On the Matrix Berlekamp-Massey Algorithm

ERICH KALTOFEN and GEORGE YUHASZ, North Carolina State University

We analyze the Matrix Berlekamp/Massey algorithm, which generalizes the Berlekamp/Massey algorithm [Massey 1969] for computing linear generators of scalar sequences. The Matrix Berlekamp/Massey algorithm computes a minimal matrix generator of a linearly generated matrix sequence and has been first introduced by Rissanen [1972a], Dickinson et al. [1974], and Coppersmith [1994]. Our version of the algorithm makes no restrictions on the rank and dimensions of the matrix sequence. We also give new proofs of correctness and complexity for the algorithm, which is based on self-contained loop invariants and includes an explicit termination criterion for a given determinantal degree bound of the minimal matrix generator.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices; computations on polynomials*; I.1.2 [**Symbolic and Algebraic Manipulations**]: Algorithms—*Analysis of algorithms*

General Terms: Algorithms

Additional Key Words and Phrases: Linear generated sequences; matrix polynomials, minimal generators; vector Berlekamp/Massey algorithm; multivariable linear control

ACM Reference Format:
Kaltofen, E. and Yuhasz, G. 2013, On the matrix Berlekamp-Massey algorithm. ACM Trans. Algor. 9, 4, Article 33 (September 2013), 24 pages.
DOI: http://dx.doi.org/10.1145/2500122

1. INTRODUCTION

The applications of the generalization of the Berlekamp/Massey algorithm for computing linear generators of scalar sequences [Berlekamp 1968; Massey 1969] to matrix sequences are manifold. The algorithm is originally introduced in multivariable control theory [Rissanen 1972b; Dickinson et al. 1974]. A recent application is to sparse linear system solving over finite fields by the Block Wiedemann algorithm [Coppersmith 1994]. Table I gives a summary of references to algorithms that compute minimal linear generators of scalar and matrix sequences. We loosely categorize the methods into approaches based on the original Berlekamp/Massey design, the extended Euclidean algorithm, the computation of a Padé approximant, a σ-basis, and the solution of the underlying linear system in Toeplitz/Hankel form. We mark those methods that reduce the time complexity from quadratic in the dimension of the linear system to essentially linear (up to poly-logarithmic factors) by use of fast polynomial arithmetic by a "*."

Clearly, all methods are related by virtue that the solve the same linear algebra problem. However, for the scalar case it was not until Dornstetter [1987] that the

This research was supported in part by the National Science Foundation of the USA under Grants CCR-0305314 and CCF-0514585.
Authors' addresses: E. Kaltofen, Department of Mathematics, North Caolina State University; G. Yuhasz, Department of Mathematics, Morehouse College, 830 Westview Dr., Atlanta, GA 30314; email: yuhasz@gmail.com.

DOI: http://dx.doi.org/10.1145/2500122

Table I. Algorithms for Computing Minimal Linear Generators

Style of algorithm	*Scalar case*	*Matrix case*
Berlekamp/Massey	[Berlekamp 1968] [Massey 1969] [Dornstetter 1987] [Giesbrecht et al. 2002], §2	[Rissanen 1972b] [Dickinson et al. 1974] [Coppersmith 1994] [Thomé 2002]* This article
Extended Euclidean	[Sugiyama et al. 1975] [Dornstetter 1987]	[Kaltofen and Villard 2004], §3.1*
Padé approximant	[Brent et al. 1980]*	[Van Barel and Bultheel 1991] [Van Barel and Bultheel 1992] [Giorgi et al. 2003]*
σ-basis	[Kaltofen and Lee 2003], §2.2	[Beckermann and Labahn 1994]* [Villard 1997a]*, [Turner 2002]
Toeplitz/Hankel solver	[Levinson 1947] [Durbin 1959]	[Cabay et al. 1990] [Kaltofen 1995], Appendix A*

*of essentially linear complexity in the degree.

Euclidean algorithm-based approach by Sugiyama et al. [1975] was reduced to the actual Berlekamp/Massey algorithm via a now classical unraveling of the doubly nested polynomial remainder chain-polynomial division loops to the single Berlekamp/Massey loop. We give a fraction-free version of the original Berlekamp/Massey algorithm in Giesbrecht et al. [2002] and in Kaltofen and Lee [2003] we give a re-interpretation as a specialized σ-basis algorithm. The first quadratic algorithm for computing linear generators in the generic case is Levinson's 21 years before Berlekamp and Massey. An essentially linear time randomized algorithm for the block Levinson-Durbin problem based on the theory of displacement rank and that is valid for all inputs and over any field is in Kaltofen [1995]. The theory of σ-bases [Beckermann and Labahn 1994] in a different manner converts the matrix-Padé problem to a scalar linear algebra problem and, as has been observed by Villard [1997a] and worked out in Turner [2002], can be employed for the computation of minimal linear generators of matrix sequences. Van Barel and Bultheel [1991, 1992] give algorithms to compute a matrix Padé approximant along a diagonal path [Van Barel and Bultheel 1991] and along any path [Van Barel and Bultheel 1992] in the Padé table. Giorgi et al. [2003] give a polynomial matrix multiplication based algorithm to compute a matrix Padé approximant.

Here we revisit the Matrix Berlekamp/Massey algorithm as described in Dickinson et al. [1974] and Coppersmith [1994]. We only consider the quadratic-time version; see Thomé [2002] for a variant of essentially linear complexity. Our objective is to give a stand-alone correctness proof based on explicit loop invariants for the algorithm which makes no restrictions: arbitrary rectangular sequences of matrices whose rank can gradually increase. Our main invariants on certain nullspace properties of the intermediate matrix polynomials (see Theorem 4.14) are reminiscent of those for σ-bases [Beckermann and Labahn 1994] and M-bases [Giorgi et al. 2003]. The arguments in Giorgi et al. [2003, Lemma 3.7], based on irreducible descriptions in Kailath [1980], are applied to Padé approximants, whereas only the approximant on the diagonal yields the minimal generator. A main difficulty is to prove that the minimal generators for sequences that are truncated at any order eventually converge to the untruncated minimal generator, for which we offer an apparently new justification (see Lemma 4.16 and its proof). We also give an explicit early termination criterion that adapts to the relation of the bound on the determinantal degree of the generator, which has to be

input, and the intermediately computed generator vector degrees (see Theorem 4.19). We assume the determinantal degree bound is correct, and when possible the algorithm may diagnose an insufficient degree bound. However, since verifying the correctness of the degree bound requires any algorithm to process infinitely many sequence elements, no algorithm can certify that the degree bound is large enough. An analysis of how an insufficient degree bound affects the Matrix Berlekamp/Massey algorithm is given. Using the termination criterion, we give a worst case complexity analysis that is applicable to any input. We also include a comparison of our algorithm and a method based on the fast power hermite padé solver of Beckermann and Labahn [1994]. Theorem 2.8 in Section 2 gives an absolute description of any linearly generated matrix sequence that seems to have never been shown before.

The remainder of this article will proceed in this order. Section 2 gives definitions and properties of linearly generated matrix sequences and their generators. Section 3 describes a version of the Matrix Berlekamp/Massey algorithm that is restricted to square matrix sequences. Section 4 gives properties and proofs of the intermediate computations of the algorithm in Section 3 culminating in a correctness proof for the algorithm. Section 5 describes the adjustments that need to be made to the algorithm in Section 3 for rectangular matrix sequences and gives justification for these changes. Performance issues such as complexity analysis and comparison with other methods are contained in Section 6.

2. LINEARLY GENERATED MATRIX SEQUENCES

Linearly generated matrix sequences are an extension and generalization of linearly generated scalar sequences. We will begin with a definition of a linearly generated matrix sequence and a scalar generator. We will then extend the notion of a generator to include vector and matrix generators. Using results from module theory and linear control theory we define minimal matrix generators. Finally, we will show the relationship between minimal matrix generators and scalar generators.

The following defines both a linearly generated matrix sequence and the scalar polynomial generator of such a sequence. Both definitions are simple generalizations of results from the theory of linearly generated scalar sequences.

Definition 2.1. A sequence of matrices $\{M_k\}_{k=0}^{\infty}$ with $M_k \in K^{N_{\text{row}} \times N_{\text{col}}}$ is linearly generated if there exists a polynomial $F(z) = \sum_{i=0}^{n} c_i z^i \in K[z]$ with $F \neq 0$ where the following holds:

$$(\forall l, l \geq 0)\colon \sum_{i=0}^{n} c_i M_{i+l} = 0^{N_{\text{row}} \times N_{\text{col}}}.$$

The polynomial F is called a scalar generator of the sequence $\{M_k\}_{k=0}^{\infty}$. The unique minimal scalar generator of the sequence $\{M_k\}_{k=0}^{\infty}$ is the monic generator of minimal degree.

This definition can be generalized by changing the coefficients from scalars into vectors in $K^{N_{\text{col}}}$. Making such a change leads to the following definition.

Definition 2.2. $F(z) = \sum_{k=0}^{n} c_k z^k \neq 0 \in K^{N_{\text{col}}}[z]$ is a right vector generator of a matrix sequence $\{M_k\}_{k=0}^{\infty}$ if the following holds:

$$(\forall l, l \geq 0)\colon \sum_{k=0}^{n} M_{k+l} c_k = 0^{N_{\text{row}}}.$$

The sequence is said to be linearly generated from the right by the vector polynomial.

An analogous definition can be made for left vector generators. We will only consider right generators and so will drop the term "right" and all generators will be right generators unless specified.

The introduction of vector generators allows us to define a matrix generator and a minimal matrix generator. We first describe the $K[z]$ module structure of the set of vector generators of a linearly generated matrix sequence.

FACT 1. *Assume that* $\{M_k\}_{k=0}^{\infty}$ *is linearly generated, and let* $W = \{F \in K^{N_{col}}[z] \mid F \text{ is a vector generator of } M\} \cup \{0\}$.

(1) W *is a* $K[z]$ *submodule of* $K^{N_{col}}[z]$.
(2) W *has rank* N_{col}.
(3) W *has a basis.*

PROOF. Let $g \in K[z]$, $g \neq 0$ be the linear generator for $\{M_k\}_{k=0}^{\infty}$. That W is a submodule is straight forward. To prove that W has rank N_{col}, consider the following vector polynomials. For all $1 \leq i \leq N_{\text{col}}$ let $g_i = g \cdot e_i$, where e_i is the standard coordinate vector. Each g_i is a vector generator of the sequence and the N_{col} vectors are linearly independent. Thus, the submodule W has rank at least N_{col}, but since W is a submodule of $F^{N_{\text{col}}}[z]$, it can have rank no larger than N_{col} [Dummit and Foote 1991, p. 371]. Therefore, W has rank N_{col}.

The fact that W has a basis is a classic result of module theory [Dummit and Foote 1991, p. 371]. Any submodule of a free module over a principal ideal domain is also a free module and so any submodule has a basis. The result proves that any submodule of a free module over a P.I.D. has a basis with certain properties. The basis is defined by vectors $w_1, w_2, \ldots, w_{N_{\text{col}}} \in K^{N_{\text{col}}}[z]$ and elements $g_1, g_2, \ldots, g_{N_{\text{col}}} \in K[z]$ where the following properties hold. The vectors $g_1 \cdot w_1, g_2 \cdot w_2, \ldots, g_{N_{\text{col}}} \cdot w_{N_{\text{col}}}$ form a basis of W; the vectors $w_1, w_2, \ldots, w_{N_{\text{col}}}$ form a basis of $K^{N_{\text{col}}}[z]$; and $g_{N_{\text{col}}} \mid g_{N_{\text{col}}-1} \mid \ldots \mid g_2 \mid g_1$. This basis is related to the Smith Normal form of the submodule W. □

We give an example of a matrix sequence that is not linearly generated and how such sequences violate Fact 1. Let $M_k = [0 \ \binom{k}{\lceil k/2 \rceil}]$, were $(M_k)_{1,2}$ is the center element of the k^{th} row of Pascal's triangle. This scalar sequence is is not linearly generated [Kaltofen 1992] and so the matrix sequence is not linearly generated. The matrix sequence has a nonzero vector generator, given by $F = [1 \quad 0]^T$. So the submodule W defined in Fact 1 is non trivial with rank at least 1. As in this proof, W is a $K[z]$ submodule of $K^2[z]$ and W has a basis. Since the center Pascal numbers are not linearly generated, then any vector generator of the sequence must have 0 in the second row, and so the rank of W is equal to 1. Thus, linearly generated matrix sequences ensure that the annihilator submodule W is full rank. We will now use the facts that W is full rank and has a basis to define right matrix generators and right minimal matrix generators.

