Row Reduction Applied to Decoding of Rank Metric and Subspace Codes

Puchinger, Sven ; Rosenkilde, Johan Sebastian Heesemann; Li, Wenhui; Sidorenko, Vladimir

Published in:
Designs, Codes and Cryptography

Link to article, DOI:
10.1007/s10623-016-0257-9

Publication date:
2017

Document Version
Peer reviewed version

Noname manuscript No.
(will be inserted by the editor)

Row Reduction Applied to Decoding of Rank Metric and Subspace Codes

Sven Puchinger · Johan S. R. Nielsen · Wenhui Li · Vladimir Sidorenko

Received: date / Accepted: date

Abstract For many algebraic codes the main part of decoding can be reduced to row reduction of a module basis, enabling general, flexible and highly efficient algorithms. Inspired by this, we develop an approach of transforming matrices over skew polynomial rings into certain normal forms. We apply this to solve generalised shift register problems, or Padé approximations, over skew polynomial rings which occur in error and erasure decoding ℓ-Interleaved Gabidulin codes. We obtain an algorithm with complexity $O(\ell\mu^2)$ where μ measures the size of the input problem. Further, we show how to list-ℓ decode Mahdavifar–Vardy subspace codes in $O(\ell r^2 m^2)$ time, where m is a parameter proportional to the dimension of the codewords' ambient space and r is the dimension of the received subspace.

Keywords Skew Polynomials · Row Reduction · Module Minimisation · Gabidulin Codes · Shift Register Synthesis · Mahdavifar–Vardy Codes

Sven Puchinger (grant BO 867/29-3), Wenhui Li and Vladimir Sidorenko (grant BO 867/34-1) were supported by the German Research Foundation "Deutsche Forschungsgemeinschaft" (DFG).

Sven Puchinger · Wenhui Li
Institute of Communications Engineering, Ulm University, Germany
E-mail: sven.puchinger@uni-ulm.de, wenhui.li@uni-ulm.de

Johan S.R. Nielsen
Department of Applied Mathematics and Computer Science, Technical University of Denmark
E-mail: jsrn@jsrn.dk

Vladimir Sidorenko
Institute for Communications Engineering, TU München, Germany, and on leave from the Institute of Information Transmission Problems (IITP), Russian Academy of Sciences
E-mail: vladimir.sidorenko@tum.de

1 Introduction

Numerous recent publications have dealt with shaping the core of various decoding algorithms for Reed–Solomon (RS) and other codes around $\mathbb{F}_q[x]$ module minimisation, lattice basis reduction or module Gröbner basis computation: three computational concepts which all converge to the same in this instance. First for the Guruswami–Sudan list decoder [2,8,26], then for Power decoding [40] and also either type of decoder for Hermitian codes [42].

The impact of this can be said to be two-fold: firstly, by factoring out coding theory from the core problem, we enable the immediate use of sophisticated algorithms developed by the computer algebra community such as [21, 63]. Secondly, the setup has proved very flexible and readily applicable in settings which were not envisioned to begin with, such as the aforementioned Power decoder for Hermitian codes, or recently for Power decoding of RS codes up to the Johnson bound [41].

The main goal of this paper is to explore the row reduction description over skew polynomial rings for decoding rank metric and subspace codes. Our algorithms improve the best known results in these applications and in several instances solves more general problems. We also believe there is a justifiable hope for similar generalisations and speed-ups as known for other code classes. Concretely, we lay a foundation by extending the core terms of weak Popov form and orthogonality defect, as well as extending the elegantly simple Mulders–Storjohann algorithm [39] to matrices over skew polynomial rings. To precisely analyse its complexity, we use the Dieudonné determinant for matrices over non-commutative rings.

We apply the established methodology in two settings, and in each setting refine the Mulders–Storjohann in a non-trivial way to obtain new algorithms with better complexity. First we solve a general form of multi-shift-register synthesis over skew polynomials, in a manner similar to [40] for $\mathbb{F}_q[x]$. This is used for fast error-erasure decoding of Interleaved Gabidulin codes. Secondly, we apply the method to the interpolation problem in the list-decoding of Mahdavifar–Vardy codes [35].

Shift-register synthesis is a classical problem with many applications in a wide variety of fields. We use the well-known polynomial description of shift-register synthesis, see e.g. [36]. Formulated as such, the problem is almost equivalent to rational function reconstruction and Padé approximation. Shift-register synthesis has several generalisations, some of which have found applications in decoding of algebraic codes, e.g. [50, 52, 62]. In the computer algebra community, Padé approximations have been studied to matrix forms in a wide generality, e.g. [5]. Over skew polynomial rings, work has been spurred on by decoding of Gabidulin and related codes, e.g. [16, 54], and the literature is still far behind the $\mathbb{F}_q[x]$ counterpart.

Gabidulin codes [10, 16, 51] are maximum rank distance codes with various applications like random linear network coding [24, 57] and cryptography [17]. They are the rank-metric analogue of RS codes. An Interleaved Gabidulin code [32, 54, 57] is a direct sum of several Gabidulin codes: these can be decoded in a collaborative manner, improving the error-correction capability beyond the usual half the minimum rank distance of Gabidulin codes. Similar to Interleaved RS codes, see [52] and its references, the core task of decoding can be reduced to a multi-sequence skew-feedback shift register synthesis problem [54].

Mahdavifar–Vardy (MV) codes [33, 35] are subspace codes which can be used for error- and erasure correction in random linear network coding. They are generalisations of the codes proposed by Kötter–Kschischang [24], who use Gabidulin codes to construct

constant-dimension subspace codes. Although obtainable rates are rather low, their decoding is of interest since they can be list-decoded using an algorithm similar to Sudan's algorithm [58] for list-decoding Reed–Solomon codes.

Previous work on normal form computation of matrices over skew or Ore rings can be found in [1, 4, 38]. However, the focus of [1, 4] was on $\mathbb{Z}$ or $K[z]$ where coefficient growth is a major concern. Over finite fields, only field operations are counted, and in this measure those previous algorithms are slower than what we present here. The row reduction algorithm presented in [38] is different from ours and has a slightly larger complexity.

This work was partly presented at the International Workshop on Coding and Cryptography (WCC) 2015 [28]. Compared to this previous work, we added the decoding of MV codes using the row reduction approach[1]. This demanded a restructuring of the presentation, where the row reduction approach is more central. It spurred a new refinement of the Mulders–Storjohann, in Section 6.2, which could be of wider interest. We also give proofs that were omitted in the conference paper due to space restrictions, and we added a formal correctness proof and analysis of the Demand–Driven algorithm. We have added an appendix containing basic facts on modules over skew polynomial rings, culminating in Theorem 14 which is crucial to our work; these facts are all known, but we found no convenient compilation in the literature.

We set basic notation in Section 2. Section 3 describes row reduction of skew polynomial matrices and presents the skew polynomial version of Mulders–Storjohann algorithm for accomplishing this. Important tools for arguing about reduced bases, especially for complexity, are given in Section 4. Section 5 shows how to solve shift register problems, arising from decoding problem of Interleaved Gabidulin codes, using row reduction of skew polynomial matrices. For these cases, we go on to refine the Mulders–Storjohann into the faster Demand–Driven algorithm. We then outline Mahdavifar–Vardy codes in Section 6 and show in Lemma 9 how to decode them using row reduction. In Section 6.2 we describe how to "walk" between row reduced forms, yielding a more efficient algorithm for the MV decoding. Appendix A contains technical proofs, and we prove general statements about modules over skew polynomial rings in Appendix B.

2 Notation and Remarks on Generality

Let K be a field. Denote by $\mathcal{R} = K[x; \theta, \delta]$ the non-commutative ring of skew polynomials over K with automorphism θ and derivation δ: elements of $\mathcal{R}$ are of the form $\sum_i a_i x^i$ with $a_i \in K$, addition is as usual, while multiplication is defined by $xa = \theta(a)x + \delta(a)$. For $a, b \in K$, δ satisfies that $\delta(a + b) = \delta(a) + \delta(b)$ and $\delta(ab) = \delta(a)b + \theta(a)\delta(b)$. Being an Ore extension, $\mathcal{R}$ is both a left and right Euclidean ring. When we say "polynomial", we will mean elements of $\mathcal{R}$ The definition of the degree of a polynomial is the same as for ordinary polynomials. By $\mathcal{R}_{<n}$ we mean polynomials in $\mathcal{R}$ with degree $< n$. See [45] for more details.

For coding theory we usually take K as a finite field $\mathbb{F}_{q^m}$ for a prime power q, $m \in \mathbb{N}$ and θ as the Frobenius automorphism $\theta(a) = a^q$ for $a \in \mathbb{F}_{q^r}$. Also, non-vanishing derivations δ are usually not considered, a notable exception being [7]. The algorithms in this paper are correct for any field, automorphism and derivation. For complexities, we count field operations, and we often assume $\delta = 0$.

[1] In the literature, the term "row reduction" is more common that "module minimisation", which we used in [28].

The outlined code constructions in this paper require the definition of an evaluation map for skew polynomials. In general there are many ways to define such a mapping (cf. [7]). We will need only the case where $\delta = 0$, for which we use the following definition for any $a \in \mathcal{R}$:

$$a(\cdot) := \mathrm{ev}_a(\cdot) \,:\, K \to K \;,\; \alpha \mapsto \sum_i a_i \theta^i(\alpha).$$

It is clear by construction that this is a group homomorphism on $(K, +)$.

In the special case where $K = \mathbb{F}_{q^m}$ and $\theta = \cdot^q$, then $\mathcal{R}$ is isomorphic to the ring of *linearised polynomials*, and the rings' evaluation maps agree. It follows that in this case, the evaluation map of a multiplication of two polynomials $f \cdot g$ is equal to the composition of their evaluation maps, i.e., $\mathrm{ev}_{ab} = \mathrm{ev}_a \circ \mathrm{ev}_b$. Furthermore, evaluation is in this case an $\mathbb{F}_q$-linear map of the $\mathbb{F}_q$-vector space $\mathbb{F}_{q^m}$. This fact motivates the definition of an *annihilator polynomial* $\mathcal{A}_\mathcal{U}$ as the monic polynomial of smallest degree having a given $\mathcal{U}$ as zero set, where $\mathcal{U}$ is an $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$. Thus, by [30, page 120], given $\mathcal{U}$, then $\mathcal{A}_\mathcal{U}$ is unique and has $\deg \mathcal{A}_\mathcal{U} = \dim_{\mathbb{F}_q} \mathcal{U}$. Also, for distinct $a_1, \ldots, a_n \in \mathbb{F}_{q^m}$ and arbitrary $b_1, \ldots, b_n \in \mathbb{F}_{q^m}$, there is a unique $c \in \mathcal{R}$ with $c(a_i) = b_i$ for all i and $\deg c < n$.

By $a \equiv b \mod c$ we denote the right modulo operation in $\mathcal{R}$, i.e., that there exists $d \in \mathcal{R}$ such that $a = b + dc$. By "modules" we will mean left $\mathcal{R}$-modules. We extensively deal with vectors and matrices over $\mathcal{R}$. Matrices are named by capital letters (e.g. V). The ith row of V is denoted by $\boldsymbol{v}_i$ and the jth element of a vector $\boldsymbol{v}$ is v_j. $v_{i,j}$ is the (i,j)th entry of a matrix V. Indices start at 0.

- The *degree of a vector* $\boldsymbol{v}$ is $\deg \boldsymbol{v} := \max_i\{\deg v_i\}$ (and $\deg \mathbf{0} = -\infty$) and the *degree of a matrix* V is $\deg V := \sum_i \{\deg \boldsymbol{v}_i\}$.
- The *max-degree* of V is $\mathrm{maxdeg}\, V := \max_i\{\deg \boldsymbol{v}_i\} = \max_{i,j}\{\deg v_{i,j}\}$.
- The *leading position* of a vector $\boldsymbol{v}$ is $\mathrm{LP}(\boldsymbol{v}) := \max\{i \,:\, \deg v_i = \deg \boldsymbol{v}\}$. Furthermore $\mathrm{LT}(\boldsymbol{v}) := v_{\mathrm{LP}(\boldsymbol{v})}$ and $\mathrm{LC}(\boldsymbol{v})$ is the leading coefficient of $\mathrm{LT}(\boldsymbol{v})$.

For complexity estimations, we assume that given $a \in K$, then computing $\theta^i(a)$ can be done in $O(1)$ time for any $i \in \mathbb{N}$. When $K = \mathbb{F}_{q^m}$ and θ is a q-powering, then this is reasonable since we can represent $\mathbb{F}_{q^m}$-elements using a normal basis over $\mathbb{F}_q$ (cf. [59, Section 2.1.2]): in this case, a^q is simply the cyclic shift of a seen as an $\mathbb{F}_q$-vector over the normal basis.

Basic notations and statements about (left) $\mathcal{R}$-modules and $\mathcal{R}$-matrices are provided in Appendix B.

3 Row Reduction of $\mathcal{R}$-matrices

In this section we discuss certain normal forms of matrices over $\mathcal{R}$. In particular, we are interested in how and in which complexity we can obtain bases of left $\mathcal{R}$-modules $\mathcal{V} \subseteq \mathcal{R}^{\ell+1}$ in the row reduced *weak Popov form*[2]. This row reduction approach enables us to present simple proofs of decoding algorithms in Section 5 and 6.

[2] There is a precise notion of "row reduced" [23][p. 384] for $\mathbb{F}[x]$ matrices. It is easy to show that being in weak Popov form implies being row reduced, but we will not formally define row reduced in this paper.

Definition 1 A matrix V over $\mathcal{R}$ is in *weak Popov form* if the leading positions of all its non-zero rows are different.

We will need to "shift" the relative importance of some columns compared to others. We therefore define for any shift $\boldsymbol{w} = (w_0, \ldots, w_\ell) \in \mathbb{N}_0^{\ell+1}$, the mapping

$$\Phi_{\boldsymbol{w}} : \mathcal{R}^{\ell+1} \to \mathcal{R}^{\ell+1},\ \boldsymbol{u} = (u_0, \ldots, u_\ell) \mapsto (u_0 x^{w_0}, \ldots, u_\ell x^{w_\ell}).$$

We can extend $\Phi_{\boldsymbol{w}}$ to $\mathcal{R}$-matrices by applying it row-wise. It is easy to see that $\Phi_{\boldsymbol{w}}(\mathcal{V})$ is also a left $\mathcal{R}$-module and $\Phi_{\boldsymbol{w}}$ is a module isomorphism. The following definition is used in Section 5 and 6 to simplify notation.

Definition 2 For any $\boldsymbol{w} = (w_0, \ldots, w_\ell) \in \mathbb{N}_0^{\ell+1}$, a matrix $V \in \mathcal{R}^{\cdot \times (\ell+1)}$ is in $\boldsymbol{w}$*-shifted weak Popov* form if $\Phi_{\boldsymbol{w}}(V)$ is in weak Popov form.

The following describes how the rows of a matrix in weak Popov form are minimal:

Lemma 1 *Let V be a matrix in weak Popov form whose rows are a basis of a left $\mathcal{R}$-module $\mathcal{V}$. Then every $\boldsymbol{u} \in \mathcal{V}^*$ satisfies $\deg \boldsymbol{u} \geq \deg \boldsymbol{v}$, where $\boldsymbol{v}$ is the row of V with $\mathrm{LP}(\boldsymbol{v}) = \mathrm{LP}(\boldsymbol{u})$.*

Proof: Let $\boldsymbol{u} \in \mathcal{V}^*$, and so $\exists a_0, \ldots, a_\ell \in \mathcal{R}$ s.t. $\boldsymbol{u} = \sum_{i=0}^{\ell} a_i \boldsymbol{v}_i$. The $\boldsymbol{v}_i$ all have different leading position, as well as the $a_i \boldsymbol{v}_i$ for $a_i \neq 0$, which in turn means that their $\psi(a_i \boldsymbol{v}_i)$ are all different. Notice that for any two $\boldsymbol{u}_1, \boldsymbol{u}_2$ with $\psi(\boldsymbol{u}_1) \neq \psi(\boldsymbol{u}_2)$, $\psi(\boldsymbol{u}_1 + \boldsymbol{u}_2)$ equals either $\psi(\boldsymbol{u}_1)$ or $\psi(\boldsymbol{u}_2)$. Applied inductively, it implies that there is an i such that $\psi(\boldsymbol{u}) = \psi(a_i \boldsymbol{v}_i)$, which gives $\mathrm{LP}(\boldsymbol{u}) = \mathrm{LP}(\boldsymbol{v}_i)$ and $\deg \boldsymbol{u} = \deg a_i + \deg \boldsymbol{v}_i$. ∎

We will also use the following *value function* for $\mathcal{R}$ vectors as a "size" of $\mathcal{R}^{\ell+1}$ vectors:

$$\begin{aligned} \psi : \mathcal{R}^{\ell+1} &\to \mathbb{N}_0 \\ \boldsymbol{v} &\mapsto \begin{cases} 0 & \text{if } \boldsymbol{v} = \boldsymbol{0} \\ (\ell+1)\deg \boldsymbol{v} + \mathrm{LP}(\boldsymbol{v}) + 1 & \text{otherwise} \end{cases} \end{aligned}$$

The following definition leads to a remarkably simple algorithm for computing a basis of modules in weak Popov form: Algorithm 1, which is an $\mathcal{R}$-variant of the Mulders–Storjohann algorithm [39] that was originally described for $K[x]$ matrices.

