Algorithms for Linearly Recurrent Sequences of Truncated Polynomials

by

Seung Gyu Hyun

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Linear recurrent sequences are those whose elements are defined as linear combinations of preceding elements, and finding recurrence relations is a fundamental problem in computer algebra. In this paper, we focus on sequences whose elements are vectors over the ring $\mathbb{A} = \mathbb{K}[x]/\langle x^d \rangle$ of truncated polynomials. Finding the ideal of their recurrence relations has applications such as the computation of minimal polynomials and determinants of sparse matrices over $\mathbb{A}$. We present three methods for finding this ideal: a Berlekamp-Massey-like approach due to Kurakin, one which computes the kernel of some block-Hankel matrix over $\mathbb{A}$ via a minimal approximant basis, and one based on bivariate Padé approximation. We propose complexity improvements for the first two methods, respectively by avoiding the computation of redundant relations and by exploiting the Hankel structure to compress the approximation problem. Then we confirm these improvements empirically through a C++ implementation, and we discuss the above-mentioned applications.

Acknowledgements

I would like to thank my supervisor Prof. Eric Schost for all his guidance throughout my university years, and especially for his patience and understanding during my undergraduate research terms. I would also like to thank Prof. Vincent Neiger for being so welcoming and friendly during my semester in France and beyond, as well as the countless hours he has spent helping me.

I would like to thank Jean for her endless love and support.

Finally, I would like to thank the readers Prof. George Labahn and Prof. Mark Giesbrecht.

Dedication

This is dedicated to my parents and brother.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Linear recurrences appear in many domains of computer science and mathematics, such as coding theory [7, 43], economics [61], and are used in many fast algorithms in computer algebra [65, 17]. The most famous example of a linearly recurrent sequence is the Fibonacci sequence, where each element is the sum of the two preceding elements. More generally, a r-dimensional sequence $\boldsymbol{s} = (S_{i_1,i_2,\ldots,i_r})$ of elements in $\mathbb{K}$, for some field $\mathbb{K}$, is said to be linearly recurrent if all elements (after some index) of $\boldsymbol{s}$ can be written as a (fixed) linear combination of preceding elements. Given such a sequence, we seek a representation of its *annihilator*, which is a (r-variate) polynomial ideal corresponding to all recurrence relations which are satisfied by the sequences; the polynomials in the annihilator are said to *cancel* the sequence.

In the simplest case, when $r = 1$, the annihilator can be represented by a single unique monic univariate polynomial of minimal degree. Continuing with the example of the Fibonacci sequence $\boldsymbol{s}_F = (S_i)_{i\geq 0} = (1, 1, 2, 3, 5, 8, \ldots)$, we can see that $S_{i+2} = S_{i+1} + S_i$ or $S_{i+2} - S_{i+1} - S_i = 0$. By replacing S_{i+j} with y^j, we get the canceling polynomial $y^2 - y - 1$ that corresponds to this relation. In general, a polynomial $\sum c_j y^j$ is said to *cancel* a sequences if the c_i's also satisfy $\sum c_i S_{i+j} = 0$, for all i (see Chapter 2 for more details). One can verify that $y^2 - y - 1$ is minimal since there exists no c that satisfies both $1 \cdot c = 1$ and $1 \cdot c = 2$, which implies that there exists no canceling polynomial of degree 1. Together with the fact that $\mathbb{K}[x]$ is a principal ideal domain, the annihilator of $\boldsymbol{s}_F$ is $\langle y^2 - y - 1\rangle$.

When $r > 1$, linear recurrences are defined in the same way, with the exception that recurrences can be defined in multiple directions. Figure 1.1 shows an example of a sequence that simultaneously satisfies $S_{i_1+2,i_2} - S_{i_1+1,i_2} - S_{i_1,i_2} = 0$ and $S_{i_1,i_2+1} - S_{i_1+1,i_2} - S_{i_1,i_2} = 0$. Again, by replacing $S_{i_1+j_1,i_2+j_2}$ with $y_1^{j_1} y_2^{j_2}$, we get the canceling polynomials $y_1^2 - y_1 - 1$ and

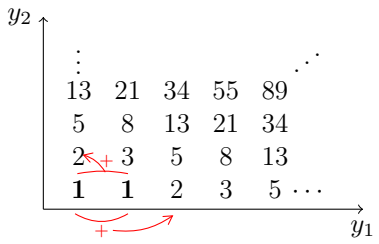

Figure 1.1: Two-dimensional linearly recurrent sequence

$y_2 - y_1 - 1$ respectively. Note that unlike the previous case, the two canceling polynomials that are not multiples of each other.

For $r = 1$, a quadractic algorithm to compute the unique monic canceling polynomial was first proposed by Berlekamp and Massey [7, 43]. For high dimensions $r > 1$, Sakata first extends the BM algorithm to dimension 2 [55] and then to the general case $r > 1$ [56]; see also Norton and Fitzpatrick's extension to $r > 1$ [22]. Recent work includes variants of Sakata's algorithm such as one which handles relations that are satisfied by several sequences simultaneously [57], approaches relating the problem to the kernel of a multi-Hankel matrix and exploiting either fast linear algebra [8] or a process similar to Gram-Schmidt orthogonalization [46], and an algorithm relying directly on multivariate polynomial arithmetic [9]. Since uniqueness of monic canceling polynomials of minimal degree does not hold for $r > 1$, algorithms must compute a suitable representation for the annihilator; the above algorithms compute a Gröbner basis or border basis of the annihilator.

In this thesis, we focus on computing recurrence relations for sequences whose elements are in $\mathbb{A}^n$, where $\mathbb{A} = \mathbb{K}[x]/\langle x^d \rangle$. When $n = 1$, such sequences correspond to bi-dimensional ($r = 2$) sequences whose elements are zero when one of the indices is greater or equal to d (see Section 2.5). By increasing $n > 1$, we want to find annihilators that simulatenously cancel all n sequences over $\mathbb{A}$. For example, setting $d = 2$ and $n = 2$, we could have a Fibonacci-like sequences

$$\left(\begin{bmatrix} 1+2x \\ 3+9x \end{bmatrix}, \begin{bmatrix} 1+3x \\ 6+3x \end{bmatrix}, \begin{bmatrix} 2+5x \\ 9+12x \end{bmatrix}, \begin{bmatrix} 3+8x \\ 12+15x \end{bmatrix}, \dots \right).$$

In this case, the canceling polynomials $y^2 - y - 1$ and $x(y^2 - y - 1)$ cancel both $(1 + 2x, 1+3x, 2+5x, 3+8x, \dots)$ and $(3+9x, 6+3x, 9+12x, 12+15x, \dots)$, and are minimal in degree; thus, the annihilator of this sequence is $\langle y^2-y-1 \rangle$ (see Section 3.1 for more details). Note that this ideal is over $\mathbb{A}[y]$; we can covert this to a bivariate ideal over $\mathbb{K}[\alpha, \beta]$ by a

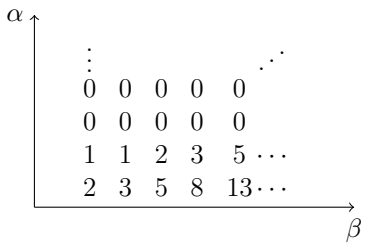

Figure 1.2: Corresponding 2-dimensional sequence for $(1+2x, 1+3x, 2+5x, 3+8x, \dots)$

"natural" mapping $x \mapsto \alpha, y \mapsto \beta$ and appending α^d. For this sequence, the resulting ideal $\langle \beta^2 - \beta - 1, \alpha^d \rangle$ also cancels the bi-dimensional sequences corresponding to the sequences $(1+2x, 1+3x, 2+5x, 3+8x, \dots)$ and $(3+9x, 6+3x, 9+12x, 12+15x, \dots)$. An example of the mapping between sequences with elements in $\mathbb{A}$ to $\mathbb{K}^2$ is shown in Fig. 1.2.

There are three main sources of motivation for this focus. Firstly, while the annihilator can be computed in cost that is optimal (up to log factors) in the size of the output in dimension 1, previous algorithms for higher dimensions are not optimal, even though this problem has been studied extensively. Although the annihilators of sequences over $\mathbb{A}^n$ can be computed via previous algorithms (by computing the annihilator of an equivalent bidimensional sequence), our approach is able to exploit the specific structure of the base ring $\mathbb{A}$ and yield algorithms with better complexities. Secondly, sequences over $\mathbb{A}^n$ arise when designing Wiedemann-like algorithms for sparse matrices with truncated polynomial entries [66]. Finally, computing the annihilators of linearly recurrent sequences forms the basis for some algorithms that convert Gröbner basis of zero-dimensional ideals from one monomial ordering to another (typically lexicographic), such as the sparse-FGLM algorithm [17] and the algorithms of Neiger et. al. [51]. The former uses uni-dimensional sequences ($r = 1$) but requires the underlying lexicographic Gröbner basis to be in shape position, meaning that the points in the variety are separated by the coordinates of the largest variable in lexicographic order. The latter extends this approach by only requiring that the ideal is supported at the origin, but the computations use multi-dimensional sequences ($r > 1$). By improving algorithms for a simpler restricted case ($r = 2$ and truncated at index d), our aim is to eventually improve algorithms for general multi-dimensional sequences; we leave this as future work.

Recurrence relations for sequences over $\mathbb{A}^n$ can be computed directly by using a specialization of Kurakin's algorithm [37, 38], as detailed in Section 3.1, where we explicitly describe the generating set of the annihilator as a lexicograhic Gröbner basis of some bivariate ideal. We derive a cost bound $O^{\sim}(\delta d(n^2\delta d + n^\omega d))$ operations in $\mathbb{K}$, where δ is

Algorithm	Complexity	Description
Kurakin	$O^{\sim}(\delta d(n^2\delta d + n^\omega d))$	computes all d possible polynomials
Lazy Kurakin	$O^{\sim}(\delta d^*(n^2\delta d + n^\omega d))$	computes $d^* \leq d$ polynomials
PM-basis	$O^{\sim}(\delta^\omega n d)$	computes kernel of a Hankel matrix over $\mathbb{A}$
Compressed PM-basis	$O^{\sim}(\delta^2 n d + \delta^\omega d)$	compresses matrix then finds kernel
Bivariate Padé	$O^{\sim}(\delta d^{\omega+1})$	bivariate extension of PM-basis

Table 1.1: Summary of proposed algorithms with costs in the number of operations in $\mathbb{K}$

the order of recurrence (see Section 2.3), and ω is an exponent for matrix multiplication over $\mathbb{K}$ [14, 41, 1]. Because the Gröbner bases computed by Kurakin's algorithm are often non-minimal, in Section 3.2, we propose a modified algorithm which aims at limiting, as much as possible, the computation of these extraneous generators. This lowers the cost to $O^{\sim}(\delta d^*(n^2\delta d + n^\omega d))$, where d^* is a number arising in the algorithm as an upper bound on the cardinality d_{opt} of minimal Gröbner bases of the annihilator. In Chapter 5, we observe empirically that d^* is often close or equal to d_{opt}.

Despite the improvement, the above cost bound still has a dependence at least quadratic in the dimension n. Our interest in the case $n \gg 1$ is motivated among others by the following fact: given a zero-dimensional ideal $\mathcal{I} \in \mathbb{K}[x, y]$, one can recover a Gröbner basis of it via $\mathcal{I} = \text{Ann}(\boldsymbol{s}) \subseteq \mathbb{K}[x, y]$ for some well chosen $\boldsymbol{s} \in \mathbb{A}^{\mathbb{N}}$ only if $\mathbb{K}[x, y]/\mathcal{I}$ has the *Gorenstein* property [42, 27]. When that is not the case, one can recover a basis of $\mathcal{I}$ via the annihilator of *several* sequences simultaneously, which means precisely $n > 1$.

For large n, we compute the annihilator via a minimal approximant basis of a block-Hankel matrix over $\mathbb{A}$ constructed from $\boldsymbol{s}$. Computing this approximant basis via the algorithm PM-Basis of [25] leads to a complexity of $O^{\sim}(\delta^\omega n d)$ operations in $\mathbb{K}$ (Section 4.1). We then propose a novel improvement of this minimal approximant basis computation, based on a randomized compression of the input matrix which leverages its block-Hankel structure, reducing the cost to $O^{\sim}(\delta^2 n d + \delta^\omega d)$ operations in $\mathbb{K}$ (Section 4.2). Furthermore, in Section 4.3 we propose an algorithm with cost quasi-linear in the order δ, whereas the above cost bounds are at least quadratic. For $d \in O(\delta)$, we compute the annihilator via the bivariate Padé approximation algorithm of [47]: this uses $O^{\sim}(d^{\omega+1}\delta)$ operations in $\mathbb{K}$, at the price of restricting to $n \in O(1)$. The complexities of the various algorithms proposed is summarized in Table 1.1.

The four above algorithms have been implemented in C++ using the libraries NTL [60] and PML [29], using Lazard's structural theorem [40] for generating examples of sequences; see Chapter 5 for more details. Our experiments on a prime field $\mathbb{K}$ highlight a good match between cost bounds and practical running times, confirming also the benefit obtained from

the improvements of both Kurakin's algorithm and the plain approximant basis approach.

Finally, in Chapter 6 we mention applications to the computation of minimal polynomials and determinants of sparse matrices over $\mathbb{A}$. To design Wiedemann-like algorithms for such matrices $A \in \mathbb{A}^{\mu \times \mu}$, we need to compute annihilators from sequences of the form $(u^T A^i v)_{i \geq 0} \in \mathbb{A}^{\mathbb{N}}$ for some vectors u and v; several such sequences may be needed, leading to the case $n > 1$.

