are integer operations and all the variables remain integral.

3.1 Minimality of the output

In Kaltofen and Yuhasz [2013], the authors proved the correctness of the general Matrix Berlekamp/Massey algorithm for field arithmetic. We will restate some of the important theorems here and give different proofs for some of the theorems that are affected by the change in the update procedure of the new algorithm. We will also restate the importance of δ and the terminating condition, which were first given in Kaltofen and Yuhasz [2013].

We begin with the following definition.

Definition 1 *Define the quantity b_{t-1} at every stage t of Algorithm 2.1 by $b_{t-1} = t+1-L_{t-1}$. The quantity b_{t-1} is the nominal degree of the polynomial $B_{t-1}(z)$. An alternate definition is $b_t = t + 2 - L_t$.*

As stated in the definition, b_t is the nominal degree of the auxiliary polynomial B_t. The value of b_t corresponds to the value β in the general Matrix Berlekamp/Massey algorithm of Kaltofen and Yuhasz [2013]. Unlike, the general algorithm, we do not need to maintain the value of b_t because it is a well defined formula involving t and L_t. In the general algorithm, β must be maintained because the nominal degree of the auxiliary (and generator) columns are not guaranteed to be equal. Algorithm 2.1 has this regularity since it only works on normalizable remainder sequences. In the general scalar algorithm, the main loop continues so long as $\beta \leq \delta$. So our algorithm maintains the same termination criteria. For if $\beta > \delta$, then $2L_{t-1} < t+1$ and so any nonsingular discrepancy will increase the degree of the generator to $\beta > \delta$.

The definition of b_t also allows us to state the next lemma. The following lemma is a restatement of Lemma 9 in Kaltofen and Yuhasz [2013]. This lemma was first used by Coppersmith as one of the conditions his version of the Matrix Berlekamp/Massey algorithm maintained [Coppersmith 1994].

Lemma 1 *At the completion of every stage t, the following holds.*

- $\mathit{Coeff}\,(l; M(z)\cdot\Lambda_t(z)) = 0^{N\times N}$ *for all l such that $L_t \leq l \leq t$.*
- $\mathit{Coeff}\,(l; M(z)\cdot B_t(z)) = 0^{N\times N}$ *for all l such that $t+2-L_t = b_t \leq l \leq t$.*

Proof. For $t=-1$, the lemma is true by default since both ranges are empty.

Suppose $t \geq -1$ and the lemma holds at t. At stage $t+1$, there are three possible updates depending on the L_t and Δ_{t+1}. We will analyze all three updating procedures to prove the lemma.

First suppose that $2L_t < t+2$ and $\Delta_{t+1} = 0^{N\times N}$. So $\Lambda_{t+1}(z) = \Lambda_t(z)$, $L_{t+1} = L_t$ and $B_{t+1}(z) = z\cdot B_t(z)$. So by induction $\mathit{Coeff}\,(l; M(z)\cdot\Lambda_{t+1}(z)) = 0^{N\times N}$ for all $L_{t+1} \leq l \leq t$. Further, since $\Delta_{t+1} = \mathit{Coeff}\,(t+1; M(z)\cdot\Lambda_{t+1}(z)) = 0^{N\times N}$, then the condition holds for $\Lambda_{t+1}(z)$. Since $B_{t+1}(z) = z\cdot B_t(z)$, then by induction we know for all l such that $t+3-L_{t+1} \leq l \leq t+1$ we see that $\mathit{Coeff}\,(l; M(z)B_{t+1}(z)) = \mathit{Coeff}\,(l-1; M(z)B_t(z)) = 0^{N\times N}$ with $t+2-L_{t+1} \leq l-1 \leq t$. So the condition holds.

Next suppose that $2L_t < t+2$ and $\Delta_{t+1} \neq 0^{N\times N}$ is nonsingular. So $B_{t+1}(z) = z\cdot\Lambda_t(z)\cdot \mathrm{adj}(\Delta_{t+1})$ and $L_{t+1} = t+2-L_t$. thus by induction, for all l such that $t+3-L_{t+1} = L_t+1 \leq l \leq t+1$, we know that $\mathit{Coeff}\,(l; M(z)B_{t+1}(z)) = \mathit{Coeff}\,(l-1; M(z)\Lambda_t(z))\cdot\mathrm{adj}(\Delta_{t+1}) = 0^{N\times N}\cdot\mathrm{adj}(\Delta_{t+1}) = 0^{N\times N}$ with $L_t \leq l-1 \leq t$. So the condition holds for $B_{t+1}(z)$. Note that $\mathit{Coeff}\,(t+2; M(z)B_{t+1}(z)) = \rho\cdot I_N$. If $L_t = 0$, then $L_{t+1} = t+2$ and so the range is empty and the condition holds for $\Lambda_{t+1}(z)$ by default. Otherwise, since $\Lambda_{t+1}(z) = \rho\cdot\Lambda_t(z)\cdot -B_t(z)\cdot\Delta_{t+1}$, then by induction, for all l such that $t+2-L_t = L_{t+1} \leq l \leq t$ we know that $\mathit{Coeff}\,(l; M(z)\Lambda_{t+1}(z)) = \rho\cdot 0^{N\times N} - 0^{N\times N}\cdot\Delta_{t+1} = 0^{N\times N}$. Further since $L_t > 0$, then $B_t(z)$ is nonzero and as above $\mathit{Coeff}\,(t+1; M(z)B_t(z)) = \rho\cdot I_N$. So we see that $\mathit{Coeff}\,(t+1; M(z)\Lambda_{t+1}(z)) = \rho\cdot\Delta_{t+1} - (\rho\cdot I_N)\cdot\Delta_{t+1} = 0^{N\times N}$. Thus the condition holds for $\Lambda_{t+1}(z)$.