Definition 3 Applying a *simple transformation i on j at position h* on a matrix V with $\deg v_{i,h} \leq \deg v_{j,h}$ means to replace $\boldsymbol{v}_j$ by $\boldsymbol{v}_j - \alpha x^\beta \boldsymbol{v}_i$, where $\beta = \deg v_{j,h} - \deg v_{i,h}$ and $\alpha = \mathrm{LC}(v_{j,h}) / \theta^\beta(\mathrm{LC}(v_{i,h}))$.

By a *simple LP-transformation i on j*, where $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(v_j)$, we will mean a simple transformation i on j at position $\mathrm{LP}(\boldsymbol{v}_i)$.

Remark 1 Note that a simple transformation i on j at position h cancels the leading term of the polynomial $v_{j,h}$. Also elementary row operations keep the row space and rank of the matrix unchanged (cf. Appendix B), so the same is true for any sequence of simple transformations.

Lemma 2 *For some $V \in \mathcal{R}^{\cdot \times (\ell+1)}$, consider a simple LP-transformation i on j, where $\boldsymbol{v}_j$ is replaced by $\boldsymbol{v}'_j$. Then $\psi(\boldsymbol{v}'_j) < \psi(\boldsymbol{v}_j)$.*

Algorithm 1 Mulders–Storjohann for $\mathcal{R}$ matrices

Input: A matrix V over $\mathcal{R}$, whose rows span the module $\mathcal{V}$.
Output: A basis of $\mathcal{V}$ in weak Popov form.
1 Until no longer possible, apply a simple LP-transformation on two rows in V.
2 **return** V.

Proof: Let $h = \operatorname{LP}(\boldsymbol{v}_i) = \operatorname{LP}(\boldsymbol{v}_j)$. Since $\deg v_{i,h} \leq \deg v_{j,h}$ then $\deg \boldsymbol{v}_i \leq \deg \boldsymbol{v}_j$, and therefore $\deg \boldsymbol{v}'_j \leq \deg \boldsymbol{v}_j$. If $\deg \boldsymbol{v}'_j < \deg \boldsymbol{v}_j$, we are done since the leading position of any vector is always less than $\ell + 1$. If $\deg \boldsymbol{v}'_j = \deg \boldsymbol{v}_j$, then $\operatorname{LP}(\boldsymbol{v}'_j) < h$: by the definition of the leading position, all terms to the right of h in both $\boldsymbol{v}_j$ and $\alpha x^\beta \boldsymbol{v}_i$—and therefore also in $\boldsymbol{v}'_j$—have degree less than $\deg \boldsymbol{v}_j$. If $\deg \boldsymbol{v}'_j = \deg \boldsymbol{v}_j$ then some entry in $\boldsymbol{v}'_j$ must have degree $\deg \boldsymbol{v}_j$, which must therefore be to the right of position h. ∎

Theorem 1 *Algorithm 1 is correct.*

Proof: By Lemma 2, the ψ-value of one row of V decreases for each simple LP-transformation. The sum of the values of the rows must at all times be non-negative so the algorithm must terminate. When the algorithm terminates there are no $i \neq j$ such that $\operatorname{LP}(\boldsymbol{v}_i) = \operatorname{LP}(\boldsymbol{v}_j)$. That is to say, V is in weak Popov form. ∎

Note that the algorithm as written is non-deterministic, since there might be multiple possible simple LP-transformations to choose from at any given iteration of the algorithm. However, the above theorem shows that any deterministic rule for choosing between simple LP-transformations would suffice for correctness. For certain input, some rules might be better than others for complexity reasons, however; we'll see an example of this in Section 6.2.

The above proof easily leads to the rough complexity estimate $O(m^2 \deg V \operatorname{maxdeg} V)$ when $\mathcal{R}$ has derivation zero, and where m is the number of columns in V. We will obtain a more fine-grained estimate in the next section, where we also restrict ourselves to matrices which are square and full rank.

4 Complexity Analysis

Lenstra [27] introduced the notion of orthogonality defect of square, full rank $K[x]$ matrices, and in [40], it was shown that it be used to describe the complexity of the Mulders–Storjohann and Alekhnovich [2] algorithms for such matrices more fine-grained than originally. This was used to improve those algorithms' asymptotic complexity estimates when the input comes from shift register problems.

The same concept cannot immediately be carried over to $\mathcal{R}$ matrices, since it is defined using the determinant. For noncommutative rings, there are no functions behaving exactly like the classical determinant, but the Dieudonné determinant [11] shares sufficiently many properties with it for our use. Simply defining this determinant requires us to pass to the field of fractions of $\mathcal{R}$.

4.1 Dieudonné Determinant and Orthogonality Defect

The following algebra is standard for noncommutative rings, so we will go through it quickly; more details can be found in [9, Chapter 1]. We know that $\mathcal{R}$ is a principal

left ideal domain which implies that it is left Ore. It therefore has a unique left field of fractions

$$\mathcal{Q} = \{s^{-1}r \,:\, r \in \mathcal{R}, s \in \mathcal{R}^*\}/(\sim) \ ,$$

where $\sim$ is the congruence relation

$$s^{-1}r \sim s'^{-1}r' \qquad \Longleftrightarrow \qquad \exists u, u' \in \mathcal{R}^* \text{ such that } ur = u'r' \text{ and } us = u's' \ .$$

The degree map on $\mathcal{R}$ can be naturally extended to $\mathcal{Q}$ by defining

$$\begin{aligned} \deg : \quad & \mathcal{Q} \to \mathbb{Z} \cup \{-\infty\} \\ & s^{-1}r \mapsto \deg r - \deg s \ . \end{aligned}$$

Let $[\mathcal{Q}^*, \mathcal{Q}^*]$ be the commutator of $\mathcal{Q}^*$, i.e. the multiplicative group generated by $\{a^{-1}b^{-1}ab \,:\, a, b \in \mathcal{Q}^*\}$. Then $\mathcal{Q}^{\mathrm{ab}} = \mathcal{Q}^*/[\mathcal{Q}^*, \mathcal{Q}^*]$ is an abelian group called the multiplicative abelianization of $\mathcal{Q}^*$. There is a canonical homomorphism

$$\begin{aligned} \phi : & \mathcal{Q}^* \to \mathcal{Q}^{\mathrm{ab}} \\ & x \mapsto x \cdot [\mathcal{Q}^*, \mathcal{Q}^*] \ . \end{aligned}$$

Since the elements $(a^{-1}b^{-1}ab) \in [\mathcal{Q}^*, \mathcal{Q}^*]$ have degree $\deg(a^{-1}b^{-1}ab) = \deg(ab) - \deg(ba) = 0$, we can pass deg through ϕ in a well-defined manner: $\deg \phi(x) = \deg x$ for all $x \in \mathcal{Q}^*$. The following lemma was proved by Dieudonné [11] and can also be found in [12, §20].

Lemma 3 *There is a function* $\det : \mathcal{Q}^{n\times n} \to \mathcal{Q}^{\mathrm{ab}}$ *s.t. for all* $A, A' \in \mathcal{Q}^{n\times n}$, $k \in \mathcal{Q}$*:*

(i) $\det I = 1$*, where* I *is the identity matrix in* $\mathcal{Q}^{n\times n}$*.*

(ii) If A' *is obtained from* A *by an elementary row operation, then* $\det A' = \det A$*.*

(iii) If A' *is obtained from* A *by multiplying a row with* k*, then* $\det A' = \phi(k) \det A$*.*

Definition 4 A function $\det(\cdot)$ with the properties of Lemma 3 is called a Dieudonné determinant.

Note that contrary to the classical determinant, a Dieudonné determinant is generally not unique. For the remainder of the paper, consider $\det(\cdot)$ to be any given Dieudonné determinant for $\mathcal{Q}^{n\times n}$.

Lemma 4 *Let* $A \in \mathcal{Q}^{n\times n}$ *be in triangular form with non-zero diagonal elements* $d_0, \ldots, d_{n-1}$*. Then* $\det A = \prod_{i=0}^{n-1} \phi(d_i)$*.*

Proof: Since $d_i \neq 0$ for all i, we can multiply the ith row of A by d_i^{-1} and get a unipotent triangular matrix A'. Any unipotent triangular matrix can be obtained by elementary row operations from the identity matrix I. Thus

$$\det A \overset{\text{Lemma 3 (iii)}}{=} \left[\prod_{i=0}^{n-1} \phi(d_i)\right] \cdot \det A' \overset{\text{Lemma 3 (ii)}}{=} \left[\prod_{i=0}^{n-1} \phi(d_i)\right] \cdot \det I \overset{\text{Lemma 3 (i)}}{=} \prod_{i=0}^{n-1} \phi(d_i).$$

■

Clearly, the notion of weak Popov form generalises readily to matrices over $\mathcal{Q}$. We will now examine how this notion interacts with the Dieudonné determinant and introduce the concept of orthogonality defect. The statements in this section are all $\mathcal{Q}$ variants of the corresponding statements for $K[x]$ matrices, see [40].

Definition 5 The orthogonality defect of $V \in \mathcal{Q}^{n\times n}$ is $\Delta(V) := \deg V - \deg \det V$.

Lemma 5 *If* $V \in \mathrm{GL}_n(\mathcal{Q})$ *is in weak Popov form, then* $\Delta(V) = 0$*.*

Proof: See Appendix A ■

4.2 Complexity of Mulders–Storjohann

We can now bound the complexity of Algorithm 1 using arguments similar to those in [40]. These are in turn, the original arguments of [39] but finer grained by using the orthogonality defect.

Theorem 2 *Algorithm 1 with a full-rank input matrix $V \in \mathcal{R}^{m\times m}$ performs at most $m[\Delta(V)+1]$ simple LP-transformations. If $\mathcal{R}$ has derivation zero, Algorithm 1 has complexity $O(m^2\Delta(V)\operatorname{maxdeg}(V))$ over K.*

Proof: By Lemma 2, every simple LP-transformation reduces the ψ-value of one row with at least 1. So the number of possible simple LP-transformations is upper bounded by the difference of values of the input matrix V and the output matrix U, the matrices values being the sum of their rows'. More precisely, the number of iterations is upper bounded by:

$$\begin{aligned}&\textstyle\sum_{i=0}^{m-1}[m\deg \boldsymbol{v}_i + \operatorname{LP}(\boldsymbol{v}_i) - \big(m\deg \boldsymbol{u}_i + \operatorname{LP}(\boldsymbol{u}_i)\big)]\\ &= \operatorname{LP}(\boldsymbol{v}_0) + m\textstyle\sum_{i=0}^{m-1}[\deg \boldsymbol{v}_i - \deg \boldsymbol{u}_i]\\ &\le m[\deg V - \deg U + 1] = m[\Delta(V)+1],\end{aligned}$$

where the last equality follows from $\deg U = \deg\det U$ due to Lemma 5 and $\deg\det U = \deg\det V$ since the determinant is invariant under simple transformations.

One simple transformation consists of calculating $\boldsymbol{v}_j - \alpha x^\beta \boldsymbol{v}_i$, so for every coefficient in $\boldsymbol{v}_i$, we must apply θ^β, multiply by α and then add it to a coefficient in $\boldsymbol{v}_j$, each being in $O(1)$. Since $\deg \boldsymbol{v}_j \le \operatorname{maxdeg}(V)$ this costs $O(m\operatorname{maxdeg}(V))$ operations in K. ∎

Since $\Delta(V) \le \deg V$, the above theorem improves the immediate complexity bound of Mulders–Storjohann whenever $OD(V) \ll \deg(V)$.

As previously mentioned, a row reduction algorithm for Ore rings was given in [38]. This algorithm is slightly slower than Algorithm 1, is somewhat more complicated to implement, and also computes only a "row reduced basis", and not one in weak Popov form.

5 Shift-Register Problems and Decoding Gabidulin Codes

We now describe how to decode Gabidulin and Interleaved Gabidulin codes from errors and erasures using the tools introduced in the preceding section. In fact, we will use the tools to solve a very general form of multi-sequence shift-register synthesis over skew polynomials. We begin by introducing the codes and describe how to decode by solving such a shift-register problem: in Section 5.1 first correcting only errors, since in Section 5.2 for erasures also.

Gabidulin codes [10, 16, 51] are rank metric codes constructed using evaluation of skew polynomials from $\mathbb{F}_{q^m}[x;\cdot^q,0]$, for some prime power q and positive integer m. In the literature, they are often described using linearised polynomials instead of skew polynomials but this is mostly a difference in notation.

A Gabidulin code with parameters n, k ($k \le n \le m$) and code locators $g_1, \ldots, g_n \in \mathbb{F}_{q^m}$, which are linearly independent over $\mathbb{F}_q$, is given by the set

$$\mathcal{G}[n,k] = \big\{\boldsymbol{c} = (f(g_1),\ldots,f(g_n)) \,:\, f \in \mathbb{F}_{q^m}[x;\cdot^q,0],\, \deg f < k\big\} \subseteq \mathbb{F}_{q^m}^n$$

It is easy to prove that $\mathcal{G}[n,k]$ has dimension k, as an $\mathbb{F}_{q^m}$ vector space. The polynomial f corresponding to a codeword $\boldsymbol{c}$ is called the message polynomial.

Recall from Section 2, the evaluation map on any polynomial in $\mathbb{F}_{q^m}[x;{\cdot}^q,0]$ is an $\mathbb{F}_q$-linear map on the m-dimensional $\mathbb{F}_q$-vector space $\mathbb{F}_{q^m}$. This is why the g_i must be linearly independent over $\mathbb{F}_q$.

For some $\boldsymbol{v} \in \mathbb{F}_{q^m}^n$, the rank of $\boldsymbol{v}$, $\mathrm{rk}(\boldsymbol{v})$, is $\dim_{\mathbb{F}_q}(\langle v_1,\ldots,v_n\rangle)$. The rank distance, $\mathrm{dist}_{\mathrm{rk}}$, of two $\mathbb{F}_{q^m}^n$ vectors is simply the rank of their difference. $\mathcal{G}[n,k]$ is a maximal-rank distance separable code, i.e. $\mathrm{dist}_{\mathrm{rk}}(\boldsymbol{c}_1,\boldsymbol{c}_2) \geq n-k+1$ for all $\boldsymbol{c}_1,\boldsymbol{c}_2 \in \mathcal{G}[n,k]$ [16].

When decoding, the error model considered is usually that of *rank errors*, i.e. the receiver knows $\boldsymbol{r} = \boldsymbol{c}+\boldsymbol{e}$ for some $\boldsymbol{c} \in \mathcal{G}[n,k]$, and *the number of errors* is $\mathrm{rk}(\boldsymbol{e})$. Decoding algorithms have long been known for retrieving $\boldsymbol{c}$ from $\boldsymbol{r}$ as long as $\mathrm{rk}(\boldsymbol{e}) \leq (n-k)/2$ [16, 31, 51].

Interleaved Gabidulin code [32, 46, 53–55, 57] is the direct sum of several Gabidulin codes. The idea is similar to interleaved codes in Hamming metric: if the errors occur in "bursts", i.e. the errors experienced by each constituent Gabidulin codeword has a large overlap, then collaboratively decoding the error can provide a gain in decoding performance, assuming random errors. The errors have large overlap when their pairwise rank distance is relatively small.