Definition 2.3. $F = \sum_{k=0}^{n} C_k z^k \neq 0 \in K^{N_{\text{col}} \times N_{\text{col}}}[z]$, $\det(F) \neq 0$, is a right matrix generator of a matrix sequence $\{M_k\}_{k=0}^{\infty}$ if the following holds:

$$(\forall l, l \geq 0)\colon \sum_{k=0}^{n} M_{k+l} C_k = 0^{N_{\text{row}} \times N_{\text{col}}}.$$

The sequence is said to be linearly generated from the right by the matrix polynomial.

Definition 2.4. The matrix polynomial $F \in K^{N_{\text{col}} \times N_{\text{col}}}[z]$ is said to be a minimal right matrix generator of $\{M_k\}_{k=0}^{\infty}$ if F is a right matrix generator and the columns of F form a basis of the $K[z]$ submodule W defined here.

This definition describes a minimal matrix generator. Similar to integral bases of algebraic number rings, the determinant of a minimal matrix generator has minimal degree.

THEOREM 2.5. *$F \in K^{N_{col} \times N_{col}}[z]$ is a minimal matrix generator of $\{M_k\}_{k=0}^{\infty}$ if and only if deg(det(F)) is minimal over all right matrix generators of $\{M_k\}_{k=0}^{\infty}$.*

Like the scalar generator case, there are many different minimal matrix generators. In the scalar case, the monic generator of minimal degree is defined to be the unique minimal generator. For the matrix generator case, we use the following definition.

Definition 2.6. The unique minimal right matrix generating polynomial of the sequence $\{M_k\}_{k=0}^{\infty}$ is the matrix polynomial $F \in K^{N_{\text{col}} \times N_{\text{col}}}[z]$ such that F is a minimal matrix generator and F is in Popov form [Popov 1970].

The next theorem shows the connection between our definition of linearly generated matrix sequences and matrix generators.

THEOREM 2.7. *If $F \in K^{N_{col} \times N_{col}}[z]$ is a minimal matrix generator of $\{M_k\}_{k=0}^{\infty}$, then $\{M_k\}_{k=0}^{\infty}$ is linearly generated. Further, the first invariant factor of F is the unique minimal scalar generator of $\{M_k\}_{k=0}^{\infty}$.*

PROOF. Let f_1 be the first or largest invariant factor of F, such that the Smith Normal form of F is given by the matrix diag$(f_1, \ldots, f_N)$, with f_i dividing f_{i-1} for all $N \geq i > 1$. There exists a matrix polynomial $\bar{F} \in K^{N \times N}[z]$ such that $F^{-1} = (1/f_1) \cdot \bar{F}$. Further, we know that for any $H \in K^{N \times N}[z]$, $F \cdot H$ is a right matrix generator of the sequence $\{M_k\}_{k=0}^{\infty}$. Therefore, the sequence has a matrix generator $F \cdot \bar{F} = f_1 \cdot F \cdot F^{-1} = f_1 \cdot I_N$. This means that f_1 must be a scalar generator of $\{M_k\}_{k=0}^{\infty}$.

To see that f_1 is a unique minimal generator, use the additional fact that f_1 is the least common denominator of the entries of F^{-1}.

Now let g be the unique minimal scalar generator of $\{M_k\}_{k=0}^{\infty}$. Since f_1 is a scalar generator of the sequence, then $g \mid f_1$. Further, the matrix polynomial $g \cdot I_N$ is a matrix generator of $\{M_k\}_{k=0}^{\infty}$. Because F is a minimal matrix generator, there exists a G such that $F \cdot G = I_N \cdot g$. So then $F^{-1} = 1/g \cdot G$ and so from the previous paragraph, $f_1 \mid g$. But f_1 is monic and so $f_1 = g$. □

An important motivation for studying linearly generated matrix sequences is the fact that the sequence $\{M_k\}_{k=0}^{\infty}$ is linearly generated if the sequence is defined by $M_k = U^T \cdot B^k \cdot V$, where U, B, and V are matrices over K. Such matrix sequences are block bilinear projections of matrix powers. These projections arise in the analysis of the block Wiedemann and block Lanczos algorithms. The results in Kaltofen [1995], Villard [1997b], and Kaltofen and Villard [2004] detail properties of the matrix sequences generated by random bilinear projections and their minimal matrix generators. We will now demonstrate that all linearly generated matrix sequences can be viewed as a bilinear projection of a matrix power sequence.

THEOREM 2.8. *The sequence $\{M_k\}_{k=0}^{\infty}$ is linearly generated if and only if there exists matrices U, B and V with entries in K such that $M_k = U^T \cdot B^k \cdot V$.*

PROOF. Proving one direction is simple. If $M_k = U^T \cdot B^k \cdot V$ and f is the characteristic polynomial of B, then f is a scalar generator of $\{M_k\}_{k=0}^{\infty}$. Thus, $\{M_k\}_{k=0}^{\infty}$ is linearly generated.

To prove the other direction, we begin by proving the 1×1 or scalar case. Suppose $\{M_k\}_{k=0}^{\infty}$ is a linearly generated 1×1 matrix (scalar) sequence and let f be a monic

scalar generator of degree m. Let B_f be the companion matrix of f:

$$B_f = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ -f_0 & -f_1 & \dots & -f_{m-2} & -f_{m-1} \end{bmatrix}.$$

Let $B = B_f$ and define $U, V \in K^{m\times 1}$ as:

$$U = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} = e_1 \quad \text{and} \quad V = \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{m-1} \end{bmatrix}.$$

Since f is assumed to be a monic generator the following is true:

$$(\forall l, l \geq 0)\colon M_{m+l} = \sum_{i=0}^{m-1} -f_i \cdot M_{i+l}.$$

Therefore, we know the following is true about the action of B^l on the vector V:

$$(\forall l, l \geq 0)\colon B^l \cdot V = \begin{bmatrix} M_l \\ M_{l+1} \\ \vdots \\ M_{m+l-1} \end{bmatrix}.$$

Since the action of U^T is to select the top row of the vector $B^l \cdot V$, then $M_k = U^T \cdot B^k \cdot V$.

Let $\{M_k\}_{k=0}^{\infty}$ be a linearly generated matrix sequence of $N_{\text{row}} \times N_{\text{col}}$ matrices and let f be the unique minimal scalar generator of the sequence of degree m. Let B_f be the companion matrix of f as defined previously. We will generalize the scalar construction above to define the matrices U, B and V. Since $\{M_k\}_{k=0}^{\infty}$ is linearly generated by f, we can fix an i and j and see that the scalar sequence defined by $\{(M_k)_{i,j}\}_{k=0}^{\infty}$ is a linearly generated scalar sequence and f is a generator. As previously mentioned, let us define the vector $V_{i,j}$ to be the following:

$$V_{i,j} = \begin{bmatrix} (M_0)_{i,j} \\ (M_1)_{i,j} \\ \vdots \\ (M_{m-2})_{i,j} \\ (M_{m-1})_{i,j} \end{bmatrix}.$$

We can now define the matrices U, B and V as block matrices. B will be a $(N_{\text{row}} \cdot m) \times (N_{\text{row}} \cdot m)$ block diagonal matrix of the form:

$$B = \text{diag}(B_f, \dots, B_f).$$

U is a sparse $(N_{\text{row}} \cdot m) \times N_{\text{row}}$ matrix given by:

$$U = \text{diag}(e_1, \dots, e_1).$$

V is an $(N_{\text{row}} \cdot m) \times N_{\text{col}}$ matrix defined in the following block form:

$$V = \begin{bmatrix} V_{1,1} & V_{1,2} & \dots & V_{1,N_{\text{col}}} \\ V_{2,1} & V_{2,2} & \dots & V_{2,N_{\text{col}}} \\ \vdots & \vdots & \vdots & \vdots \\ V_{N_{\text{row}},1} & V_{N_{\text{row}},2} & \dots & V_{N_{\text{row}},N_{\text{col}}} \end{bmatrix}.$$

Due to the block nature of the matrices U, B and V, and the fact that the scalar sequences embedded in the i, j entries of the matrices in $\{M_k\}_{k=0}^{\infty}$ are linearly generated by f, then $M_k = U^T \cdot B^k \cdot V$. □

3. MATRIX BERLEKAMP/MASSEY ALGORITHM

In this section, we present a version of the Matrix Berlekamp/Massey algorithm that will compute a minimal (right) matrix generating polynomial of a matrix sequence. Other versions of this algorithm can be found in Lobo [1995], and Coppersmith [1994]. The algorithm given here removes a condition found in the versions previously. Both Coppersmith [1994] and Lobo [1995] require that the initial matrix in the sequence be full rank, but by changing the initialization and updating procedure, we have removed this requirement.

In the previous section, the matrix sequences were generically rectangular. There were no assumptions made on the relationship between the row dimension N_{row} and the column dimension N_{col}. All of the properties of minimal matrix generators are valid over all linearly generated matrix sequences of all possible dimensions. The row and column dimension of the matrix sequence will define the dimension and number of variables needed during the Matrix Berlekamp/Massey algorithm. Further, the invariants and proof of the Matrix Berlekamp/Massey algorithm are dependent on the dimensions of the matrix sequence. For the purpose of simplification, we present a description and proof of the Matrix Berlekamp/Massey algorithm for square matrix sequences, ie $N_{\text{row}} = N_{\text{col}}$. In a later section, a description and proof of the rectangular algorithm will be given. Because we are assuming $N_{\text{row}} = N_{\text{col}}$, we will drop the subscripts and just let N be the row and column dimension of the matrix sequence.

In the algorithm description, we will refer to the following matrix rank, which is computed implicitly (see Lemma 4.5 on page 11).

Definition 3.1. By r_t we denote the rank of $[M_0\ M_1 \cdots M_t]$, and we set $r_{-1} = 0$.

If M_0 is singular, r_t can increase beyond $t = 0$. The algorithm then performs an "on-the-fly" initialization (see step GE24 on page 9).

3.1. Algorithm *Matrix Berlekamp/Massey*

Input. $M(z) \in K^{N\times N}[[z]]$ with $M_i \in K^{N\times N}$ the coefficient of z^i
$\delta \geq 0$ an upper bound on the determinantal degree of the matrix generator.
Output. $F(z) \in K^{N\times N}[z]$ a minimal matrix generator.
In the case that the input δ is not an upper bound for the determinantal degree, the algorithm may return "insufficient bound" or an incorrect F.
Variables. $f(z) \in K^{N\times 2N}[z]$ where the first N columns $f_1, \dots, f_N$ represent the reversal of the current generator and the last N columns $f_{N+1}, \dots, f_{2N}$ are what we call the *auxiliary polynomials*.
$d_1, d_2, \dots, d_{2N}$. the nominal degree of each column. We have throughout the algorithm that $\deg(f_j) \leq d_j$. Since f_j are reversed polynomials, d_j are upper bounds for the degrees of their reversals $z^{d_j} f_j(z^{-1})$.