Finally, suppose $2L_t \geq t+2$. Its obvious that if we prove the condition after step 7, then the scalar adjustment of step 11 will not affect the condition. In step 7, we see that $\Lambda_{t+1}(z) = \rho\cdot\Lambda_t(z) - B_t(z)\cdot\Delta_{t+1}$ and $L_{t+1} = L_t$. So as in the previous case, induction implies that for all l such that $L_t = L_{t+1} \leq l \leq t$ we know that $\mathit{Coeff}\,(l; M(z)\Lambda_{t+1}(z)) =$

$\rho \cdot 0^{N\times N} - 0^{N\times N} \cdot \Delta t + 1 = 0^{N\times N}$. Further, as in the previous case, we see that $Coeff(t+1; M(z)\Lambda_{t+1}(z)) = \rho \cdot \Delta_{t+1} - (\rho \cdot I_N) \cdot \Delta_{t+1} = 0^{N\times N}$. So the condition holds for $\Lambda_{t+1}(z)$. Like the first case, we have $B_{t+1}(z) = z \cdot B_t(z)$. So the proof of the condition is the same and the condition holds for $B_{t+1}(z)$.

Thus the conditions hold at stage $t+1$ for every possible update, and so by induction the conditions hold at every stage. ■

Since Lemma 1 holds for our fraction free Matrix Berlekamp/Massey algorithm, then the subsequent lemmas from Kaltofen and Yuhasz [2013] also hold for the algorithm. Thus, Theorem 4, Lemma 12, Lemma 13 and Theorem 5 imply that Algorithm 2.1 has the following output options. The algorithm returns a minimal matrix generator for a prefix of the matrix sequence $\{M_k\}_{k=0}^{\infty}$. This generator is the only possible generator with degree less than δ for any continuation of of the sequence. Thus the returned generator is the unique generator for the given matrix sequence prefix and the given δ. Finally, if the algorithm returns "singular sequence" then this is a certificate that the sequence is not a normalizable remainder sequence. Theorem 5 of Kaltofen and Yuhasz [2013] also states that the fraction free Matrix Berlekamp/Massey algorithm will process at most a prefix of 2δ elements of the sequence.

Both the general Matrix Berlekamp/Massey algorithm of Kaltofen and Yuhasz [2013] and Algorithm 2.1 have error outputs, but the error outputs of each algorithm are not the same. As previously stated, if Algorithm 2.1 return "singular sequence", then the algorithm has diagnosed that the sequence is not a normalizable remainder sequence. This condition is not a requirement of the general Matrix Berlekamp/Massey algorithm since that algorithm computes a minimal matrix generator for arbitrary matrix sequences. Therefore the general algorithm does not produce the "singular sequence" output. The general algorithm's error output is "insufficient bound". Algorithm 2.1 does not produce this error because the δ of the algorithm is different than the δ of the general algorithm, which we will denote as $\bar{\delta}$. The two values are related by the formula $\bar{\delta} = N\delta$. In the general algorithm, the terminating value $\bar{\delta}$ is a condition not on the degree of the generator but the determinantal degree. Further, during an update, the determinantal degree could increase past $\bar{\delta}$ while the loop condition had been satisfied. This is not possible in Algorithm 2.1 since every column of the generator has the same nominal degree, meaning the determinantal degree will always be $N \cdot L_t$ at every stage t. The enforcement of the sequence being normalizable, means that Algorithm 2.1 would return "insufficient bound" only if $N \cdot L_t > N \cdot \delta$, which implies that $L_t > \delta$. Assuming that the degree was increased at stage t, then this implies that $L_t = t + 1 - L_{t-1} > \delta$. This condition violates the loop condition and so the algorithm would have ended before the update. Thus Algorithm 2.1 cannot diagnose if δ is too small, but will instead compute a generator that is valid for the given δ and a certain prefix of the matrix sequence.

3.2 Integrality

Having established that Algorithm 2.1 computes a minimal matrix generator, we are left to show that every intermediate value remains integral. To do this, we will show that h and $\Lambda_{t-1}(0)$ are determinants of integral matrices, generalizing the subresultant based identities of [Giesbrecht et al. 2002] to matrix polynomials by giving a direct argument. Being determinants obviously implies integrality of those values, but by applying Cramer's rule, this implies the integrality of every intermediate value.