Sakata's 2-dimensional algorithm shares similarities with the case $n = 1$ of Kurakin's algorithm, and has the same complexity $O(\delta^2 d^2)$ [55, Thm. 3]. Apart from this, to the best of our knowledge previous work has $n = 1$ and considers r-dimensional sequences over $\mathbb{K}$ for an arbitrary $r \geq 2$ [8, 9, 46]. Complexity in this r-variate context is often expressed using the degree D of the considered zero-dimensional ideal; here, $\delta \leq D \leq \delta d$ and a minimal Gröbner basis or a border basis will have at most $\min(\delta, d) + 1$ elements. The Scalar-FGLM algorithm has cost $O\tilde{}(d_{\text{opt}} \delta^\omega d)$ [8, Prop. 16]. Both the Artinian border basis and Polynomial-Scalar-FGLM algorithms [46, 9] cost $O(D^2 \delta d)$, which is $O(\delta^3 d)$ in the most favourable case $D = \delta$, and $O(\delta^3 d^3)$ when $D \in \Theta(\delta d)$ (which will be the case in our experiments, see Chapter 5). In all cases, a better complexity bound can be achieved by one of our algorithms outlined above.

Chapter 2

Background

2.1 Basic algorithmic tools

The basic building blocks of our algorithms are univariate polynomial operations: addition, truncation, and multiplication. While addition and truncation of polynomials of degree $< d$ are straightforward and have linear complexities, naive polynomial multiplication requires quadratic operations over $\mathbb{K}$. The first subquadratic multiplication algorithm is the famous Karatsuba's algorithm [34], which reduced the complexity to $O(d^{\lg 3})$. Methods such as multi-point evaluation/interpolation and Fast Fourier Transform are quasi-linear in d [23]. Note that this implies all ring operations over $\mathbb{K}[x]/\langle x^d \rangle$ can be done in $O\tilde{}(d)$, where $O\tilde{}(\cdot)$ omits logarithmic factors.

We also frequently use polynomial matrix operations. Matrices of size $n \times n$ can be multiplied naively in $O(n^3)$ operations over the base ring of the matrix. Fast matrix multiplication has been extensively studied, with improvements still being made [1]. We define $2 \leq \omega \leq 3$ as the exponent of matrix multiplication – that is, we can multiply two matrices of size $n \times n$ in $O(n^\omega)$ ring operations. Since all ring operations over $\mathbb{K}[x]/\langle x^d \rangle$ are quasi-linear, this brings the cost of polynomial matrix multiplication of size $n \times n$ and degree $< d$ as $O\tilde{}(n^\omega d)$.

2.2 Linearly recurrent sequences over $\mathbb{K}$

While we focus on sequences over $\mathbb{A} = \mathbb{K}[x]/\langle x^d \rangle$, we begin by reviewing sequences with elements in $\mathbb{K}$ (hereafter referred to as scalar sequences). Note that properties of linearly

recurrent sequences over $\mathbb{A}^n$ also apply to scalar sequences since setting $d = n = 1$ will yield a scalar sequence.

A scalar sequence $\boldsymbol{s} = (S_0, S_1, \dots)$ is said to be linearly recurrent if it satisfies either

1. There exists $e \in \mathbb{Z}_{>0}$ and c_j's such that $c_e S_{i+e} - \sum_{j=0}^{e-1} c_j S_{i+j} = 0$, for all $i \geq 0$.
2. There exist polynomials P and N, with $\deg P > \deg N$, such that $S = \sum_{i \geq 0} S_i y^{-i-1} = \frac{N}{P}$.

For any coefficients c_j's that satisfies the first property, we define $\sum_{i=0}^{e} c_i y^i$ as the corresponding *canceling polynomial*. We also say a canceling polynomial is a *generating polynomial* if it is monic (since it describes a rule to generate $\boldsymbol{s}i$, $i \geq e$). Using the notion of canceling polynomials, we can relate the two properties: $S = \frac{N}{P}$, N and P polynomials, if and only if P is a canceling polynomial. The minimal possible degree of the canceling polynomials is called the *order* of the sequence.

Over fields, all canceling polynomials of a linearly recurrent sequence are multiples of the unique monic canceling polynomial of minimal degree since $\mathbb{K}[x]$ is a principal ideal domain; thus, the task of finding the annihilator can be reduced to finding a single polynomial. There are two main approaches for computing the minimal polynomial, which mirrors the two properties above. First, we can build a linear system and compute coefficients c_j's of the minimal polynomial, which corresponds to a nullspace element of this system. Second, we can rewrite $S = N/P$ as $PS - N = 0$ and find P (of minimal degree) along with a corresponding N that satifisies this equation.

We conclude this section through a concrete example for both approaches. Using the Fibonacci sequence $\boldsymbol{s}_F$ again, we first truncate this sequence to 4 terms, since it is known that the order is 2 (see Lemma 3 for details about partial sequences). First, to set up the system, we need c_0, c_1 such that

$$\begin{aligned} 2 + c_1 + c_0 &= 0 \\ 3 + 2c_1 + c_0 &= 0 \end{aligned}$$

or written in matrix form

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ 1 \end{bmatrix} = 0.$$

We can see that the resulting matrix has constant anti-diagonals (also known as a Hankel matrix), which can be solved in quasi-linear time via fast structured algorithms [52]. If we instead apply (structured) Gaussian elimination, this yields an algorithm close to the Berlekamp-Massey algorithm [7, 43].

On the other hand, we can solve for Q and N in $Q(1+y+2y^2+3y^3)-N=0 \bmod y^4$, which can be written in matrix form as

$$\begin{bmatrix}1+y+2y^2+3y^3 & -1\end{bmatrix}\begin{bmatrix}Q\\N\end{bmatrix}=0 \bmod y^4.$$

The basis of all pairs (Q,N) that satisfies the above can be computed via a right approximant basis of order 4 in quasi-linear time as well (see Section 2.6). For scalar sequences, the minimal approximant basis consists of two pairs (Q_1,N_1) and (Q_2,N_2); reversing the Q with the smaller degree gives us the minimal polynomial.

2.3 Linearly recurrent sequences over $\mathbb{K}[x]/\langle x^d\rangle$

We consider the set $\mathcal{S}=(\mathbb{A}^n)^{\mathbb{N}}$ of *(vector) sequences* over the ring $\mathbb{A}=\mathbb{K}[x]/\langle x^d\rangle$ for some $d\in\mathbb{Z}_{>0}$, that is, sequences $\boldsymbol{s}=(S_0,S_1,\ldots)$ with each S_k in $\mathbb{A}^n$. Such a sequence is said to be *linearly recurrent* if there exist $\gamma\in\mathbb{N}$ and $p_0,\ldots,p_\gamma\in\mathbb{A}$ with p_γ invertible such that

$$p_0S_k+\cdots+p_{\gamma-1}S_{k+\gamma-1}+p_\gamma S_{k+\gamma}=0 \text{ for all } k\geq 0; \tag{2.1}$$

the *order* of $\boldsymbol{s}$ is the smallest such γ, denoted by δ hereafter. A polynomial $p_0+\cdots+p_\gamma y^\gamma$ in $\mathbb{A}[y]$ is said to *cancel* $\boldsymbol{s}$ if $p_0,\ldots,p_\gamma$ satisfies Eq. (2.1) (without requiring that p_γ be invertible). The set of canceling polynomials forms an ideal $\mathrm{Ann}(\boldsymbol{s})$ in $\mathbb{A}[y]$, called the *annihilator* of $\boldsymbol{s}$. Thus $\boldsymbol{s}$ is linearly recurrent of order δ if and only if there is a monic polynomial of degree δ in $\mathrm{Ann}(\boldsymbol{s})$: such polynomials are called *generating polynomials* of $\boldsymbol{s}$. Unlike for sequences over fields, here there may be canceling polynomials of degree less than δ, which prevents uniqueness of generating polynomials; and there are sequences which are not linearly recurrent but still admit a nonzero canceling polynomial (i.e. $\mathrm{Ann}(\boldsymbol{s})\neq\{0\}$).

Example 1. *Consider $\mathbb{A}=\mathbb{K}[x]/\langle x^2\rangle$ and the sequence $\boldsymbol{s}=(1,1+x,1,1+x,1,1+x,\ldots)$ in $\mathbb{A}^{\mathbb{N}}$. Note that $x\boldsymbol{s}=(x,x,x,x,\ldots)$. This sequence has order $\delta=2$, a generating polynomial is y^2-1, and a canceling polynomial of degree less than 2 is $x(y-1)$. One can verify that $\mathrm{Ann}(\boldsymbol{s})=\langle y^2-1,x(y-1)\rangle$; in particular $y^2+x(y-1)-1$ is also a generating polynomial. For any sequence $\boldsymbol{s}$ in $\mathbb{K}^{\mathbb{N}}$ which is not linearly recurrent, the sequence $x\boldsymbol{s}$ in $\mathbb{A}^{\mathbb{N}}$ is not linearly recurrent but is canceled by x, i.e. $x\in\mathrm{Ann}(x\boldsymbol{s})\setminus\{0\}$.*

Like for sequences over fields, here canceling polynomials can be characterized as denominators of the (vector) generating series of the sequence, defined as $G = \sum_{k\geq 0} S_k y^{-k-1}$ in $(\mathbb{A}[[y^{-1}]])^n$. In what follows, the elements of $\mathbb{A}[y]^n$ are called polynomials, and for $g = (g_1, \ldots, g_n) \in \mathbb{A}[y]^n$ we define $\deg(g) = \max_{1\leq j\leq n} \deg(g_j)$.

Lemma 2. *Let $\boldsymbol{s} \in \mathcal{S}$, let G be its generating series, and let $p \in \mathbb{A}[y]$. Then, $p \in \mathrm{Ann}(\boldsymbol{s})$ if and only if the series $pG \in (\mathbb{A}[[y^{-1}]])^n$ is a polynomial, in which case $\deg(pG) < \deg(p)$.*

Proof. One has $pG = \sum_{0\leq j\leq \gamma} \sum_{k\geq -j} p_j S_{k+j} y^{-k-1}$, where $\gamma = \deg(p)$ and $p = p_0 + \cdots + p_\gamma y^\gamma$. Thus all terms of pG have degree less than γ, and pG is a polynomial if and only if its term in y^{-k-1} vanishes for all $k \geq 0$, i.e. if and only if Eq. (2.1) holds. □

2.4 Partial sequences

In this thesis, we want to compute a generating set for $\mathrm{Ann}(\boldsymbol{s})$, for a linearly recurrent $\boldsymbol{s} \in \mathcal{S}$, but for algorithms we typically only have access to a finite number of terms of the sequence. Suppose we have access to the partial sequence $\boldsymbol{s}_e = (S_0, \ldots, S_{e-1})$ in $\mathcal{S}_e = (\mathbb{A}^n)^e$, for some $e \in \mathbb{Z}_{>0}$. Similar to Eq. (2.1), a polynomial $p_0 + \cdots + p_\gamma y^\gamma$ of degree $\gamma < e$ cancels $\boldsymbol{s}_e$ if

$$p_0 S_k + \cdots + p_\gamma S_{k+\gamma} = 0 \text{ for all } 0 \leq k < e - \gamma. \tag{2.2}$$

Like for sequences over fields, here polynomials of degree γ which cancel $\boldsymbol{s}_e$ also cancel the whole sequence $\boldsymbol{s}$, provided the discrepancy between e and γ is sufficiently large (namely, $e \geq \gamma + \delta$).

Lemma 3. *Let $\boldsymbol{s} \in \mathcal{S}$ be linearly recurrent of order δ. For any $e \in \mathbb{Z}_{>0}$ and any $p \in \mathbb{A}[y]$ with $\deg(p) \leq e - \delta$, one has $p \in \mathrm{Ann}(\boldsymbol{s})$ if and only if p cancels $\boldsymbol{s}_e$.*

Proof. Obviously, any polynomial $p \in \mathrm{Ann}(\boldsymbol{s})$ also cancels $\boldsymbol{s}_e$, for any $e \in \mathbb{Z}_{>0}$ greater than the degree of p. Now let $p \in \mathbb{A}[y] \setminus \{0\}$ such that $\gamma = \deg(p) \leq e - \delta$ and p cancels $\boldsymbol{s}_e$. Since $e - \gamma \geq \delta$, Eq. (2.2) yields $\sum_{0\leq i\leq \gamma} p_i S_{k+i} = 0$ for $0 \leq k < \delta$. Furthermore, since $\boldsymbol{s}$ is linearly recurrent of order δ, there exists $y^\delta - \sum_{0\leq j<\delta-1} q_j y^j \in \mathbb{A}[y]$ which cancels $\boldsymbol{s}$, meaning that $S_{k+i} = \sum_{0\leq j<\delta} q_j S_{k-\delta+j+i}$ for any $k \geq \delta$. Therefore we get

$$\sum_{0\leq i\leq \gamma} p_i S_{k+i} = \sum_{0\leq j<\delta} q_j \sum_{0\leq i\leq \gamma} p_i S_{k-\delta+j+i}.$$

Using this identity, it follows by induction on $k \geq \delta$ that the relation $\sum_{0\leq i\leq \gamma} p_i S_{k+i} = 0$ also holds for all $k \geq \delta$. Hence $p \in \mathrm{Ann}(\boldsymbol{s})$. □

2.5 Bivariate interpretation and generating sets

Uni-dimensional sequences of vectors in $\mathbb{A}^n$ as above can be interpreted as two-dimensional sequences of vectors in $\mathbb{K}^n$, that is, sequences $\boldsymbol{\sigma} = (\zeta_{i,j})_{i,j\geq 0}$ in $\mathfrak{S} = (\mathbb{K}^n)^{\mathbb{N}^2}$. This is based on the natural injection $\varphi : \mathbb{A}[y] \to \mathbb{K}[\alpha, \beta]$ with $(\varphi(x), \varphi(y)) = (\alpha, \beta)$.