More formally, let $\mathcal{G}_1 = \mathcal{G}[n,k_1],\ldots,\mathcal{G}_\ell = \mathcal{G}[n,k_\ell]$ be Gabidulin codes with $\mathcal{G}_i$ having code locators $g_{i,1},\ldots,g_{i,n}$. The corresponding Interleaved Gabidulin code is defined as

$$\mathcal{IG}[n,k_1,\ldots,k_\ell] = \bigoplus_{i=1}^{\ell} \mathcal{G}_i = \{\boldsymbol{c} = (\boldsymbol{c}_1,\ldots,\boldsymbol{c}_\ell) \,:\, \boldsymbol{c}_i \in \mathcal{G}_i\} \subseteq \left(\mathbb{F}_{q^m}^n\right)^\ell \cong \mathbb{F}_{q^m}^{n\ell}$$

5.1 Correcting errors only

Assume that an error of the form $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ occurs, with

$$\boldsymbol{e} = (\boldsymbol{e}_1,\ldots,\boldsymbol{e}_\ell) = (e_{1,1},\ldots,e_{1,n},e_{2,1},\ldots,e_{\ell,n}) \in \mathbb{F}_{q^m}^{n\ell}.$$

Now we consider *number of errors* as $t := \dim \mathcal{E}$, where

$$\mathcal{E} = \langle e_{1,1},\ldots,e_{1,n},e_{2,1},\ldots,e_{\ell,n}\rangle_{\mathbb{F}_q}$$

In the following, we describe an strategy for decoding Interleaved Gabidulin codes. Our formulation combines the ideas of [60, §3.2] and [54].

Recall from Section 2 the notion of "annihilator polynomial" $\mathcal{A}_{\mathcal{U}}$ for an $\mathbb{F}_q$-linear space $\mathcal{U} \subset \mathbb{F}_{q^m}$. Define the (unknown) error span polynomial $\Lambda = \mathcal{A}_{\mathcal{E}}$ and for all $i = 1,\ldots,\ell$ the (known) polynomials $G_i = \mathcal{A}_{\langle g_{i,1},\ldots,g_{i,n}\rangle_{\mathbb{F}_q}}$. For all $\boldsymbol{r}_i$, we define $\hat{r}_i$ to be the unique polynomial in $\mathbb{F}_{q^m}[x;{\cdot}^q,0]_{<n}$ such that

$$\hat{r}_i(g_{i,j}) = r_{i,j} \quad \forall j = 1,\ldots,n.$$

Note that if $n|m$, we can choose the $g_{i,j}$'s to be a basis of $\mathbb{F}_{q^n} \subseteq \mathbb{F}_{q^m}$ for any i, yielding $G_i = x^n - 1$.

The following theorem states a relation between $\Lambda, \hat{r}_i, G_i$ and the information polynomials f_i. It uses a formulation which is similar to Gao's key equation [20] for Reed–Solomon codes and is an extension of [59, Section 3.2.2] to Interleaved Gabidulin codes. As soon as a solution $(\Lambda, \Lambda f_1,\ldots,\Lambda f_\ell)$ of this key equation is found, the f_i can be obtained simply by division, so no further steps are necessary in contrast to syndrome-based decoding algorithms, e.g. [53].

Theorem 3 (Key Equation)

$$\Lambda\hat{r}_i \equiv \Lambda f_i \mod G_i \quad \forall i = 1, \ldots, \ell \tag{1}$$

Proof: For all i, j, by definition of $\hat{r}_i$ and Λ, we obtain

$$\big[\Lambda(\hat{r}_i - f_i)\big](g_{i,j}) = \Lambda(\hat{r}_i(g_{i,j}) - f_i(g_{i,j})) = \Lambda(r_{i,j} - c_{i,j}) = \Lambda(e_{i,j}) = 0,$$

Thus $T = \Lambda(\hat{r}_i - f_i)$ is a polynomial which has $\langle g_{i,1}, \ldots, g_{i,n}\rangle_{\mathbb{F}_q}$ in its zero set. By considering T's remainder when dividing on the right by $G_i = \mathcal{A}_{\langle g_{i,1},\ldots,g_{i,n}\rangle_{\mathbb{F}_q}}$, and by the minimality of $\deg G_i$, we conclude $G_i \mid T$, proving the congruence. ∎

The key equation motivates us to find a solution $(\lambda, \omega_1, \ldots, \omega_\ell)$ to a linearised version of the above equation:

$$\begin{aligned} \lambda\hat{r}_i &\equiv \omega_i \mod G_i \\ \deg\lambda &> \deg\omega_i - k_i \qquad\qquad \text{for } i = 1, \ldots, \ell\ , \end{aligned}$$

and such that $\deg\lambda$ is minimal. It is shown in [53] for an equivalent key equation that this strategy succeeds with high probability, if $t \leq \frac{\ell}{\ell+1}(n - \frac{1}{\ell}\sum_{i=1}^{\ell} k_i)$. When successful, f_i can then be extracted by right-dividing ω_i by λ.

5.2 Errors and erasures

Erasures in rank metric naturally occur in error correction of subspace codes constructed as lifted rank metric codes [55, 57]. Compared to erasures in Hamming metric, we need to distinguish between two different types of erasures, *row erasures* and *column erasures*. These terms stem from the interpretation of errors as matrices [57, 59], where row erasures correspond to an error matrix whose column space is given in form of a basis and in case of column erasures, a basis of the row space of the error matrix is given.

There exist several descriptions and decoding algorithms for correcting errors and erasures for Gabidulin codes [14, 18, 19, 48, 49, 55, 57, 59] and interleaved Gabidulin codes [29], [59, Section 4.4]. Here, we use the erasure description of [15], which is similar to [29], and combine them with the results of [59, Section 3.2.3] to obtain a Gao key equation for error and erasure correction of Interleaved Gabidulin codes.

For simplicity, we restrict ourselves to the case $n = m$. Thus, for every $i = 1, \ldots, \ell$, $\mathcal{G}_i = \{g_{i,1}, \ldots, g_{i,n}\}$ is a basis of $\mathbb{F}_{q^m}$ and has a unique dual basis $\mathcal{G}_i^{\perp} = \{g_{i,1}^{\perp}, \ldots, g_{i,n}^{\perp}\}$ [37]. Also, $G_i = x^m - 1$ for all i.

Consider again receiving $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$. The "erasures" correspond to side-information at the receiver on the structure of $\boldsymbol{e}$, whose nature we now explain. We consider that the error $\boldsymbol{e} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_\ell) \in (\mathbb{F}_{q^m}^n)^\ell$ is decomposed into three components $\boldsymbol{e} = \boldsymbol{e}^E + \boldsymbol{e}^R + \boldsymbol{e}^C$, where

- $\boldsymbol{e}^E$ is the *full error* of $\operatorname{rank}(\boldsymbol{e}^E) = \tau$
- $\boldsymbol{e}^R$ is the *row erasure* of $\operatorname{rank}(\boldsymbol{e}^R) = \varrho$
- $\boldsymbol{e}^C$ is the *column erasure* of $\operatorname{rank}(\boldsymbol{e}^C) = \gamma$

By rank decomposition, we can thus rewrite any sub-part $\boldsymbol{e}_i \in \mathbb{F}_{q^m}^n$ of the error as

$$\boldsymbol{e}_i = \sum_{j=1}^{\tau} a_j^E \mathbf{b}_{i,j}^E + \sum_{j=1}^{\varrho} a_j^R \mathbf{b}_{i,j}^R + \sum_{j=1}^{\gamma} a_j^C \mathbf{b}_{i,j}^C,$$

where for all i, j we consider at the receiver that:

- $a_j^E \in \mathbb{F}_{q^m}$ and $\mathbf{b}_{i,j}^E \in \mathbb{F}_q^n$ are both unknown,
- $a_j^R \in \mathbb{F}_{q^m}$ is known and $\mathbf{b}_{i,j}^R \in \mathbb{F}_q^n$ is unknown,
- $a_j^C \in \mathbb{F}_{q^m}$ is unknown and $\mathbf{b}_{i,j}^C \in \mathbb{F}_q^n$ is known.

Using only known variables, we can define the following ($b_{i,j,\kappa}^C$ is the κth entry of $\mathbf{b}_{i,j}^C$):

$$\begin{aligned} d_{i,j}^C &= \sum_{\kappa=1}^{n} b_{i,j,\kappa}^C g_\kappa^\perp \in \mathbb{F}_{q^m} \ , \\ \Gamma_i^C &= \mathcal{A}_{\langle d_{i,1}^C,\ldots,d_{i,\tau}^C\rangle_{\mathbb{F}_q}} \in \mathbb{F}_{q^m}[x;\cdot^q,0] \ , \\ \Lambda^R &= \mathcal{A}_{\langle a_1^R,\ldots,a_\varrho^R\rangle_{\mathbb{F}_q}} \in \mathbb{F}_{q^m}[x;\cdot^q,0], \end{aligned}$$

as well as the unknown *error-span polynomial*:

$$\Lambda^E = \mathcal{A}_{\langle \Lambda^R(a_1^E),\ldots,\Lambda^R(a_\tau^E)\rangle_{\mathbb{F}_q}}.$$

Finally, for any $a = \sum_{i=0}^{m-1} a_i x^i \in \mathbb{F}_{q^m}[x;\cdot^q,0]_{<m}$, we define its *full q-reverse* as $\bar{a} = \sum_{i=0}^{m-1} \bar{a}_i x^i$, where $\bar{a}_i = a_{-i \bmod m}^{q^i}$ (cf. [55]).

Using all of this we can prove the following key equation.

Theorem 4 (Error-Erasure Key Equation)

$$\Lambda^E \Lambda^R \hat{r}_i \overline{\Gamma_i^C} \equiv \Lambda^E \Lambda^R f_i \overline{\Gamma_i^C} \mod x^m - 1 \quad \forall\, i = 1,\ldots,\ell$$

Proof: See Appendix A. ∎

Since $\Lambda^R \hat{r}_i \overline{\Gamma_i^C}$ is known and $\deg(\Lambda^E \Lambda^R f_i \overline{\Gamma_i^C}) < \deg \Lambda^E + \varrho + k_i + \deg(\overline{\Gamma_i^C})$, we can reduce the problem to the linearised problem of finding $(\lambda, \omega_1, \ldots, \omega_\ell) \in \mathcal{R}^{\ell+1}$ with $\deg \lambda$ minimal such that

$$\begin{aligned} \lambda \cdot (\Lambda^R \hat{r}_i \overline{\Gamma_i^C}) &\equiv \omega_i \mod x^m - 1, \\ \deg \lambda &> \deg \omega_i - (k_i + \varrho + \deg(\overline{\Gamma_i^C})). \end{aligned}$$

A similar key equation was proved in [29] and it is claimed [29, Theorem 5] that the strategy of solving the linearised problem will succeed in finding Λ^E with probability greater than $1 - \frac{4}{q^m}$ if

$$\ell < \tau \leq \frac{\ell}{\ell+1}(\overline{d} - 1),$$

where $\overline{d} = \frac{1}{\ell}\sum_{i=1}^{\ell}(n - k_i + 1 - \varrho - \gamma)$ and with probablity 1 if

$$\tau < \frac{1}{2}\min_i\{n - k_i + 1 - \varrho - \gamma\}.$$

5.3 Shift Register Synthesis Problems

For the remainder of this section, we consider the following Multi-Sequence generalised Linear Skew-Feedback Shift Register (MgLSSR) synthesis problem, which as special cases includes both types of key equations introduced in the previous subsections.

Problem 1 (MgLSSR) Given skew polynomials $s_i, g_i \in \mathcal{R}$ and non-negative integers $\gamma_i \in \mathbb{N}_0$ for $i = 1, \ldots, \ell$, find skew polynomials $\lambda, \omega_1, \ldots, \omega_\ell \in \mathcal{R}$, with λ of minimal degree such that the following holds:

$$\begin{aligned} \lambda s_i &\equiv \omega_i \mod g_i \\ \deg \lambda + \gamma_0 &> \deg \omega_i + \gamma_i \end{aligned}$$

The solutions we describe works over any skew polynomial ring $\mathcal{R}$, and not just $\mathbb{F}_{q^m}[x; \cdot^q, 0]$. Over $K[x]$ the analogous problem is studied as multi-sequence shift-register synthesis [13], simultaneous Padé approximation [3], or vector rational function reconstruction [44].

We will first show how to solve such problems using row reduction by Mulders–Storjohann. We then present a refinement, the Demand–Driven algorithm, for the matrices arising from the above problem. The latter algorithm has the same complexity as [54], but there only the case where the g_i are all powers of x, and where all $\gamma_i = 0$ is handled.

In the sequel we consider a particular instance of Problem 1, so $\mathcal{R}$, $\ell \in \mathbb{N}$, and $s_i, g_i \in \mathcal{R}$, $\gamma_i \in \mathbb{N}_0$ for $i = 1, \ldots, \ell$ are arbitrary but fixed. We assume $\deg s_i \leq \deg g_i$ for all i since taking $s_i := (s_i \bmod g_i)$ yields the same solution to Problem 1.

Denote by $\mathcal{M}$ the set of all vectors $\boldsymbol{v} \in \mathcal{R}^{\ell+1}$ satisfying just the congruence requirement, i.e.,

$$\mathcal{M} := \left\{ (\lambda, \omega_1, \ldots, \omega_\ell) \in \mathcal{R}^{\ell+1} \mid \lambda s_i \equiv \omega_i \mod g_i \ \forall i = 1, \ldots, \ell \right\}. \tag{2}$$

Lemma 6 *$\mathcal{M}$ with component-wise addition and left multiplication by elements of $\mathcal{R}$ forms a left module over $\mathcal{R}$. The rows of M form a basis of $\mathcal{M}$, where*

$$M = \begin{pmatrix} 1 & s_1 & s_2 & \ldots & s_\ell \\ 0 & g_1 & 0 & \ldots & 0 \\ 0 & 0 & g_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & g_\ell \end{pmatrix} \tag{3}$$

Proof: To show that $\mathcal{M}$ is free, then since $\mathcal{R}^{\ell+1}$ is a left module over $\mathcal{R}$ and $\mathcal{M} \subset \mathcal{R}^{\ell+1}$, it suffices to show that $\mathcal{M}$ is closed under addition and left multiplication, by Theorem 12 in Appendix B. If $\boldsymbol{v} \in \mathcal{M}$, it follows from (2) that

$$v_0 s_i = v_i + a_i g_i \quad \forall i \in \{1, \ldots, \ell\} \tag{4}$$

for some $a_i \in \mathcal{R}$. If $\boldsymbol{v}, \boldsymbol{v}' \in \mathcal{M}$ then (4) implies that $(v_0 + v_0')s_i = (v_i + v_i') + (a_i + a_i')g_i$, $i = 1, \ldots, \ell$, which means that $\boldsymbol{v} + \boldsymbol{v}' \in \mathcal{M}$. If $\boldsymbol{v} \in \mathcal{M}$ and $b \in \mathcal{R}$ then from (4) we get $bv_0 s_i = bv_i + ba_i g_i$ and thus, $b\boldsymbol{v} \in \mathcal{M}$. Therefore $\mathcal{M}$ is indeed a left $\mathcal{R}$ module.

For the basis claim, then clearly the left span of M is a subset of $\mathcal{M}$. To see that $\mathcal{M}$ is a subset of the left span of M, consider again (4) for some $\boldsymbol{v} \in \mathcal{M}$ and the $a_i \in \mathcal{R}$ which must exist. That means

$$\boldsymbol{v} = v_0 \boldsymbol{u}_0 - a_1 \boldsymbol{u}_1 - \cdots - a_\ell \boldsymbol{u}_\ell \ ,$$

where $\boldsymbol{u}_0, \ldots, \boldsymbol{u}_\ell$ are the rows of M. Lastly, the rows of M must be linearly independent since M is in triangular form. ■

The above gives a simple description of all solutions of the congruence requirement of Problem 1 in the form of the row span of an explicit matrix M. The following theorem implies that computing the weak Popov form of M is enough to solve Problem 1. The strategy is formalised in Algorithm 2.