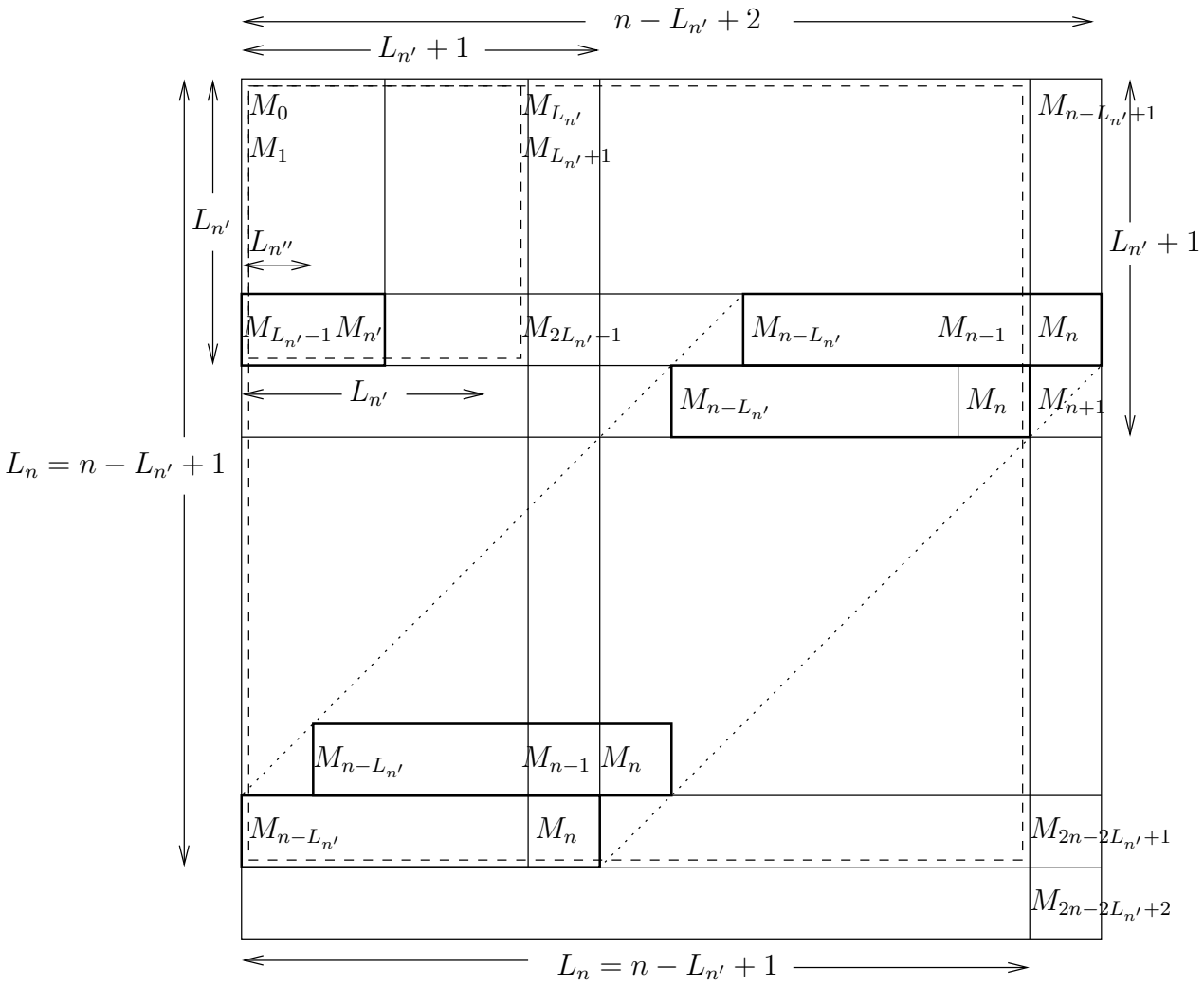

Figure 1: Berlekamp/Massey algorithm

Figure 1 follows the execution as in [Kaltofen and Lee 2003, Fig. 1]. For $t = n$ a non-zero discrepancy Δ_n has caused execution of Step FFMBM5. The generator degree has changed last at $t = n'$, i.e., $2L_{n-1} = 2L_{n'} \leq n$. So L_n is increased to $L_n = n+1-L_{n'} > L_{n'}$, and subsequent iterations execute either Step FFMBM7 or Step FFMBM8 until $t = 2L_n - 1 = 2n - 2L_{n'} + 1 > n$ without changing L_t, i.e., "complete the square." Additionally, at $t = 2L_n - 1$ the divisions of Step FFMBM11 adjust the generator polynomial and the auxiliary scalars.

Lemma 2 *In Algorithm 2.1 at the end of Step* FFMBM*11, before Step* FFMBM*12 that increments t, the following statements are always true.*

1. *The scalar $h = (-1)^{N \cdot \chi}$ times the determinant of the the principal $(NL_t) \times (NL_t)$ block Hankel matrix H_{L_t} generated by $\{M_k\}_{k=0}^{\infty}$, where $\chi = \sum_{\gamma'} \lfloor \gamma'/2 \rfloor$ with the summation taken over each value of γ' that has been computed in Step* FFMBM*5 so far. In addition, that determinant is non-zero.*

2. $\Lambda_t(0) = h\, I_N$.

Proof. We will use induction to prove the lemma. Subscripted quantities ρ_i, γ_i, ϵ_i, etc. refer, like L_i, to the values these variables have at the end of the iteration for $t = i$ at Step FFMBM12 before t is incremented. If the sequence is all zero matrices, Step FFMBM11 is never executed and the trivial identity matrix generator is returned. Thus the lemma is true.