Here we recall from [55, 22] that a polynomial $q = \sum_{i,j} q_{ij}\alpha^i\beta^j$ in $\mathbb{K}[\alpha, \beta]$ is said to cancel a sequence $\boldsymbol{\sigma} = (\zeta_{i,j})_{i,j\geq 0} \in \mathfrak{S}$ if

$$\sum_{i,j\geq 0} q_{ij}\zeta_{i+k_1,j+k_2} = 0 \text{ for all } k_1, k_2 \geq 0.$$

Then, let $\boldsymbol{s} = (S_0, S_1, \ldots) \in \mathcal{S}$, and define $\boldsymbol{\sigma} = (\zeta_{i,j})_{i,j\geq 0} \in \mathfrak{S}$ such that $\zeta_{i,j} \in \mathbb{K}^n$ is the coefficient of degree $d - 1 - i$ of the truncated polynomial vector $S_j \in \mathbb{A}^n$ if $i < d$, and $\zeta_{i,j} = 0$ otherwise. Then, a polynomial $p \in \mathbb{A}[y]$ cancels $\boldsymbol{s}$ if and only if the polynomial $\varphi(p)$ cancels $\boldsymbol{\sigma}$. Furthermore, the set of polynomials in $\mathbb{K}[\alpha, \beta]$ which cancel $\boldsymbol{\sigma}$ is an ideal of $\mathbb{K}[\alpha, \beta]$ which contains α^d, and this ideal is zero-dimensional if and only if $\boldsymbol{s}$ is linearly recurrent.

In what follows, we define $\bar{\varphi}(\mathcal{I}) = \langle\{\varphi(p) \mid p \in \mathcal{I}\} \cup \{\alpha^d\}\rangle$ for any ideal $\mathcal{I}$ of $\mathbb{A}[y]$, providing a correspondence between the ideals of $\mathbb{A}[y]$ and those of $\mathbb{K}[\alpha, \beta]$ containing α^d. For insight into possible "nice" generating sets for $\mathrm{Ann}(\boldsymbol{s})$, we consider the lexicographic order $\preccurlyeq_{\mathrm{lex}}$ with $\alpha \preccurlyeq_{\mathrm{lex}} \beta$, and use the fact that Gröbner bases of the ideals in $\mathbb{K}[\alpha, \beta]$ for this order are well understood [40]. Below, unless mentioned otherwise, we use $\preccurlyeq_{\mathrm{lex}}$ when some term order is needed, e.g. for leading terms and Gröbner bases.

Gröbner bases, originally introduced by Buchberger [13], are generating sets of ideals over a multivariate polynomial ring $\mathbb{K}[y_1, \ldots, y_n]$ with particularly desirable properties. Formally, a generating set G of an ideal $I \subseteq \mathbb{K}[y_1, \ldots, y_n]$ is a Gröbner basis (wrt a monomial ordering) if G satisfies any of the following with respect to some monomial ordering:

1. the ideal generated by the leading terms of all elements of I is equal to the ideal generated by the leading terms of the elements in G
2. the leading term of any element in I is divisible by the leading term of some element in G
3. the result of multivariate division for any element in $\mathbb{K}[y_1, \ldots, y_n]$ by G is unique (regardless of the order of reduction by elements in G)

4. the result of multivariate division of any element of I by G results in a remainder of 0.

The last two properties allow us to define reduction of elements in $\mathbb{K}[y_1, \ldots, y_n]$ with I. For sequences, reduction of monomials of high powers by the annihilator forms the basis for fast algorithms for computing elements of linear sequences (via multivariate extensions of Fiduccia's algorithm [18]).

We conclude this section by giving a bound on the cardinality of the minimal Gröbner basis. Consider a zero-dimensional ideal $\mathcal{I}$ in $\mathbb{K}[\alpha, \beta]$ that contains a power of α and let $\mathcal{G}$ be its reduced Gröbner basis. Let

$$(\beta^{e_0}, \alpha^{d_1}\beta^{e_1}, \ldots, \alpha^{d_{t-1}}\beta^{e_{t-1}}, \alpha^{d_t})$$

be the leading terms of the elements of $\mathcal{G}$ listed in decreasing order, i.e. the e_i's are decreasing and the d_i's are increasing. We set $d_0 = e_t = 0$, and for $1 \leq i \leq t$ we set $\delta_i = d_i - d_{i-1}$, so that $d_i = \delta_1 + \cdots + \delta_i$. Similarly, for $0 \leq i < t$ we set $\varepsilon_i = e_i - e_{i+1}$. Then write $\mathcal{G} = \{g_0, \ldots, g_t\}$, with g_i having leading term $\alpha^{d_i}\beta^{e_i}$; in particular $g_t = \alpha^{d_t} = \alpha^{\delta_1+\cdots+\delta_t}$ and g_0 is monic in β.

Lazard's Theorem states the following [40]: for $0 \leq i \leq t$ one can write $g_i = \alpha^{d_i}\hat{g}_i$, with $\hat{g}_i$ monic of degree e_i in β. In addition, for $0 \leq i < t$, $\hat{g}_i = g_i/\alpha^{d_i}$ is in the ideal generated by

$$\langle \hat{g}_{i+1}, \alpha^{\delta_{i+2}}\hat{g}_{i+2}, \ldots, \alpha^{\delta_{i+2}+\cdots+\delta_t}\rangle = \left\langle \frac{g_{i+1}}{\alpha^{d_{i+1}}}, \frac{g_{i+2}}{\alpha^{d_{i+1}}}, \ldots, \frac{g_t}{\alpha^{d_{i+1}}} \right\rangle;$$

in particular, α^{δ_1} divides $g_1, \ldots, g_t$. Lazard also proved that a set of polynomials which satisfies these conditions is necessarily a minimal Gröbner basis.

With the above notation, a minimal Gröbner basis of $\mathcal{I}$ has cardinality $t + 1$, with $t \leq \min(e_0, d_t)$ since $0 = d_0 < d_1 < \cdots < d_t$ and $0 = e_t < \cdots < e_1 < e_0$. Since for the reduced Gröbner basis $\mathcal{G}$ each polynomial g_i is represented by at most $e_0 d_t$ coefficients in $\mathbb{K}$, the total size of $\mathcal{G}$ in terms of field elements is at most $e_0 d_t \min(e_0, d_t)$. Finer bounds for the cardinality and size of $\mathcal{G}$ could be given using the vector space dimension $\dim_{\mathbb{K}}(\mathbb{K}[\alpha, \beta]/\mathcal{I})$.

2.6 Univariate and bivariate approximations

For a univariate polynomial matrix $F \in \mathbb{K}[x]^{\mu\times\nu}$ and a positive integer d, we consider a free $\mathbb{K}[x]$-module of rank μ defined as

$$\mathcal{A}_d(F) = \{p \in \mathbb{K}[x]^{1\times\mu} \mid pF = 0 \bmod x^d\};$$

its elements are called *approximants for F at order d* [64, 3]. Bases of such submodules can be represented as $\mu \times \mu$ nonsingular matrices over $\mathbb{K}[x]$ and are usually computed in so-called *reduced* forms [67] or the corresponding canonical *Popov* forms [53]. Extensions of these forms have been defined to accommodate degree weights or degree constraints, and are called *shifted* reduced or Popov forms [64, 3, 5].

Given a vector $v \in \mathbb{K}[x]^{1\times m}$, the *pivot index* of v is the largest index j such that $\deg v_j = \deg v$, where the degree of a vector is the maximal degree of its entries. A matrix $M \in \mathbb{K}[x]^{n\times n}$ is said to be in (row-wise) *weak Popov form* if it has no zero row and the pivot indices are all distinct for each row. A weak Popov form is *ordered* if its pivot indices (arranged by the rows) are strictly increasing. Finally, M is in *Popov form* if it is in weak Popov form, the corresponding pivot entries are monic, and for each column, all entries have degree less than the corresponding pivot entry in that column (note that since we are only considering square matrices, each column must contain a pivot entry). A matrix in Popov form is canonical and have rows which have the minimal possible degrees. We can also define a column-wise Popov form by defining the pivot column-wise.

The algorithm PM-Basis [25] computes an approximant basis in shifted reduced form in time $O\tilde{}(\mu^{\omega-1}(\mu+\nu)d)$; using essentially two calls to this algorithm, one recovers the unique approximant basis in shifted Popov form within the same cost bound [33].

More generally, in the bivariate case with $F \in \mathbb{K}[\alpha,\beta]^{\mu\times\nu}$ and $(d,e) \in \mathbb{Z}_{>0}$, the set

$$\mathcal{A}_{d,e}(F) = \{p \in \mathbb{K}[\alpha,\beta]^{1\times\mu} \mid pF = 0 \bmod (\alpha^d, \beta^e)\}$$

is a $\mathbb{K}[\alpha,\beta]$-submodule of $\mathbb{K}[\alpha,\beta]^{1\times\mu}$ whose elements are called *approximants for F at order (d,e)*. Such submodules are usually represented by a $\preccurlyeq$-Gröbner basis for some term order $\preccurlyeq$ on $\mathbb{K}[\alpha,\beta]^{1\times\mu}$; for definitions of term orders and Gröbner bases for submodules we refer to [15]. For $\nu \leq \mu$ algorithms based on an iterative approach or on efficient linear algebra yield cost bounds in $O\tilde{}(\mu(\nu de)^2+(\nu de)^3)$ and $O\tilde{}(\mu(\nu de)^{\omega-1}+(\nu de)^\omega)$ operations in $\mathbb{K}$ respectively [21, 50], whereas a recent divide and conquer approach costs $O\tilde{}((M^\omega + M^2\nu)de)$, where $M = \mu\min(d,e)$ [47, Prop. 5.5]; in these cases the output is a minimal Gröbner basis.

Chapter 3

Kurakin's Algorithm

3.1 Kurakin's algorithm over $\mathbb{A}$

In [37], Kurakin gives an algorithm based on the Berlekamp-Massey algorithm that computes the annihilators of a partial sequence over a ring R (and modules over R) that can be decomposed as a disjoint union $R = \{0\} \cup R_0 \cup \cdots \cup R_{d-1}$ where

$$R_i = \{r_i r^* \mid r^* \in R \text{ invertible}\} \text{ for some } r_i \in R.$$

In this paper we consider $R = \mathbb{A} = \mathbb{K}[x]/\langle x^d \rangle$; in this case the canonical choice is $r_i = x^i$, with

$$R_i = \{x^i p^* \mid p^* \in \mathbb{A} \text{ with nonzero constant term}\}.$$

Consider a partial sequence $\boldsymbol{s}_e \in \mathcal{S}_e$ of a linearly recurrent $\boldsymbol{s} \in \mathcal{S}$ of order δ. Kurakin's algorithm computes d polynomials $P_i \in \mathbb{A}[y]$, $i = 0, \ldots, d-1$, such that P_i is a canceling polynomial of $\boldsymbol{s}_e$ that has leading coefficient x^i and is minimal in degree among all canceling polynomials with leading coefficient x^i. Furthermore, one has $\mathrm{Ann}(\boldsymbol{s}) = \langle P_0, \ldots, P_{d-1} \rangle$ provided $e \geq 2\delta$ [38, Thm. 1].

We first define three operations on sequences. Given a partial sequence $\boldsymbol{s}_e$ and $c \in \mathbb{A}$, $c \cdot \boldsymbol{s}_e$ denotes multiplying c to every element in $\boldsymbol{s}_e$, while $y^j \cdot \boldsymbol{s}_e$ denotes a shift of j elements — that is, removing the first j elements. Given another partial sequence $\hat{\boldsymbol{s}}_{\hat{e}}$, the sum $\boldsymbol{s}_e + \hat{\boldsymbol{s}}_{\hat{e}}$ returns the first $\min(e, \hat{e})$ elements of the two sequences added together element-wise.

Kurakin's algorithm iterates on $s = 0, \ldots, e-1$, keeping track of polynomials $P_{i,s}$ as well as partial sequences $\boldsymbol{s}_{e,i,s} = P_{i,s} \cdot \boldsymbol{s}_e = \sum_{j=0}^{e-s} P_{i,s}[j] \cdot y^j \cdot \boldsymbol{s}_e$, where $P_{i,s}[j]$ is the j-th

coefficient of $P_{i,s}$. An invariant is that the leading coefficient of $P_{i,s}$ is x^i for all s. For each $s = 0, \ldots, e-1$, the algorithm essentially attempts to either create a zero by using the partial sequences from previous iterations with equal number of leading zeros (similar to Gaussian elimination), or shift the sequence if we cannot cancel this element.

At each iteration s, let $\mathcal{I}[k]$ be the $\mathbb{A}$-submodule of $\mathbb{A}^n$ generated by the elements $\boldsymbol{s}_{e,i,s'}[k]$ for all $i = 0, \ldots, d-1$ and $s' < s$ such that $\boldsymbol{s}_{e,i,s'}$ has k leading zeros. Furthermore, let $\mathcal{P}[k, j]$ and $\mathcal{S}[k, j]$ be the corresponding polynomial and partial sequence to the j-th element in the basis of $\mathcal{I}[k]$, $\mathcal{I}[k, j]$. At iteration s, if $\boldsymbol{s}_{e,i,s}$ has k leading zeros and $\boldsymbol{s}_{e,i,s}[k] \in \mathcal{I}[k]$, then we can find coefficients such that $\boldsymbol{s}_{e,i,s}[k] - \sum_j c_j \mathcal{I}[k, j] = 0$ and $\boldsymbol{s}_{e,i,s} - \sum_j c_j \mathcal{S}[k, j]$ results in a sequence with at least $k+1$ zeros since both sequences had k leading zeros and we canceled $\boldsymbol{s}_{e,i,s}[k]$. The algorithm terminates when all $\boldsymbol{s}_{e,i,s} = 0$ (see Algorithm 1).