Theorem 5 *Let $\boldsymbol{w} = (\gamma_0, \ldots, \gamma_\ell)$. If V is a basis of $\mathcal{M}$ in $\boldsymbol{w}$-shifted weak Popov form, the row $\boldsymbol{v}$ of M with $\mathrm{LP}(\Phi_{\boldsymbol{w}}(\boldsymbol{v})) = 0$ is a solution to Problem 1.*

Proof: By Lemma 6 then $\boldsymbol{v}$ satisfies the congruence requirement of Problem 1. For the degree restriction of Problem 1, note that any $\boldsymbol{u} \in \mathcal{M}$ satisfies this restriction if and only if $\mathrm{LP}(\Phi_{\boldsymbol{w}}(\boldsymbol{u})) = 0$, since $\deg u_i + \gamma_i = \deg(\Phi_{\boldsymbol{w}}(\boldsymbol{u})_i)$. Furthermore, if this is the case, then $\deg(\Phi_{\boldsymbol{w}}(\boldsymbol{u})) = \deg u_0 + \gamma_0$. Thus, not only must $\boldsymbol{v}$ satisfy the degree restriction, but by Lemma 1, then v_0 also has minimal possible degree. ■

Algorithm 2 Solve Problem 1 by Row Reduction

Input: $s_i, g_i \in \mathcal{R}$ for $i = 1, \ldots, \ell$ and a shift $\boldsymbol{w} = (\gamma_0, \ldots, \gamma_\ell)$.
Output: Solution $\boldsymbol{v} = (\lambda, \omega_1, \ldots, \omega_\ell)$ of Problem 1.
1 Set up M as in (3).
2 Compute V as a $\boldsymbol{w}$-shifted weak Popov form of M.
3 **return** the row $\boldsymbol{v}$ of V having $\mathrm{LP}(\Phi_{\boldsymbol{w}}(\boldsymbol{v})) = 0$.

The complexity of Algorithm 2 is determined by Line 2. We can compute a $\boldsymbol{w}$-shifted weak Popov form of M by applying Algorithm 1 to $\Phi_{\boldsymbol{w}}(M)$, and then applying $\Phi_{\boldsymbol{w}}^{-1}$ to the result. In the following we investigate the computational complexity of Algorithm 2 if one does this. In Section 5.4 we modify Algorithm 1 to arrive at a new, often more efficient algorithm for computing the weak Popov form when starting with matrices of the special form that $\Phi_{\boldsymbol{w}}(M)$ has.

Remark 2 Using known properties for the weak Popov form, it is easy to prove how the polynomials returned by Algorithm 1 can be used as a basis for *all* solutions to the shift register problem. For decoding Gabidulin codes, this can be used to determine whether the found solution corresponds to a unique decoding, without incurring an additional cost. Another use is when the error has rank greater than $(n-k)/2$, where such a basis can be used to exhaustively search through the higher-degree solutions to the key equation, thereby correcting more errors (at a high computational cost).

In the following, let $\mu := \max_i\{\gamma_i + \deg g_i\}$. We can assume that $\gamma_0 < \mu$ since otherwise M is already in $\boldsymbol{w}$-shifted weak Popov form.

Theorem 6 *Over $\mathcal{R}$ with derivation zero, Algorithm 1 with input matrix $\Phi_{\boldsymbol{w}}(M)$ performs at most $(\ell+1)(\mu-\gamma_0+1)$ simple LP-transformations and performs $O(\ell^2\mu^2)$ operations over K.*

Proof: Since $\Phi_{\boldsymbol{w}}(M)$ is upper triangular, the degree of its determinant is

$$\deg \det \Phi_{\boldsymbol{w}}(M) = \deg\left(\phi(x^{\gamma_0}) \textstyle\prod_{i=0}^{\ell} \phi(g_i x^{\gamma_i})\right) = \gamma_0 + \textstyle\sum_{i=0}^{\ell}(\gamma_i + \deg g_i).$$

The degrees of the rows of $\Phi(M)$ are

$$\deg \Phi_{\boldsymbol{w}}(\boldsymbol{m}_0) = \max_i\{\gamma_i + \deg s_i\} \quad \text{and} \quad \deg \Phi_{\boldsymbol{w}}(\boldsymbol{m}_i) = \gamma_i + \deg g_i \quad \text{for } i \geq 1.$$

Thus, $\Delta(\Phi_{\boldsymbol{w}}(M)) = \max_i\{\gamma_i + \deg s_i\} - \gamma_0 \leq \mu - \gamma_0$. With $\operatorname{maxdeg}(\Phi_{\boldsymbol{w}}(M)) \leq \max_i\{\gamma_i + \deg s_i\} = \mu$, the statement follows from Theorem 2. ∎

5.4 Demand-Driven Algorithm

It was observed in [40] that the Mulders–Storjohann algorithm over $K[x]$ admits a "demand–driven" variant when applied to matrices coming from shift register problems: the algorithm stores and computes only the necessary coefficients of the working matrix. This means a much lower memory requirement, as well as a better complexity under certain conditions. Over $\mathcal{R}$, Algorithm 1 admits exactly the same speedup.

The central observation is that due to the special form of M, during the Mulders–Storjohann algorithm the first column suffices for reconstructing the rest. The Demand–Driven algorithm, Algorithm 3, therefore calculates just the first element of a vector whenever doing a simple LP-transformation. A bit more information, like degree and leading coefficient of the remaining rows, is also cached, in order to know which simple LP-transformations to apply. We begin with a technical lemma.

Lemma 7 *Consider Algorithm 1 with input $\Phi(M)$, and let $\tilde{g}_j = g_j x^{\gamma_j}$. Consider a variant where, we after a simple LP-transformation i on j, which replaces $\boldsymbol{v}_j$ with $\boldsymbol{v}'_j$, we instead replace it with $\boldsymbol{v}''_j = (v'_{j,0}, v'_{j,1} \bmod \tilde{g}_1, \ldots, v'_{j,\ell} \bmod \tilde{g}_\ell)$. This does not change the correctness of the algorithm or the upper bound on the number of simple LP-transformations performed.*

Proof: Correctness follows if we can show that each of the ℓ modulo reductions could have been achieved by a series of row operations on the current matrix V after the simple LP-transformation producing $\boldsymbol{v}'_j$. For each $h \geq 1$, let $\boldsymbol{g}_h = (0, \ldots, 0, \tilde{g}_h, 0, \ldots, 0)$, with position h non-zero.

During the algorithm, we will let J_h be a subset of the current rows in V having two properties: that $\boldsymbol{g}_h$ can be constructed as an $\mathcal{R}$-linear combination of the rows in J_h; and that each $\boldsymbol{v} \in J_h$ has $\psi(\boldsymbol{v}) \leq \psi(\boldsymbol{g}_h)$. Initially, $J_h = \{\boldsymbol{g}_h\}$.

After simple LP-transformations on rows not in J_h, the h'th modulo reduction is therefore allowed, since $\boldsymbol{g}_h$ can be constructed by the rows in J_h. On the other hand, consider a simple LP-transformation i on j where $\boldsymbol{v}_j \in J_h$, resulting in the row $\boldsymbol{v}'_j$. Then the h'th modulo reduction has no effect since $\psi(\boldsymbol{v}'_j) < \psi(\boldsymbol{v}_j) \leq \psi(\boldsymbol{g}_h)$. Afterwards, J_h is updated as $J_h = J_h \setminus \{\boldsymbol{v}_j\} \cup \{\boldsymbol{v}'_j, \boldsymbol{v}_i\}$. We see that J_h then still satisfies the two properties, since $\psi(\boldsymbol{v}_i) \leq \psi(\boldsymbol{v}_j) \leq \psi(\boldsymbol{g}_h)$.

Since $\psi(\boldsymbol{v}''_j) \leq \psi(\boldsymbol{v}'_j)$ the proof of Theorem 2 shows that the number of simple LP-transformations performed is still not greater than $(\ell + 1)[\Delta(V) + 1]$. ∎

Theorem 7 *Algorithm 3 is correct.*

Proof: We first prove that an intermediary algorithm, Algorithm 4, is correct using the correctness of Algorithm 1, and then prove the correctness of Algorithm 3 using Algorithm 4 and Lemma 7.

Starting from Algorithm 1, then Algorithm 4 is obtained by two simple modifications. For the first modifications, note that initially, when $V := \Phi_{\boldsymbol{w}}(M)$, then $\operatorname{LP}(\boldsymbol{v}_h) = h$ for

Algorithm 3 Demand–Driven algorithm for MgLSSR

Input: $\tilde{s}_j \leftarrow s_{1,j}x^{\gamma_j}$, $\tilde{g}_j \leftarrow g_j x^{\gamma_j}$ for $j = 1, \ldots, \ell$.
Output: The first column of a basis of $\mathcal{M}$ in $\boldsymbol{w}$-shifted weak Popov form.
1 $(\eta, h) \leftarrow (\deg, \mathrm{LP})$ of $(x^{\gamma_0}, \tilde{s}_1, \ldots, \tilde{s}_\sigma)$.
2 **if** $h = 0$ **then return** $(1, 0, \ldots, 0)$.
3 $(\lambda_0, \ldots, \lambda_\ell) \leftarrow (x^{\gamma_0}, 0, \ldots, 0)$.
4 $\alpha_j x^{\eta_j} \leftarrow$ the leading monomial of $\tilde{g}_j$ for $j = 1, \ldots, \ell$.
5 **while** $\deg \lambda_0 \leq \eta$ **do**
6 $\quad \alpha \leftarrow$ coefficient to x^η in $(\lambda_0 \tilde{s}_h \mod \tilde{g}_h)$.
7 $\quad$ **if** $\alpha \neq 0$ **then**
8 $\quad\quad$ **if** $\eta < \eta_h$ **then** swap $(\lambda_0, \alpha, \eta)$ and $(\lambda_h, \alpha_h, \eta_h)$.
9 $\quad\quad \lambda_0 \leftarrow \lambda_0 - \alpha/\theta^{\eta-\eta_h}(\alpha_h)x^{\eta-\eta_h}\lambda_h$.
10 $\quad (\eta, h) \leftarrow (\eta, h-1)$ **if** $h > 1$ **else** $(\eta - 1, \ell)$.
11 **return** $\left(\lambda_0 x^{-\eta_0}, \ldots, \lambda_\ell x^{-\eta_0}\right)$.

Algorithm 4 Intermediary algorithm for the correctness proof of Algorithm 3

Input: $V \leftarrow \Phi_{\boldsymbol{w}}(M)$.
Output: A basis V' of $\mathcal{M}$ in $\boldsymbol{w}$-shifted weak Popov form.
1 $(\eta, h) \leftarrow (\deg, \mathrm{LP})$ of $\boldsymbol{v}_0$.
2 **if** $h = 0$ **then return** $\Phi_{\boldsymbol{w}}^{-1}(V)$.
3 **while** $\deg v_{0,0} \leq \eta$ **do**
4 $\quad \alpha \leftarrow$ coefficient to x^η in $v_{0,h}$.
5 $\quad$ **if** $\alpha \neq 0$ **then**
6 $\quad\quad \eta_h \leftarrow \deg \boldsymbol{v}_h$.
7 $\quad\quad \alpha_h \leftarrow$ coefficient to x^{η_h} in $v_{h,h}$.
8 $\quad\quad$ **if** $\eta < \eta_h$ **then** swap $(\boldsymbol{v}_0, \alpha, \eta)$ and $(\boldsymbol{v}_h, \alpha_h, \eta_h)$.
9 $\quad\quad \boldsymbol{v}_0 \leftarrow \boldsymbol{v}_0 - \alpha/\theta^{\eta-\eta_h}(\alpha_h)x^{\eta-\eta_h}\boldsymbol{v}_h$.
10 $\quad (\eta, h) \leftarrow (\eta, h-1)$ **if** $h > 1$ **else** $(\eta - 1, \ell)$.
11 **return** $\Phi_{\boldsymbol{w}}^{-1}(V)$.

$h \geq 1$, and therefore the only possible simple LP-transformation must involve $\boldsymbol{v}_0$. We can maintain this property as a loop invariant throughout the algorithm by swapping $\boldsymbol{v}_0$ and $\boldsymbol{v}_{\mathrm{LP}(\boldsymbol{v}_0)}$ when applying a simple LP-transformation $\mathrm{LP}(\boldsymbol{v}_0)$ on 0.

The second modification is to keep an upper bound on the $(\deg, \mathrm{LP})$ of $\boldsymbol{v}_0$ throughout the algorithm: we initially simply compute the value and store as (η, h). Whenever we have applied a simple LP-transformation on $\boldsymbol{v}_0$ resulting in $\boldsymbol{v}_0'$, we know by Lemma 2 that $\psi(\boldsymbol{v}_0') < \psi(\boldsymbol{v}_0)$. Therefore, either $\deg \boldsymbol{v}_0' < \eta$ or $\deg \boldsymbol{v}_0' = \eta \wedge \mathrm{LP}(\boldsymbol{v}_0') < h$. This is reflected in a corresponding decrement of (η, h) in order of the ψ it bounds.

As a loop invariant we therefore have $\psi(\boldsymbol{v}_0) \leq \eta(\ell + 1) + h$. After an iteration, if this inequality is sharp, it simply implies that the α computed in the following iteration will be 0, and (η, h) will be correspondingly decremented once more. Note that we never set $h = 0$: when $\mathrm{LP}(\boldsymbol{v}_0) = 0$ then V must be in weak Popov form (since $\mathrm{LP}(\boldsymbol{v}_h) = h$ for $h > 0$). At this point, the while-loop will be exited since $\deg v_0 > \eta$.

Algorithm 4 is then simply the implementation of these modifications, and writing out in full what the simple LP-transformation does to $\boldsymbol{v}_0$.

This proves that Algorithm 4 is operationally equivalent to Algorithm 1. Obtaining Algorithm 3 from Algorithm 4 is relatively straightforward. The idea is simply to store only the necessary part of V and compute the rest on demand. Firstly, by Lemma 7 correctness would be maintained if the simple LP-transformation on Line 9 of Algorithm 4 was followed by the ℓ modulo reductions. In that case, we would have $v_{0,h} = (v_{0,0}\tilde{s}_h$

mod $\tilde{g}_h$), so only storing $v_{0,0}$ suffice for reconstructing $\boldsymbol{v}_0$. For simple LP-transformations, we then need the entire first column of V which is stored in Algorithm 3 as $(\lambda_0, \ldots, \lambda_\ell)$.

Line 6 of Algorithm 3 is now the computation of the needed coefficient of $v_{0,h}$ at the latest possible time, instead of an inspection in an already computed polynomial.

As $\deg \boldsymbol{v}_h$ is used in Line 6 of Algorithm 4, we need to store and maintain this between iterations; this is the variables $\eta_1, \ldots, \eta_\ell$. To save some redundant computation of coefficients, the x^{η_h}-coefficient of $v_{h,h}$ is also stored as α_h.

This proves that Algorithm 3 is operationally equivalent to Algorithm 4, which finishes the proof of correctness. ∎

It is interesting to remark that a direct proof of correctness of Algorithm 3 without using Algorithm 1 is rather cumbersome, and would likely have to establish technical properties on the $(\lambda_h \tilde{s}_h \bmod \tilde{g}_h)$ during the algorithm.

Now we turn to the computational complexity of Algorithm 3. This will depend entirely on how Line 6 is realised, as is described by the following proposition. The algorithm will turn out to be faster on special cases of the g_h, such as when they are powers of x, and we elaborate on such cases and the general case afterwards. Recall that we let $\mu := \max_i\{\gamma_i + \deg g_i\}$.