Assume now that the algorithm encounters its first non-zero M_k, where $k \geq 0$. Then $t = k$, $\Delta_k = M_k$ and Step FFMBM5 is executed. If M_k is non-singular, the algorithm continues. In Figure 1 we then have $L_{n'} = 0$, $n = k$, and $L_{2n+1} = L_n = n + 1$, and $\gamma_n = L_n - L_{n'} = n + 1$. Step FFMBM11 occurs at $t = 2n + 1$ and the $(N(n + 1)) \times (N(n + 1))$ block Hankel matrix has the form

$$H_{L_{2n+1}} = \begin{bmatrix} 0^{N\times N} & 0^{N\times N} & \cdots & 0^{N\times N} & M_k \\ 0^{N\times N} & 0^{N\times N} & \iddots & M_k & \\ \vdots & \iddots & \iddots & & \\ 0^{N\times N} & M_k & & & \\ M_k & & & & \end{bmatrix},$$

whose determinant is $(-1)^{N\lfloor (n+1)/2\rfloor}\rho_n^{n+1} = (-1)^{N\cdot\chi_{2n+1}} h_{2n+1}$, because $\lfloor (n + 1)/2\rfloor$ block row exchanges put the matrix in block upper triangular form, where each block row exchange needs N row exchanges. Since $B_t(z)$ is always a multiple of z, $\Lambda(0)$ has been multiplied by ρ_n in Step FFMBM7 exactly ϵ_{2n+1} times, and subsequently in Step FFMBM11 an additional $\gamma_n - \epsilon_{2n+1}$ times, yielding $\Lambda_{2n+1}(0) = \rho_n^{\gamma} I_N = h_{2n+1} I_N$.

Assume that the lemma is true for $t = j' = 2L_{n'} - 1$, the last time the end of Step FFMBM11 was reached (see again Figure 1). We suppose that the first next non-zero, non-singular discrepancy is found at iteration $t = n > 2L_{n'} - 1$. Then the square is completed at iteration $t = j = 2n - 2L_{n'} + 1 > n$. We subscript the variable h by j' and j, depending for which t we have completed Step FFMBM11. Let $\Lambda_{j'}(z) = \sum_{\kappa=0}^{L_{n'}} \lambda_{j',\kappa} z^{\kappa}$, where $\lambda_{j',\kappa} \in D^{N\times N}$. When post-multiplying the $(NL_n)\times(NL_n)$ block Hankel matrix H_{L_n} (see Figure 1) by the $(L_nN)\times(L_nN)$ block upper triangular matrix

$$U_{L_n} = \begin{bmatrix} I_N & 0 & 0 & \ldots & 0 & \lambda_{j',L_{n'}} & 0 & \ldots \\ 0 & I_N & 0 & \ldots & 0 & \lambda_{j',L_{n'}-1} & \lambda_{j',L_{n'}} & \ldots \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & I_N & 0 & \lambda_{j',2} & \lambda_{j',3} & \ldots \\ 0 & 0 & 0 & \ddots & I_N & \lambda_{j',1} & \lambda_{j',2} & \ldots \\ 0 & 0 & 0 & \ddots & 0 & \lambda_{j',0} & \lambda_{j',1} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & \lambda_{j',0} \end{bmatrix}$$

(there are $\gamma_n = L_n - L_{n'}$ block columns containing matrix coefficients $\lambda_{j',\kappa}$), we get

$$\begin{bmatrix} H_{L_{n'}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta_n \\ 0 & \vdots & \iddots & * \\ 0 & \Delta_n & & \end{bmatrix}.$$

As in the induction step, this matrix requires $\lfloor \gamma_n/2 \rfloor$ block row exchanges for the matrix to be transformed to block upper triangular form, where each block row exchange requires N row exchanges. By hypothesis $\det(\lambda_{j',0}) = h_{j'}^N$. Therefore

$$\det(H_{L_n}) = \frac{\det(H_{L_{n'}})\ (-1)^{N\lfloor \gamma_n/2 \rfloor} \det(\Delta_n)^{\gamma_n}}{\det(U_{L_n})} = (-1)^{N\lfloor \gamma_n/2 \rfloor} \frac{(-1)^{N\cdot\chi_{j'}} h_{j'} \rho_n^{\gamma_n}}{h_{j'}^{\gamma_n N}} = (-1)^{N\cdot\chi_j} h_j,$$

which establishes Part 1.

For Part 2, by hypothesis $\Lambda_{j'}(0) = h_{j'} I_N$ and, similarly to the induction basis above, that coefficient gets multiplied by $\rho_{n'}$ once in Step FFMBM5 and by ρ_n exactly ϵ times in Step FFMBM7 before Step FFMBM11. Therefore

$$\Lambda_j(0) = \frac{\rho_n^{\gamma_n - \epsilon_j} \rho_n^{\epsilon_j} \rho_{n'}}{\rho_{n'} h_{j'}^{\gamma_n N}} \Lambda_{j'}(0) = \frac{\rho_n^{\gamma_n}}{h_{j'}^{\gamma_n N}}\ h_{j'} I_n = h_j I_n. \qquad \blacksquare$$

We now show that every intermediate value is integral at every stage of Algorithm 2.1.

Theorem 1 *The quantities $\Lambda_t(z)$, $B_t(z)$, ρ, Δ_t, g and h of Algorithm 2.1 are integral for all $t \geq 0$.*

Proof. We continue the notation from Lemma 2. We will again use induction on t to prove the theorem. We only need to worry about the computations at $t = 2n-1$, where n is a stage that step FFMBM5 is performed, since this is the time that step FFMBM11 is performed. All eliminations in steps FFMBM5 and FFMBM7 contain no divisions. So if the variables are integral at the completion of stage $2n-1$, then at every other step, the values must remain integral. Further this implies that Δ_t and $\mathrm{adj}(\Delta_t)$ are always integral if $\Lambda_t(z)$ is always integral. Thus g and ρ are always integral since they are $\det(\Delta_t)$ for some t at every stage. If the sequence is all zero matrices then no update is ever performed, so the theorem is true by initialization.