$\tau \in K^{2N\times 2N}$. a linear transformation
$\Delta \in K^{N\times 2N}$. the discrepancy matrix
β. the minimum of the nominal degrees of the auxiliary polynomials
σ. the sum of the nominal degrees of the generator polynomials
μ. the maximum of the nominal degrees of the generator polynomials

MBM1 $f \leftarrow [I_N\ 0^{N\times N}]$
$d_1 \leftarrow d_2 \leftarrow \cdots \leftarrow d_N \leftarrow 0$; $d_{N+1} \leftarrow d_{N+2} \leftarrow \cdots \leftarrow d_{2N} \leftarrow 1$
$t \leftarrow -1$
$\beta \leftarrow 1$
$\sigma \leftarrow 0$
$\mu \leftarrow 0$.
MBM2 while $\beta < \delta - \sigma + \mu + 1$ do steps MBM3 through MBM9
MBM3 $t \leftarrow t + 1$
MBM4 $\Delta \leftarrow Coeff(t; M(z) \cdot f(z))$
Note that $\Delta = [C\ L^{[t-1]}]$ where C, $L^{[t-1]} \in K^{N\times N}$ and $L^{[t-1]}$ is lower triangular with rank r_{t-1} computed in the previous step MBM5 below. By definition, we set $L^{[-1]} = 0^{N\times N}$.
MBM5 Call Algorithm 3.2 below to compute τ. As a side-effect, $d_1, \ldots, d_{2N}$ are updated.
Let $\Delta \cdot \tau = [Z\ L^{[t]}]$. We have that $Z \in K^{N\times N}$ has $r_t - r_{t-1}$ nonzero columns and if j is the index of a nonzero column of Z, then $d_j = t + 1$. Furthermore, $L^{[t]} \in K^{N\times N}$ is lower triangular of rank r_t.
MBM6 $d_{N+i} = d_{N+i} + 1$ for all $i = 1, 2, \ldots N$
MBM7 $\beta \leftarrow \min_{N+1\le i\le 2N} d_i$; $\mu \leftarrow \max_{1\le i\le N} d_i$; $\sigma \leftarrow \sum_{i=1}^{N} d_i$
These updates are most efficiently performed during Algorithm 3.2.
MBM8 if $\sigma \ge \delta + 1$, then return "insufficient bound"
MBM9 $f(z) \leftarrow f(z) \cdot \tau \cdot \mathrm{diag}(I_N, z \cdot I_N)$
MBM10 $F \leftarrow [z^{d_1} f_1(z^{-1})\ z^{d_2} f_2(z^{-1}) \ldots z^{d_N} f_N(z^{-1})]$
MBM11 return F

3.2. Auxiliary Gaussian Elimination Algorithm

Input. $\Delta \in K^{N\times 2N}$ where Δ has rank R and the last N columns of Δ have rank r.
$d_1, d_2, \ldots, d_{2N}$ nominal degrees of the columns.
Output. $\tau \in K^{2N\times 2N}$ a transformation such that $\Delta\tau = [Z\ L]$. Z will be a matrix with $R - r$ nonzero columns. L is a lower triangular matrix of rank R.
$d_1, d_2, \ldots, d_{2N}$ a reordering of the nominal degrees. Note that no d_j for $1 \le j \le N$ is decreased.

GE1 $\tau \leftarrow I_{2N}$
GE2 $\Gamma \leftarrow \{1, 2, \ldots, N\}$
GE3 for $i = 1$ to N do steps GE4 through GE24
In Dickinson et al. [1974, p. 36] four types of eliminations and updates are distinguished. We will annotate which steps of our Gaussian elimination algorithm correspond to each case. Except for Case 1 of Dickinson et al. [1974], the updates defined in each case can be performed as i ranges from 1 to N. For a fixed i however, only one type of update will be performed. Case 1 in Dickinson et al. [1974] requires the same type of update for each i.
GE4 $\Pi_i \leftarrow \{j \in \Gamma \text{ and } \Delta_{i,j} \ne 0\} \cup \{N + i\}$
GE5 $l \leftarrow l \in \Pi_i \mid d_l = \min_{i\in\Pi_i} d_i$
There may be different choices for the column index l of the pivot element $\Delta_{i,l}$. Our current implementation employs the following rules. Assume that d_r and

d_s are equal and minimal with $r, s \in \Pi_i$ and $r < s$. If $1 \le r < s \le N$, then $l = r$. If $1 \le r \le N$ and $N + 1 \le s \le 2N$, then $l = s$.

GE6 $\Pi_i \leftarrow \Pi_i \setminus \{l\}$

GE7 for all $j \in \Pi_i$ do steps GE8 through GE24

Case 1 of Dickinson et al. [1974] occurs if Π_i is empty for all $1 \le i \le N$. Then $\tau = I_{2N}$ and so none of the generating vectors are updated. This situation only occurs when C defined in step MBM4 is $0^{N \times N}$.

GE8 if $l = N + i$ do steps GE9 and GE10

These steps perform a minor change of the generating vectors, as defined in Case 3 of Dickinson et al. [1974]. The nominal degrees of all the generating vectors are unchanged.

GE9 Perform the column operation $\Delta_j \leftarrow \Delta_j - \frac{\Delta_{i,j}}{\Delta_{i,N+i}} \Delta_{N+i}$

Now $\Delta_{i,j} = 0$.

GE10 Perform the column operation $\tau_j \leftarrow \tau_j - \frac{\Delta_{i,j}}{\Delta_{i,N+i}} \tau_{N+i}$

GE11 if $l < N + i$ do steps GE12 through GE24

At this stage, the pivot column is a previous generator column, and so σ will be increased by step GE20 or step GE23. If $\sigma - d_l + d_{N+i} > \delta$, then any further calculations are unnecessary since step MBM8 will return "insufficient bound".

GE12 if $j < N + i$, do steps GE13 and GE14

GE13 Perform the column operation $\Delta_j \leftarrow \Delta_j - \frac{\Delta_{i,j}}{\Delta_{i,l}} \Delta_l$

Now $\Delta_{i,j} = 0$.

GE14 Perform the column operation $\tau_j \leftarrow \tau_j - \frac{\Delta_{i,j}}{\Delta_{i,l}} \tau_l$

GE15 if $j = N + i$, do steps GE16 through GE24

GE16 if $\Delta_{i,N+i} \neq 0$, do steps GE17 through GE20

The update performed here is defined as a major change in Case 4 of Dickinson et al. [1974]. The nominal degree of one vector generator will be increased and an auxiliary vector will be replaced by a former generating vector.

GE17 Perform the column operation $\Delta_{N+i} \leftarrow \frac{-\Delta_{i,l}}{\Delta_{i,N+i}} \Delta_{N+i} + \Delta_l$

GE18 Perform the column operation $\tau_{N+i} \leftarrow \frac{-\Delta_{i,l}}{\Delta_{i,N+i}} \tau_{N+i} + \tau_l$

GE19 Switch column l with column $N + i$ in Δ and τ

GE20 Switch d_l and d_{N+i}.

Now $\Delta_{i,l} = 0$ and d_{N+i} is minimized.

GE21 if $\Delta_{i,N+i} = 0$, do steps GE22 through GE24

GE22 Perform the column operation $\tau_{N+i} \leftarrow \tau_{N+i} + \tau_l$

GE23 Switch d_l and d_{N+i}.

GE24 $\Gamma \leftarrow \Gamma \setminus \{l\}$

Note that steps GE22 through GE24 initialize a new column in the auxiliary polynomials. If $\Delta_{i,N+i} = 0$, then a new component as defined by Case 2 of Dickinson et al. [1974] has been found and a new auxiliary vector must be added. The corresponding discrepancy is skipped over via $d_l = t + 1$, necessitated by an increase of r_t. If M_0 is nonsingular [Coppersmith 1994], the initialization only occurs for $t = 0$.

GE25 Return τ and $d_1, d_2, \ldots, d_{2N}$.

4. PROPERTIES AND CORRECTNESS OF THE MATRIX BERLEKAMP/MASSEY ALGORITHM

We now establish several properties of the computed quantities that remain invariant during the iterations. In order to distinguish the values of variables at different

iterations, we will use the superscript $[t]$ for the value of a variable for a given loop index t at step MBM2.

LEMMA 4.1. *For each iteration at step* MBM2 $\deg(f_j^{[t]}) \le d_j^{[t]}$.

PROOF. Assuming $\deg(0^N) \le 0$, then this is true by construction on initialization.

We proceed by induction. Suppose $t \ge -1$ and $\deg(f_j^{[t]}) \le d_j^{[t]}$ for all $1 \le j \le 2N$. Steps MBM5 and MBM6 update the nominal degrees while step MBM9 updates f. These steps can be performed within the iteration steps GE3 and GE7. Note that each auxiliary column f_{N+i} is updated only once after which multiplication by z and incrementing d_{N+i} can be performed. So we can analyze each column operation performed by Algorithm 3.2 to prove the property is maintained.

During each iteration of GE7, l is the index of the pivot column. Algorithm 3.2 performs two types of column operations, adding a scalar multiple of the pivot column into another column and switching column l with column $N+i$. The latter operation preserves the property, since the column switch in step GE19, is followed by a nominal degree switch in step GE20. Since l is minimal in nominal degree, then steps GE10, GE14, GE18, and GE22 preserve the property in the following manner: $\deg(f_j^{[new]}) \le \max\{\deg(f_j^{[old]}), \deg(f_l)\} \le \max\{d_j, d_l\} = d_j$. Step GE23 minimizes the nominal degree of the auxiliary polynomial but it preserves the property since $\deg(f_l) \le d_l \le d_{N+i}$. Finally, when column $N+i$ is multiplied by z, its degree increases by 1, but its nominal degree is also incremented by 1 and so the property is maintained. So for $1 \le j \le 2N$, $\deg(f_j^{[t+1]}) \le d_j^{[t+1]}$. □

LEMMA 4.2. *For each iteration at step* MBM2*, one has* $\sum_{i=1}^{2N} d_j = N \cdot (t+2)$.

PROOF. For $t = -1$, the property is true by the initialization in step MBM1. Because the last N columns of $f(z)$ have their nominal degrees incremented by 1 in step MBM6, the summation holds by induction for all $t \ge 0$. □

LEMMA 4.3. *For each iteration at step* MBM2*, one has*

$$\det([f_1(0) \quad f_2(0) \quad \cdots \quad f_N(0)]) \neq 0.$$

PROOF. This is true by construction on initialization. At step MBM9, the elementary column operations encoded in τ are applied to the constant coefficient, thus leaving it nonsingular. □

LEMMA 4.4. *For each iteration at step* MBM2 *the following equivalence is true for each matrix L produced by Algorithm* 3.2 *in step* MBM5*:*

$$(\forall i, 1 \le i \le N)\colon L_{i,i} \neq 0 \iff L_i \neq 0^N \iff f_{N+i} \neq 0^N,$$

where L_i is the ith column of L.

PROOF. At every stage, $L_{i,i} \neq 0 \Longrightarrow L_i \neq 0^N$ is apparent. Also, since $L_{N+i} = \mathit{Coeff}(t+1; M(z) \cdot f_{N+i}(z))$, $L_i \neq 0^N \Longrightarrow f_{N+i} \neq 0^N$. We will show by induction that at every stage $t \ge -1$, $f_{N+i} \neq 0^N \Longrightarrow L_{i,i} \neq 0$.

If $t = -1$, then for all $1 \le i \le N$, $f_{N+i}^{[-1]} = 0^N$. By definition $L^{[-1]} = 0^{N \times N}$ and so the induction basis is immediate.

Suppose $t \ge -1$ and $f_{N+i}^{[t]} \neq 0^N \Longrightarrow L_{i,i}^{[t]} \neq 0$ for all i. Let $1 \le i \le N$ be such that $f_{N+i}^{[t+1]} \neq 0^N$. If $f_{N+i}^{[t+1]} = z \cdot f_{N+i}^{[t]}$ then $L_{i,i}^{[t+1]} = L_{i,i}^{[t]} \neq 0$ by the induction hypothesis. So now we consider the possibility that $f_{N+i}^{[t+1]} \neq z \cdot f_{N+i}^{[t]}$.

If $f_{N+i}^{[t+1]} \neq z \cdot f_{N+i}^{[t]}$, then we know that $f_{N+i}^{[t+1]} = z \cdot \sum_{j=1}^{N+i-1}(\alpha_j f_j^{[t]})$. Since auxiliary columns are never interchanged in Algorithm 3.2, then there exists $1 \leq j \leq N$ such that $\alpha_j \neq 0$. Because column $N+i$ of $f^{[t]}$ was replaced then at stage i of Algorithm 3.2, there was an $1 \leq l \leq N$ such that $\Delta_{i,l}^{[t+1]} \neq 0$ and column l had minimal nominal degree. Noted that $\Delta^{[t+1]}$ here is not the original matrix passed to Algorithm 3.2, but the eliminated form being computed by the algorithm. So for all $1 \leq m < i$, we have $\Delta_{m,l}^{[t+1]} = 0$. Since column l of $\Delta^{[t+1]}$ has a discrepancy in row i and has minimal nominal degree, then either step GE19 or step GE22 will place the contents of column l into column $N+i$ of $\Delta^{[t+1]} \cdot \tau^{[t+1]}$. At the completion of Algorithm 3.2, column i of $L^{[t+1]}$ is equal to column $N+i$ of $\Delta^{[t+1]} \cdot \tau^{[t+1]}$. Thus, $L_{i,i}^{[t+1]} \neq 0$ since $\Delta_{i,l}^{[t+1]} \neq 0$, completing the induction argument. □

LEMMA 4.5. *For each iteration at step* MBM2 *the nonzero columns of L form a basis for the column space of* $[M_0 \quad M_1 \quad \cdots \quad M_t]$.