Proposition 1 *Over $\mathcal{R}$ with derivation zero, Algorithm 3 has computational complexity $O(\ell\mu^2 + \sum_{h=1}^{\ell}\sum_{\eta=0}^{\mu-1} T_{h,\eta})$, where $T_{h,\eta}$ bounds the complexity of running Line 6 for those values of h and η.*

Proof: Clearly, all steps of the algorithm are essentially free except Line 6 and Line 9. Observe that every iteration of the while-loop decrease an *upper bound* on the value of row 0, whether we enter the if-branch in Line 7 or not. So by the arguments of the proof of Theorem 6, the loop will iterate at most $O(\ell\mu)$ times in which each possible value of $(h, \eta) \in \{1, \ldots, \ell\} \times \{0, \ldots, \mu - 1\}$ will be taken at most once. Each execution of Line 9 costs $O(\mu)$ since the λ_j all have degree at most μ. ∎

The cost of Line 6 will depend on how we realise it. We introduce some notation to describe this. For any $f \in \mathcal{R}$, we let $f[i]$ denote the coefficient to x^i in f, with $f[i] := 0$ if $i < 0$ or $i > \deg f$. Note first that for any η, we have

$$(\lambda \tilde{s}_h \bmod \tilde{g}_h)[\eta] = (\lambda s_h \bmod g_h)[\eta - \gamma_h] \ ,$$

and thus, we can essentially ignore the shift when considering how to compute Line 6. Now, for any $g \in \mathcal{R}$, let Ξ_g be the following matrix, where $t = \deg g$:

$$\Xi_g = \left(\begin{array}{c|ccc} \multicolumn{4}{c}{\boldsymbol{I}_{t\times t}} \\ \hline (x^t \bmod g)[0] & \cdots & (x^t \bmod g)[t-1] \\ \vdots & & \vdots \\ (x^{2t-2} \bmod g)[0] & \cdots & (x^{2t-2} \bmod g)[t-1] \end{array}\right) \in K^{(2t-1)\times t}$$

Consider an execution of Line 4 for given $h, \eta, \lambda_0, \tilde{s}_h$ and $\tilde{g}_h$, and denote by $t_h = \deg \tilde{g}_h$ and by $u = \lambda_0 \tilde{s}_h \bmod \tilde{g}_h$. Clearly $u[\eta] = 0$ when $\eta \geq t_h$. For lower values of η, we have the following equality:

$$(u[0], \ldots, u[t_h - 1]) = \boldsymbol{\lambda}\, \mathrm{II}_h\, \Xi_{\tilde{g}_h} \ ,$$

where $\boldsymbol{\lambda} = (\lambda[0], \ldots \lambda[t_h - 1])$, and where Π_h is given by

$$\Pi_h = \begin{pmatrix} \tilde{s}_h[0] & \cdots & \tilde{s}_h[t_h-1] & & \\ & \theta(\tilde{s}_h[0]) & \cdots & \theta(\tilde{s}_h[t_h-1]) & \\ & & \ddots & & \ddots \\ & & & \theta^{t_h-1}(\tilde{s}_h[0]) & \cdots \; \theta^{t_h-1}(\tilde{s}_h[t_h-1]) \end{pmatrix} \in K^{t_h \times (2t_h-1)}$$

When the $\Xi_{\tilde{g}_h}$ are sparse, this immediately leads to an efficient realisation of Line 6:

Lemma 8 *In the context of Proposition 1, assume that $\Xi_{\tilde{g}_h}$ has been precomputed and cached. Then we can set $\sum_{\eta=0}^{\mu-1} T_{\eta,h} \in O(\mathrm{wt}(\Xi_{\tilde{g}_h})\mu)$ where $\mathrm{wt}(\Xi_{\tilde{g}_h})$ denotes the number of non-zero entries of $\Xi_{\tilde{g}_h}$.*

Proof: By the above discussion, $u[\eta] = \boldsymbol{\lambda}\,\Pi_h \boldsymbol{\xi}_{h,\eta}$, where $\boldsymbol{\xi}_{h,\eta}$ is the ηth column of $\Xi_{\tilde{g}_h}$ for $\eta < t_h - 1$, and $u[\eta] = 0$ otherwise. To compute $u[\eta]$ in the first case, we can first compute those elements of $\boldsymbol{\lambda}\Pi_h$ that correspond to non-zero elements of $\boldsymbol{\xi}_{h,\eta}$, and then do a dot-product. The cost of this will be in $O((\mathrm{wt}(\boldsymbol{\xi}_{h,\eta})(\mu+1))$. The lemma's claim follows directly. ∎

In many applications—e.g. for decoding Interleaved Gabidulin codes as in Section 5—one would use the same g_h for multiple s_h; in this case, an actual precomputation of the $\Xi_{\tilde{g}_h}$ makes sense. The computation of all the $\Xi_{\tilde{g}_h}$ can be done in $O(\ell\mu^2)$, however, so it would in any case not incur an asymptotic cost to always perform it just before Algorithm 3.

Corollary 1 *Let $\mathcal{R}$ have derivation zero. For any $g \in \mathcal{R}$ where $\deg_2(g) < \frac{1}{2}\deg(g)$, then*

$$\mathrm{wt}(\Xi_g) < \deg g \cdot \mathrm{wt}(g) + \deg_2 g \cdot \min(\mathrm{wt}(g)^2,\ \deg_2 g)$$

where $\deg_2(g)$ denotes the second-largest non-zero monomial of g, and $\mathrm{wt}(g)$ denotes the number of non-zero monomials.[3]

Proof: Write $g = g_t x^t + g'$, where $\deg g' = \deg_2 g$, and let $w = \mathrm{wt}(g') = \mathrm{wt}(g) - 1$. Then $(x^t \bmod g) = -g_t^{-1} g'$, so the t'th row of Ξ_g has weight w. Similarly, $x^{t+i} \equiv -x^i g_t^{-1} g' \mod g$, so for any $i < t - \deg_2 g$ then $(x^{t+i} \bmod g) = -x^i g_t^{-1} g'$, and therefore the $(t+i)$'th row of Ξ_g also has weight w. This is depicted on Figure 1.

For the $(t+i)$'th row when $i \geq t - \deg_2 g$, then $x^{t+i} \bmod g$ can be written as $h_\top x^i + h_\perp$, where $\deg h_\top < t - i$, $\mathrm{wt}(h_\top) < w$, and

$$\deg h_\perp < \deg_2 g + i - (t - \deg_2 g) < 2\deg_2 g\ .$$

Furthermore, $\mathrm{wt}(h_\perp) < (w - \mathrm{wt}(h_\top))w$: this is because each non-zero monomial of g' becomes at most w monomials when they have been shifted to degree t and reduced modulo g.

In total, the number on non-zero entries in Ξ_g is upper-bounded by

$$t + tw + \min(w^2 \deg_2 g,\ 2(\deg_2 g)^2)$$

∎

[3] In Theorem 3 of the WCC extended abstract [28], a complexity estimate was given for Algorithm 3 which unfortunately is slightly incorrect: in the terms of this paper, it claims $\mathrm{wt}(\Xi_g) \in O(\deg g \cdot \mathrm{wt}(g))$ when $\deg_2 g < \deg g/2$. As seen by this corollary, that is not generally true, e.g. if $\deg_2 g \in O(\deg g)$ while $\mathrm{wt}(g) \in \Omega(1)$ and $\mathrm{wt}(g)^2 \in o(\deg g)$.

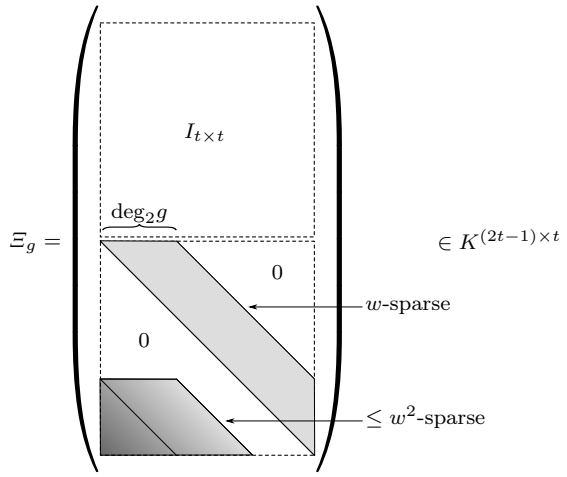

Figure 1 Depiction of Ξ_g in the proof of Corollary 1.

For decoding of Interleaved Gabidulin codes, two important cases are $g_i = x^k$ (syndrome decoding [54]) and $g_i = x^n - 1$ (Gao-type decoding as in Section 5 or [60, §3.2]), and the above immediately implies that these cases are fast:

Corollary 2 *For $\mathcal{R}$ with derivation zero, then Algorithm 3 can be realised in complexity $O(\ell\mu^2)$ if each g_h has the form $x^t + c$ for some $t \in \mathbb{Z}_+$ and $c \in K$.*

Remark 3 Algorithm 3 also has good *memory complexity* when Line 6 is realised using Lemma 8: if the Ξ_g are sparse matrices, they require only memory $O(\mathrm{wt}(\Xi_g))$ to store, and the rest of Algorithm 3 clearly uses only $O(\ell\mu)$ memory. This can be compared with the memory complexity $O(\ell^2\mu)$ of Algorithm 1.

For completely general g_h, Proposition 1 clearly implies that Algorithm 3 can be done in $O(\ell\mu^3)$. We can do slightly better however:

Proposition 2 *For $\mathcal{R}$ with derivation zero, Algorithm 3 can be realised in complexity $O(\ell\,\mathsf{MM}(\mu))$, where $O(\mathsf{MM}(\mu))$ denotes the cost of multiplying two $K^{\mu\times\mu}$ matrices.*

Proof: Consider $u, \boldsymbol{\lambda}, \Pi_h$ as in the proof of Lemma 8. If we first compute $\Pi_h \Xi_{\tilde{g}_h}$, then afterwards $u[\eta]$ can be computed in time $O(\mu)$ as a dot-product of $\boldsymbol{\lambda}$ and the ηth column of $\Pi_h \Xi_{\tilde{g}_h}$. Thus, if we prepend Algorithm 3 with the computation of the ℓ matrices $\Pi_h \Xi_{\tilde{g}_h}$, costing a total of $O(\ell\mathsf{MM}(\mu))$, the execution of Algorithm 3 itself will cost only a further $O(\ell\mu^2)$ by Proposition 1. ∎

Remark 4 When all g_i are powers of x, Algorithm 3 bears a striking similarity to the Berlekamp–Massey-variant for multiple shift-registers [54]. By Corollary 2, the algorithms have the same running time in this case. Using the language of modules, we obtain a more general algorithm with a conceptually simpler proof, and we can much more readily realise algebraic properties of the algorithm, as mentioned in Remark 2.

6 Decoding Mahdavifar–Vardy Codes

Mahdavifar–Vardy (MV) codes [33, 35] are subspace codes constructed by evaluating powers of skew polynomials at certain points. We will describe how one can use row reduction to carry out the most computationally intensive step of the MV decoding algorithm given in [35]. The decoding of MV codes is heavily inspired by the Guruswami–Sudan algorithm for Reed–Solomon codes [22]; our row reduction approach is similarly inspired by fast module-based algorithms for realising the Guruswami–Sudan, [6, 8, 26].

We only give an overview of MV codes and the other parts of the decoding procedure here.

Let $n, m \in \mathbb{Z}_+$ be such that $1 \leq n \leq m$. Consider the space $\mathcal{W} = \langle \alpha_1, \ldots, \alpha_n \rangle \oplus \mathbb{F}_{q^{nm}}^{\ell}$, where the $\alpha_i \in \mathbb{F}_{q^{nm}}$ are n carefully selected elements specified in [35, Section IV.A]. They satisfy that all the elements $\alpha_i^{q^j}$ for $i = 1, \ldots, n$ and $j = 0, \ldots, m-1$ are linearly independent over $\mathbb{F}_q$, but apart from that their precise expression will not be important for us. $\mathcal{W}$ is then an $n(\ell m+1)$-dimensional $\mathbb{F}_q$-vector space. The Grassmanian $\mathrm{Gr}(n, \mathcal{W})$ is the set of all n-dimensional $\mathbb{F}_q$-subspaces of $\mathcal{W}$. A Mahdavifar–Vardy code $\mathcal{C}_{\mathrm{MV}} \subseteq \mathrm{Gr}(n, \mathcal{W})$ of dimension $k \leq nm$ is defined as

$$\mathcal{C}_{\mathrm{MV}} = \left\{ \langle \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \rangle \,:\, \boldsymbol{v}_i = (\alpha_i, f(\alpha_i), f^2(\alpha_i), \ldots, f^{\ell}(\alpha_i)) \wedge f \in \mathbb{F}_q[x; \cdot^q, 0]_{<k} \right\}.$$

That is, the codewords of $\mathcal{C}_{\mathrm{MV}}$ are n-dimensional $\mathbb{F}_q$-subspaces of the ambient space $\mathcal{W}$, i.e. $\mathcal{C}_{\mathrm{MV}} \subseteq \mathrm{Gr}(n, \mathcal{W})$, As with Gabidulin codes, the polynomial $f \in \mathbb{F}_q[x; \cdot^q, 0]_{<k}$, from which a given codeword is obtained, is called the *message polynomial*. We stress that the code will consist of $\mathbb{F}_q$ vector spaces spanned by $\mathbb{F}_{q^{nm}}$-elements, while message polynomials are over $\mathbb{F}_q$. Note in particular that $\mathbb{F}_q[x; \cdot^q, 0] \cong \mathbb{F}_q[x]$.

We endow $\mathrm{Gr}(n, \mathcal{W})$, and therefore our code, with the *subspace metric* [24]:

$$\mathrm{dist}_{\mathrm{S}}(\mathcal{U}, \mathcal{V}) = \dim_{\mathbb{F}_q}(\mathcal{U} \oplus \mathcal{V}) - \dim_{\mathbb{F}_q}(\mathcal{U} \cap \mathcal{V}) \,, \qquad \mathcal{U}, \mathcal{V} \in \mathrm{Gr}(n, \mathcal{W})$$

As error model we use the operator channel as in [35], i.e., for some $0 \leq t \leq \ell nm$ and $0 \leq \rho \leq n$, the receiver obtains the result of the codeword through the following probabilistic mapping:

$$\begin{aligned} \mathcal{C}_{\mathrm{MV}} &\to \mathrm{Gr}(r = n - \rho + t, \mathcal{W}) \\ \mathcal{V} &\mapsto \mathcal{U} := \mathcal{V}' \oplus \mathcal{F}, \end{aligned}$$

where $\mathcal{V}'$ is an $(n-\rho)$-dimensional subspace of $\mathcal{V}$ and $\dim \mathcal{F} = t$ with $\mathcal{F} \cap \mathcal{C}_{\mathrm{MV}} = \{0\}$; we say that ρ *erasures* and t *errors* occurred. Note that $r = n - \rho + t$ is immediately known by the receiver as $\dim \mathcal{U}$, while ρ and t independently are unknown.

Mahdavifar and Vardy describe a decoding algorithm for MV codes consisting of two steps [35, Section IV.C], strongly inspired by the Guruswami–Sudan algorithm [22]:

1. Find a non-zero formal polynomial $Q \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0][y]$ satisfying certain interpolation properties.
2. Find the message polynomial $f \in \mathbb{F}_q[x; \cdot^q, 0]_{<k}$ as a root of Q.

More precisely, consider a decoding instance where $\mathcal{U}$ is the received subspace, and let $\{(x_i, y_{1,i}, \ldots, y_{\ell,i}) \,:\, i = 1, \ldots, r\}$ be a basis of $\mathcal{U}$. The "interpolation step" is now to

find a Q of the following form[4]:

$$Q(X, Y_1, \ldots, Y_\ell) = Q_0(X) + \sum_{i=1}^{\ell} Q_i(Y_i) \ ,$$

with all $Q_i \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$, and which satisfies:

$$Q(x, y_1, \ldots, y_\ell) = 0 \qquad \forall\, (x, y_1, \ldots, y_\ell) \in \mathcal{P}, \tag{5}$$

$$\deg Q_i < \chi - 1 - i(k-1) \qquad \forall\, i = 0, \ldots, \ell, \tag{6}$$

where χ and $\mathcal{P}$ are given by

$$\chi = \left\lceil \frac{mr+1}{\ell+1} + \frac{1}{2}\ell(k-1) \right\rceil$$
$$\mathcal{P} = \left\{ (x_i^{q^j}, y_{1,i}^{q^j}, \ldots, y_{\ell,i}^{q^j}) \ : \ {}^{i=1,\ldots,r}_{j=0,\ldots,m-1} \right\}.$$

All the $x_i^{q^j}$ are distinct, due to $x_i \in \langle \alpha_1, \ldots, \alpha_n \rangle$ and the properties of the α_i, as shown in [35, Lemma 9].