Let M_k be the first nonzero discrepancy where $k \geq 0$. At $t = 2k-1$, Step 11 is performed. Since $g_t = h_t = 1$, then there is no division and so all of the values are integral.

Assume that the theorem is true for $t = j' = 2L_{n'} - 1$, the last time the end of Step FFMBM11 was reached (see again Figure 1). We suppose that the first next non-zero, non-singular discrepancy is found at iteration $t = n > 2L_{n'} - 1$. Then the square is completed at iteration $t = j = 2n - 2L_{n'} + 1 > n$. Since $B_j(z) = z^{\gamma_t+1} \cdot \Lambda_{n-1}(z) \cdot \mathrm{adj}(\Delta_n)$, then the induction hypothesis proves that $B_j(z)$ is integral. Before the division in step FFMBM11, we know by induction that Λ is integral. Lemma 2 says that h_j is integral since it only differs from the determinant of the $L_{j+1} \times L_{j+1}$ block Hankel matrix H_j defined in Lemma 2 by a sign. Also h_j is nonzero since h_j and ρ_j are nonzero. Let $\hat{H}$ be the $L_j \times L_j + 1$ rectangular

block Hankel matrix defined by $\{M_k\}_{k=0}^{\infty}$ (see Figure 1). Lemma 1 implies that the $L_j + 1$ block coefficient vector λ^k [Kaltofen and Yuhasz 2013, Definition 7], defined by the kth column of Λ_j is a nullspace vector of $\hat{H}$. Since $\Lambda(0) = \pm\det(H_j) \cdot I_N$, then the last N rows of $\lambda^k = \pm\det(H_j) \cdot e^k$ where e^k is the kth standard basis vector. Thus the first $L_j \cdot N$ entries of λ^k are equal to $\pm\det(H_j) \cdot H_j^{-1} \cdot b^k$ where b^k is the kth column of the $L_j + 1$ block column of $\hat{H}$ (see Figure 1). Thus Cramer's rule implies that λ^k is integral for $1 \leq k \leq N$. So all of the coefficients of $\Lambda_j(z)$ are integral at the completion of stage j. Therefore the theorem holds at every stage t of Algorithm 2.1. ■

4 Block Hankel Systems With Arbitrary Right Sides

Figure 2 follows the execution of the Matrix Berlekamp/Massey algorithm as a block Hankel matrix solver with arbitrary right side Y. If we assume that the finite sequence of matrices defining the block Hankel matrix is a normalizable remainder sequence, then the Matrix Berlekamp/Massey algorithm can be extended to a linear solver with the addition of one block (solution) vector. The solution block column vector $\widetilde{\Lambda}$ is produced for each non-singular leading principal block Hankel coefficient matrix. When Algorithm 2.1 increases the generator degree in Step 5, then the next blocks $Y_{L_{n'}}, \ldots, Y_{L_n-1}$ can be solved for like the corresponding M blocks. In Figure 2, the columns containing Y are not actually a part of the block Hankel matrix but are there to illustrate when each block of Y is processed. Note the $\widetilde{\Lambda}_n$ is $\widetilde{\Lambda}_{L_{n'}-1}$ padded at the bottom by $L_n - L_{n'}$ zero blocks. A block discrepancy $\widetilde{\Delta}_t$ at Y_t, $L'_n \leq t \leq L_n - 1$, of the current solution $\widetilde{\Lambda}_{t-1}$ for $Y_0, \ldots, Y_{t-1}$ is removed by $\widetilde{\Lambda}_t \leftarrow \rho \cdot \widetilde{\Lambda}_{t-1} - \text{adj. coeff. col. vector}(B_{t-1}) \cdot \widetilde{\Delta}_t$. The fraction free division at Step 11 is the same, with a separate count for zero discrepancies.

5 Examples and Conclusions

5.1 Scalar Example

We give two simple scalar examples to illustrate how the improved pseudo-division implementation in our algorithm results in smaller intermediate values than that of the fraction free Berlekamp/Massey algorithm of Giesbrecht et al. [2002]. The first sequence shows how our algorithm builds the pseudo-division step-by-step and how that affects the coefficient size. Let the sequence $\{a_k\}_{k=0}^{\infty}$ be defined as:

$$0, 0, 0, 5, 5, 10, 15, 25, 40, 65, 105, 172, 275, 445, 720, 1165, \ldots,$$

a variation of the Fibonacci sequence. This Fibonacci sequence has an error at a_{11}. This error will be corrected if we make δ large enough. By inputting $\delta = 14$, Algorithm 2.1 will correct the error by ignoring the first 11 sequence elements. The following table gives the value of $\Lambda_t(0)$ for both algorithms.