We track the subiterations by the index t for analysis; this does not play a role in the algorithm. Kurakin shows that the total number of subiterations across all s is $O(e)$ per polynomial, bringing the total to $O(ed)$ ([37, Thm. 2]). However, the analysis of the runtime in [37] treats all ring operations (including computing solution to line 12 of Algorithm 1) as constant time operations, which is unrealistic over $\mathbb{A}^n$. Thus, we will give a cost analysis in terms of number of field operations over $\mathbb{K}$.

We note that, since $\mathbb{A}^n$ is a free $\mathbb{K}[x]$-module of rank n (with a basis given by the canonical vectors of length n) and $\mathbb{K}[x]$ is a principal ideal domain, any of its $\mathbb{K}[x]$-submodule is free of rank at most n. As a consequence, the number of generators of $\mathcal{I}[k]$ is at most n. This will allow us to bound the cost for solving submodule membership as well as the equation $\boldsymbol{s}^{(t)}_{e,s,i}[k] - \sum_j c_j \mathcal{I}[k, j] = 0$ by finding the right approximant basis of

$$F = \begin{bmatrix} \mathcal{I}[k, 0] & \cdots & \mathcal{I}[k, n-1] & s_{e,s,i}[k] \end{bmatrix}.$$

The following lemma shows that the same approximant basis can be used to reduce $\mathcal{I}$.

Lemma 4. *Let $u_1, \ldots, u_{n+1} \in \mathbb{K}[x]^n$, then there exists a non-trivial solution to $\sum c_i u_i = 0$, $c_i \in \mathbb{K}[x]$. Furthermore, there exists a solution such that at least one c_i has a non-zero constant term.*

Proof. Working over the field of fractions over $\mathbb{K}[x]$, there must be a solution to $\sum d_i u_i = 0$ since there are $n+1$ vectors of size n; this implies that there exists a non-trivial solution to $\sum c_i u_i = 0$ such that no c_i is a fraction by multiplying both sides of $\sum d_i u_i = 0$ by the least common multiple of the d_i's. For the second part of the statement, suppose that all c_i's have zero constant terms and let j be the lowest valuation of the c_i's. Then we have that $\sum c_i u_i = x^j (\sum c'_i u_i) = 0$, such that $c'_i \in \mathbb{K}[x]$ and some c_i has a nonzero constant term. □

Algorithm 1 KURAKIN($\boldsymbol{s}_e$)

Input: partial sequence $\boldsymbol{s}_e$

Output: minimal canceling polynomials of $\boldsymbol{s}_e$

1: **for** $i = 0, \ldots, d-1$ **do**
2: set $P_{i,0} = x^i$ and $\boldsymbol{s}_{e,i,0} = x^i \boldsymbol{s}_e$
3: set k to be index of first non-zero element of $\boldsymbol{s}_{e,i,0}$
4: **if** $\boldsymbol{s}_{e,i,0}[k] \neq 0$ **then**
5: add $\boldsymbol{s}_{e,i,0}[k], P_{i,0}, \boldsymbol{s}_{e,i,0}$ to $\mathcal{I}[k], \mathcal{P}[k], \mathcal{S}[k]$ resp.
6: **for** $s = 1, \ldots, e-1$ **do**
7: **for** $i = 0, \ldots d-1$ **do**
8: set $t = 0$; $P_{i,s}^{(t)} = yP_{i,s-1}$; and shift $\boldsymbol{s}_{e,i,s}^{(t)} = y \cdot \boldsymbol{s}_{e,i,s-1}$
9: **if** $\boldsymbol{s}_{e,i,s}^{(t)} = 0$ **then** continue to next i
10: set k to be the first non-zero index of $\boldsymbol{s}_{e,i,s}^{(t)}$
11: **if** $\boldsymbol{s}_{e,i,s}^{(t)}[k] \notin \mathcal{I}[k]$ **then** continue to next i
12: solve for c_j's such that $\boldsymbol{s}_{e,s,i}^{(t)}[k] - \sum_j c_j \mathcal{I}[k,j] = 0$
13: set $\boldsymbol{s}_{e,i,s}^{(t+1)} = \boldsymbol{s}_{e,i,s}^{(t)} - \sum_j c_j \mathcal{S}[k,j]$
14: set $P_{i,s}^{(t+1)} = P_{i,s}^{(t)} - \sum_j c_j \mathcal{P}[k,j]$
15: go to line 9 with $t = t+1$
16: **for** $i = 0, \ldots, d-1$ **do**
17: set $s_{e,i,s} = s_{e,i,s}^{(t)}$ and $P_{i,s} = P_{i,s}^{(t)}$
18: set k to be the index of first non-zero element of $s_{e,i,s}$
19: **if** $s_{e,i,s}[k] \notin \mathcal{I}[k]$ **then**
20: add $\boldsymbol{s}_{e,i,s}[k], P_{i,s}, \boldsymbol{s}_{e,i,s}$ to $\mathcal{I}[k], \mathcal{P}[k], \mathcal{S}[k]$ resp.
21: reduce the basis of $\mathcal{I}[k]$ if needed
22: **for** $i = 0, \ldots, d-1$ **do**
23: return $P_{i,s}$ that makes $\boldsymbol{s}_{e,i,s} = 0$ for the first time

The above lemma implies that if F has $n+1$ columns, then there is a column in the right approximant basis such that at least one entry has a nonzero constant term (and thusly invertible in $\mathbb{A}$); thus, we can always reduce F to have at most n columns by removing the corresponding element from $\mathcal{I}[k]$. Since F has n rows and at most $n+1$ columns, we can compute this in cost $O^{\sim}(n^{\omega}d)$ [33]. At lines 13 and 14, $S[k,j]$ and $P[k,j]$ have length and degree at most e resp., making the cost of these lines $O^{\sim}(n(ned)) = O^{\sim}(n^2ed)$. Finally, using the fact that the total number of subiterations is bounded by $O(ed)$, we arrive at the total cost $O^{\sim}(ed(n^2ed + n^{\omega}d))$.

We conclude by showing that the output of Algorithm 1 is indeed a basis of $\mathrm{Ann}(\boldsymbol{s})$ and that it forms a lexicographical Gröbner basis.

Theorem 5. *For each $i \in \{0, \ldots, d-1\}$, let P_i be a canceling polynomial of $\boldsymbol{s}$ with leading coefficient x^i that is minimal in degree among all polynomials with leading coefficient x^i. Then one has $\mathrm{Ann}(\boldsymbol{s}) = \langle P_0, \ldots, P_{d-1}\rangle$. Furthermore, $\{\varphi(P_0), \cdots, \varphi(P_{d-1}), \alpha^d\}$ forms a Gröbner basis of $\bar{\varphi}(\mathrm{Ann}(\boldsymbol{s}))$ with respect to the lexicographic term order with $\alpha \preccurlyeq_{\mathrm{lex}} \beta$.*

Proof. Suppose that there exists some $Q \in \mathbb{A}[y]$ with leading coefficient x^t that is in $\mathrm{Ann}(\boldsymbol{s})$ but $Q \notin \langle P_0, \ldots, P_{d-1}\rangle$. Note that for any polynomial in $\mathbb{A}[y]$, we can always make the leading coefficient to be some x^t by pulling out the minimal power of x from the leading coefficient and multiplying by its inverse. Now, since we assumed minimality of degrees for P_i's, $\deg(Q) > \deg(P_t)$ and $Q' = Q - y^{\deg Q - \deg P_t}P_t \in \mathrm{Ann}(\boldsymbol{s})$ has degree less than Q. By normalizing the leading coefficient of Q' to be some $x^{t'}$, we can repeat the same process and keep decreasing the degree. This process must terminate when we encounter some Q' with leading coefficient $x^{t'}$ such that $\deg Q' < \deg P_{t'}$, or $Q' = 0$. Both cases lead to contradictions; thus, such Q cannot exist and $\mathrm{Ann}(\boldsymbol{s}) = \langle P_0, \ldots, P_{d-1}\rangle$.

Next, let $\mathcal{G} = \{g_0, \ldots, g_k\}$, $g_i \in \mathbb{K}[\alpha, \beta]$ with leading coefficient x^{d_i}, be the minimal reduced (lexicographic) Gröbner basis of $\bar{\varphi}(\mathrm{Ann}(\boldsymbol{s}))$. We can turn $\mathcal{G}$ into another non-minimal Gröbner basis by adding the polynomials $a^c g_i$, for $c = 1, \ldots, d_{i+1} - 1$; we define the resulting basis as $\mathcal{G}' = \{g'_0, \cdots, g'_d\}$, with $g'_d = \alpha^d$ and each g'_i has leading term $\alpha^i\beta^{r_i}$. Furthermore, define u_i as the degree of P_i such that $\varphi(P_i)$ has leading term $\alpha^i\beta^{u_i}$.

For $i = 0, \ldots, d$, we have that $u_i \geq r_i$, otherwise $\mathcal{G}'$ would not reduce $\varphi(P_i)$ to zero, which $\mathcal{G}'$ must since $\varphi(P_i) \in \bar{\varphi}(\mathrm{Ann}(\boldsymbol{s}))$. We also have that $u_i \leq r_i$ due to the assumed minimality of degree for P_i's. Thus, the leading terms of $\{\varphi(P_0), \ldots, \varphi(P_{d-1}), \alpha^d\}$ generate the leading terms of $\bar{\varphi}(\mathrm{Ann}(\boldsymbol{s}))$. □

3.2 Lazy algorithm based on Kurakin's

Kurakin's algorithm requires that we keep track of all d possible generators, regardless of the actual number of generators needed. For example, consider $\boldsymbol{s} = (1, 1, 2, 3, 5, \dots) \in \mathbb{A}^{\mathbb{N}}$ with $\mathrm{Ann}(\boldsymbol{s}) = \langle y^2 - y - 1 \rangle$: Kurakin's algorithm returns $\{x^i(y^2 - y - 1), 0 \leq i < d\}$. In this section, we outline a modified version of Kurakin's algorithm that attempts to avoid as many extraneous computations as possible.

In the previous example, we can see that the polynomials associated with x^i, $i \geq 1$, were not useful. The next definition aims to qualify precisely the usefulness of the monomial x^i.

Definition 6. *Let $P_{i,s}$ and $\boldsymbol{s}_{e,i,s}$ be the polynomial and sequence at the end of step s associated with monomial x^i. A monomial x^{i_2} is* useful *wrt to x^{i_1}, $i_1 < i_2$, at step s if at least one of two conditions is true at the end of s:*

U1. $P_{i_2,s} \neq x^{i_2 - i_1} P_{i_1,s}$

U2. let k_{i_1} and k_{i_2} be the index of the first non-zero element of $\boldsymbol{s}_{e,i_1,s}$ and $\boldsymbol{s}_{e,i_2,s}$ resp., then $k_{i_1} \neq k_{i_2}$

Suppose a monomial x^{i_2} is not useful wrt x^{i_1} at step s, then by negating condition U1, we have $P_{i_2,s} = x^{i_2 - i_1} P_{i_1,s}$. Due to negation of U2, $\boldsymbol{s}_{e,i_2,s}$ is the zero sequence if and only if $\boldsymbol{s}_{e,i_1,s}$ is the zero sequence; so either we return $P_{i_2,s} = x^{t_2 - t_1} P_{i_1,s}$ or we do not terminate at this step for both monomials. Finally, since $k_{i_1} = k_{i_2}$ and $\boldsymbol{s}_{e,i_2,s} = x^{i_2 - i_1} \boldsymbol{s}_{e,i_1,s}$, we always have that $\boldsymbol{s}_{e,i_2,s}[k_{i_2}] = x^{i_2 - i_1} \boldsymbol{s}_{e,i_1,s}[k_{i_1}] \in (\langle \boldsymbol{s}_{e,i_1,s}[k_{i_1}] \rangle \cup \mathcal{I}[k_{i_1}])$, meaning we can safely ignore $\boldsymbol{s}_{e,i_2,s}[k_{i_2}]$ when updating $\mathcal{I}[k_{i_2}]$ at the end of step s. Thus, the negation of usefulness conditions U1 and U2 implies that any computation associated with x^{i_2} is not needed at step s.

However, as defined, U1 and U2 do not impose any conditions about the subiterations (indexed by t). The next lemma gives a different characterization of the usefulness conditions in terms of t.