In the second step, root-finding, one should then find the list of all $f \in \mathbb{F}_q[x; \cdot^q, 0]_{<k}$ satisfying $Q(X, f(X), f^2(X), \ldots, f^\ell(X)) = 0$.

[35, Lemma 8] shows that a satisfactory Q always exists. Furthermore, [35, Theorem 12] states that if

$$\ell\rho + t < n\ell - \frac{1}{2}\ell(\ell+1)\frac{k}{m} \ , \tag{7}$$

then the original message polynomial f is found among the roots of Q. (7) can therefore be considered as an error-erasure decoding radius of the decoding algorithm.

Mahdavifar and Vardy describe a linearised version of the Roth–Ruckenstein algorithm [35] for finding the roots of Q. We have to point out that the complexity analysis of that algorithm has severe issues[5], and it is outside the scope of this paper to amend them; however, the algorithm possibly has a complexity of $O(\ell^2 rmk)$.

The interpolation step can be accomplished using a skew-variant of the Kötter–Nielsen–Høholdt algorithm [43], given by Xie, Lin, Yan and Suter [61], with a reported complexity $O(\ell^2 m^2 rn)$ over $\mathbb{F}_{q^{nm}}$.

In the following, we give an alternative method for solving the interpolation step using the framework of Section 3. The method will also have complexity $O(\ell r^2 m^2)$ over $\mathbb{F}_{q^{nm}}$. Since $r < \ell n$, this is at least as fast as the algorithm of [61]. The algorithm first finds a basis of the module spanning all possible interpolation polynomials, and then brings it into a shifted weak Popov form to obtain an interpolation polynomial with low enough weighted degree. Algorithm 1 can be directly applied for this, but to obtain a better complexity, we show in Section 6.2 how a better complexity can be obtained by "walking" between weak Popov forms through a sequence of shifts.

[4] Here, we explicitly add the different formal variables to the $\mathbb{F}_{q^m}[x; \cdot^q, 0]$ polynomials.

[5] There are two problems: 1) It is not proven that the recursive calls will not produce many spurious "pseudo-roots" which are sifted away only at the leaf of the recursions; and 2) the cost analysis ignores the cost of computing the shifts $Q(X, Y^q + \gamma Y)$. Issue 1 is necessary to resolve for assuring polynomial complexity. For the Roth–Ruckenstein this is proved as [50, Proposition 6.4], and an analogous proof might carry over. Issue 2 is critical since these shift dominate the complexity: assuming the algorithm makes a total of $O(\ell k)$ recursive calls to itself, then $O(\ell k)$ shifts need to be computed, each of which costs $O(\ell \deg_x Q) \subset O(\ell r m)$.

Note that Mahdavifar and Vardy presented in [34] a code construction which can be decoded "with multiplicities" for allowing a better decoding radius and rate. We are not considering that construction here, though it is interesting as potential future work.

6.1 Interpolation step of Mahdavifar–Vardy decoding using Row Reduction

Let $\mathcal{M}$ be the set of vectors $(Q_0, Q_1, \ldots, Q_\ell) \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]^{\ell+1}$ corresponding to all formal multivariate polynomials satisfying the interpolation requirement (5) (but not necessarily the degree requirement). We will speak of "evaluation" of such elements of $\mathbb{F}_{q^{nm}}[x; \cdot^q, 0]^{\ell+1}$ at points in $\mathbb{F}_{q^m}^{\ell+1}$ by considering them as $(\ell+1)$-ary formal polynomials over $\mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$ of the form of Q.

Lemma 9 *$\mathcal{M}$ is a left $\mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$-module.*

Proof: $\mathcal{M}$ is closed under addition since $a(\alpha) + b(\alpha) = (a+b)(\alpha)$ for all $a, b \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$ and $\alpha \in \mathbb{F}_{q^m}$. Let $f \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$, $\boldsymbol{u} = (Q_0, Q_1, \ldots, Q_\ell) \in \mathcal{M}$. Then the evaluation of the multivariate polynomial representation of $f \cdot \boldsymbol{u} = (f \cdot Q_0, \ldots, f \cdot Q_\ell) \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]^{\ell+1}$ at a point $(x, y_1, \ldots, y_L) \in \mathcal{P}$ is

$$\begin{aligned}(f \cdot Q_0)(x) + \sum_{i=1}^{\ell} (f \cdot Q_i)(y_i) &= f\left(Q_0(x)\right) + \sum_{i=1}^{\ell} f\left(Q_i(y_i)\right) \\ &= f\left(Q_0(x) + \sum_{i=1}^{\ell} Q_i(y_i)\right) = f(0) = 0\end{aligned}$$

Thus, $f \cdot \boldsymbol{u} \in \mathcal{M}$. ∎

To complete the interpolation step, we therefore need to find a low weighted-degree element of $\mathcal{M}$. Using row reduction, we will in fact find the minimal. Since we know that there must exist a satisfactory Q, the minimal that we will find must in particular be satisfactory.

In order to apply the row reduction approach, we first need to find a basis of the module $\mathcal{M}$. Note that by Theorem 12 on page 31, then $\mathcal{M}$ must admit a basis, since it is a sub-module of $\mathbb{F}_{q^{nm}}[x; \cdot^q, 0]^{\ell+1}$. The following definitions turn out to be useful for explicitly describing such a basis. For each $j \in \{1, \ldots, \ell\}$, let $R_j \in \mathbb{F}_{q^{nm}}[x; \cdot^q, 0]$ be such that

$$R_j\left(x_i^{q^v}\right) = y_{j,i}^{q^v} \quad \forall i = 1, \ldots, r,\ \forall v = 0, \ldots, m-1$$

and let

$$G := \mathcal{A}_{\langle x_1, x_1^q, x_1^{q^2}, \ldots, x_1^{q^{m-1}}, x_2, x_2^q, \ldots, x_r^{q^{m-1}} \rangle},$$

using the annihilator polynomial notation from Section 2. By the remarks in that section, and since all the $x_i^{q^v}$ are distinct, such R_j must exist, and we can choose them such that $\deg R_j < rm$. Also $\deg G = rm$.

Lemma 10 *The rows of M form a basis of $\mathcal{M}$:*

$$M = \begin{pmatrix} \boldsymbol{m}_0 \\ \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \\ \vdots \\ \boldsymbol{m}_\ell \end{pmatrix} = \begin{pmatrix} G & 0 & 0 & \ldots & 0 \\ -R_1 & 1 & 0 & \ldots & 0 \\ -R_2 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -R_\ell & 0 & 0 & \ldots & 1 \end{pmatrix}$$

Proof: "$\supseteq$": All rows $\boldsymbol{m}_j$ are in $\mathbb{F}_{q^{nm}}[x;\cdot^q,0]^{\ell+1}$. It remains to show that they all vanish at the points $\mathcal{P}$. Let $(x_i^{q^v}, y_{1,i}^{q^v}, \ldots, y_{\ell,i}^{q^v}) \in \mathcal{P}$. Then, by definition,

$$\begin{aligned} \boldsymbol{m}_0: &\quad G(x_i^{q^v}) = 0 \\ \boldsymbol{m}_j: &\quad 1(y_{j,i}^{q^v}) - R_j(x_i^{q^v}) = y_{j,i}^{q^v} - R_j(x_i^{q^v}) = 0\ , \quad j = 1, \ldots, \ell \end{aligned}$$

"$\subseteq$": Let $\boldsymbol{v}_\ell := (Q_0, \ldots, Q_\ell) \in \mathcal{M}$. Then we can write

$$\begin{aligned} \boldsymbol{v}_{\ell-1} &:= \quad \boldsymbol{v}_\ell - v_{\ell,\ell} \cdot \boldsymbol{m}_\ell &&= (v_{\ell-1,0}, \ldots, v_{\ell-1,\ell-1}, 0) \\ \boldsymbol{v}_{\ell-2} &:= \boldsymbol{v}_{\ell-1} - v_{\ell-1,\ell-1} \cdot \boldsymbol{m}_{\ell-1} &&= (v_{\ell-2,0}, \ldots, v_{\ell-2,\ell-2}, 0, 0) \\ &\ \ \vdots \\ \boldsymbol{v}_0 &:= \quad \boldsymbol{v}_1 - v_{1,1} \cdot \boldsymbol{m}_1 &&= (v_{0,0}, 0, \ldots, 0). \end{aligned}$$

Since $\boldsymbol{v}_\ell \in \mathcal{M}$, and each $\boldsymbol{m}_j \in \mathcal{M}$, we conclude that all the $\boldsymbol{v}_j \in \mathcal{M}$ and in particular $\boldsymbol{v}_0 \in \mathcal{M}$. Thus for any $(x_i, \ldots) \in \mathcal{P}$ we must have $v_{0,0}(x_i) = 0$, and so all of $\langle x_1, x_1^q, x_1^{q^2}, \ldots, x_1^{q^{m-1}}, x_2, x_2^q, \ldots, x_r^{q^{m-1}} \rangle$ are zeroes of $v_{0,0}$. This means G must right-divide $v_{0,0}$: for otherwise, the division would yield a non-zero remainder $B \in \mathbb{F}_{q^{nm}}[x;\cdot^q,0]$ with $\deg B < \deg G$ but with the same zero set, contradicting the minimality of G.

Summarily, $\boldsymbol{v}_0 = A \cdot \boldsymbol{m}_0$ for some $A \in \mathbb{F}_{q^{nm}}[x;\cdot^q,0]$, and hence $\boldsymbol{v}_\ell$ is an $\mathbb{F}_{q^{nm}}[x;\cdot^q,0]$-linear combination of the rows of M. ∎

Theorem 8 *Let $\boldsymbol{w} = (0, (k-1), \ldots, \ell(k-1))$ and let V be a basis of $\mathcal{M}$ in $\boldsymbol{w}$-shifted weak Popov form. If $\boldsymbol{v}$ is a row of V with minimal $\boldsymbol{w}$-shifted degree $\deg \Phi_{\boldsymbol{w}}(\boldsymbol{v})$, then it is an interpolation polynomial satisfying* (5) *and* (6)*.*

Proof: Any row of V satisfies (5) because it is in $\mathcal{M}$. According to [35, Lemma 8] there is an interpolation polynomial $\boldsymbol{u} = (Q_0, Q_1, \ldots, Q_\ell)$ satisfying the degree conditions (6). By the choice of $\boldsymbol{v}$ and by Lemma 1 on page 5, then $\deg \Phi_{\boldsymbol{w}}(\boldsymbol{v}) \leq \deg \Phi_{\boldsymbol{w}}(\boldsymbol{u})$. But then if $h = \mathrm{LP}(\boldsymbol{u})$ we have that for any i:

$$\deg\left(v_i x^{i(k-1)}\right) \leq \deg \Phi_{\boldsymbol{w}}(\boldsymbol{u}) = \deg\left(Q_h x^{h(k-1)}\right) \leq m$$

Hence, $\boldsymbol{v}$ satisfies (6) and is a desired interpolation polynomial. ∎

Algorithm 5 outlines the implied decoding procedure of MV codes. The following theorem provides a complexity estimate of Algorithm 5, using the Mulders–Storjohann algorithm for row reduction.

Algorithm 5 Find Interpolation Polynomial by Row Reduction

Input: Interpolation points $\mathcal{P}$ and $\boldsymbol{w} = (0, (k-1), \ldots, \ell(k-1))$.
Output: Interpolation polynomial satisfying (5) and (6).
1 Set up M as in (3).
2 Compute $\boldsymbol{w}$-shifted weak Popov form V of M.
3 **return** row $\boldsymbol{v}$ which has minimal $\boldsymbol{w}$-shifted degree $\deg \Phi_{\boldsymbol{w}}(\boldsymbol{v})$.

Theorem 9 *Algorithm 5 has complexity $O(\ell^3 r^2 m^2)$ if Line 2 is computed using Algorithm 1.*

Proof: The computation of the minimal subspace polynomial G can be done in time $O((\deg G)^2) = O(r^2m^2)$, [57], and the ℓ many R_j's can each be computed in time $O((\deg(R_j)^2) \subseteq O(r^2m^2)$ using the algorithm from [56], yielding in total $O(\ell r^2m^2)$. The degrees of the nonzero entries in $\Phi_{\boldsymbol{w}}(M)$ are component-wise upper bounded by:

$$\begin{pmatrix} rm & & & \\ rm & (k-1) & & \\ rm & & 2(k-1) & \\ \vdots & & & \ddots \\ rm & & & \ell(k-1) \end{pmatrix}$$

Since $\chi - 1 - \ell(k-1) > \deg Q_\ell \geq 0$, we must have

$$\frac{rm+1}{\ell+1} > \frac{1}{2}\ell(k-1)+1 \quad \text{so} \quad rm > \frac{1}{2}\ell(\ell+1)(k-1) + \ell > \ell(k-1),$$

and thus, $\operatorname{maxdeg}(\Phi_{\boldsymbol{w}}(M)) = rm$ and

$$\Delta(\Phi_{\boldsymbol{w}}(M)) \leq \sum_{i=1}^{\ell}(rm - i(k-1)) = \ell rm - \frac{\ell(\ell+1)}{2}(k-1) \leq \ell rm.$$

Using Theorem 2, we obtain a complexity in

$$O\left(\ell^2 \Delta(\Phi_{\boldsymbol{w}}(M)) \operatorname{maxdeg}(\Phi_{\boldsymbol{w}}(M))\right) \subseteq O(\ell^3 r^2 m^2)$$

over $\mathbb{F}_{q^{nm}}$. ∎

We explain the next section how to gain a factor ℓ^2 in complexity by exploiting the initial structure and the small determinant of M.

6.2 Weak Popov Walking

Let M be as in Lemma 10 and $\boldsymbol{w} = \big(0, (k-1), \ldots, \ell(k-1)\big)$. We saw in the previous section that $\Delta(\Phi_{\boldsymbol{w}}(M))$ is large compared to $\deg M$, leading to the bad complexity of Algorithm 1. Consider now $\boldsymbol{w}^+ = \boldsymbol{w} + (0, rm, \ldots, rm)$: then $\Phi_{\boldsymbol{w}^+}(M)$ is in weak Popov form and consequently has an orthogonality defect of zero. That is, by a relatively small change in the shift compared to the sought, M is already in weak Popov form.

We will use these two observations—that $\deg\det M$ is small while M is already in a shifted weak Popov form close to the sought—to obtain an algorithm which is in a sense a special case of the Mulders–Storjohann on such input: we carefully select in which order to apply simple transformations, and this gives a better bound on how many transformations to do and how expensive they are.

In this section we will extensively discuss vectors under different shifts. To ease the notation somewhat, we therefore introduce shifted versions of the following operators: $\operatorname{LP}_{\boldsymbol{w}}(\boldsymbol{v}) := \operatorname{LP}(\Phi_{\boldsymbol{w}}(\boldsymbol{v}))$ as well as $\deg_{\boldsymbol{w}}(\boldsymbol{v}) := \deg \Phi_{\boldsymbol{w}}(\boldsymbol{v})$.

We begin by Algorithm 6 that efficiently "walks" from a weak Popov form according to the shift $\boldsymbol{w}$ into one with the shift $\boldsymbol{w} + (1, 0, \ldots, 0)$. The approach can readily be generalised to support incrementing any index, and not just index 0, but since we need only this case and since it simplifies notation, we restrict ourselves to that.

Algorithm 6 Weak Popov Walking

Input: Shift $\boldsymbol{w} \in \mathbb{Z}^m$ and matrix $V \in \mathcal{R}^{m \times m}$ in $\boldsymbol{w}$-shifted weak Popov form.
Output: Matrix in $\hat{\boldsymbol{w}}$-shifted weak Popov form spanning the same $\mathcal{R}$-row space as V, where $\hat{\boldsymbol{w}} = \boldsymbol{w} + (1, 0, \ldots, 0)$.

1 $h_i \leftarrow \mathrm{LP}_{\boldsymbol{w}}(\boldsymbol{v}_i)$, for $i = 0, \ldots, m-1$.
2 $I \leftarrow$ indexes i such that $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\boldsymbol{v}_i) = 0$.
3 $[i_1, \ldots, i_s] \leftarrow I$ sorted such that $h_{i_1} < h_{i_2} < \ldots < h_{i_s}$.
4 $t \leftarrow i_1$.
5 **for** $i = i_2, \ldots, i_s$ **do**
6 **if** $\deg v_{t,0} \leq \deg v_{i,0}$ **then**
7 Apply a simple transformation t on i at position 0 in V.
8 **else**
9 Apply a simple transformation i on t at position 0 in V.
10 $t \leftarrow i$.
11 **return** V.