Notice that the $|\Lambda_t(0)|$ of Algorithm 2.1 is smaller or equal to the corresponding value of Giesbrecht et al. [2002]. The iterations where the two algorithms have the same value are the iterations where the Step 11 is executed or iterations where a zero discrepancy occurs. In

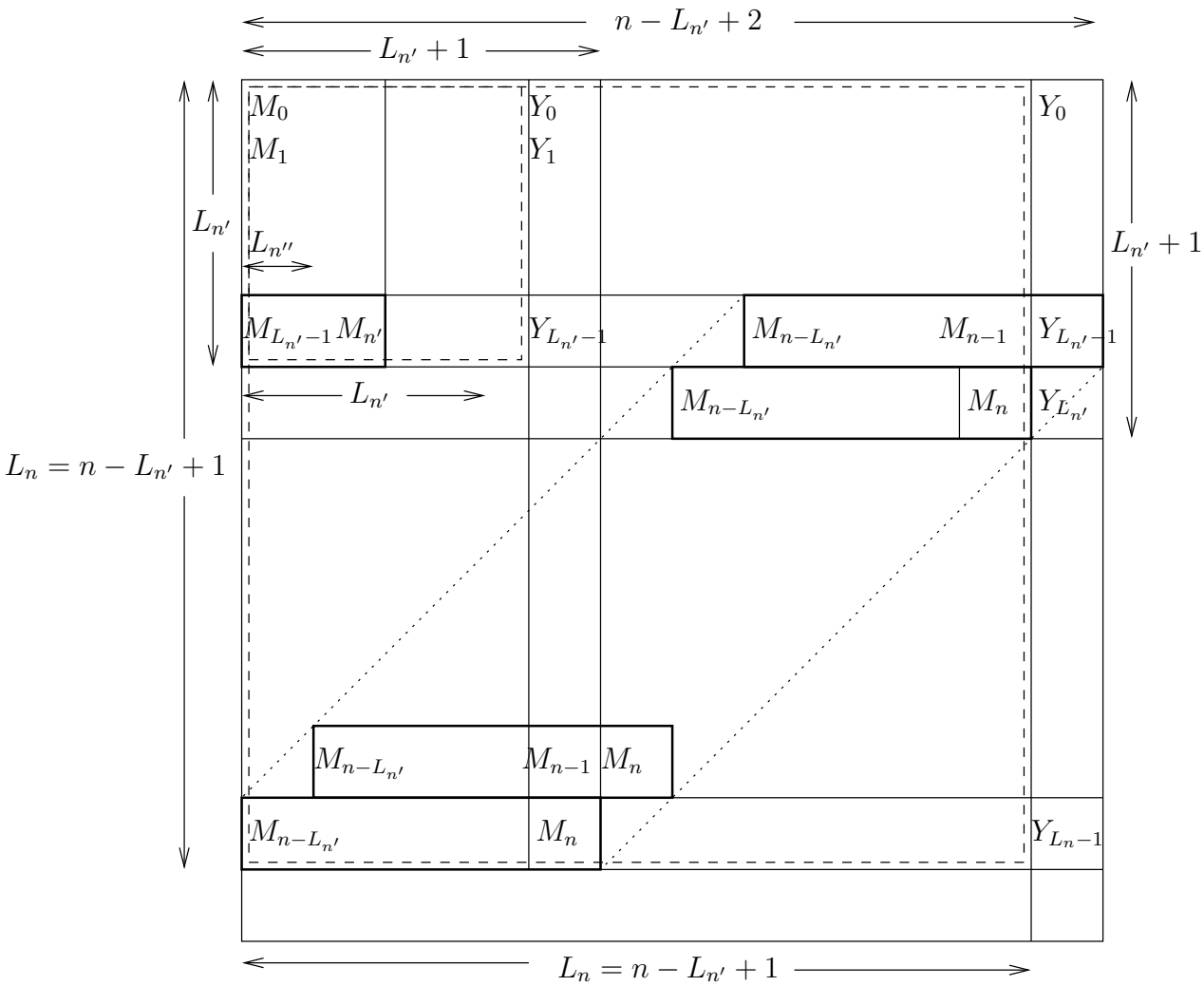

Figure 2: Block-Hankel Linear System Solver

these steps, the value of $|\Lambda_t(0)|$ is a known determinant and so the values must be equal. In every other step, when the pseudo-division of the algorithm in Giesbrecht et al. [2002] is being performed, our algorithm is smaller. This difference is also true for the other coefficients of Λ_t. The polynomial computed by our algorithm is $-102400z^{14} + 102400z^{13} + 102400z^{12}$. The algorithm of [Giesbrecht et al. 2002] computes the negative polynomial of our algorithm.

The second example shows how the improved pseudo-division of [Hearn 1972] allows us to skip over zero discrepancies during the pseudo-division in our algorithm. We define the sequence $\{a_k\}_{k=0}^{\infty}$ to be defined as:

$$0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 5, 0, 0, 0, 0, 10, 0, 0, 0, 0, 15, \ldots,$$

another variation of the Fibonacci sequence. This variation of the Fibonacci sequence is the standard sequence, multiplied by 5 with four zeroes inserted between each Fibonacci sequence element. These zeroes lead to zero discrepancies during the pseudo-division of both

Table 1: Example 1

Iteration	GKL	Our Algorithm	Iteration	GKL	Our Algorithm
1	1	1	15	≈–7.6e13	≈6.1e12
2	1	1	16	10000	10000
3	1	1	17	≈9.5e12	≈1.25e7
4	–625	1	18	760000	–760000
5	–625	5	19	≈2.3e18	≈5.7e11
6	–625	25	20	4040000	–4040000
7	–625	125	21	≈–4.2e20	≈1.6e13
8	625	625	22	–25881600	–25881600
9	625	625	23	≈1.1e23	≈6.6e14
10	625	625	24	166722816	166722816
11	625	625	25	≈–1.1e25	≈2.7e16
12	≈–7.6e13	3125	26	–430074880	430074880
13	≈–7.6e13	3906250	27	≈1.8e22	≈1.8e17
14	≈–7.6e13	≈4.8e9	28	102400	–102400

our algorithm and the algorithm of Giesbrecht et al. [2002]. The following table gives the value of $\Lambda_t(0)$ for both algorithms.