PROOF. Suppose $t = -1$. By definition, $L^{[-1]} = 0^{N \times N}$, meaning there are no nonzero columns. Since the matrix M_{-1} is empty, its column space is empty. Therefore the invariant holds.

Suppose $t \geq -1$ and suppose the lemma holds for t. Step MBM4 defines $\Delta^{[t+1]} = \left[C^{[t+1]} \quad L^{[t]}\right]$. The induction hypothesis allow us to write $C^{[t+1]}$ as

$$\begin{aligned} C^{[t+1]} &= M_{t+1}[f_1(0) \ldots f_N(0)] + \sum_{i=1}^{t+1} M_{t+1-i} Coeff(i; [f_1(z) \ldots f_N(z)]) \\ &= M_{t+1}[f_1(0) \ldots f_N(0)] + L^{[t]} \Xi \quad \text{for some } \Xi \in K^{N \times N}. \end{aligned}$$

By hypothesis and Lemma 4.3 we deduce that the columns of $\Delta^{[t+1]}$ span the columnspace of $[M_0 \cdots M_{t+1}]$. Algorithm 3.2 performs elementary column operations so the columns of $[Z \quad L^{[t+1]}]$ span the columnspace of $[M_0 \cdots M_{t+1}]$ as well. We need to argue that the nonzero columns in Z are not needed. In Step GE21, any nonzero column of Z replaces by Lemma 4.4 a zero column of L and is then removed from further updates in Step GE24. Therefore, the remaining columns in Z and the columns of L still span $[M_0 \cdots M_{t+1}]$. However, at the conclusion of Algorithm 3.2 all remaining columns in Z are zero. Since $L^{[t+1]}$ is triangular, its nonzero columns are linearly independent. □

LEMMA 4.6. *For each iteration at step* MBM2*, we have* r_t *(see Definition* 3.1 *on page* 7*)* $= |\{j \mid N+1 \leq j \leq 2N \text{ and } f_j \neq 0^N\}|$ *(the number of nonzero auxiliary polynomials).*

PROOF. For all $N+1 \leq j \leq 2N$, Lemma 4.4 implies that $f_j^{[t]} \neq 0^N$ if and only if $L_{j-N}^{[t]} \neq 0^N$. Lemma 4.5 implies that $L^{[t]}$ has exactly r_t nonzero columns. So there are exactly r_t indices j such that $N+1 \leq j \leq 2N$ and $f_j^{[t]} \neq 0^N$. □

LEMMA 4.7. *For each iteration at step* MBM2 *for all l such that* $N+1 \leq l \leq 2N$, $f_l = 0^N$ *if and only if* $d_l = t+2$.

PROOF. If $t = -1$, then $d_l^{[-1]} = 1 = t+2$ and $f_l^{[-1]} = 0^N$.

Let $t \geq -1$ and suppose the invariant holds for t.

The update of f performed in step MBM9 dependent on the computation of $\tau^{[t+1]}$ by Algorithm 3.2 in step MBM5 forces two possibilities for the update of $f_l^{[t+1]}$. Either $f_l^{[t+1]} = z \cdot f_l^{[t]}$ or $f_l^{[t+1]} = z \cdot \sum_i (\alpha_i \cdot f_i^{[t]})$ where $i \leq l$, $f_i^{[t]} \neq 0^N$ and $\alpha_i \neq 0$ for all i. The proof of Lemma 4.4 illustrates that Algorithm 3.2 adjusts the column $L_{l-N}^{[t]}$ if and only

if $L_{l-N}^{[t+1]} \neq 0^N$. So $L_{l-N}^{[t+1]} = L_{l-N}^{[t]} = 0^N$ implies that $f_l^{[t+1]} = z \cdot f_l^{[t]}$. $L_{l-N}^{[t+1]} \neq 0^N$ implies that $f_l^{[t+1]} = z \cdot \sum_i (\alpha_i \cdot f_i^{[t]})$. Further, Lemma 4.4 implies $L_{l-N}^{[t+1]} = 0^N$ if and only if $f_l^{[t+1]} = 0^N$.

So $f_l^{[t+1]} = 0^N$ if and only if $L_{l-N}^{[t+1]} = 0^N$ if and only if $L_{l-N}^{[t]} = 0^N$. By Lemma 4.4 and the previous paragraph, $f_l^{[t+1]} = 0^N$ implies that $f_l^{[t]} = 0^N$ and $f_l^{[t+1]} = z \cdot f_l^{[t]}$. Thus, $d_l^{[t+1]} = d_l^{[t]} + 1 = t + 2 + 1 = t + 3$ by the induction hypothesis.

Conversely, $f_l^{[t+1]} \neq 0^N$ if and only $L_{l-N}^{[t+1]} \neq 0^N$. So $f_l^{[t+1]} = z \cdot \sum_i (\alpha_i \cdot f_i^{[t]})$ and the induction hypothesis implies that $d_i^{[t]} < t + 2$ for all i. Thus, the $d_l^{[t+1]} \neq t + 3$. □

LEMMA 4.8. *For each iteration at step* MBM2 *we have*

$$\sum_{1 \leq j \leq 2N \text{ and } f_j \neq 0^N} d_j = r_t \cdot (t + 2).$$

PROOF. The invariant holds at $t = -1$ by initialization. By definition $r_{-1} = 0$ and the N nonzero columns of $f^{[-1]}$ are the first N columns and each column has nominal degree 0.

By Lemma 4.5, there are $N - r_t$ zero columns, each of which by Lemma 4.7 has nominal degree $t + 2$. Therefore, by using Lemma 4.2, we have the following equation:

$$\begin{aligned} \sum_{1 \leq j \leq 2N \text{ and } f_j \neq 0^N} d_j &= \left(\sum_{j=1}^{2N} d_j \right) - (N - r_t) \cdot (t + 2) \\ &= N \cdot (t + 2) - (N - r_t) \cdot (t + 2) = r_t \cdot (t + 2). \quad \square \end{aligned}$$

LEMMA 4.9. *For each iteration at step* MBM2*, one has*

$$(\forall j, 1 \leq j \leq 2N)(\forall l, d_j \leq l \leq t)\text{: } \mathit{Coeff}(l; M(z) \cdot f_j(z)) = 0^N. \tag{1}$$

PROOF. This invariant is one of the main conditions in Coppersmith [1994, p. 338].

The invariant is true by default at initialization, that is, $t = -1$ since every column j has $d_j^{[-1]} > -1$ and so the range is empty.

Let $t \geq -1$ and assume the invariant holds at t. Let j be such that $1 \leq j \leq 2N$. We consider two cases, $j \leq N$ and $j > N$.

If $j > N$, then $f_j^{[t+1]} = z \cdot \sum_i \alpha_i f_i^{[t]}$ and $d_j^{[t+1]} = d_m^{[t]} + 1$ where m is the column of maximal nominal degree in the linear combination. The induction hypothesis implies that for all i in the linear combination and all l such that $d_m^{[t]} \leq l \leq t$ we have $\mathit{Coeff}(l; M(z) \cdot f_i^{[t]}(z)) = 0^N$. Therefore, for all l such that $d_j^{[t+1]} \leq l \leq t + 1$, we have $\mathit{Coeff}(l; M(z) \cdot f_j^{[t+1]}(z)) = \mathit{Coeff}(l - 1; M(z) \cdot \sum_i \alpha_i f_i^{[t]}(z)) = 0^N$.

If $j \leq N$, then $f_j^{[t+1]} = \sum_i \alpha_i \cdot f_i^{[t]}$. If $d_j^{[t+1]} = t + 2$, then the condition holds trivially as in the base case. If $d_j^{[t+1]} \leq t + 1$, then $d_j^{[t+1]} = d_m^{[t]}$, where m is the column in the linear combination with maximal nominal degree. The induction hypothesis implies that for all i in the linear combination and all l such that $d_m^{[t]} \leq l \leq t$ then $\mathit{Coeff}(l; M(z) \cdot f_i^{[t]}(z)) = 0^N$. Therefore, $\mathit{Coeff}(l; M(z) \cdot f_j^{[t+1]}(z)) = 0^N$ for all l such that $d_j^{[t+1]} \leq l \leq t$. Further, since $d_j^{[t+1]} < t + 2$, then column j of $Z^{[t+1]}$ is 0^N. Thus, $\mathit{Coeff}(t + 1; M \cdot f_j^{[t+1]}(z)) = Z_j^{[t+1]} = 0^N$. So for all l such that $d_j^{[t+1]} \leq l \leq t + 1$, we have $\mathit{Coeff}(l; M \cdot f_j^{[t+1]}(z)) = 0^N$.

Thus, (1) holds at stage $t + 1$ and by induction the invariant is true for every t. □

We will make use of the following vectors.

Definition 4.10. If $v \in K[z]^N = \sum_{i=0}^{m} v_i z^i$, $v_i \in K^N$, then let $CoeffVec(d; v)$ where $d \geq m$ be a block vector in $K^{N(d+1)}$ of the following form:

$$CoeffVec(d; v) = \begin{bmatrix} 0^N \\ \vdots \\ 0^N \\ v_m \\ v_{m-1} \\ \vdots \\ v_0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} d-m \text{ blocks} \\ \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} m+1 \text{ blocks} \end{matrix}.$$

These vectors are generalizations of coefficient vectors. One can think of them as embedding the coefficient vector of a vector polynomial into a larger coefficient vector. Further, the block entries of the coefficient vector are vectors in K^N.

LEMMA 4.11. *For each iteration at step* MBM2 *for* $d = \min\{\max_{1 \leq j \leq 2N}\{d_j\}, t+1\}$*, the set of vectors in* $K^{N(d+1)}$ *defined by*

$$U_t = \{CoeffVec(d; z^i f_j) \mid 1 \leq j \leq 2N \text{ and } f_j \neq 0^N, 0 \leq i \leq d - d_j\},$$

is linearly independent.

PROOF. We proceed by induction on t.

So for $t = -1$ at initialization, $d = 0$ and the vectors in U_{-1} are the columns of the matrix I_N.

Let $t \geq -1$ and suppose the invariant holds at stage t. After the $t+1$ stage of the algorithm, we can construct a matrix τ that is derived from $\tau^{[t]}$ and is a product of the corresponding elementary column operations on the shifted coefficient vectors of the generator and the auxiliary polynomials. Let $\tilde{U}_{t+1} = U_t \tau$. Here, U_t is considered to be a matrix with the appropriate columns. By hypothesis, the columns of $\tilde{U}_{t+1}$ are linearly independent.

If d is unchanged at the $t+1$ stage, then $U_{t+1} \subset \tilde{U}_{t+1}$. So U_{t+1} is linearly independent.

If d changes, then it can only increase by one. Further, since N polynomials did not increase in nominal degree, we must have $|U_{t+1}| = |U_t| + N$. We can reconstruct U_{t+1} as the following matrix:

$$\begin{bmatrix} 0^{N \times N} & \tilde{U}_{t+1} \\ F & 0^{N \times |U_t|} \end{bmatrix},$$

where

$$F \in K^{Nd \times N} = \begin{bmatrix} CoeffVec(d-1; f_1^{[t+1]}) & \cdots & CoeffVec(d-1; f_N^{[t+1]}) \end{bmatrix}.$$

We leave out the proof that this is in fact U_{t+1}.