Lemma 7. *If x^{i_2} is useful wrt to x^{i_1} at some step s, then at some subiteration t of step s, one of u1, u2, u3 is true at the start of t:*

u1. $P^{(t)}_{i_2,s} \neq x^{i_2 - i_1} P^{(t)}_{i_1,s}$

u2. if $P^{(t)}_{i_2,s} = x^{i_2 - i_1} P^{(t)}_{i_1,s}$, *then* $k^{(t)}_{i_2} \neq k^{(t)}_{i_1}$

u3. if $P^{(t)}_{i_2,s} = x^{i_2-i_1}P^{(t)}_{i_1,s}$ *and* $k^{(t)}_{i_2} = k^{(t)}_{i_1}$*, then* $\boldsymbol{s}^{(t)}_{e,i_1,s}[k^{(t)}_{i_1}] \notin \mathcal{I}[k^{(t)}_{i_1}]$ *and* $s^{(t)}_{e,i_2,s}[k^{(t)}_{i_1}] \in \mathcal{I}[k^{(t)}_{i_1}]$

Proof. We prove that if u1, u2, and u3 are false for every subiteration t and s, then U1 and U2 are false for x^{i_2} wrt x^{i_1}. Suppose the conditions u1, u2, and u3 are all false for every subiteration t at s. The negation of u1 forces $P^{(t)}_{i_2,s} = x^{i_2-i_1}P^{(t)}_{i_1,s}$ at the start of t, which sets the hypothesis of u2 true, implying $k^{(t)}_{i_2} = k^{(t)}_{i_1}$. Finally, since the hypothesis of u3 holds, we must have $s^{(t)}_{e,i_1,s}[k^{(t)}_{i_1}] \in \mathcal{I}[k^{(t)}_{i_1}]$ or $s^{(t)}_{e,i_2,s}[k^{(t)}_{i_1}] \notin \mathcal{I}[k^{(t)}_{i_1}]$. The two are mutually exclusive since $\boldsymbol{s}^{(t)}_{e,i_2,s} = x^{i_2-i_1}\boldsymbol{s}^{(t)}_{e,i_1,s}$, if $s^{(t)}_{e,i_1,s}[k^{(t)}_{i_1}] \in \mathcal{I}[k^{(t)}_{i_1}]$, then $s^{(t)}_{e,i_2,s}[k^{(t)}_{i_1}] \in \mathcal{I}[k^{(t)}_{i_1}]$. When $s^{(t)}_{e,i_1,s}[k^{(t)}_{i_1}] \in \mathcal{I}[k^{(t)}_{i_1}]$, we can update

$$P^{(t+1)}_{i_1,s} = P^{(t)}_{i_1,s} - \sum c_j \mathcal{I}[k^{(t)}_{i_1}, j]$$
$$P^{(t+1)}_{i_2,s} = x^{i_2-i_1}P^{(t)}_{i_1,s} - x^{i_2-i_1}\sum c_j \mathcal{P}[k^{(t)}_{i_1}, j] = x^{i_2-i_1}P^{(t+1)}_{i_1,s},$$

which was already implied by the assumption that u1 is false for all t. On the other hand, when $s^{(t)}_{e,i_2,s}[k^{(t)}_{i_1}] \notin \mathcal{I}[k^{(t)}_{i_1}]$, we also have $\boldsymbol{s}^{(t)}_{e,i_1,s}[k^{(t)}_{i_1}] \notin \mathcal{I}[k^{(t)}_{i_1}]$, so the subiterations terminate and we must have $P_{i_2,s} = x^{i_2-i_1}P_{i_1,s}$ with $k_{i_2} = k_{i_1}$. This implies U1 and U2 also do not hold for step s. □

While the converse is not true, we say a monomial x^{i_2} is *potentially useful* wrt x^{i_1} when at some step s and subiteration t, at least one of the conditions u1, u2, and u3 holds. Using this, we can construct a Lazy variant of Kurakin's algorithm (see Algorithm 2). While most of the algorithm remains the same as Algorithm 1, we only iterate through the potentially useful monomials, rather than $i = 0, \ldots, d-1$.

Proposition 8. *The output Algorithm 2 is an equivalent ideal to Algorithm 1 and costs* $\tilde{O}(ed^*(n^2ed + n^\omega d))$*, where* $d^* \leq d$ *is the number of potentially useful monomials.*

Proof. At each subiteration, we check to see if there exists $i' > i, i' \notin \mathcal{U}$ such that $x^{i'}$ satisfies one of u2 or u3, and add the smallest such i' to $\mathcal{U}$. Note that we need not check u1 since if u1 holds, then either u2 or u3 must have been true at some previous subiteration, thus i' is already included in $\mathcal{U}$. Condition u2 can be checked in $O(n)$ by checking the valuations of all entries in $\boldsymbol{s}_{e,i,s}[k]$ at lines 4 and 10. Condition u3 can be checked in $O(\log d)$ membership computations via a binary search to find the minimal i' such that $x^{i'-i}\boldsymbol{s}_{e,i,s}[k] \in \mathcal{I}[k]$ when $\boldsymbol{s}_{e,i,s}[k] \notin \mathcal{I}[k]$ on line 11. Thus, the complexity for the subiterations do not change in terms of $\tilde{O}(\cdot)$. Defining $d^* = |\mathcal{U}| \leq d$, this brings the total cost to $\tilde{O}(ed^*(n^2ed + n^\omega d))$. Since we only omit monomials that never satisfy any conditions u1 through u3, by the contrapositive of Lemma 7, we only omit monomials that are not useful. □

Algorithm 2 LazyKurakin($\boldsymbol{s}_e$)

Input: partial sequence $\boldsymbol{s}_e$

Output: minimal canceling polynomials of $\boldsymbol{s}_e$

1: set $\mathcal{U} = \{1\}$ and $\mathcal{U}_0 = \mathcal{U}$
2: **while** $\mathcal{U}_0$ not empty **do**
3: set $i = \mathcal{U}_0[0]$ and pop $\mathcal{U}_0[0]$
4: set $P_i = x^i$ and $\boldsymbol{s}_{e,i,0} = x^i \boldsymbol{s}_e$
5: set k to be index of first non-zero element of $\boldsymbol{s}_{e,i,0}$
6: add $\boldsymbol{s}_{e,i,0}[k], P_{i,0}, \boldsymbol{s}_{e,i,0}$ to $\mathcal{I}[k], \mathcal{P}[k], \mathcal{S}[k]$ resp.
7: **if** there exists i' such that $i' + i < d$ and $x^{i'} \boldsymbol{s}_{e,i,0}[k] = 0$ **then**
8: add $i' + i$ to $\mathcal{U}$ and $\mathcal{U}_0$
9: **for** $s = 1, \ldots, e-1$ **do**
10: $\mathcal{U}_s = \mathcal{U}$
11: **while** $\mathcal{U}_s$ is not empty **do**
12: set $i = \mathcal{U}_s[0]$ and pop $\mathcal{U}_s[0]$
13: set $t = 0$; $P_{i,s}^{(t)} = yP_{i,s-1}$; and shift $\boldsymbol{s}_{e,i,s}^{(t)} = y \cdot \boldsymbol{s}_{e,i,s-1}$
14: **if** $\boldsymbol{s}_{e,i,s}^{(t)} = 0$ **then** continue to next i
15: set k to be the first non-zero index of $\boldsymbol{s}_{e,i,s}^{(t)}$
16: **if** $\boldsymbol{s}_{e,i,s}^{(t)}[k] \notin \mathcal{I}[k]$ **then**
17: **if** there exists i' such that $i' + i < d$ and $x^{i'} \boldsymbol{s}_{e,i,s}^{(t)}[k] \in \mathcal{I}[k]$ **then**
18: set $P_{e,i'+i,s}^{0} = x^{i'} P_{e,i,s}$ and $\boldsymbol{s}_{e,i'+i,s}^{0} = x^{i'} \boldsymbol{s}_{e,i,s}$
19: add $i' + i$ to $\mathcal{U}$ and $\mathcal{U}_s$
20: continue to next i
21: solve for c_j's such that $\boldsymbol{s}_{e,s,i}^{(t)}[k] - \sum_j c_j \mathcal{I}[k,j] = 0$
22: set $\boldsymbol{s}_{e,i,s}^{(t+1)} = \boldsymbol{s}_{e,i,s}^{(t)} - \sum_j c_j \mathcal{S}[k,j]$
23: set $P_{i,s}^{(t+1)} = P_{i,s}^{(t)} - \sum_j c_j \mathcal{P}[k,j]$
24: go to line 14 with $t = t + 1$
25: **for** $i \in \mathcal{U}$ **do**
26: set $s_{e,i,s} = s_{e,i,s}^{(t)}$ and $P_{i,s} = P_{i,s}^{(t)}$
27: set k to be the index of first non-zero element of $s_{e,i,s}$
28: **if** $s_{e,i,s}[k] \notin \mathcal{I}[k]$ **then**
29: add $\boldsymbol{s}_{e,i,s}[k], P_{i,s}, \boldsymbol{s}_{e,i,s}$ to $\mathcal{I}[k], \mathcal{P}[k], \mathcal{S}[k]$ resp.
30: reduce the basis of $\mathcal{I}[k]$ if needed
31: **for** $i = 0, \ldots, d-1$ **do**
32: return $P_{i,s}$ that makes $\boldsymbol{s}_{e,i,s} = 0$ for the first time

While we do not know how far d^* is from the number d_{opt} of polynomials in the minimal lexicographic Gröbner basis of $\bar{\varphi}(\text{Ann}(\boldsymbol{s}))$, we have observed empirically that d^* is often equal or close to d_{opt} (see Section 5).

3.3 Example of Kurakin and Lazy Kurakin

In order to highlight the different between Kurakin's algorithm and our lazy variant, we will run through a small example with $d = 3$ and input (truncated) sequence

$$\boldsymbol{s}_4 = (1 + x + x^2, 1 + 2x + 2x^2, 1 + 3x + 3x^2, 1 + 4x + 4x^2)$$

whose annihilator is $\boldsymbol{s}$ is $\langle y^2 - 2y + 1, x^2(y - 1)\rangle$. Kurakin's algorithm first initializes $P_{i,0} = x^i$ and since $x^i\boldsymbol{s}$ does not have a leading zero for all $i = 0, 1, 2$, we initialize only $\mathcal{I}[0] = \langle 1 + x + x^2\rangle, \mathcal{P}[0] = [1], \mathcal{S}[0] = [\boldsymbol{s}_{4,0,0}]$.

At $s = 1$, $i = 0$, $t = 0$:

- $\boldsymbol{s}^0_{4,0,1} = (1 + 2x + 2x^3, 1 + 3x + 3x^2, 1 + 4x + 4x^2)$
- $P^0_{0,1} = y$

Since $1 + 2x + 2x^2 \notin \mathcal{I}[0]$, we move to next i.

At $s = 1$, $i = 1$, $t = 0$

- $\boldsymbol{s}^0_{4,1,1} = (x + 2x^2, x + 3x^2, x + 4x^2)$
- $P^0_{1,1} = xy$

Since $x + 2x^2 \notin \mathcal{I}[0]$, we move to next i.

At $s = 1$, $i = 2$, $t = 0$

- $\boldsymbol{s}^0_{4,2,1} = (x^2, x^2, x^2)$
- $P^0_{1,1} = x^2y$

Now, $x^2 \in \mathcal{I}[0]$, with solution $x^2 - x^2\mathcal{I}[0][0] = 0$. The update $\boldsymbol{s}^0_{4,2,1} - x^2\mathcal{S}[0][0] = 0$, so $P_2 = x^2 - x^2\mathcal{P}[0][0] = x^2 - x^2 \cdot 1 = x^2(y-1)$.

We now move on to the second part of the iteration (on s) to update $\mathcal{I}$. Since $1 + 2x + 2x^2 \notin \mathcal{I}[0]$, we add it to $\mathcal{I}[0]$. Now, $x + 2x^2$ is in the updated $\mathcal{I}[0]$ and since there are no other non-zero leading terms, we move to the next iteration.

At $s = 2$, $i = 0$, $t = 0$:

- $\boldsymbol{s}^0_{4,0,2} = (1 + 3x + 3x^2, 1 + 4x + 4x^2)$
- $P^0_{0,1} = y^2$

We have $1 + 3x + 3x^2 \in \mathcal{I}[0]$ with $(1 + 3x + 3x^2) - 2\mathcal{I}[0][1] + \mathcal{I}[0][0] = 0$. Now, $\boldsymbol{s}^0_{4,0,2} - 2\mathcal{S}[0][1] + \mathcal{S}[0][0] = 0$ so $P_0 = y^2 - 2\mathcal{P}[0][1] + \mathcal{P}[0][0] = y^2 - 2y + 1$.

At $s = 2$, $i = 1$, $t = 0$:

- $\boldsymbol{s}^0_{4,1,2} = (x + 3x^2, x + 4x^2)$
- $P^0_{1,1} = xy^2$

We have (from the solution above) $(x + 3^2) - 2x\mathcal{I}[0][1] + x\mathcal{I}[0][0] = 0$; thus, $P_1 = xP_0$.

Since all P_i's are canceling polynomials, we terminate the algorithm. During the iterations, we can see that all the computations we do for $i = 1$ could have been deduced by computations for $i = 0$. Next, we will see how our lazy variant of Kurakin's algorithm attempts to avoid these extraneous computations. We begin by setting the set of potentially useful monomials $\mathcal{U}$ to be empty. Now, since $\boldsymbol{s}_e[0]$ has a nonzero constant, we only add 1 to $\mathcal{U}$. As before, we also initialize $\mathcal{I}[0] = \langle 1 + x + x^2 \rangle, \mathcal{P}[0] = [1], \mathcal{S}[0] = [\boldsymbol{s}_{4,0,0}]$.

At $s = 1$, $\mathcal{U} = [1]$. We start by popping 1:

- $\boldsymbol{s}^0_{4,0,1} = (1 + 2x + 2x^3, 1 + 3x + 3x^2, 1 + 4x + 4x^2)$
- $P^0_{0,1} = y$

Now, $1 + 2x + 2x^3 \notin \mathcal{I}[0]$, but $x^2(1 + 2x + 2x^3) \in \mathcal{I}[0]$, so we add x^2 to $\mathcal{U}$ with $P_{2,0} = x^2 P_{0,0}$.

At $s = 1$, $\mathcal{U} = [x^2]$. We pop x^2:

- $\boldsymbol{s}^0_{4,2,1} = (x^2, x^2, x^2)$

- $P^0_{1,1} = x^2 y$

We have $x^2 \in \mathcal{I}$ and $P_2 = x^2(y-1)$ as above.