Table 2: Example 2

Iteration	GKL	Our Algorithm	Iteration	GKL	Our Algorithm
1	1	1	11	–15625	5
2	1	1	12	15625	15625
3	1	1	13	15625	15625
4	1	1	14	15625	15625
5	1	1	15	15625	15625
6	–15625	1	16	≈–2.9e25	78125
7	–15625	1	17	≈–2.9e25	78125
8	–15625	1	18	≈–2.9e25	78125
9	–15625	1	19	≈–2.9e25	78125
10	–15625	1	20	9765625	9765625

As the table shows, our algorithm builds the pseudo-division step by step and and skips over zero discrepancies during the pseudo-division steps. This makes the intermediate values much smaller. As in the previous case, the two algorithms have the same coefficients during the iterations where $\Lambda_t(0)$ is a known determinant. The polynomial computed by both algorithms for this sequence is $9765625z^{10} - 9765625z^5 - 9765625$.

5.2 Scalar Monic Minimal Generators

During the design of the scalar version of Algorithm 2.1, the following theorem was discovered. As the algorithm was tested on various sequences, a pattern emerged. For every integer sequence, the minimal generator F, computed by Algorithm 2.1 had a very intriguing content. Every coefficient of the generator was divisible by the leading term of F. Therefore, by dividing F by its leading term, we will call it h, then F/h is monic and so it must be the unique minimal generator of the sequence denoted by $f_{\min}$. Thus since h exactly divides every coefficient of F, then the unique minimal generator is integral. The two examples in the previous section both exhibit this behavior. So we have the following theorem, a generalization of a result by Pierre Fatou in [Fatou 1906, page 368].

Theorem 2 *If $\{a_k\}_{k=0}^{\infty}$ is a linearly generated sequence with $a_i \in D$ where D is a unique factorization domain, then the monic minimal generator $f_{\min} \in D[z]$.*

Proof. Let F be the field of quotients of D and let $a(z) \in D[[z]]$ be the power series $a(z) = \sum_{i\geq 0} a_i z^i$. Without loss of generality, we assume that the unique monic minimal generator of $\{a_k\}_{k=0}^{\infty}$, $f_{\min}$, is such that $f_{\min}(0) \neq 0$. Otherwise, $f_{\min}$ is divisible by z^i for some $i > 0$. This factor of z^i skips over the first i sequence elements in the linear generation. As such we can divide $f_{\min}$ by z^i and skip the necessary sequence elements to continue the proof.

Since $\{a_k\}_{k=0}^{\infty}$ is linearly generated, then there exists two polynomials $A(z), B(z) \in F[z]$ such that $\frac{A(z)}{B(z)} = a(z)$ and $A(z), B(z)$ are relatively prime. This is a known result of linearly generated (linearly recurrent) sequences and rational functions. Further, we know that $B(z)$ is a minimal recurrence of $\{a_k\}_{k=0}^{\infty}$ and so the reverse polynomial of $B(z)$ is a minimal generator of $\{a_k\}_{k=0}^{\infty}$. By clearing out denominators, we can assume that $A(z), B(z) \in D[z]$ and A, B remain relatively prime. So $A(z) = B(z)a(z)$. We now proceed to show that B is the unique minimal recurrence.

Suppose there exists $d \in D$, d not a unit, such that d is a common divisor of the coefficients of $B(z)$. Then since $A(z) = B(z)a(z)$, we see that d is a common divisor of all the coefficients of $A(z)$, contradicting the relative primeness of A and B. So no such d can exist.

Since A and B are relatively prime, there exists $U(z), V(z) \in D[z]$ such that $UA + VB = \alpha \in D \setminus \{0\}$. Thus the rational function $G = \frac{\alpha}{B(z)} = U(z)\frac{A(z)}{B(z)} + V(z) = U(z)a(z) + V(z) \in D[[z]]$. If $c \in D \setminus \{0\}$ is a common divisor of the coefficients of G, then $B(z)G(z) = \alpha$ implies that $c \mid \alpha$. Therefore we can clear out the common divisors of G and call it G^*. Let $\alpha^* = B(z)G^*(z)$.

Finally we show that $B(0) = 1$. Suppose there exists $p \in D$ a prime such that $p \mid B(0)$. Then $B(z)G^*(z) = \alpha^*$ implies that $B(0)G^*(0) = \alpha^*$ and so $p \mid \alpha^*$. Now consider $\overline{B}(z)\overline{G^*}(z) = 0 \in D/pD[[z]]$. Since D is a unique factorization domain, then D/pD is an integral domain and so $D/pD[[z]]$ is also an integral domain. So $\overline{B} = 0$ or $\overline{G^*} = 0$. Thus p is a common divisor of the coefficients of B or G^*. This is a contradiction from the previous paragraphs and so there is no $p \in D$ that divides $B(0)$ and so $B(0)$ is a unit of D. Therefore we can assume that $B(0) = 1$.