Theorem 10 *Algorithm 6 is correct.*

Proof: Denote for the sake of the proof V as the input and $\hat{V}$ as the output of the algorithm. Note that each row is the target of a simple transformation at most once, and always by a row of V which has not yet been modified. Thus if $\boldsymbol{v}_i, \hat{\boldsymbol{v}}_i$ are the rows of V respectively $\hat{V}$, then either $\hat{\boldsymbol{v}}_i = \boldsymbol{v}_i$ or $\hat{\boldsymbol{v}}_i$ is the result of a simple transformation on $\boldsymbol{v}_i$ by another row of V. The algorithm performs a single sweep of simple transformations. We will see that the $\hat{\boldsymbol{w}}$-shifted leading positions after this sweep will be a permutation of the h_i, implying that $\hat{V}$ is in $\hat{\boldsymbol{w}}$-shifted weak Popov form.

Note first that for any vector $\boldsymbol{v} \in \mathcal{R}^m$ with $\mathrm{LP}_{\boldsymbol{w}}(\boldsymbol{v}) \neq \mathrm{LP}_{\hat{\boldsymbol{w}}}(\boldsymbol{v})$, then $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\boldsymbol{v}) = 0$, since only the degree of the 0'th position of $\Phi_{\hat{\boldsymbol{w}}}(\boldsymbol{v})$ is different from the corresponding position of $\Phi_{\boldsymbol{w}}(\boldsymbol{v})$. Therefore, for each $i \in \{0, \ldots, m-1\} \setminus I$ we have $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_i) = h_i$, and of course for each $i \in I$ we have $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\boldsymbol{v}_i) = 0$. Furthermore, for each $i \in I$ we must have

$$\deg v_{i,0} + w_0 = \deg v_{i,h_i} + w_{h_i} \tag{8}$$

Consider first an index $i \in I$ for which Line 7 was run, and let t be as at that point. We will establish that $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_i) = h_i$. Since $\deg v_{t,0} \leq \deg v_{i,0}$, applying the simple transformation makes sense. Due to (8) then

$$\deg v_{t,k} + w_k < \deg v_{t,h_t} + w_{h_t} = \deg v_{t,0} + w_0 \ ,$$

for $k > h_t$. By the order of the $i_\star$, then $h_t < h_i$, and so the simple transformation will have

$$\deg \hat{v}_{i,k} + w_k \leq \max\big(\deg v_{i,k},\ \deg v_{t,k} + (\deg v_{i,0} - \deg v_{t,0})\big) + w_k < \deg v_{i,h_i} + w_{h_i} \ ,$$

for $k > h_i$. Since the degree at position 0 went down, we get $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_i) = h_i$.

Consider now an $i \in I$ for which Line 9 was run, and let again t be as at that point, before the reassignment. The situation is completely reversed according to before, so by the same arguments $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_t) = h_i$.

For the value of t at the end of the algorithm, then clearly $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_t) = 0$ since the row was not modified. Since we necessarily have $h_{i_1} = 0$, then $\mathrm{LP}_{\hat{\boldsymbol{w}}}(\hat{\boldsymbol{v}}_t) = h_{i_1}$.

Thus we conclude that the $\hat{\boldsymbol{w}}$-leading positions of the $\hat{\boldsymbol{w}}_i$ are a permutation of the h_i. But the h_i were all different, and so $\hat{V}$ is in $\hat{\boldsymbol{w}}$-shifted weak Popov form. ∎

Proposition 3 *If $\mathcal{R}$ has derivation zero, then Algorithm 6 performs at most*

$$O\big(m \deg\det(V) + \textstyle\sum_{i=0}^{m-1}(2i+1-m)w_i + m^2\big)$$

operations over $\mathcal{R}$.

Proof: We will bound the total number of non-zero monomials which are involved in simple transformations. All the computation of the algorithm is in the addition of two such monomials, so this will bound the complexity.

Assume w.l.o.g. that $w_0 \leq w_1 \leq \ldots \leq w_{m-1}$, and since the input matrix V was in $\boldsymbol{w}$-shifted weak Popov form, assume also w.l.o.g that we have sorted the rows such that $\mathrm{LP}_{\boldsymbol{w}}(\boldsymbol{v}_i) = i$. Since $\Delta(\Phi_{\boldsymbol{w}}(V)) = 0$ we have

$$\deg\det \Phi_{\boldsymbol{w}}(V) = \deg_{\boldsymbol{w}} V \qquad \text{that is} \qquad \deg_{\boldsymbol{w}} V = \deg\det V + \sum \boldsymbol{w}\ .$$

We can therefore consider what is the assignment of $\deg_{\boldsymbol{w}}$ to the individual rows of V which maximises the possible number of monomials in V. We can not have $\deg_{\boldsymbol{w}} v_i < w_i$ since $\mathrm{LP}_{\boldsymbol{w}}(v_i) = i$. It is easy to see that the worst-case assignment is then to have exactly $\deg_{\boldsymbol{w}} v_i = w_i$ for $i = 0, \ldots, m-2$ and $\deg_{\boldsymbol{w}} v_{m-1} = \deg\det V + w_{m-1}$. In this case, the number of monomials can then be bounded as

$$\begin{aligned}
&\left(\sum_{i=0}^{m-2}\big(1+\sum_{j=0}^{i-1}(w_i - w_j + 1)\big)\right) + \left(m(\deg\det V + w_{m-1}) - \sum_{i=0}^{m-1} w_i + 1\right)\\
&\leq m^2 + \left(\sum_{i=0}^{m-2} i w_i - \sum_{j=0}^{m-2}(m-2-j)w_j\right) + \left(m\deg\det V + m w_{m-1} - \sum_{i=0}^{m-1} w_i\right)\\
&= m^2 + m\deg\det V + \sum_{i=0}^{m-1}(2i+1-m)w_i\ .
\end{aligned}$$

■

In particular, if $\sum_{i=0}^{m-1}(2i+1-m)w_i \in O(m\deg\det V)$, then the cost is bounded by $O(m\deg\det V + m^2)$. That is to say, as long as the w_i are fairly *balanced* wrt. each other, then they essentially are handled for free. Similar balancing conditions are seen in other fast algorithms for $K[x]$ matrices, e.g. in [63].

Remark 5 Note that Algorithm 6 is essentially a special case of the Mulders–Storjohann, Algorithm 1, where we carefully select which simple transformations to apply. This results in a factor m fewer simple transformations to apply, as well as a better handle on the size of the simple transformations, leading to the improved complexity.

Algorithm 7 Find Interpolation Polynomial by Weak Popov Walk

Input: Interpolation points $\mathcal{P}$ and $\boldsymbol{w} = (0, (k-1), \ldots, \ell(k-1))$.
Output: Interpolation polynomial satisfying (5) and (6).

```
1 V ← M as in (3).
2 w = (0, (k−1), 2(k−1), ..., ℓ(k−1)).
3 w' = w + (0, rm, rm, ..., rm).
4 for i = 0, ..., rm − 1 do
5     V ← WeakPopovWalk(V, w).
6     w' ← w' + (1, 0, ..., 0).
7 return row v which has minimal w-shifted degree deg_w v.
```

Proposition 4 *Algorithm 7 is correct. It has complexity* $O(\ell r^2 m^2)$.

Proof: Note that M is in $\boldsymbol{w}'$-shifted weak Popov form, where $\boldsymbol{w}'$ is as on Line 3. Thus by the correctness of Algorithm 6, then V at the end of the algorithm must be in $\big(\boldsymbol{w} + (rm, \ldots, rm)\big)$-shifted weak Popov form. Then it is clearly also in $\boldsymbol{w}$-shifted weak Popov form. By Theorem 8 the returned row must be a satisfactory interpolation polynomial.

For the complexity, the algorithm simply performs rm calls to WeakPopovWalk. The quantity $\sum_{i=0}^{m-1}(2i+1-m)w'_i$ is greatest at the first iteration, where it is:

$$\sum_{i=1}^{m-1}(2i+1-m)(rm+i(k-1)) \approx \ell rm + \tfrac{2}{3}\ell^3(k-1) - \tfrac{1}{2}\ell^2 \ .$$

Since $rm > \frac{1}{2}\ell(\ell+1)(k-1)$, as we established in the proof of Theorem 9, the above is bounded by $O(\ell rm)$. Since $\deg\det(V) = \deg\det(M) = rm$ then by Proposition 3 each of the calls to WeakPopovWalk therefore costs at most $O(\ell rm)$. ∎

Bounding $r < \ell n$, we obtain a complexity of $O(\ell^2 m^2 rn)$, matching [61].

7 Conclusion

In this paper, we have explored the notion of row reduction of polynomial matrices in the setting of matrices over skew polynomial rings. For ordinary polynomial rings, row reduction has proven a useful strategy for obtaining numerous flexible, efficient while conceptually simple decoding algorithms for Reed–Solomon and other code families. Our results introduce the methodology and tools aimed at bringing similar benefits to Gabidulin, Interleaved Gabidulin, Mahdavifar–Vardy, and other skew polynomial-based codes. We used those tools in two settings. We solved a general form of multiple skew-shift register synthesis, and applied this for error-erasure decoding of Interleaving Gabidulin codes in complexity $O(\ell\mu^2)$. For Mahdavifar–Vardy codes, we gave an interpolation algorithm with complexity $O(\ell r^2 m^2)$.

We extended and analysed the simple and generally applicable Mulders–Storjohann algorithm to the skew polynomial setting. In both the studied settings, the complexity of that algorithm was initially not satisfactory, but it served as a crucial step in developing more efficient algorithms. For multiple skew-shift register synthesis, we were able to solve a much more general problem than previously, but retaining good complexity. For the Mahdavifar–Vardy codes, the improved algorithm was in the shape of a versatile "Weak Popov Walk", which could potentially apply to many other problems. In all previously studied cases, we matched the best known complexities [54, 61].

Over $K[x]$, row reduction, and the related concept of order bases, have been widely studied and sophisticated algorithms have emerged, e.g. [2, 21, 63]. We hope that future work will enable similar speed-ups for reducing $\mathcal{R}$-matrices.

References

1. Abramov, S.A., Bronstein, M.: On solutions of linear functional systems. In: Proc. of ISSAC, pp. 1–6. New York, NY, USA (2001)
2. Alekhnovich, M.: Linear Diophantine equations over polynomials and soft decoding of Reed–Solomon codes. IEEE Trans. Inf. Theory **51**(7), 2257–2265 (2005)

3. Baker, G., Graves-Morris, P.: Padé approximants, vol. 59. Cambridge Univ. Press (1996)
4. Beckermann, B., Cheng, H., Labahn, G.: Fraction-free row reduction of matrices of Ore polynomials. J. Symb. Comp. **41**(5), 513–543 (2006)
5. Beckermann, B., Labahn, G.: A Uniform Approach for the Fast Computation of Matrix-Type Padé Approximants. SIAM J. Matr. Anal. Appl. **15**(3), 804–823 (1994)
6. Beelen, P., Brander, K.: Key equations for list decoding of Reed–Solomon codes and how to solve them. J. Symb. Comp. **45**(7), 773–786 (2010)
7. Boucher, D., Ulmer, F.: Linear codes using skew polynomials with automorphisms and derivations. Designs, Codes and Cryptography **70**(3), 405–431 (2014)
8. Cohn, H., Heninger, N.: Ideal forms of Coppersmith's theorem and Guruswami–Sudan list decoding. arXiv **1008.1284** (2010)
9. Cohn, P.M.: Skew Field Constructions, vol. 27. Cambridge Univ. Press (1977)
10. Delsarte, P.: Bilinear forms over a finite field, with applications to coding theory. J. Comb. Th. **25**(3), 226–241 (1978)
11. Dieudonné, J.: Les déterminants sur un corps non commutatif. Bull. Soc. Math. France **71**, 27–45 (1943)
12. Draxl, P.K.: Skew Fields, vol. 81. Cambridge Univ. Press (1983)
13. Feng, G.L., Tzeng, K.K.: A Generalization of the Berlekamp-Massey Algorithm for Multisequence Shift-Register Synthesis with Applications to Decoding Cyclic Codes. IEEE Trans. Inf. Theory **37**(5), 1274–1287 (1991)
14. Gabidulin, E., Paramonov, A., Tretjakov, O.: Rank errors and rank erasures correction. In: Proc. of ICCT, vol. 30, pp. 11–19 (1991)
15. Gabidulin, E.M.: Rank-metric codes and applications. Moscow Institute of Physics and Technology (State University). http://iitp.ru/upload/content/839/Gabidulin.pdf
16. Gabidulin, E.M.: Theory of codes with maximum rank distance. Problemy Peredachi Informatsii **21**(1), 3–16 (1985)
17. Gabidulin, E.M., Paramonov, A., Tretjakov, O.: Ideals over a non-commutative ring and their application in cryptology. In: Eurocrypt, pp. 482–489 (1991)
18. Gabidulin, E.M., Pilipchuk, N., et al.: A new method of erasure correction by rank codes. In: Proc. of IEEE ISIT, p. 423 (2003)
19. Gabidulin, E.M., Pilipchuk, N.I.: Error and erasure correcting algorithms for rank codes. Designs, Codes and Cryptography **49**(1-3), 105–122 (2008)
20. Gao, S.: A new algorithm for decoding Reed-Solomon codes. In: Communications, Information and Network Security, pp. 55–68. Springer (2003)
21. Giorgi, P., Jeannerod, C., Villard, G.: On the complexity of polynomial matrix computations. In: Proc. of ISSAC, pp. 135–142 (2003)
22. Guruswami, V., Sudan, M.: Improved Decoding of Reed–Solomon Codes and Algebraic-Geometric Codes. IEEE Trans. Inf. Theory **45**(6), 1757–1767 (1999)
23. Kailath, T.: Linear Systems. Prentice-Hall (1980)
24. Koetter, R., Kschischang, F.R.: Coding for errors and erasures in random network coding. IEEE Trans. Inf. Theory **54**(8), 3579–3591 (2008)
25. Lang, S.: Algebra (2002)
26. Lee, K., O'Sullivan, M.E.: List decoding of Reed–Solomon codes from a Gröbner basis perspective. J. Symb. Comp. **43**(9), 645 – 658 (2008)
27. Lenstra, A.: Factoring multivariate polynomials over finite fields. J. Comp. Syst. Sc. **30**(2), 235–246 (1985)
28. Li, W., Nielsen, J.S.R., Puchinger, S., Sidorenko, V.: Solving shift register problems over skew polynomial rings using module minimisation. In: Proc. of WCC (2015)
29. Li, W., Sidorenko, V., Silva, D.: On transform-domain error and erasure correction by Gabidulin codes. Designs, Codes and Cryptography **73**(2), 571–586 (2014)
30. Lidl, R., Niederreiter, H.: Finite Fields, vol. 20. Cambridge Univ. Press (1997)
31. Loidreau, P.: A welch–berlekamp like algorithm for decoding gabidulin codes. In: Coding and Cryptography, pp. 36–45. Springer (2006)
32. Loidreau, P., Overbeck, R.: Decoding rank errors beyond the error correcting capability. In: Proc. of ACCT, pp. 186–190 (2006)
33. Mahdavifar, H.: List decoding of subspace codes and rank-metric codes. dissertation, University of California, San Diego (2012)
34. Mahdavifar, H., Vardy, A.: Algebraic list-decoding of subspace codes with multiplicities. In: Allerton Conf. Comm., Ctrl. and Comp., pp. 1430–1437 (2011)
35. Mahdavifar, H., Vardy, A.: Algebraic list-decoding of subspace codes. IEEE Trans. Inf. Theory **59**(12), 7814–7828 (2013)