Now, $\mathcal{U}$ is empty, so we move to the second part of the iteration. Again, since $1 + 2x + 2x^2 \notin \mathcal{I}[0]$, we add it to $\mathcal{I}[0]$. Finally, we populate $\mathcal{U} = [1, x^2]$ with all the potentially useful monomials we found before.

At $s = 2$, $\mathcal{U} = [1, x^2]$. We pop 1:

- $\boldsymbol{s}^0_{4,0,2} = (1 + 3x + 3x^2, 1 + 4x + 4x^2)$
- $P^0_{0,1} = y^2$

We have $1 + 3x + 3x^2 \in \mathcal{I}[0]$ with $(1 + 3x + 3x^2) - 2\mathcal{I}[0][1] + \mathcal{I}[0][0] = 0$. Now, $\boldsymbol{s}^0_{4,0,2} - 2\mathcal{S}[0][1] + \mathcal{S}[0][0] = 0$ so $P_0 = y^2 - 2\mathcal{P}[0][1] + \mathcal{P}[0][0] = y^2 - 2y + 1$.

At this point, we can terminate the algorithm since we have already found a canceling polynomial that leads with x^2. We can see that we did not do any computations involving x. Indeed, in the output of Kurakin's algorithm, we had $P_1 = xP_0$, so P_1 was redundant; thus, we have found an equivalent set of generators for the annihilator of $\boldsymbol{s}_e$.

Chapter 4

Approximation Approaches

4.1 Via Univariate Approximant Bases

Extending the classical theory of linearly recurrent sequences over the field $\mathbb{K}$, another approach is to consider the left kernel of the block-Hankel matrix

$$H_{\boldsymbol{s},e} = \begin{bmatrix} S_0 & S_1 & \cdots & S_{e-1} \\ S_1 & S_2 & \iddots & S_e \\ \vdots & \iddots & \iddots & \vdots \\ S_e & S_{e+1} & \cdots & S_{2e-1} \end{bmatrix} \in \mathbb{A}^{(e+1)\times(en)}.$$

Indeed, if e is large enough, vectors in this kernel represent polynomials which cancel $\boldsymbol{s}$, and which even generate all of $\mathrm{Ann}(\boldsymbol{s})$.

Lemma 9. *Let $\boldsymbol{s} \in \mathcal{S}$ be linearly recurrent of order δ, and define*

$$\mathcal{K}_{\boldsymbol{s},e} = \{p = p_0 + \cdots + p_e y^e \in \mathbb{A}[y] \mid [p_0 \ \cdots \ p_e] H_{\boldsymbol{s},e} = 0\}$$

for $e \in \mathbb{N}$. Assume $e \geq \delta$. Then $\mathcal{K}_{\boldsymbol{s},e} = \mathrm{Ann}(\boldsymbol{s}) \cap \mathbb{A}[y]_{\leq e}$, and in particular $\mathcal{K}_{\boldsymbol{s},e}$ is a generating set of $\mathrm{Ann}(\boldsymbol{s})$.

Proof. Let $p = p_0 + \cdots + p_e y^e \in \mathbb{A}[y]$ and $\gamma = \deg(p) \leq e$. Then $p \in \mathcal{K}_{\boldsymbol{s},e}$ if and only if $[p_0 \ \cdots \ p_e] H_{\boldsymbol{s},e} = 0$, and by definition of canceling partial sequences this exactly means that p cancels $\boldsymbol{s}_{e+\gamma}$. Now, $\deg(p) = \gamma \leq e + \gamma - \delta$ holds under the assumption $e \geq \delta$,

hence p cancels $\boldsymbol{s}_{e+\gamma}$ if and only if $p \in \mathrm{Ann}(\boldsymbol{s})$ by Lemma 3. It follows that $\mathcal{K}_{\boldsymbol{s},e}$ generates $\mathrm{Ann}(\boldsymbol{s})$, since there exists a generating set of $\mathrm{Ann}(\boldsymbol{s})$ whose polynomials all have degree at most δ. □

Computing the left kernel of $H_{\boldsymbol{s},e}$ can be done via univariate approximation. Indeed, calling $F \in \mathbb{K}[x]^{(e+1)\times(en)}$ the natural lifting of $H_{\boldsymbol{s},e}$, an approximant basis of F at order d gives a generating set of that left kernel. As recalled in Section 2.6, using PM-Basis, a basis of $\mathcal{A}_d(F)$ in shifted reduced or Popov form can be computed in $O^{\sim}(e^{\omega-1}(e+en)d) = O^{\sim}(e^{\omega}nd)$ operations in $\mathbb{K}$.

4.2 Speed-up by compression using structure

Now we show that, when n is large, one can speed up the above approach by a randomized "compression" of the matrix $H_{\boldsymbol{s},e}$. Precisely, taking a random constant matrix $C \in \mathbb{K}^{(en)\times(e+1)}$ and performing the right-multiplication FC, one obtains a square $(e+1)\times(e+1)$ matrix such that $\mathcal{A}_d(F) = \mathcal{A}_d(FC)$ holds with good probability. The cost of the approximant basis computation is thus reduced to $O^{\sim}(e^{\omega}d)$ operations in $\mathbb{K}$, and the right-multiplication can be done efficiently by leveraging the block-Hankel structure of F.

Theorem 10. *Algorithm 3 takes as input an integer $d \in \mathbb{Z}_{>0}$, vectors $F_0, \ldots, F_{\mu+e-2} \in \mathbb{K}[x]^{1\times n}$ of degree less than d, and a shift $w \in \mathbb{Z}_{>0}^{\mu}$, and uses $O^{\sim}(\mu end + \mu^{\omega}d)$ operations in $\mathbb{K}$ to compute a w-Popov matrix $P \in \mathbb{K}[x]^{\mu\times\mu}$ of degree at most d. It chooses at most μen elements independently and uniformly at random from a subset of $\mathbb{K}$ of cardinality κ, and P is the w-Popov basis of $\mathcal{A}_d(F)$ with probability at least $1-\frac{\mu}{\kappa}$, where F is the block-Hankel matrix*

$$F = \begin{bmatrix} F_0 & F_1 & \cdots & F_{e-1} \\ F_1 & F_2 & \iddots & F_e \\ \vdots & \iddots & \iddots & \vdots \\ F_{\mu-1} & F_{\mu} & \cdots & F_{\mu+e-2} \end{bmatrix} \in \mathbb{K}[x]^{\mu\times(en)}. \tag{4.1}$$

When applied to the computation of $\mathrm{Ann}(\boldsymbol{s})$ with $\mu = e+1$, the cost becomes $O^{\sim}(e^2nd + e^{\omega}d)$. Below we focus on the case of interest $\mu \le en$, since when $en \in O(\mu)$ this w-Popov approximant basis is computed deterministically by PM-Basis at a cost of $O^{\sim}(\mu^{\omega}d)$ operations in $\mathbb{K}$. Our approach is based on the following two lemmas.

Lemma 11. *Let $F \in \mathbb{K}[x]^{\mu\times\nu}$ and $d \in \mathbb{Z}_{>0}$. Let $C \in \mathbb{K}[x]^{\nu\times r}$ and $K \in \mathbb{K}[x]^{\nu\times(\nu-r)}$, for some $r \in \{0,\ldots,\nu\}$, such that $FK = 0$ and $[C(0) \;\; K(0)] \in \mathbb{K}^{\nu\times\nu}$ is invertible. Then, $r \geq \rho$ where ρ is the rank of F, and $\mathcal{A}_d(F) = \mathcal{A}_d(FC)$.*

Proof. Let $N = [C \;\; K] \in \mathbb{K}[x]^{\nu\times\nu}$. The assumption that $N(0)$ is invertible ensures that N is nonsingular (since $\det(N)(0) = \det(N(0)) \neq 0$), and therefore K has full rank $\nu - r$. The assumption that the columns of K are in the right kernel of F, which has rank $\nu - \rho$, implies that $\nu - r \leq \nu - \rho$ and therefore $r \geq \rho$.

The inclusion $\mathcal{A}_d(F) \subset \mathcal{A}_d(FC)$ is obvious. For the other inclusion, let $p \in \mathcal{A}_d(FC)$, i.e. there exists $q \in \mathbb{K}[x]^{1\times r}$ such that $pFC = x^d q$. It follows that $pFN = x^d[q \;\; 0]$, and thus

$$pF = x^d[q \;\; 0]N^{-1} = \frac{x^d[q \;\; 0]\mathrm{Adj}(N)}{\det(N)}$$

where $\mathrm{Adj}(N) \in \mathbb{K}[x]^{\nu\times\nu}$ is the adjugate of N. Our assumption $\det(N)(0) \neq 0$ means that x^d and $\det(N)$ are coprime, hence $\det(N)$ divides $[q \;\; 0]\mathrm{Adj}(N)$, and $pF = 0 \bmod x^d$ follows. □

Lemma 12. *Let $F \in \mathbb{K}[x]^{\mu\times\nu}$ with rank ρ and $\mu \leq \nu$, and let $r \in \{\rho,\ldots,\mu\}$. Let $\mathcal{R}$ be a finite subset of $\mathbb{K}$ of cardinality $\kappa \in \mathbb{Z}_{>0}$, and let $C \in \mathbb{K}^{\nu\times r}$ with entries chosen independently and uniformly at random from $\mathcal{R}$. Then, the probability that there exists $K \in \mathbb{K}[x]^{\nu\times(\nu-r)}$ such that $[C \;\; K(0)]$ is invertible and $FK = 0$ is at least $1 - \frac{r}{\kappa}$; furthermore if $\mathbb{K}$ is finite and $\mathcal{R} = \mathbb{K}$, this probability is at least $\prod_{i=1}^{r}(1 - \kappa^{-i})$.*

Proof. Consider a right kernel basis $B \in \mathbb{K}[x]^{\nu\times(\nu-\rho)}$ for F. Then B has unimodular row bases [69, Lem. 3.1], implying that there exists $V \in \mathbb{K}[x]^{(\nu-\rho)\times\nu}$ such that $VB = I_{\nu-\rho}$. In particular $V(0)B(0) = I_{\nu-\rho}$ and therefore $B(0)$ has full rank $\nu - \rho$. Define $K \in \mathbb{K}[x]^{\nu\times(\nu-r)}$ as the matrix formed by the first $\nu - r$ columns of B (recall $\nu - r \leq \nu - \rho$ by assumption). Then $FK = 0$. Furthermore $K(0)$ has rank $\nu - r$, hence the DeMillo-Lipton-Schwartz-Zippel lemma implies that $[C \;\; K(0)] \in \mathbb{K}^{\nu\times\nu}$ is singular with probability at most r/κ [16, 59, 70]. If $\mathbb{K}$ is finite and $\mathcal{R} = \mathbb{K}$ then $[C \;\; K(0)]$ is invertible with probability exactly $\prod_{i=1}^{r}(1 - \kappa^{-i})$. □

These lemmas lead to Algorithm 3 and Theorem 10; indeed computing FC has quasi-linear cost $O^{\sim}(\mu end)$ thanks to the block-Hankel structure of F, and then the call PM-Basis(d, FC, w) costs $O^{\sim}(\mu^{\omega} d)$ operations as recalled in Section 2.6.

Note that $1 - r/\kappa \geq 3/4$ as soon as $\kappa \geq 4\mu$ (which implies $\kappa \geq 4r$); furthermore $\prod_{i=1}^{r}(1 - \kappa^{-i}) \geq 3/4$ already for $\kappa = 7$. The randomization is of the Monte Carlo type,

Algorithm 3 HANKEL-PM-BASIS(d, F, w)

Input: integers $d, \mu, e, n \in \mathbb{Z}_{>0}$, vectors $F_0, \ldots, F_{\mu+e-2} \in \mathbb{K}[x]^{1\times n}$ of degree less than d, a shift $w \in \mathbb{Z}_{>0}^{\mu}$

Output: a w-Popov matrix $P \in \mathbb{K}[x]^{\mu\times\mu}$ of degree at most d

1: $F \in \mathbb{K}[x]^{\mu\times(en)} \leftarrow$ form the block-Hankel matrix as in Eq. (4.1)
2: **if** $\mu \geq en$ **then return** PM-BASIS(d, F, w)
3: Choose $r \in \{\rho, \ldots, \mu\}$ where ρ is the rank of F (by default, choose $r = \mu$ if no information is known on ρ)
4: Fill a matrix $C \in \mathbb{K}^{(en)\times r}$ with entries chosen uniformly and independently at random from a subset of $\mathbb{K}$ of cardinality κ
5: Compute $FC \in \mathbb{K}[x]^{\mu\times r}$ (exploiting the Hankel structure of F)
6: **return** PM-BASIS(d, FC, w)

since the algorithm may return P which is not a basis of $\mathcal{A}_d(F)$. Still, since the expected w-Popov basis P of $\mathcal{A}_d(F)$ is unique, one can easily increase the probability of success by repeating the randomized computation and following a majority rule. Another approach is to rely on the non-interactive, Monte Carlo certification protocol of [26], which has lower cost than Algorithm 3 but requires a larger field $\mathbb{K}$; this first asks to compute the coefficient of degree d of PF, which here can be done via bivariate polynomial multiplication in time $O\tilde{}(\mu end)$ thanks to the structure of F. For a given output P, this certification can be repeated for better confidence in P (in which case the coefficient of degree d of PF needs only be computed once).