Now consider the following linear system:

$$\begin{bmatrix} 0^{N \times N} & \tilde{U}_{t+1} \\ F & 0^{N \times |U_t|} \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_N \\ c_{N+1} \\ \vdots \\ c_{N+|U_t|} \end{bmatrix} = 0^{N(d+1)}.$$

By Lemma 4.3, we know that $c_1 = c_2 = \cdots = c_N = 0$ and the induction hypothesis tells us that $c_{N+1} = \cdots = c_{N+|U_t|} = 0$. Further, since U_{t+1} is equal to the columns of the matrix above, U_{t+1} is linearly independent. □

COROLLARY 4.12. *Let $d' \le d$ as defined in Lemma 4.11. The set of vectors CoeffVec$(d'; z^i f_j(z))$, where $1 \le j \le 2N$ and $f_j \neq 0^N$, $0 \le i \le d' - d_j$, is linearly independent.*

PROOF. The vectors are formed from vectors in U_t by removing top segments that are zero. □

LEMMA 4.13. *For each iteration at step* MBM2 *for all j, d, i with $1 \le j \le 2N$, $d_j \le d \le t$ and $0 \le i \le d - d_j$, CoeffVec$(d; z^i f_j(z))$ is in the right nullspace of the following block Hankel matrix:*

$$H = \begin{bmatrix} M_0 & M_1 & \cdots & M_d \\ M_1 & M_2 & \cdots & M_{d+1} \\ \vdots & \vdots & \vdots & \vdots \\ M_{t-d} & \cdots & \cdots & M_t \end{bmatrix}.$$

PROOF. This proof relies on Lemma 4.9. First, let $c_0, c_1, \ldots, c_{d_j}$ be the coefficients of f_j. Then, the following holds:

$$H \cdot \mathit{CoeffVec}(d; z^i f_j) = H \cdot \begin{bmatrix} 0^N \\ \vdots \\ 0^N \end{bmatrix} \Big\} d - d_j - i \quad \begin{bmatrix} c_{d_j} \\ c_{d_j-1} \\ \vdots \\ c_0 \end{bmatrix} \Big\} d_j + 1 \quad \begin{bmatrix} 0^N \\ \vdots \\ 0^N \end{bmatrix} \Big\} i$$

$$= \begin{bmatrix} \sum_{l=0}^{d_j} M_{d-i-l} c_l \\ \sum_{l=0}^{d_j} M_{d+1-i-l} c_l \\ \vdots \\ \sum_{l=0}^{d_j} M_{t-i-l} c_l \end{bmatrix}$$

$$= \begin{bmatrix} \mathit{Coeff}((d-i; M(z)\, f_j(z)) \\ \mathit{Coeff}(d+1-i; M(z)\, f_j(z)) \\ \vdots \\ \mathit{Coeff}(t-i; M(z)\, f_j(z)) \end{bmatrix}$$

Since $d_j \le d-i < d+1-i < \cdots < t-i < t+1$, Lemma 4.9 implies that every coefficient in this vector is 0^N. □

THEOREM 4.14. *For each iteration at step* MBM2*, let $d = \min\{\max\{\mu, \beta - 1\}, t\}$. Then, the right nullspace of the block Hankel matrix given by*

$$H_t = \begin{bmatrix} M_0 & M_1 & \cdots & M_d \\ M_1 & M_2 & \cdots & M_{d+1} \\ \vdots & \vdots & \vdots & \vdots \\ M_{t-d} & \cdots & \cdots & M_t \end{bmatrix} \tag{2}$$

has a basis defined by the set

$$V_t = \{CoeffVec(d; z^i f_j(z)) \mid 1 \le j \le 2N \text{ and } d_j \le d, 0 \le i \le d - d_j\}. \tag{3}$$

Further, for all l with $N+1 \le l \le 2N$ and $d < d_l \le t$, any wider and shallower block Hankel matrix given by

$$H_{t,d_l} = \begin{bmatrix} M_0 & M_1 & \cdots & M_{d_l} \\ M_1 & M_2 & \cdots & M_{d_l+1} \\ \vdots & \vdots & \vdots & \vdots \\ M_{t-d_l} & \cdots & \cdots & M_t \end{bmatrix} \tag{4}$$

has a right nullspace basis

$$V_{t,d_l} = \{CoeffVec(d_l; z^i f_j(z)) \mid 1 \le j \le 2N \text{ and } d_j \le d_l, 0 \le i \le d_l - d_j\}. \tag{5}$$

Proof. By Corollary 4.12 on page 14 the vectors in V_t and V_{t,d_l} are linearly independent, and by Lemma 4.13 on page 14 they are in the nullspace of H_t and H_{t,d_l}, respectively. We proceed by induction to show that the vectors indeed span the corresponding nullspaces.

If $t = -1$, then the invariant is trivially true. Obviously, since we may set $H_{-1} = [0]$ whose nullspace basis is the empty set. But $d = t = -1$ and $d_j^{[-1]} > -1$ for all j, so $V_{-1} = \emptyset$.

The case $t = 0$ can be proven as a base case as well. Such a proof would be a specialized version of the proof for any $t \ge 0$ where $d = t$. Thus, we will use $t = -1$ as our base case for induction.

Suppose $t \ge -1$ and the theorem holds for t. To prove the theorem holds at $t+1$, we must break the update of the algorithm into the various cases which can take place. Let $\mu^{[t+1]}$ and $\beta^{[t+1]}$ as computed by Algorithm 3.1 and d as stated in the theorem. The first separation of cases is a condition on d. The case $d = t+1$ can be viewed as an extension of the base case. If $d < t+1$, we will then break the proof down into two cases, each based on $\mu^{[t+1]}$ and $\beta^{[t+1]}$. These cases are: $\mu^{[t+1]} + 1 < \beta^{[t+1]}$, $\mu^{[t+1]} + 1 \ge \beta^{[t+1]}$. The latter has two subcases, $\mu^{[t]} = \mu^{[t+1]}$ and $\mu^{[t]} \ne \mu^{[t+1]}$. The four cases are exhaustive.

Case 1. $d = t+1$. Then $H_{t+1} = \begin{bmatrix} M_0 & M_1 & \cdots & M_{t+1} \end{bmatrix}$. By Lemma 4.6, H_{t+1} has rank r_{t+1}. We show that $|V_{t+1}| = N(t+2) - r_{t+1}$, which is the full dimension of the nullspace of H_{t+1} and proves that case. Obviously, the cardinality of V_{t+1} depends on counting the non-negative integers in the range from 0 to $t + 1 - d_j^{[t+1]}$ for each nonzero column $f_j^{[t+1]}$. This is true even for any j such that $d_j^{[t+1]} > t+1$. If $d_j^{[t+1]} > t+1$, then $d_j^{[t+1]} = t+2$ and the number of non-negative integers in the range of this theorem is $t + 1 - d_j^{[t+1]} + 1 = t + 2 - (t+2) = 0$, and so can be included in a full summation without causing problems. By Lemma 4.3 on page 10 and Lemma 4.6 on page 11, there are $N + r_{t+1}$ nonzero columns of $f^{[t+1]}$. These facts and Lemma 4.8 on page 12 give us the following:

$$\begin{aligned} |V_{t+1}| &= \sum_{1 \le j \le 2N \text{ and } f_j^{[t+1]} \ne 0^N} \left(t + 1 - d_j^{[t+1]} + 1\right) \\ &= (N + r_{t+1})(t+2) - \sum_{1 \le j \le 2N \text{ and } f_j^{[t+1]} \ne 0^N} d_j^{[t+1]} \\ &= (N + r_{t+1})(t+2) - r_{t+1}(t+3) = N(t+2) - r_{t+1}. \end{aligned}$$

Case 2. $d < t+1$ and $\mu^{[t+1]}+1 < \beta^{[t+1]}$. Then, $d = \beta^{[t+1]}-1$. The condition $\mu^{[t+1]}+1 < \beta^{[t+1]}$ implies that $\Delta^{[t+1]} = [0^{N\times N} \quad L]$, for otherwise Algorithm 3.2 would have made a corresponding update on the nominal degrees. Therefore, $\mu^{[t+1]} = \mu^{[t]}$ and the first N columns of $f(z)$ are unchanged during the update. Furthermore, $\beta^{[t+1]} = \beta^{[t]}+1$ and d is incremented by 1, so H_{t+1} is formed from H_t by adding on right block column. Now, for all j with $1 \le j \le N$ the range $0 \le i \le d - d_j^{[t+1]}$ has increased in cardinality by 1, so there are N more vectors in V_{t+1}. Since that is the maximal increase for the nullspace, V_{t+1} forms a basis.

The same proof applies to all $H_{t+1,\eta}$, where $d < \eta = d_j^{[t+1]} = d_j^{[t]}+1 < t+2$ for some $N+1 \le j \le 2N$.

Case 3. $d < t+1$, $\mu^{[t+1]}+1 \ge \beta^{[t+1]}$ and $\mu^{[t+1]} = \mu^{[t]}$. Then, $d = \mu^{[t]}$ and has not changed, so H_{t+1} is formed from H_t by adding at the bottom one block row. Let $w \in K^{N(d+1)}$ be an element of the right nullspace of H_{t+1}. Consider w to be a degree d coefficient vector with a corresponding zero constant coefficient. By Lemma 4.3 on page 10, we may assume that the corresponding constant coefficient of w is zero.

Then w is also in the right nullspace of H_t. From the induction hypothesis, w can be written as a linear combination of vectors from V_t. Therefore let

$$w = \sum \alpha_\rho v_\rho, \tag{6}$$

where for all ρ, $v_\rho \in V_t$. Thus $v_\rho = \mathit{CoeffVec}(d; z^i f_j^{[t]}(z))$, for some j with $1 \le j \le 2N$ and some i in the proper range. We first show that for all ρ we have $i \ge 1$. The constant coefficients of the auxiliary polynomials are always zero, so by Lemma 4.3 the generating polynomials must be shifted by at least z so the corresponding constant coefficient of w can be zero. For j with $N+1 \le j \le 2N$ and $f_j^{[t]} \ne 0$ we know that $\mathit{Coeff}(t+1; M(z)\cdot f_j^{[t]}(z)) = L_j^{[t]}$, which are nonzero and linearly independent (see Lemma 4.4). Because the shifted vectors corresponding to the columns of $f^{[t]}$ by hypothesis zero the new block row in H_{t+1}, if any $\mathit{CoeffVec}(d; f_j^{[t]})$ were in the linear combination, then w could not be a nullspace element of H_{t+1}. Note that the argument shows that at least one vector of V_t cannot be in the nullspace of H_{t+1}, hence the rank of H_{t+1} is at least one more than the rank of H_t.

We can carry out the steps MBM9 and MBM6 in Algorithm 3.1 within the iteration steps GE3 and GE7 in Algorithm 3.2. Note that each column f_{N+i} is updated only once, after which multiplication by z can be performed. We shall show that at each iteration at step GE7 the vector w in (6) remains in the span of V in (3). We shall use the superscripts [*old*] and [*new*] for the contents of variables before and after each iteration in step GE7. From the updates $f_j^{[new]} = f_j^{[old]} + \gamma f_l$ in steps GE10 and GE14, we obtain

$$\mathit{CoeffVec}\big(d; z^i f_j^{[old]}\big) = \mathit{CoeffVec}\big(d; z^i f_j^{[new]}\big) - \gamma\,\mathit{CoeffVec}\big(d; z^i f_l\big). \tag{7}$$

Since $d_l \le d_j^{[old]} = d_j^{[new]}$, for all i in the range $1 \le i \le d - d_j^{[old]}$ the two vectors on the right-side of (7) are in the new set V_{new} of (3). Hence, all vectors in V_{old} of (3) are still spanned by V_{new} and w remains in the span of V_{new}.

From the updates

$$f_l^{[new]} = \tfrac{1}{\gamma} f_{N+i}^{[old]} + f_l^{[old]},\ d_l^{[new]} = d_{N+i}^{[old]}, \quad f_{N+i}^{[new]} = z f_l^{[old]},\ d_{N+i}^{[new]} = d_l^{[old]} + 1$$

in steps GE18–GE20 and subsequent multiplication with z, we obtain $\mathit{CoeffVec}(d; z^i f_l^{[old]}) = \mathit{CoeffVec}(d; z^{i-1} f_{N+i}^{[new]})$ and

$$\mathit{CoeffVec}\big(d; z^i f_{N+i}^{[old]}\big) = \gamma\,\mathit{CoeffVec}\big(d; z^i f_l^{[new]}\big) - \gamma\,\mathit{CoeffVec}\big(d; z^{i-1} f_{N+i}^{[new]}\big). \tag{8}$$

Again since $d_l^{[old]} \le d_{N+i}^{[old]}$, the required old range for i is included in the ranges of i and $i-1$ of the new right-side vectors in (8).