So $a(z) = \frac{A(z)}{B(z)}$ and $B(0) = 1$. Further B is a minimal recurrence of $\{a_k\}_{k=0}^{\infty}$ and $B \in D[z]$. So the reverse of B is $f_{\min}$ and $f_{\min} \in D[z]$. ■

Theorem 2 allows us to compute the unique minimal generator of a scalar sequence in a unique factorization domain using Algorithm 2.1. By performing a final division by h before returning the generator, we can compute $\pm f_{\min}$. This result does not extend to general matrix sequences. A counterexample is the following sequence:

$$M_i = \begin{bmatrix} 3 \cdot Fibo_i & 2 \cdot Fibo_i \\ 0 & m_i \end{bmatrix},$$

where $m_i = -2$ for $i \equiv 2 \mod 3$ and $m_i = 1$ otherwise. This sequence has a unique right minimal matrix generator given by:

$$f(z) = \begin{bmatrix} z^2 - z - 1 & -\frac{4}{3}z - \frac{4}{3} \\ 0 & z^2 + z + 1 \end{bmatrix}.$$

We know that f is the unique right minimal matrix generator because it is in (descending degree) column Popov form. Algorithm 2.1 confirms that this sequence is a normalizable remainder sequence.

5.3 Conclusion

We present a fraction free Matrix Berlekamp/Massey algorithm. The algorithm works for any scalar sequence and any normalizable remainder sequence. If the sequence is not a normalizable remainder sequence, then the algorithm may return a certificate that is is not. The scalar form of Algorithm 2.1 maintains smaller intermediate values than the previous known algorithm. Theorem 2 allows us to compute the unique minimal generators of sequences in unique factorization domains such as Z or $F[X]$ using our fraction free scalar algorithm. A fraction free variant of the general Matrix Berlekamp/Massey algorithm is still unknown.

References

Bareiss, E. H. Sylvester's identity and multistep integers preserving Gaussian elimination. *Math. Comp.*, 22:565–578, 1968.

Beckermann, Bernhard, Cabay, Stanley, and Labahn, George. Fraction-free computation of matrix pade systems. In *International Symposium on Symbolic and Algebraic Computation*, pages 125–132, 1997. URL citeseer.ist.psu.edu/beckermann97fractionfree.html.

Beckermann, Bernhard and Labahn, George. Fraction-free computation of matrix rational interpolants and matrix GCDs. *SIAM J. Matrix Anal. Applic.*, 22(1):114–144, 2000.

Berlekamp, E. R. *Algebraic Coding Theory.* McGraw-Hill Publ., New York, 1968.

Brown, W. S. On Euclid's algorithm and the computation of polynomial greatest common divisors. *J. ACM*, 18:478–504, 1971.

Brown, W. S. The subresultant PRS algorithm. *ACM Trans. Math. Software*, 4:237–249, 1978.

Brown, W. S. and Traub, J. F. On Euclid's algorithm and the theory of subresultants. *J. ACM*, 18:505–514, 1971.

Collins, George E. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14: 128–142, 1967.

Coppersmith, D. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. Comput.*, 62(205):333–350, 1994.

Corless, Robert M., Jeffrey, David J., and Zhou, Wenqin. Fraction-free forms of LU and QR matrix factors. In *Proc. Transgressive Computing*, pages 443–446, Granada, 2006.

Dickinson, Bradley W., Morf, Martin, and Kailath, Thomas. A minimal realization algorithm for matrix sequences. *IEEE Trans. Automatic Control*, AC-19(1):31–38, February 1974.

Dornstetter, J. L. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory*, IT-33(3):428–431, 1987.

Fatou, P. Séries trigonométriques et séries de Taylor. *Acta Mathematica*, 30(1):335–400, December 1906.

Giesbrecht, Mark, Kaltofen, Erich, and Lee, Wen-shin. Algorithms for computing the sparsest shifts for polynomials via the Berlekamp/Massey algorithm. In Mora, T., editor, *Proc. 2002 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'02)*, pages 101–108, New York, N. Y., 2002. ACM Press. ISBN 1-58113-484-3.

Hearn, Anthony C. An improved non-modular polynomial gcd algorithm. *SIGSAM Bull.*, (23):10–15, 1972. ISSN 0163-5824.

Kaltofen, Erich and Lee, Wen-shin. Early termination in sparse interpolation algorithms. *J. Symbolic Comput.*, 36(3–4):365–400, 2003. Special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: EKbib/03/KL03.pdf.

Kaltofen, Erich and Villard, Gilles. On the complexity of computing determinants. *Computational Complexity*, 13(3-4):91–130, 2004. URL: EKbib/04/KaVi04_2697263.pdf; Maple 7 worksheet URL: EKbib/04/KaVi04_2697263.mws.

Kaltofen, Erich and Yuhasz, George. On the matrix Berlekamp-Massey algorithm. *ACM Trans. Algorithms*, 9(4), September 2013. URL: EKbib/06/KaYu06.pdf.

Massey, J. L. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory*, IT-15: 122–127, 1969.

Nakos, George C., Turner, Peter R., and Williams, Robert M. Fraction-free algorithms for linear and polynomial equations. *SIGSAM Bull.*, 31(3):11–19, 1997. ISSN 0163-5824.

Zhou, Wenqin and Jeffrey, David J. Fraction-free matrix factors: new forms for LU and QR factors. *Frontiers of Computer Science in China*, 2(1):67–80, 2008. ISSN 1673-7350.