4.3 Via bivariate Padé approximation

In this section, we propose another approach which directly uses the interpretation of canceling polynomials as denominators of the generating series of the sequence (see Lemma 2). The next lemma describes more precisely the link between the annihilator and these denominators when we have access to a partial sequence, that is, denominators of the generating series truncated at some order. One can also view this lemma as a description of the kernel of the univariate Hankel matrix $H_{\boldsymbol{s},e}$ via bivariate Padé approximation.

Lemma 13. *Let $\boldsymbol{s} \in \mathcal{S}$ be linearly recurrent of order δ, and for $e \in \mathbb{N}$ define $G = \sum_{j<2e} S_j y^{2e-1-j} \in \mathbb{A}[y]^n$ and*

$$\mathcal{P}_{\boldsymbol{s},e} = \{p \in \mathbb{A}[y]_{\leq e} \mid pG = q \bmod y^{2e} \text{ for some } q \in \mathbb{A}[y]^n_{<e}\}.$$

Assume $e \geq \delta$. Then $\mathcal{P}_{\boldsymbol{s},e} = \mathrm{Ann}(\boldsymbol{s}) \cap \mathbb{A}[y]_{\leq e}$, and in particular $\mathcal{P}_{\boldsymbol{s},e}$ is a generating set of $\mathrm{Ann}(\boldsymbol{s})$; furthermore for any $p \in \mathcal{P}_{\boldsymbol{s},e}$ the corresponding $q \in \mathbb{A}[y]^n_{<e}$ satisfies $\deg(q) < \deg(p)$.

Proof. Let $p = p_0 + \cdots + p_\gamma y^\gamma \in \mathbb{A}[y]_{\leq e}$ where $\gamma = \deg(p)$. Then $p \in \mathcal{P}_{\boldsymbol{s},e}$ if and only if the coefficient of pG of degree $2e-1-k$ is zero for $0 \leq k < e$. Since $\gamma \leq e \leq 2e-1-k$, this coefficient is

$$\mathrm{Coeff}(pG, 2e-1-k) = \sum_{i=0}^{\gamma} p_i S_{2e-1-(2e-1-k-i)} = \sum_{i=0}^{\gamma} p_i S_{k+i} = 0.$$

Thus we have proved $\mathcal{P}_{\boldsymbol{s},e} = \mathcal{K}_{\boldsymbol{s},e}$, and Lemma 9 shows the claims in this lemma except the last one. Let $p \in \mathcal{P}_{\boldsymbol{s},e}$ and define q as the polynomial in $\mathbb{A}[y]^n_{<e}$ such that $pG = q \bmod y^{2e}$. Since $p \in \mathrm{Ann}(\boldsymbol{s})$, Lemma 2 shows that pG is a polynomial. On the other hand the definitions of G and G yield $pG = y^{2e} pG - p\sum_{j\geq 2e} S_j y^{2e-1-j}$. Hence $-p\sum_{j\geq 2e} S_j y^{2e-1-j}$ is a polynomial, and since it has degree less than γ, and thus in particular less than $2e$, it is equal to q. □

From G, define $F \in \mathbb{K}[\alpha,\beta]^{1\times n}$ of bi-degree less than $(d, 2e)$ via the morphism φ from Section 2.5. Equip $\mathbb{K}[\alpha,\beta]$ with the lexicographic order $\preccurlyeq_{\mathrm{lex}}$, and let $\preccurlyeq$ be the corresponding term over position order on $\mathbb{K}[\alpha,\beta]^{n+1}$. Then a minimal $\preccurlyeq$-Gröbner basis of the submodule of simultaneous Padé approximants

$$\{(p,q) \in \mathbb{K}[\alpha,\beta] \times \mathbb{K}[\alpha,\beta]^{1\times n} \mid pF = q \bmod (x^d, y^{2e})\}$$

is computed in $O^{\sim}((n^\omega \min(d,e)^\omega + n^3 \min(d,e)^2)de)$ operations, using the algorithm of [47] (see also Section 2.6) with input matrix of size $(n+1)\times n$ formed by stacking the identity I_n below F. Lemma 13 shows that from this $\preccurlyeq$-Gröbner basis one can find a minimal $\preccurlyeq_{\mathrm{lex}}$-Gröbner basis of $\bar{\varphi}(\mathrm{Ann}(\boldsymbol{s}))$ by selecting p for each (p,q) in the basis such that $\deg_\beta(q) < \deg_\beta(p)$.

While the PM-Basis approach had cost quasi-linear in d and n, the method here is most efficient in an opposite parameter range: for $n \in O(1)$ and $d \leq e$ the above cost bound becomes $O^{\sim}(d^{\omega+1}e)$.

Chapter 5

Experimental Results

In this section, we compare timings for the algorithms in Sections 3.1, 3.2 and 4.1, implemented in C++ using the libraries NTL [60] and PML [29] which provide high-performance support for univariate polynomials and polynomial matrices. We leave the implementation of the bivariate algorithm of Section 4.3 as future work. To control the cardinality and shape of the Gröbner basis, we use Lazard's structural theorem (see Section 2.5). The shape of the monomial staircase is randomized with maximal β-degree δ and α^d included in the basis. After generating a random Gröbner basis $\mathcal{G}$ of target degree and size, we use it to generate n sequences (with $e = 2\delta$ terms), using random initial conditions. Finally, we compute the annihilator of the sequence, which may not necessarily recover $\mathcal{G}$ itself (see Section 6.1). Runtimes are showed below.

As we claim in Section 3.2, d^* is often close or equal to d_{opt}. More interestingly, Lazy Kurakin outperforms Kurakin more than d/d^* would suggest. For example, for $\delta = 256, d = 64, d_{\text{opt}} = 49$, then $d/d^* \approx 1.2$ but Kurakin is 23 times slower than Lazy Kurakin. This is because the cost bound $O\tilde{}(ed^*(n^2ed + n^\omega d))$ for Lazy Kurakin assumes that d^* polynomials are tracked from the beginning of the algorithms. However, due to its lazy nature, polynomials are often added later in the algorithm and the bound of ed^* subiterations may significantly overestimate the true number of subiterations.

When δ, d, n are fixed, Kurakin's algorithm performs worse for $d_{\text{opt}} = 1$ than $d_{\text{opt}} > 1$, although this is a favourable case for Lazy Kurakin. In this case, Kurakin's algorithm computes $P_i = x^i P_0$ so there cannot be any early termination. Additionally, the size of the staircase is maximal ($D = ed$), so this is also the worst case for algorithms whose complexity depends directly on D. Lazy Kurakin's algorithm somewhat remedies this by using the extra structure of $\mathbb{A}$ and adding monomials in a lazy fashion. (When it is

n	d	δ	d_{opt}	$D/d\delta$	K	LK	d^*	PM-B	HPM
1	64	256	1	1	62.8	0.93	1	1.06	NA
1	64	256	49	0.62	38.0	1.65	53	2.10	NA
1	128	512	16	0.92	> 100	12	17	20.5	NA
1	128	32	12	0.91	7.85	0.078	12	0.029	NA
1	256	32	14	0.94	27.3	0.12	14	0.08	NA
1	256	128	27	0.92	> 100	1.28	27	1.60	NA
1	512	256	29	0.96	> 100	8.65	29	27.8	NA
2	17	256	2	0.5	14.1	0.91	2	0.33	0.29
3	12	512	4	0.4	6.93	1.40	4	2.47	1.86
8	16	256	1	1	54.1	3.16	1	0.56	0.25
32	16	256	1	1	> 100	39.8	1	2.79	0.35
64	16	128	1	1	> 100	> 100	1	1.02	0.13

Table 5.1: Runtimes, in seconds, of algorithms Kurakin, Lazy Kurakin, direct PM-Basis, and Hankel-PM-Basis, observed on AMD Ryzen 5 3600X 6-Core CPU with 16 GB RAM over $\mathbb{K} = \mathbb{F}_{9001}$.

known that $\text{Ann}(\boldsymbol{s}) = \langle P \rangle$, it is possible to design an algorithm that is quasilinear in e via structured system solving, see Section 6.2).

For scalar sequences over $\mathbb{A}$, i.e. $n = 1$, Lazy Kurakin's algorithm seems to be the best choice when δ is large compared to d, whereas PM-Basis seems to be the best choice in the converse. When $e = 2\delta = d$, Lazy Kurakin outperforms PM-Basis, given that d^* is small. This is predicted by the theoretical complexities, as the former has complexity $O^{\sim}(e^3 d^*)$, while the latter has complexity $O^{\sim}(e^{\omega+1})$.

For $n > 1$, PM-Basis and Hankel-PM-Basis clearly outperform Kurakin and Lazy Kurakin. This is as predicted since the complexity of the former depends linearly on n, while the latter has a factor n^ω. The theoretical improvement of Hankel-PM-Basis over PM-Basis is observed empirically, especially for the two cases of $n = 32, 64$.

Chapter 6

Applications to Sparse Matrices

In this section, we outline two applications to sparse matrices $A \in \mathbb{A}^{n \times n}$: first, the computation of minimal polynomials of A, which are polynomials of minimal degree that cancel the matrix sequence $\boldsymbol{s}_A = (A^0, A^1, A^2, \ldots)$; second, the computation of the determinant of A. In what follows, we assume A has sparsity $O(n)$, i.e. it has $O(n)$ nonzero entries, and that the representation of A allows us to compute matrix-vector products at cost $O\tilde{}(nd)$. Our approach is based on Wiedemann's [66], designed for matrices over fields.

6.1 Minimal polynomials of sparse matrices

Given a matrix A, the well-known Cayley-Hamilton theorem states that A cancels its own characteristic polynomial. This implies that the sequence of successive powers of A is linearly recurrent, and a polynomial of minimal degree that cancels this sequence is said to be a minimal polynomial of A. A different view one can take is that such canceling polynomials must cancel the n^2 linearly recurrent sequences $((A^i)_{j_1,j_2})_{i \geq 0}$ simultaneously for $1 \leq j_1, j_2 \leq n$. Then, as usual, we want to compute a Gröbner basis of the ideal of these canceling polynomials, denoted by $\mathrm{Ann}(A) \subseteq \mathbb{K}[\alpha, \beta]$.

Over $\mathbb{A}$, trying to deduce $\mathrm{Ann}(A)$ from $\mathrm{Ann}((u^T A^i v)_{i \geq 0})$, for random vectors $u, v \in \mathbb{A}^{n \times 1}$, presents a problem when $\mathrm{Ann}(A)$ does not have the *Gorenstein* property [42, 27]. When $\mathrm{Ann}(A)$ has the Gorenstein property, it has been showed that $\mathrm{Ann}(A)$ can be recovered, with high probability, by using a bidimensional sequence with random initial conditions, provided $\mathbb{K}$ has large characteristic [8]. When it does not have the property, $\mathrm{Ann}(A)$ is still recoverable with a similar approach, but using several sequences [49]. Over

various commutative rings, the problem of computing minimal polynomials of a matrix have been studied in [12, 28, 54]. However, the algorithms given in these works do not exploit sparsity.

Given matrix A as above, we start by choosing random $u_1, v \in \mathbb{A}^n$ and generating $\boldsymbol{s}_{A,1} = (u_1^T A^i v)_{0 \le i < 2n}$. Next, we apply one of the algorithms in the previous sections to compute $\mathrm{Ann}(\boldsymbol{s}_{A,1})$. If $\mathrm{Ann}(\boldsymbol{s}_{A,1}) = \mathrm{Ann}(A)$, which can be checked probabilistically by checking if $\mathrm{Ann}(\boldsymbol{s}_{A,1})$ also cancels some validation sequence $((u')^T A^i v)_{0 \le i < 2n}$, we terminate the process. Otherwise, we double the number of sequences by doubling the number of random u_i's and generating $\boldsymbol{s}_{A,1}, \ldots, \boldsymbol{s}_{A,2^s}$. The cost of the process is $O\tilde{}(\tau n^2 d + \mathcal{L}(n, d, \tau))$, where τ is the number of sequences used and $\mathcal{L}(n, d, \tau)$ is the cost of finding the annihilators of a partial sequence of length n in $(\mathbb{K}[x]/\langle x^d \rangle)^\tau$. Note that this process must terminate. The crudest bound is when $\tau > n^2$ since then we could simply compute $\mathrm{Ann}(A)$ directly. Another slightly more refined bound for the number of generic linear forms needed is $\tau \le D$, where D is the size of the staircase of $\mathrm{Ann}(A)$ [49, Prop. 1].

6.2 Determinant of sparse matrices

The determinant of a matrix is easily obtained from its minimal polynomial when the latter is equal to the characteristic polynomial. Wiedemann [66] calls such matrices *nonderogatory* and shows that preconditioning any matrix $B \in \mathbb{K}^{n \times n}$ with a random diagonal matrix D results in a nonderogatory matrix with high probability. We will show that the same preconditioning can be applied to matrices over $\mathbb{A}$. Here, a particular role will be played by sequences $\boldsymbol{s} \in (\mathbb{A}^n)^{\mathbb{N}}$ such that $\mathrm{Ann}(\boldsymbol{s}) = \langle P \rangle$, for some monic $P \in \mathbb{A}[y]$. Indeed, the next theorem shows that it is sufficient for the constant part of A to be nonderogatory in $\mathbb{K}$ for A to be nonderogatory in $\mathbb{A}$ and for the sequence of its powers to satisfy this property.