Finally, steps GE22–GE24 cannot occur because $\mu^{[t+1]} = d < t+1$. Therefore, at the conclusion of all updates for f, the vector w remains the linear span of V_{t+1}, as was to be shown.

We now show that for all η such that $\mu^{[t+1]} = d < \eta = d_j^{[t+1]} < t+2$ where $N+1 \le j \le 2N$, the set defined in (5) on page 15 is a nullspace basis of (4). We know that $\eta = d_j^{[t+1]} = d_\xi^{[t]} + 1$ for some $1 \le \xi \le 2N$. Further, since $d_\xi^{[t]} + 1 > \mu^{[t+1]} = \mu^{[t]}$, then $d_\xi^{[t]} \ge \mu^{[t]}$. Therefore, $j = \xi$ and $f_j^{[t+1]} = z \cdot f_j^{[t]}$, for otherwise we would have $d = \mu^{[t+1]} > \mu^{[t]}$. So $\eta = d_j^{[t+1]} = d_j^{[t]} + 1 > \mu^{[t+1]}$. Since $\eta - 1 \ge \mu^{[t]}$, then the induction hypothesis implies that the set $V_{t,\eta-1}$ is a nullspace basis of the matrix $H_{t,\eta-1}$. Note that $H_{t,\eta-1} = H_t$ and $V_{t,\eta-1} = V_t$ is possible. $H_{t+1,\eta}$ has one more block column than $H_{t,\eta-1}$ and so the nullspace of $H_{t+1,\eta}$ has dimension at most $|V_{t,\eta-1}| + N$. It is sufficient then to show that $|V_{t+1,\eta}| = |V_{t,\eta-1}| + N$.

The definition of $V_{t,\eta-1}$ in (5) implies the following:

$$|V_{t,\eta-1}| = \sum_{\{\zeta \,|\, d_\zeta^{[t]} \le \eta-1\}} \left(\eta - 1 - d_\zeta^{[t]} + 1\right).$$

For all $1 \le k \le N$, $d_k^{[t+1]} = d_{\varpi(k)}^{[t]} \le \mu^{[t+1]} \le \eta - 1$ where ϖ is an bijection from $\{1, 2, \ldots, 2N\}$ into $\{1, 2, \ldots, 2N\}$. So for each k, the range between η and $d_k^{[t+1]}$ is one more than the range between $\eta - 1$ and $d_{\varpi(k)}^{[t]} = d_k^{[t+1]}$. Also for any auxiliary column s with $d_s^{[t+1]} \le \eta$, then the range between $d_{\varpi(s)}^{[t]}$ and $\eta - 1$ is equal to the range between $d_s^{[t+1]} = d_{\varpi(s)}^{[t]} + 1$ and η. Thereforem, we have the following equation:

$$|V_{t+1,\eta}| = \sum_{\{\zeta \,|\, d_\zeta^{[t+1]} \le \eta\}} \left(\eta - d_\zeta^{[t+1]} + 1\right) = N + \sum_{\{\zeta \,|\, d_{\varpi(\zeta)}^{[t]} \le \eta-1\}} \left(\eta - 1 - d_{\varpi(\zeta)}^{[t]} + 1\right) = |V_{t,\eta-1}| + N.$$

Thus, $V_{t+1,\eta}$ exhausts the nullspace of $H_{t+1,\eta}$.

Case 4. $d < t+1$. $\mu^{[t+1]} + 1 \ge \beta^{[t+1]}$ and $\mu^{[t+1]} > \mu^{[t]}$. Then, $d = \mu^{[t+1]} = d_l t$ for some l with $N+1 \le l \le 2N$. Let $\eta = d_l^{[t]}$. We know that $\eta > \max_{1 \le j \le N}\{d_j^{[t]}\}$. By the induction hypothesis, we have a right nullspace basis $V_{t,\eta}$ of the block Hankel matrix $H_{t,\eta}$, which is H_{t+1} without the last block row. So we may use the same argument as in Case 3, replacing H_t and V_t, with $H_{t,\eta}$ and $V_{t,\eta}$.

Finally, for all $\lambda = d_j^{[t+1]} > d = \eta$, the set $V_{t+1,\lambda}$ exhausts the nullspace of $H_{t+1,\lambda}$. Again, we use the same argument in Case 3 by replacing $\mu^{[t+1]} = \mu^{[t]}$ with η and η with λ. □

LEMMA 4.15. *For each iteration at step* MBM2 *we have the following. Consider* $\bar{M}_0 = M_0, \ldots, \bar{M}_t = M_t$, $\bar{F}(z) = z^\mu [f_1(z^{-1}) \ldots f_N(z^{-1})] \in K^{N \times N}[z]$ *and*

$$\bar{M}_{t+\theta} = -\left(\sum_{i=0}^{\mu-1} \bar{M}_{t+\theta-\mu+i} \mathit{Coeff}(i; \bar{F})\right) [f_1(0) \cdots f_N(0)]^{-1} \quad \text{for } \theta = 1, 2, \ldots.$$

Then $F(z) = [z^{d_1} f_1(z^{-1})\ z^{d_2} f_2(z^{-1}) \cdots z^{d_N} f_N(z^{-1})]$ *is the minimal matrix generator for* $\bar{M}_0, \bar{M}_1, \ldots$

PROOF. Since $[f_1(0) \cdots f_N(0)] = \mathit{Coeff}(\mu; \bar{F})$ the polynomial $\bar{F}$ is a matrix generator in the sense of Definition 2.3. In particular, $C^{[t+1]} = C^{[t+2]} = \cdots = 0^{N \times N}$ in step MBM4. Now let n be the degree of the minimal generator, denoted by G. If $\mu + 1 > \beta$ or $d < n$, then

continue the algorithm until $d \geq n$ and $\beta \geq \mu + 1$. Because the generator polynomials do not change any further on the sequence $\{\bar{M}_i\}$, $\min\{\beta - 1, t\}$ will eventually catch up with both. Let G_k be any column of G, let $e = \deg(G_k)$ and let $g(z) = z^e G_k(z^{-1})$. Because g is a generator vector of $\{\bar{M}_i\}$, $CoeffVec(d; g(z))$ is in the nullspace of H_t on page 14. Since $d \geq n$ and $\mu \leq \beta - 1$, by Theorem 4.14, we have

$$CoeffVec(d; z^{d-e} g(z)) = \sum_{j=1}^{N} \sum_{i=0}^{d-d_j} \alpha_{j,i} CoeffVec(d; z^i f_j(z)) \quad \text{for some } \alpha_{i,j} \in K.$$

Hence, $G_k(z) = \sum_j \sum_i \alpha_{j,i} z^{d-d_j-i} F_j(z)$, where F_j is the jth column of F. Hence, every column of G is in the $K[z]$ submodule of F and F is a minimal generator. □

LEMMA 4.16. *Suppose that $\{M_i\}$ is linearly generated. In Algorithm* 3.1*, there exists a T such that for all $t \geq T$ the matrix polynomial*

$$F(z) = \left[z^{d_1} f_1(z^{-1}) \;\; z^{d_2} f_2(z^{-1}) \;\; \cdots \;\; z^{d_N} f_N(z^{-1}) \right]$$

is the minimal matrix generator for $\{M_i\}$.

PROOF. We introduce the notion of *defect* (cf. Beckermann and Labahn [1994]) for each $1 \leq j \leq 2N$: $\mathrm{dfct}_j^{[t]} = t - d_j^{[t]}$. We now follow the defect of each vector polynomial f_j throughout the algorithm. When a column is placed into the auxiliary part of f and shifted in steps GE19 and MBM9 its defect remains unchanged when reaching step MBM5 again, because the nominal degree is incremented in step MBM6 and t is incremented in step MBM3. The corresponding column that is switched into the generator part has its defect incremented by 1 due to the increment of t, as have all other generator vectors.

Now let ν be the degree of the first invariant factor of the minimal matrix generator, which by Theorem 2.7 is the scalar minimal generator. We claim that the defects of the generator columns eventually grow to ν. The minimal defect $\min_{1 \leq j \leq N}\{\mathrm{dfct}_j^{[t]}\}$ is $t - \mu^{[t]}$. When μ grows, in Case 1 or 4 in the proof of Theorem 4.14, the minimal defect can shrink. However, the corresponding column f_l used in the update has its defect incremented. Since initially the defects are -1 and -2, after at most $N(\nu + 2)$ cases where $\mu^{[t+1]} > \mu^{[t]}$ the defects of all generator columns must be at least ν. In particular, then $\mu \leq t$.

Suppose now that at $t = T$ the defect of all generator polynomials is at least ν. We complete the proof by showing that in Lemma 4.15 we have $\bar{M}_{T+\theta} = M_{T+\theta}$ for all $\theta \geq 1$. For $t = T$ in Theorem 4.14, we have $d \geq \mu$ and $T \geq \mu + \nu$. Furthermore, from (3) on page 14,

$$M_{T-\kappa}[f_1(0) \cdots f_N(0)] + \sum_{i=0}^{\mu-1} M_{T-\kappa-\mu+i} Coeff(i; \bar{F}) = 0^{N \times N} \quad \text{for } \kappa = 0, 1, \ldots, \nu - 1.$$

Let $z^\nu - \sum_{k=0}^{\nu-1} c_k z^{\nu-k}$ be the minimal scalar generator for $\{M_i\}$. Then

$$\begin{aligned} 0^{N \times N} &= \sum_{\kappa=0}^{\nu-1} c_\kappa M_{T-\kappa}[f_1(0) \cdots f_N(0)] + \sum_{\kappa=0}^{\nu-1} \sum_{i=0}^{\mu-1} c_\kappa M_{T-\kappa-\mu+i} Coeff(i; \bar{F}) \\ &= M_{T+1}[f_1(0) \cdots f_N(0)] + \sum_{i=0}^{\mu-1} M_{T+1-\mu+i} Coeff(i; \bar{F}). \end{aligned}$$

Therefore, $\bar{M}_{T+1} = M_{T+1}$. For larger θ, we use induction. □

LEMMA 4.17. *After each iteration of steps* MBM3 *through* MBM7 *of Algorithm* 3.1, $\beta^{[t+1]} + \sigma^{[t+1]} > \beta^{[t]} + \sigma^{[t]}$.

PROOF. During each iteration of steps MBM3 through MBM7, σ can never decrease so that $\sigma^{[t+1]} \geq \sigma^{[t]}$ for all $t \geq -1$.

If $\sigma^{[t+1]} = \sigma^{[t]}$, then steps GE20 and GE23 will not be performed by the call to Algorithm 3.2 in step MBM5. So $\beta^{[t+1]} = \beta^{[t]} + 1$, and thus $\beta^{[t+1]} + \sigma^{[t+1]} > \beta^{[t]} + \sigma^{[t]}$.

If $\sigma^{[t+1]} > \sigma^{[t]}$, then during step MBM5, step GE20 or step GE23 were performed at least once. Therefore, $\beta^{[t+1]} = \beta^{[t]} + 1$, or $\beta^{[t+1]} = d_j^{[t]} + 1$ for some $1 \leq j \leq N$. In the former case, then $\beta^{[t+1]} + \sigma^{[t+1]} = \beta^{[t]} + 1 + \sigma^{[t+1]} > \beta^{[t]} + \sigma^{[t]}$. In the latter case, then $\beta^{[t+1]} + \sigma^{[t+1]} = d_j^{[t]} + 1 + \sigma^{[t+1]} \geq d_j^{[t]} + 1 + \sigma^{[t]} - d_j^{[t]} + \beta^{[t]} > \beta^{[t]} + \sigma^{[t]}$. □

LEMMA 4.18. *Given any input matrix sequence* $\{M_k\}_{k=0}^{\infty}$ *and any bound* δ, *Algorithm* 3.1 *returns "insufficient bound" or a candidate minimal generator* F *after processing at most* 2δ *elements.*

PROOF. Suppose that Algorithm 3.1 is run until $t = 2\delta - 1$, thus processing the first 2δ. Since $\beta^{[-1]} + \sigma^{[-1]} = 1$, then Lemma 4.17 implies that $\beta^{[2\delta-1]} + \sigma^{[2\delta-1]} \geq 2\delta + 1$. If $\sigma \geq \delta + 1$, then "insufficient bound" is returned. If $\sigma \leq \delta$, then $\beta^{[2\delta-1]} + \sigma^{[2\delta-1]} \geq 2\delta + 1 \geq \delta + \mu^{[2\delta-1]} + 1$ since $\mu \leq \sigma$ by definition. Thus, β fails the condition of MBM2 and so Algorithm 3.1 will return a candidate minimal generator F. □

THEOREM 4.19. *Suppose* $\{M_k\}_{k=0}^{\infty}$ *is linearly generated by a minimal matrix generator with minimal degree* d *and determinantal degree* $\delta_M \leq \delta$. *Then, Algorithm* 3.1 *returns a minimal generator of* $\{M_k\}_{k=0}^{\infty}$ *after processing at most* $d + \delta$ *elements. Otherwise,* $\{M_k\}_{k=0}^{\infty}$ *is not linearly generated or has minimal generator of determinantal degree* $\delta_M > \delta$. *In either case, Algorithm* 3.1 *returns "insufficient bound" or an incorrect generator after processing at most* $\min(\bar{d}, \delta) + \delta$ *elements where* $\bar{d}$ *is the degree of the generator candidate computed at* $t = 2\delta - 1$ *(see Lemma* 4.15*).*

PROOF. We will show that Algorithm 3.1 returns a minimal matrix generator of a linearly generated sequence if the bound δ is large enough. We also give a tighter bound on the number of elements that need to be processed before terminating.

We begin by assuming the sequence is linearly generated and $\delta_M \leq \delta$. Since Lemma 4.16 states that Algorithm 3.1 will find a minimal generator, we can assume that $\mu \leq d$ for all t. We proceed as in Lemma 4.18 and assume the algorithm is run until $t = d + \delta - 1$. Then, $\beta + \sigma \geq d + \delta + 1$ and so $\sigma > \delta$ or $\beta \geq \delta - \sigma + \mu + 1$. If $\sigma > \delta$, then step MBM8 returns "insufficient bound". If $\sigma \leq \delta$, then the condition of step MBM2 is violated. So the algorithm will terminate and return either "insufficient bound" or a candidate minimal generator.

If "insufficient bound" is returned, then $\sigma \geq \delta + 1$. Lemma 4.16 implies that there exists a T such that if Algorithm 3.1 is computed to stage $T > t = d + \delta - 1$ or beyond, then $F(z)$ is a minimal matrix generator. Since $\sigma^{[t]}$ is an increasing sequence, then $\sigma^{[T]} \geq \sigma > \delta$. If we can show that $\delta_M = \deg(\det(F(z))) = \sigma^{[T]} > \delta$, then we have proven $\delta_M > \delta$ and reached a contradiction. The upper bound $\deg(\det(F(z))) \leq \sigma^{[T]}$ is straightforward since Lemma 4.1 and Lemma 4.3 imply that $\deg(F_j(z)) = d_j$ for all $1 \leq j \leq N$. The multilinear property of the determinant allows us to write $\det(F(z))$ in the following manner:

$$\det(F(z)) = z^{\sigma^{[T]}} \cdot \det([f_1(0) \cdots f_N(0)]) + \text{lower order terms.}$$

By Lemma 4.3, we know $\det(\begin{bmatrix} f_1(0) & \cdots & f_N(0) \end{bmatrix}) \neq 0$ and so $\deg(\det(F(z))) = \sigma^{[T]}$. Therefore, $\delta_M > \delta$ and so we have a contradiction.

If "insufficient bound" is not returned, then we have $\delta \geq \sigma$, $F(z)$ a possible minimal generator and $\beta - \mu > \delta - \sigma$. Since $\delta - \sigma \geq 0$, then $\beta > \mu$. Suppose $F(z)$ is not the minimal generator of $\{M_k\}_{k=0}^{\infty}$. Lemma 4.16 proves that there is a $T \geq t = d+\delta-1$ such that Algorithm 3.1 returns a minimal generator. Since $F(z)$ is not a generator, then as the algorithm is run until T, there must be a nonzero discrepancy. Since $\beta > \mu$, then σ will be increased such that $\sigma^{[T]} \geq \sigma + \beta - \mu > \delta$. As in the previous case, this implies that $\delta_M > \delta$ and a contradiction has been found. Therefore, $F(z)$ is a minimal generator of $\{M_k\}_{k=0}^{\infty}$.

Otherwise, assume $\{M_k\}_{k=0}^{\infty}$ is not linearly generated or is linearly generated with determinantal degree $\delta_M > \delta$. Suppose Algorithm 3.1 is run until $t = \min(\bar{d}, \delta) + \delta - 1$. So $\beta + \sigma \geq \min(\bar{d}, \delta) + \delta + 1$. If $\sigma > \delta$, then "insufficient bound" is returned. If $\sigma \leq \delta$ then we know that $\mu \leq \min(\bar{d}, \delta)$ by definition of μ and $\bar{d}$ and so $\beta \geq \delta - \sigma + \mu + 1$. So Algorithm 3.1 will return a possible generator F with determinantal degree σ. Since $\{M_k\}_{k=0}^{\infty}$ is not linearly generated or is linearly generated with determinantal degree $\delta_M > \delta \geq \sigma$, then F is an incorrect generator. □

Remark 4.20. Note that Theorem 4.19 is also applicable in the scalar case for the classical Berlekamp/Massey algorithm: if a degree bound δ for the linear generator is given, the Berlekamp/Massey algorithm can stop after processing $d + \delta$ sequence elements, where d is the actual degree of the generator. That early termination property is useful, for example, in the Wiedemann algorithm for solving sparse linear systems over finite fields [Wiedemann 1986; Kaltofen and Saunders 1991].

Theorem 4.19 gives an exhaustive analysis of the way Algorithm 3.1 terminates for all possible input. If the the bound δ is sufficient, then a minimal generator is returned. If the bound δ is too small, then the algorithm may return "insufficient bound" or an incorrect candidate generator. We ignore the possibility that the sequence is not linearly generated since only a finite number of sequence elements are processed and the sequence can always be extended to a linearly generated sequence (see Lemma 4.15). It is important to note that no algorithm, including Algorithm 3.1, can determine if the bound δ is sufficient. Such verification requires infinitely many sequence elements be processed, since the determinantal degree can increase at any stage. The correctness of the output of Algorithm 3.1 is dependent on δ, and the algorithm must assume the bound δ is sufficient. If "insufficient bound" is returned, then the algorithm has proven that the bound δ is too small since the proof of Theorem 4.19 shows that the sequence can have no matrix generator of determinantal degree less than or equal to δ. Otherwise, Algorithm 3.1 returns a minimal generator of the completed sequence from Lemma 4.15 and assumes that the input sequence and the completed sequence are the same, which is true if the bound δ is sufficient. The output of Algorithm 3.1 is always correct for a given input δ that is assumed to be sufficient.

5. MATRIX BERLEKAMP/MASSEY AND RECTANGULAR MATRIX SEQUENCES

Algorithms 3.1 and 3.2 and the subsequent proofs assume that the matrix sequence $\{M_k\}_{k=0}^{\infty}$ consists of square matrices. In this section, we will detail the necessary changes that allow the algorithms to compute a minimal right generator of a rectangular matrix sequence. We revive our notation N_{row} and N_{col} to denote the row and column dimensions of the sequence. Rather than rewriting the algorithms, we only detail the differences between the square and rectangular algorithms. Finally, we give justification for our changes by appealing to the square algorithm and its output given a special input.

The input to Algorithm 3.1 will now be $M \in K^{N_{\text{row}} \times N_{\text{col}}}[[z]]$. The output of the algorithm will be $F(z) \in K^{N_{\text{col}} \times N_{\text{col}}}[z]$. The variable $f(z)$ is now an $N_{\text{col}} \times (N_{\text{col}} + N_{\text{row}})$ matrix. For the entire algorithm, the quantity $2N$ will be replaced by $N_{\text{col}} + N_{\text{row}}$. The matrix Δ will

now have dimensions $N_{\text{row}} \times (N_{\text{col}} + N_{\text{row}})$. At Step MBM1, f is initialized to be the matrix $[I_{N_{\text{col}}}\ 0^{N_{\text{col}} \times N_{\text{row}}}]$. The nominal degrees will be initialized thusly, $d_i = 0$ for all $i \leq N_{\text{col}}$ and $d_i = 1$ otherwise. In steps MBM4 and MBM5, $L^{[t]} \in K^{N_{\text{row}} \times N_{\text{row}}}$. Also $C^{[t]}$, $Z^{[t]} \in K^{N_{\text{row}} \times N_{\text{col}}}$. The diagonal block matrix in Step MBM9 will be changed to $\text{diag}(I_{N_{\text{col}}}, z \cdot I_{N_{\text{row}}})$. In steps MBM6, MBM7, and MBM10, N is changed to N_{col}.

The changes that must be made to Algorithm 3.2 are as follows. As previously mentioned, $2N$ becomes $N_{\text{col}} + N_{\text{row}}$. In Step GE3, N will be changed to N_{row}. Everywhere else, in Step GE2 and steps GE4 through GE24, N is changed to N_{col}.

The specific changes given above are a result of the general changes that must occur when moving from square the rectangular matrix sequences. As in Section 2, the minimal right generator is a square matrix of dimension N_{col}. Thus, the algorithm must reflect this dimension change. Because the sequence has row dimension N_{row}, then the algorithm must maintain N_{row} auxiliary vectors. This reflects the possible rank of the columns of the matrix sequence. The number of generator columns and auxiliary columns affects the initialization and computation of f and its associated variables. The differentiation between generator and auxiliary columns also forces the changes in Algorithm 3.2.

We now give justification for the changes listed previously. Rather than redo the complete proof of the square algorithm, we will use the square algorithm to prove the rectangular algorithm. Given the rectangular matrix sequence $\{M_k\}_{k=0}^{\infty}$, we will construct a square matrix $\{\bar{M}_k\}_{k=0}^{\infty}$. The square sequence will have dimension $\max\{N_{\text{row}}, N_{\text{col}}\}$. If $N_{\text{row}} > N_{\text{col}}$, then $\bar{M}_k = [M_k\ 0^{N_{\text{row}} \times (N_{\text{row}} - N_{\text{col}})}]$. Otherwise $\bar{M}_k$ is M_k padded by $N_{\text{col}} - N_{\text{row}}$ zero rows. Now we consider the execution and output of Algorithm 3.1 when the sequence $\{\bar{M}_k\}_{k=0}^{\infty}$ is given as input. We assume that the input δ is the same for both sequences.

Suppose $N_{\text{col}} > N_{\text{row}}$. Then, the square sequence contains the original sequence as full column submatrices. Thus, any generator of $\{\bar{M}_k\}_{k=0}^{\infty}$ is a generator of $\{M_k\}_{k=0}^{\infty}$. The inverse is also true since the added rows are zero rows. Thus, the output of Algorithm 3.1 is a minimal generator of $\{M_k\}_{k=0}^{\infty}$, when $\{\bar{M}_k\}_{k=0}^{\infty}$ is given as input. However, the square algorithm contains too many auxiliary columns. At every stage t, the bottom $N_{\text{col}} - N_{\text{row}}$ rows of $\Delta^{[t]}$ are zero. Thus, the last $N_{\text{col}} - N_{\text{row}}$ auxiliary columns will remain $0^{N_{\text{col}}}$ throughout the algorithm. It is therefore, unnecessary to compute those columns and those rows of $\Delta^{[t]}$. By changing the algorithms as listed here, these computations are eliminated.