Theorem 14. *Let $A_0 \in \mathbb{K}^{n \times n}$ be the constant part of A (i.e. for $x = 0$). If A_0 is nonderogatory, then $\mathrm{Ann}(A) = \langle P \rangle$ for some monic $P \in \mathbb{A}[y]$ of degree n.*

Proof. Let $P \in \mathbb{A}[y]$ be the minimal monic polynomial of the sequence $\boldsymbol{s}_A = (A^0, A^1, A^2, \ldots)$, then $\deg(P) \le n$ since A is $n \times n$. Now, A_0 is nonderogatory, so any canceling polynomial must have degree $\ge n$; thus, $\deg(P) = n$. Furthermore, if there exists another polynomial Q of degree n and leading coefficient x^i such that $Q \ne x^i P$, then $Q - x^i P$ is a canceling polynomial of degree less than n, contradicting the previous statement. Thus, $P, xP, \ldots, x^{d-1}P$ are minimal in degree and, by Theorem 5, $\mathrm{Ann}(A) = \langle P, xP, \ldots, x^{d-1}P \rangle = \langle P \rangle$. □

The above theorem allows us to use the same preconditioner as in [66]: a random constant diagonal matrix D. The preconditioning ensures that the ideal of canceling polynomial is generated by a single monic polynomial; thus, $\bar{\varphi}(\mathrm{Ann}(AD))$ is Gorenstein and requires only a single linear form to be recovered. Furthermore, when it is known that the ideal is generated by a single polynomial, we can recover this polynomial in $O^{\sim}(nd)$ by taking advantage of the fact that the constant part of the leading $n \times n$ submatrix of $H_{\boldsymbol{s},2n}$ is an invertible Hankel matrix [11]. Once we have P, we can compute $\det(A) = P(0)(\prod_i D_{i,i})^{-1}$. Under our sparsity assumption, the cost of this method is $O^{\sim}(n^2 d)$ for computing $(u^T A^i v)_{i \leq 2n}$, $O^{\sim}(nd)$ for computing P, and $O^{\sim}(n+d)$ for recovering the determinant from P, leading to the total cost of $O^{\sim}(n^2 d)$ operations in $\mathbb{K}$. This is to be compared with computing the determinant of A "at full precision", i.e. by seeing A as a matrix over $\mathbb{K}[x]$, and then truncating the result modulo x^d: this costs $O^{\sim}(n^\omega d)$ operations in $\mathbb{K}$ [39].

Chapter 7

Future Works and Conclusion

A possible future direction is to investigate if our methods can be used to improve algorithms over $\mathbb{K}[x]$. More precisely, given a sequence over $\mathbb{K}[x]^n$, our algorithms can find the annihilator of this sequences at precision d and we want to 'lift' this solution to sufficient precision $d' > d$ to apply rational reconstruction. For many matrix algorithms over $\mathbb{K}[x]$ of size $m \times m$ and maximal degree d, we typically want the precision to be $O(md)$ as this is the maximal degree of the determinant. Via high-order lifting [63], one could solve such systems at 'full precision' at cost $O^{\sim}(m^{\omega}d)$. Assuming the sparsity assumption and applying the algorithms of Chapter 6 at precision $O(md)$ yields costs at least cubic in m. A problem with such approach is that while we improve the cost of computing solutions to sparse systems at fixed d, the cost of lifting the solution becomes the bottleneck at high precisions.

Linearly recurrent sequences are fundamental to many areas in computer science and mathematics, and there are many important open questions for sequences over rings. By tailoring our focus to sequences over $\mathbb{A}^n$, we propose algorithms that have better complexities than more general algorithms applied to our context. While Lazy Kurakin's algorithm attempts reduce the number of output polynomials, approximation algorithms reduce the dependence in one of the dimensions at the cost of the other dimension. Our experiments show that the theoretical complexities are reflected in practise, particularly between the different parameter regimes. We also present applications to sparse matrices over $\mathbb{K}[x]/x^d$, as linearly recurrent sequences naturally arise when designing Wiedemann-like algorithms.

References

[1] J. Alman and V. Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings SODA 2021*, pages 522–539, 2021.

[2] B. Beckermann. A reliable method for computing M-Padé approximants on arbitrary staircases. *J. Comput. Appl. Math.*, 40(1):19–42, 1992.

[3] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM J. Matrix Anal. Appl.*, 15(3):804–823, 1994.

[4] B. Beckermann and G. Labahn. Recursiveness in matrix rational interpolation problems. *J. Comput. Appl. Math.*, 77(1):5–34, 1997.

[5] B. Beckermann, G. Labahn, and G. Villard. Shifted normal forms of polynomial matrices. In *ISSAC'99*, pages 189–196. ACM, 1999.

[6] C. Berkesch and F.-O. Schreyer. Syzygies, finite length modules, and random curves. In *Commutative Algebra and Noncommutative Algebraic Geometry*, pages pp. 25–52. Mathematical Sciences Research Institute Publications (Vol. 67), 2015.

[7] E. Berlekamp. Nonbinary bch decoding (abstr.). *IEEE Trans. Inf. Theory*, 14(2):242–242, 1968.

[8] J. Berthomieu, B. Boyer, and J.-C. Faugère. Linear algebra for computing Gröbner bases of linear recursive multidimensional sequences. *J. Symb. Comput.*, 83:36–67, 2017.

[9] J. Berthomieu and J.-C. Faugère. A polynomial-division-based algorithm for computing linear recurrence relations. In *ISSAC'18*, pages 79–86, 2018.

[10] D. Bini and V. Y. Pan. *Polynomial and Matrix Computations (Vol. 1): Fundamental Algorithms*. Birkhauser Verlag, 1994.

[11] A. Bostan, C.-P. Jeannerod, and É. Schost. Solving structured linear systems with large displacement rank. *Theor. Comput. Sci.*, 407(1):155–181, 2008.

[12] W. C. Brown. Null ideals of matrices. *Communications in Algebra*, 33(12):4491–4504, 2005.

[13] Bruno Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation*, 41(3-4):475–511, 2006.

[14] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[15] D. A. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry (second edition).* Springer-Verlag New-York, New York, NY, 2005.

[16] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.*, 7(4):193–195, 1978.

[17] Jean-Charles Faugère and Chenqi Mou. Sparse fglm algorithms. *Journal of Symbolic Computation*, 80:538–569, 2017.

[18] Charles M Fiduccia. An efficient formula for linear recurrences. *SIAM Journal on computing*, 14(1):106–112, 1985.

[19] P. Fitzpatrick. A theoretical basis for Padé approximation. *Irish Math. Soc. Bull*, 30:6–17, 1993.

[20] P. Fitzpatrick. On the key equation. *IEEE Trans. Inf. Theory*, 41(5):1290–1302, 1995.

[21] P. Fitzpatrick. Solving a Multivariable Congruence by Change of Term Order. *J. Symb. Comput.*, 24(5):575–589, 1997.

[22] P. Fitzpatrick and G. H. Norton. Finding a basis for the characteristic ideal of an n-dimensional linear recurring sequence. *IEEE Trans. Inf. Theory*, 36(6):1480–1487, 1990.

[23] Joachim Gathen and Jürgen Gerhard. *Modern computer algebra (2. ed.).* 01 2003.

[24] Patrizia Gianni and Teo Mora. Algebrric solution of systems of polynomirl equations using groebher bases. In *International Conference on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 247–257. Springer, 1987.

[25] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *ISSAC'03*, pages 135–142. ACM, 2003.

[26] P. Giorgi and V. Neiger. Certification of minimal approximant bases. In *ISSAC'18*, pages 167–174. ACM, 2018.

[27] W. Gröbner. Über irreduzible Ideale in kommutativen Ringen. *Mathematische Annalen*, 110(1):197–222, 1935.

[28] C. Heuberger and R. Rissner. Computing J-ideals of a matrix over a principal ideal domain. *Linear Algebra Appl.*, 527:12–31, 2017.

[29] S. G. Hyun, V. Neiger, and É. Schost. Implementations of efficient univariate polynomial matrix algorithms and application to bivariate resultants. In *ISSAC'19*, pages 235–242. ACM, 2019.

[30] Seung Gyu Hyun, Vincent Neiger, Hamid Rahkooy, and Éric Schost. Sparse fglm using the block wiedemann algorithm. *ACM Commun. Comput. Algebra*, 52(4):123–125, May 2019.

[31] C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard. Fast computation of minimal interpolation bases in Popov form for arbitrary shifts. In *ISSAC'16*, pages 295–302. ACM, 2016.

[32] C.-P. Jeannerod, V. Neiger, É. Schost, and G. Villard. Computing minimal interpolation bases. *J. Symb. Comput.*, 83:272–314, 2017.

[33] C.-P. Jeannerod, V. Neiger, and G. Villard. Fast computation of approximant bases in canonical form. *J. Symb. Comput.*, 98:192–224, 2020.

[34] Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences, 1962.

[35] M. Kuijper and K. Schindelar. The predictable leading monomial property for polynomial vectors over a ring. In *Proceedings ISIT 2010*, pages 1133–1137, 2010.

[36] M. Kuijper and K. Schindelar. Minimal Gröbner bases and the predictable leading monomial property. *Linear Algebra Appl.*, 434(1):104–116, 2011.

[37] V. L. Kurakin. The Berlekamp–Massey algorithm over finite rings, modules, and bimodules. *Discrete Mathematics and Applications*, 8(5):441–474, 1998.

[38] V. L. Kurakin. Construction of the annihilator of a linear recurring sequence over finite module with the help of the Berlekamp-Massey Algorithm. In *FPSAC 2000*, pages 476–483. Springer, 2000.

[39] G. Labahn, V. Neiger, and W. Zhou. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. 42:44–71, 2017.

[40] D. Lazard. Ideal bases and primary decomposition: Case of two variables. *J. Symb. Comput.*, 1(3):261–270, 1985.

[41] F. Le Gall. Powers of tensors and fast matrix multiplication. In *ISSAC'14*, pages 296–303. ACM, 2014.

[42] F. S. Macaulay. Modern algebra and polynomial ideals. In *Math. Proc. Camb. Philos. Soc*, volume 30, pages 27–46. Cambridge University Press, 1934.

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

[44] H.M. Möller, T. Mora, and C. Traverso. Gröbner bases computation using syzygies. In *Papers from the international symposium on Symbolic and algebraic computation*, pages 320–328, 1992.

[45] T. Mora. The FGLM problem and Möller's algorithm on zero-dimensional ideals. In M. Sala, S. Sakata, T. Mora, C. Traverso, and L. Perret, editors, *Gröbner Bases, Coding, and Cryptography*, pages 27–45, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[46] B. Mourrain. Fast algorithm for border bases of Artinian Gorenstein algebras. In *ISSAC'17*, pages 333–340. ACM, 2017.

[47] S. Naldi and V. Neiger. A divide-and-conquer algorithm for computing Gröbner bases of syzygies in finite dimension. In *ISSAC'20*, pages 380–387. ACM, 2020.

[48] V. Neiger. *Bases of relations in one or several variables: fast algorithms and applications*. PhD thesis, École Normale Supérieure de Lyon, November 2016.

[49] V. Neiger, H. Rahkooy, and É. Schost. Algorithms for zero-dimensional ideals using linear recurrent sequences. In *CASC 2017*, pages 313–328. Springer, 2017.

[50] V. Neiger and É. Schost. Computing syzygies in finite dimension using fast linear algebra. *J. Complexity*, 60:101502, 2020.

[51] Vincent Neiger, Hamid Rahkooy, and Éric Schost. Algorithms for zero-dimensional ideals using linear recurrent sequences. In *International Workshop on Computer Algebra in Scientific Computing*, pages 313–328. Springer, 2017.

[52] Victor Pan. *Structured matrices and polynomials. Unified superfast algorithms*. 01 2001.

[53] V. M. Popov. Invariant description of linear, time-invariant controllable systems. *SIAM Journal on Control*, 10(2):252–264, 1972.

[54] R. Rissner. Null ideals of matrices over residue class rings of principal ideal domains. *Linear Algebra Appl.*, 494:44–69, 2016.

[55] S. Sakata. Finding a minimal set of linear recurring relations capable of generating a given finite two-dimensional array. *J. Symb. Comput.*, 5(3):321–337, 1988.

[56] S. Sakata. Extension of the berlekamp-massey algorithm to n dimensions. *Information and Computation*, 84(2):207–239, 1990.

[57] S. Sakata. The bms algorithm. In *Gröbner Bases, Coding, and Cryptography*, pages 143–163. Springer, 2009.

[58] F-O. Schreyer. *Die Berechnung von Syzygien mit dem verallgemeinerten Weierstraßschen Divisionssatz*. PhD thesis, Master's thesis, Fakultät für Mathematik, Universität Hamburg, 1980.

[59] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[60] V. Shoup. NTL: A library for doing number theory, version 11.4.3. `http://www.shoup.net`, 2020.

[61] Nancy L Stokey. *Recursive methods in economic dynamics*. Harvard University Press, 1989.

[62] A. Storjohann. Notes on computing minimal approximant bases. In *Challenges in Symbolic Computation Software*, Dagstuhl Seminar Proceedings, 2006.

[63] Arne Storjohann. High-order lifting and integrality certification. *Journal of Symbolic Computation*, 36(3):613 – 648, 2003. ISSAC 2002.

[64] M. Van Barel and A. Bultheel. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numer. Algorithms*, 3:451–462, 1992.

[65] Gilles Villard. On computing the resultant of generic bivariate polynomials. In *Proceedings of the 2018 acm international symposium on symbolic and algebraic computation*, pages 391–398, 2018.

[66] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, 32(1):54–62, 1986.

[67] W. A. Wolovich. *Linear Multivariable Systems*, volume 11 of *Applied Mathematical Sciences*. Springer-Verlag New-York, 1974.

[68] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symb. Comput.*, 47(7):793–819, 2012.

[69] W. Zhou and G. Labahn. Computing column bases of polynomial matrices. In *ISSAC'13*, pages 379–386. ACM, 2013.

[70] R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM'79*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.