If $N_{\text{row}} > N_{\text{col}}$, then the output of Algorithm 3.1 given $\{\bar{M}_k\}_{k=0}^{\infty}$ has incorrect dimensions to be a minimal generator of $\{M_k\}_{k=0}^{\infty}$. Since the last $N_{\text{row}} - N_{\text{col}}$ columns of the matrix sequence are zero columns, then the last $N_{\text{row}} - N_{\text{col}}$ columns of $C^{[t]}$ are zero columns. This means that the column $f_i^{[t]}$ is unchanged since initialization and so $d_i^{[t]} = 0$ for all $N_{\text{col}} + 1 \leq i \leq N_{\text{row}}$. Therefore, if $\bar{F}$ is the output of Algorithm 3.1, when given $\{\bar{M}_k\}_{k=0}^{\infty}$ as input, we know that $\bar{F} = \text{diag}(F, I_{N_{\text{row}} - N_{\text{col}}})$, with $F \in K^{N_{\text{col}} \times N_{\text{col}}}[z]$. F is a minimal right generator of $\{M_k\}_{k=0}^{\infty}$ since the $\deg(\det(\bar{F})) = \deg(\det(F))$. So Algorithm 3.1 computes the minimal generator of $\{M_k\}_{k=0}^{\infty}$ when given $\{\bar{M}_k\}_{k=0}^{\infty}$ as input. The rectangular algorithm described above eliminates the computation of the last $N_{\text{row}} - N_{\text{col}}$ generator columns. These columns do not need to be computed since they are precomputed by the nature of the padded sequence.

The augmentations of Algorithm 3.1 and Algorithm 3.2 described here eliminate the unnecessary computations performed when a rectangular matrix sequence is converted to a square matrix sequence by padding the sequence with zeroes. The proof of the original algorithm in Section 4 implies that the rectangular algorithm is correct. This completely generalizes the Matrix Berlekamp/Massey algorithm and implies that given

any linearly generated matrix sequence and a proper bound on the determinantal degree of a minimal generator, the Matrix Berlekamp/Massey algorithm will compute a minimal generator.

6. PERFORMANCE

We end with a discussion of the complexity of Algorithm 3.1 and empirical evidence of its performance versus another method for computing minimal matrix generators. The results of Lemma 4.18 and Theorem 4.19 on page 19 give a bound on the number of elements Algorithm 3.1 processes before terminating. By amortizing the cost of each iteration, the bound will lead to a worst case complexity for the algorithm. Finally, we provide the results of a comparison between Algorithm 3.1 and a method based on the fast power hermite padé solver of Beckermann and Labahn [1994].

THEOREM 6.1. *Given any bound δ and any matrix sequence $\{M_k\}_{k=0}^{2\delta-1}$ Algorithm* 3.1 *has a worst-case complexity of $O(\delta^2 N^2 + \delta N^3)$ field operations, where N is the matrix dimension of the sequence.*

PROOF. We will analyze steps MBM4, MBM5, and MBM9 of Algorithm 3.1. These steps represent the major field operation costs of each iteration. In step MBM4, we assume L is computed by step MBM5 of the previous iteration, and so only C is computed. This assumption allows the cost of steps MBM4 and MBM9 to be given in terms of δ. The results of Lemma 4.18 and Theorem 4.19 state that Algorithm 3.1 processes at most 2δ elements of the sequence.

Since step MBM5 is Gaussian elimination, it has a complexity of $O(N^3)$ field operations. We will amortize the costs of steps MBM4 and MBM9 by analyzing the computation cost column by column. Using matrix times vector products and matrix column operations instead of matrix multiplication will result in the stated complexity. Computing column j of C requires $d_j + 1$ matrix times vector products and d_j vector additions for each $1 \le j \le N$. By summing over the range of j, then step MBM4 has a cost of $\sigma N^2 + N^3 + \sigma N$ field operations. Since $\sigma \le \delta$, then each iteration of step MBM4 has a complexity of $O(\delta N^2 + N^3)$ field operations. To analyze step MBM9, we use matrix column operations instead of matrix multiplication. The transformation matrix τ computed in step MBM5 has two types of column operations. Each column operation is either a column switch or an addition of one polynomial vector and a second polynomial vector of a lesser degree times a scalar. Each generator column requires at most N of the latter column operations and so τ has at most N^2 of these column operations. Each of these operations has a complexity of $O(d_j N + N)$ field operations where $1 \le j \le N$. So the total cost of computing column j of f has a complexity of $O(d_j N^2 + N^2)$. The auxiliary columns will be computed as a result of the computations of the generator columns. By summing over $1 \le j \le N$, then the total cost of step MBM9 has a complexity of $O(\sigma N^2 + N^3)$ field operations. During step MBM5, σ is updated. If $\sigma \le \delta$, then the total cost of step MBM9 has a complexity of $O(\delta N^2 + N^3)$ field operations. If $\sigma > \delta$, then step MBM8 will return "insufficient bound" and step MBM9 will not be performed.

Algorithm 3.1 processes each element of the sequence with worst case complexity of $O(\delta N^2 + N^3)$ field operations and it processes at most 2δ elements. So Algorithm 3.1 has a worst case complexity of $O(\delta^2 N^2 + \delta N^3)$ field operations. □

We now compare the performance of Algorithm 3.1 with an algorithm based on the fast power hermite padé solver of Beckermann and Labahn [1994]. Turner [2002] describes the relationship between the minimal matrix generator of a linearly generated matrix sequence and a σ-basis of an associated Padé system. Turner implemented his methods in Maple and provided a copy to the authors. We altered his original code

Table II. Arithmetic Operation Comparison

Matrix Dimension	Experiment 6		Experiment 6	
	Algorithm 3.1	[Turner 2002]	Algorithm 3.1	[Turner 2002]
1	29946	99034	30720	100155
2	64665	380339	64133	384443
3	101948	868197	104877	907893
4	143652	1562988	140967	1557380
5	185175	2450470	180659	2456240
6	228962	3559226	234787	3696432
7	280705	4871117	301006	5350720
8	327771	6373395	343334	6888984
9	394147	8061927	423928	9409654
10	433936	10009119	429072	9992999

to work over finite fields and added an operation count so the comparison could be performed. A Maple implementation of Algorithm 3.1 was used for comparison.

We test the algorithms using random square matrix sequences over GF2 with matrix dimensions varying from 1 to 10 and a fixed degree bound for each dimension. For each dimension, 10 random sequences are constructed and the minimal matrix generator of each sequence is computed. During each computation the number of arithmetic operations performed by each algorithm is calculated and an average for each dimension is determined. The sequences are constructed using two methods described here.

Experiment A For each dimension i, the sequences are constructed from bilinear block projections. A random 100×100 matrix A, and two random $100 \times i$ matrices X and Y are used to create the sequence $\{M_k\}_{k=0}^{\infty} = X^T A^k Y$. The degree bound for each i is fixed at 100.

Experiment B For each dimension i, random matrix generators and initial sequences are used to construct each sequence. A matrix generator of degree $\lceil 100/i \rceil$ is constructed from random coefficients and a random initial sequence of length $\lceil 100/i \rceil$ is defined. Then using Definition 2.3, the sequence is completed. The degree bound for each dimension i is $\lceil 100/i \rceil i \geq 100$.

The results of the comparison given in Table II show that Algorithm 3.1 performs significantly faster than the σ-bases methods of Turner [2002]. For most dimensions the results for each experiment are similar. For some dimensions, such as dimension 9, the results of Experiment B are much larger than Experiment A. This is due to the larger degree bound used in Experiment B, which in the case of dimension 9 is 108 as compared to 100 for Experiment A.

REFERENCES

Beckermann, B. and Labahn, G. 1994. A uniform approach for fast computation of matrix-type Padé approximants. *SIAM J. Matrix Anal. Applic. 15*, 3, 804–823.

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

Brent, R. P., Gustavson, F. G., and Yun, D. Y. Y. 1980. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *J. Algorithms 1*, 259–295.

Cabay, S., Choi, D. K., and Labahn, G. 1990. Inverses of block Hankel and block Toeplitz matrices. *SIAM Journal of Computing 19*, 98–123.

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

Dickinson, B. W., Morf, M., and Kailath, T. 1974. A minimal realization algorithm for matrix sequences. *IEEE Trans. Automatic Control* AC-19, *1*, 31–38.

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

DUMMIT, D. S. AND FOOTE, R. M. 1991. *Abstract Algebra*. Prentice Hall, Inc., Englewood Cliffs, NJ 07632.

DURBIN, J. 1959. The fitting of time-series models. *Rev. Int. Stat. Inst. 28*, 229–249.

GIESBRECHT, M., KALTOFEN, E., AND LEE, W.-S. 2003. Algorithms for computing sparsest shifts of polynomials in power, chebychev, and pochhammer bases. *J. Symbolic Comput. 36*, 3–4, (Special issue (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL://EKbib/03/GKL03.pdf.) 401–424.

GIESBRECHT, M., KALTOFEN, E., AND LEE, W.-S. 2002. Algorithms for computing the sparsest shifts for polynomials via the Berlekamp/Massey algorithm. In *Proceedings of the 2002 International Symposium on Symbolic Algebraic Comput. (ISSAC'02)*, T. Mora (Ed.). ACM, New York, 101–108.

GIORGI, P., JEANNEROD, C.-P., AND VILLARD, G. 2003. On the complexity of polynomial matrix computations. In *Proceedings of the 2003 International Symposium on Symbolic Algebraic Comput. (ISSAC'03)*, J. R. Sendra Ed., ACM, New York, 135–142.

KAILATH, T. 1980. *Linear systems*. Prentice-Hall.

KALTOFEN, E. 1992. On computing determinants of matrices without divisions. In *Proceedings of the 1992 International Symposium on Symbolic Algebraic Computing (ISSAC'92)*, P. S. Wang (Ed.). ACM Press, New York, 342–349.

KALTOFEN, E. 1995. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Math. Comput. 64*, 210, 777–806.

KALTOFEN, E. AND LEE, W.-S. 2003. Early termination in sparse interpolation algorithms. *J. Symbolic Comput. 36*, 3–4, Special issue (ISSAC 2002). 365–400.

KALTOFEN, E. AND SAUNDERS, B. D. 1991. On Wiedemann's method of solving sparse linear systems. In *Proceedings of the (AAECC-9)*. H. F. Mattson, T. Mora, and T. R. N. Rao Eds., Lecture Notes in Computer Science, vol. 539. Springer-Verlag, Berlin, Germany, 29–38.

KALTOFEN, E. AND VILLARD, G. 2004. On the complexity of computing determinants. *Comput. Complex. 13*, 3–4, 91–130.

LEVINSON, N. 1947. The Wiener RMS (Root-Mean-Square) error criterion in the filter design and prediction. *J. Math. Phys. 25*, 261–278.

LOBO, A. A. 1995. Matrix-Free Linear System Solving and Applications to Symbolic Computation. Ph.D. dissertation. Rensselaer Polytechnic Instit., Troy, NY.

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

POPOV, V. M. 1970. Some properties of control systems with irreducible matrix transfer functions. In *Lecture Notes in Mathematics*. Vol. 144, Springer-Verlag, Berlin, 169–180.

RISSANEN, J. 1972a. *Realizations of Matrix Sequences*. Technical rep. RJ-1032. IBM T. J. Watson Research Center, Yorktown Heights, New York.

RISSANEN, J. 1972b. *Realizations of Matrix Sequences*. Technical rep. RJ-1032. IBM T. J. Watson Research Center, Yorktown Heights, New York.

SUGIYAMA, Y., KASAHARA, M., HIRASAWA, S., AND NAMEKAWA, T. 1975. A method for solving key equation for decoding Goppa codes. *Inf. Cont. 27*, 87–99.

THOMÉ, E. 2002. Subquadratic Computation of Vector Generating Polynomials and Improvements of the Block Wiedemann Method. *J. Symbolic Comput. 33*, 5, 757–775.

TURNER, W. J. 2002. Black box linear algebra with the LINBOX library. Ph.D. dissertation. North Carolina State University, Raleigh, North Carolina.

VAN BAREL, M. AND BULTHEEL, A. 1991. The computation of non-perfect Padé-Hermite approximants. *Numer. Algo. 1*, 285–304.

VAN BAREL, M. AND BULTHEEL, A. 1992. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numer. Algo. 3*, 451–462.

VILLARD, G. 1997a. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In *ISSAC 97: Proceedings of the 1997 International Symposium on Symbolic Algebraic Computation*, W. Küchlin Ed., ACM, New York, 32–39.

VILLARD, G. 1997b. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Rapport de Recherche 975 IM. Institut d'Informatique et de Mathématiques Appliquées de Grenoble.

WIEDEMANN, D. 1986. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory* IT-32, 54–62.

Received December 2011; revised February 2013; accepted March 2013