36. Massey, J.L.: Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory **15**(1), 122–127 (1969)
37. Menezes, A.J., Blake, I.F., Gao, X., Mullin, R.C., Vanstone, S.A., Yaghoobian, T.: Applications of finite fields, vol. 199. Springer (2013)
38. Middeke, J.: A computational view on normal forms of matrices of Ore polynomials. Ph.D. thesis, Research Institute for Symbolic Computation (RISC) (2011)
39. Mulders, T., Storjohann, A.: On lattice reduction for polynomial matrices. J. Symb. Comp. **35**(4), 377–401 (2003)
40. Nielsen, J.S.R.: Generalised multi-sequence shift-register synthesis using module minimisation. In: Proc. of IEEE ISIT, pp. 882–886 (2013). Extended version at http://arxiv.org/abs/1301.6529.
41. Nielsen, J.S.R.: Power decoding Reed–Solomon codes up to the Johnson radius. In: Proc. of ACCT (2014)
42. Nielsen, J.S.R., Beelen, P.: Sub-Quadratic Decoding of One-Point Hermitian Codes. IEEE Trans. Inf. Theory **61**(6), 3225–3240 (2015)
43. Nielsen, R.R., Høholdt, T.: Decoding Reed–Solomon codes beyond half the minimum distance. In: Coding Theory, Cryptography and Related Areas, p. 221–236. Springer (1998)
44. Olesh, Z., Storjohann, A.: The vector rational function reconstruction problem. In: Proc. of WWCA, pp. 137–149 (2006)
45. Ore, O.: Theory of non-commutative polynomials. Annals of Mathematics **34**(3), 480–508 (1933)
46. Overbeck, R.: Public key cryptography based on coding theory. Ph.D. thesis, TU Darmstadt (2007)
47. Pete L. Clark: Non-commutative algebra. Public (2012)
48. Richter, G., Plass, S.: Error and erasure decoding of rank-codes with a modified Berlekamp-Massey algorithm. ITG Fachbericht pp. 203–210 (2004)
49. Richter, G., Plass, S.: Fast decoding of rank-codes with rank errors and column erasures. In: Proc. of IEEE ISIT, pp. 398–398 (2004)
50. Roth, R., Ruckenstein, G.: Efficient Decoding of Reed–Solomon Codes Beyond Half the Minimum Distance. IEEE Trans. Inf. Theory **46**(1), 246 –257 (2000)
51. Roth, R.M.: Maximum-rank array codes and their application to crisscross error correction. IEEE Trans. Inf. Theory **37**(2), 328–336 (1991)
52. Schmidt, G., Sidorenko, V.R., Bossert, M.: Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs. IEEE Trans. Inf. Theory **55**(7), 2991–3012 (2009)
53. Sidorenko, V., Bossert, M.: Decoding interleaved Gabidulin codes and multisequence linearized shift-register synthesis. In: Proc. of IEEE ISIT, pp. 1148–1152. IEEE (2010)
54. Sidorenko, V., Jiang, L., Bossert, M.: Skew-feedback shift-register synthesis and decoding interleaved Gabidulin codes. IEEE Trans. Inf. Theory **57**(2), 621–632 (2011)
55. Silva, D.: Error control for network coding. Ph.D. thesis, University of Toronto (2009)
56. Silva, D., Kschischang, F.R.: Rank-metric codes for priority encoding transmission with network coding. In: Proc. of CWIT, pp. 81–84. IEEE (2007)
57. Silva, D., Kschischang, F.R., Koetter, R.: A rank-metric approach to error control in random network coding. IEEE Trans. Inf. Theory **54**(9), 3951–3967 (2008)
58. Sudan, M.: Decoding of Reed Solomon codes beyond the error-correction bound. J. Complexity **13**(1), 180–193 (1997)
59. Wachter-Zeh, A.: Decoding of block and convolutional codes in rank metric. Ph.D. thesis, Universität Ulm (2013)
60. Wachter-Zeh, A., Zeh, A.: Interpolation-based decoding of interleaved Gabidulin codes. In: Proc. of WCC, pp. 528–538 (2013)
61. Xie, H., Lin, J., Yan, Z., Suter, B.W.: Linearized Polynomial Interpolation and Its Applications. IEEE Trans. Signal Process. **61**(1), 206–217 (2013)
62. Zeh, A., Gentner, C., Augot, D.: An Interpolation Procedure for List Decoding Reed-Solomon Codes Based on Generalized Key Equations. IEEE Trans. Inf. Theory **57**(9), 5946–5959 (2011)
63. Zhou, W., Labahn, G.: Efficient algorithms for order basis computation. J. Symb. Comp. **47**(7), 793–819 (2012)

A Technical Proofs

A.1 Proof of Lemma 5

Proof: Let $V \in \mathrm{GL}_n(\mathcal{Q})$. It is obvious that for each i then $\mathrm{LT}(\boldsymbol{v}_i) \neq 0$ because otherwise $\boldsymbol{v}_i = (0, \ldots, 0)$ and V would not be invertible. We can assume w.l.o.g that $\mathrm{LP}(\boldsymbol{v}_i) = i$ because if not, we can change the order of the rows (elementary row operation) of V and obtain a matrix with the same determinant and degree. We consider a sequence of matrices $V^{(k)}$ for $k = 0, \ldots, n-1$, where $V^{(0)} = V$ and $V^{(k+1)}$ is obtained from $V^{(k)}$ by the following elementary row operations:

$$\boldsymbol{v}_i^{(k+1)} = \boldsymbol{v}_i^{(k)} - v_{ik}^{(k)} \big(v_{kk}^{(k)}\big)^{-1} \boldsymbol{v}_k^{(k)} \quad \forall i = k+1, \ldots, n-1 \tag{9}$$

We can prove the following properties for any $k = 0, \ldots, n-1$ by induction.

$$\mathrm{LP}(\boldsymbol{v}_i^{(k)}) = \quad i \quad \forall i = 0, \ldots, n-1 \tag{10}$$

$$\deg v_{ii}^{(k)} = \deg v_{ii} \ \forall i = 0, \ldots, n-1 \tag{11}$$

$$v_{ij}^{(k)} = \quad 0 \quad \forall j < i \wedge j < k \tag{12}$$

For $k = 0$, these properties are fulfilled because $V^{(0)} = V$. Now suppose (10) - (12) are true for some $k \in \{0, \ldots, n-2\}$. Then (12) follows for $k+1$ because

$$v_{ik}^{(k+1)} = v_{ik}^{(k)} - v_{ik}^{(k)} \big(v_{kk}^{(k)}\big)^{-1} v_{kk}^{(k)} = v_{ik}^{(k)} - v_{ik}^{(k)} = 0 \quad \forall i > k.$$

Due to $\mathrm{LP}(\boldsymbol{v}_k) = k$, $\deg v_{kj}^{(k)} < \deg v_{kk}^{(k)}$ for $j > k$ and it follows that

$$\deg \big(v_{ik}^{(k)} \cdot (v_{kk}^{(k)})^{-1} \cdot v_{kj}^{(k)}\big) = \deg v_{ik}^{(k)} + \deg v_{kj}^{(k)} - \deg v_{kk}^{(k)} < \deg v_{ik}^{(k)} \overset{\mathrm{LP}(v_i)=i}{\leq} \deg v_{ii}^{(k)}.$$

Hence, for $j > k$ it holds that

$$\deg v_{ij}^{(k+1)} = \begin{cases} -\infty < \deg v_{ii} = \deg v_{ii}^{(k+1)} \quad (\text{since } v_{ij}^{(k+1)} = 0), & j \leq k \\ \deg \big(\underbrace{v_{ij}^{(k)}}_{\deg \leq \deg v_{ii}^{(k)}} - \underbrace{v_{ik}^{(k)} \cdot (v_{kk}^{(k)})^{-1} \cdot v_{kj}^{(k)}}_{\deg < \deg v_{ii}^{(k)}} \big) \leq \deg v_{ii}^{(k)} = \deg v_{ii}^{(k+1)}, & k < j < i \\ \deg \big(v_{ii}^{(k)} - v_{ik}^{(k)} \cdot (v_{kk}^{(k)})^{-1} \cdot v_{ki}^{(k)}\big) = \deg v_{ii}^{(k)} = \deg v_{ii}, & j = i \\ \deg \big(\underbrace{v_{ij}^{(k)}}_{\deg < \deg v_{ii}^{(k)}} - a_{ik}^{(k)} \cdot (v_{kk}^{(k)})^{-1} \cdot v_{kj}^{(k)} \big) < \deg v_{ii}^{(k)} = \deg v_{ii}^{(k+1)}, & i < j \end{cases}$$

This implies that $\mathrm{LP}(\boldsymbol{v}_i) = i$ (10) and $\deg v_{ii}^{(k)} = \deg v_{ii}$ (11) for $k+1$. Hence, $V^{(n-1)}$ is in upper triangular form, its diagonal elements have degree $\deg v_{ii}^{(n-1)} = \deg v_{ii}$ and by Lemma 4

$$\deg \det V^{(n-1)} = \deg \det \big(\prod_{k=0}^{n-1} \phi\big(v_{ii}^{(n-1)}\big)\big) = \sum_{i=0}^{n-1} \deg \phi\big(v_{ii}^{(n-1)}\big) = \sum_{i=0}^{n-1} \deg v_{ii}.$$

Since $V^{(n-1)}$ is obtained from V by elementary row operations, it follows that

$$\Delta(V) = \deg V - \deg \det V = \sum_{k=0}^{n-1} \deg \boldsymbol{v}_k - \deg \det V^{(n-1)} = \sum_{k=0}^{n-1} \deg v_{ii} - \sum_{i=0}^{n-1} \deg v_{ii} = 0$$

■

A.2 Proof of Theorem 4

Due to the properties of the interpolation, we can split $\hat{r}_i - f_i =: \hat{e}_i$ into three parts

$$\hat{e}_i = \hat{e}_i^E + \hat{e}_i^R + \hat{e}_i^C$$

with $\hat{e}_i, \hat{e}_i^E, \hat{e}_i^R, \hat{e}^C \in \mathcal{R}_{<n}$ such that $\hat{e}_i(g_{i,j}) = \hat{r}(g_{i,j}) - f_i(g_{i,j}) = r_{i,j} - c_{i,j} = e_{i,j}$ and $\hat{e}_i^E(g_{i,j}) = e_{i,j}^E$, $\hat{e}_i^R(g_{i,j}) = e_{i,j}^R$, and $\hat{e}_i^C(g_{i,j}) = e_{i,j}^C$ for all $i = 1, \ldots, \ell$ and $j = 1, \ldots, n$

Lemma 11 *$\hat{e}_i^C(\overline{\Gamma_i^C}(g_{i,j})) = 0$ for all $i = 1, \ldots, \ell$ and $j = 1, \ldots, n$.*

Proof: The proof is exactly the same as for the statement of [59, Lemma A.1], using the relation of the matrix representation of the linear maps $\hat{e}_i^C(\cdot)$ and $\overline{\hat{e}_i^C}(\cdot)$ in certain bases (cf. [55, Lemma 6.3]). ∎

We recall the statement of the theorem:

Theorem 11

$$\Lambda^E \Lambda^R \hat{r}_i \overline{\Gamma_i^C} \equiv \Lambda^E \Lambda^R f_i \overline{\Gamma_i^C} \mod x^m - 1 \quad \forall\, i = 1, \ldots, \ell$$

Proof: Subtracting the right-hand side from the left-hand, we split the expression into parts as in $\hat{e}_i$, and evaluate at each $g_{i,j}$:

$$\begin{aligned}
&(\Lambda^E \Lambda^R \hat{r}_i \overline{\Gamma_i^C} - \Lambda^E \Lambda^R f_i \overline{\Gamma_i^C})(g_{i,j}) \\
&= \underbrace{(\Lambda^E \Lambda^R \hat{e}_i^E \overline{\Gamma_i^C})(g_{i,j})}_{=:A} + \underbrace{(\Lambda^E \Lambda^R \hat{e}_i^R \overline{\Gamma_i^C})(g_{i,j})}_{=:B} + \underbrace{(\Lambda^E \Lambda^R \hat{e}_i^C \overline{\Gamma_i^C})(g_{i,j})}_{=:C}.
\end{aligned}$$

We now have

$$\begin{aligned}
A &= (\Lambda^E \Lambda^R \hat{e}_i^E \overline{\Gamma_i^C})(g_{i,j}) = \Lambda^E(\Lambda^R(\hat{e}_i^E(\overline{\Gamma_i^C}(g_{i,j})))) = \Lambda^E(\Lambda^R(\hat{e}_i^E(\sum_{\kappa=1}^{m} \alpha_{i,\kappa} g_{i,\kappa}))) \\
&\overset{\alpha_{i,\kappa} \in \mathbb{F}_q}{=} \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \Lambda^E(\Lambda^R(\hat{e}_i^E(g_{i,\kappa}))) = \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \Lambda^E(\Lambda^R(e_{i,\kappa}^E)) = \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \Lambda^E(\Lambda^R(\sum_{j=1}^{\tau} a_{i,j}^E b_{i,j,\kappa}^E)) \\
&\overset{b_{i,j,\kappa}^E \in \mathbb{F}_q}{=} \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \sum_{j=1}^{\tau} b_{i,j,\kappa}^E \underbrace{\Lambda^E(\Lambda^R(a_{i,j}^E))}_{=0} = 0 \ .
\end{aligned}$$

$$B = \Lambda^E(\Lambda^R(\hat{e}_i^R(\overline{\Gamma_i^C}(g_{i,j})))) = \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \Lambda^E(\Lambda^R(\hat{e}_i^R(g_{i,\kappa}))) = \sum_{\kappa=1}^{m} \alpha_{i,\kappa} \sum_{j=1}^{\tau} b_{i,j,\kappa}^R \Lambda^E(\underbrace{\Lambda^R(a_{i,j}^R)}_{=0}) = 0 \ .$$

$$C = (\Lambda^E \Lambda^R \hat{e}_i^C \overline{\Gamma_i^C})(g_{i,j}) = \Lambda^E(\Lambda^R(\underbrace{\hat{e}_i^C(\overline{\Gamma_i^C}(g_{i,j}))}_{=0, \text{ Lemma 11}})) = \Lambda^E(\Lambda^R(0)) = 0 \ .$$

We obtain $(\Lambda^E \Lambda^R \hat{e}_i \overline{\Gamma_i^C})(g_{i,j}) = 0$ for all i, j, so since $G_1 = \ldots = G_\ell$, all G_i must divide $\Lambda^E \Lambda^R \hat{e}_i \overline{\Gamma_i^C}$, and the claim follows. ∎

B Basics on $\mathcal{R}$-modules

In this section, we prove statements about $\mathcal{R}$-modules. Most of the results are well-known or obvious for commutative rings. However, we have not found a compact source for these statements which include the case of skew polynomial rings and derives them from algebraic principles.

We say that vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n \in \mathcal{R}^n$ are (left) *linearly independent* over $\mathcal{R}$ if from a vanishing linear combination $a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2, \ldots, a_n\boldsymbol{v}_n = 0$ it follows that $a_1 = a_2 = \cdots = a_n = 0$. For a left module $\mathcal{M}$ and $B \subseteq \mathcal{M}$, by $\langle B \rangle_{\mathcal{R}}$ we denote the (left) span of B: the set of (left) $\mathcal{R}$-linear combinations of elements of B. B is a generating set of $\mathcal{M}$ if $\langle B \rangle_{\mathcal{R}} = \mathcal{M}$. A basis of a (left) module $\mathcal{M}$ is a generating set $B \subseteq \mathcal{M}$ which is (left) linearly independent over $\mathcal{R}$. We say a module $\mathcal{M}$ is free if it admits a basis.

We define the (left) *rank* of a matrix as the maximum number of rows which are linearly independent over $\mathcal{R}$. Note that any subset of linearly independent rows of cardinality equal to the rank is a basis of the (left) row space $\langle M \rangle_{\mathcal{R}}$ of a matrix M, which is a (left) module. Since $\mathcal{R}$ is a left (right) Euclidean domain [45], it is a left (right) principal ideal domain.

The statement and proof of the following theorem is similar to [25, Theorem 7.1], with the difference that we use the weaker assumption that $\mathcal{R}$ is in general only left (right) principal and not a commutative principal ideal domain.

Theorem 12 *Let F be a free left (right) module over $\mathcal{R}$ which has a finite basis $\{x_1, \ldots, x_n\}$ for some $n \in \mathbb{N}$, and $M \subset F$ a left (right) submodule. Then M is free and there exists a basis of M which has $\leq n$ elements.*

Proof: We prove the claim by constructing a basis of F inductively. Before we start, we have to define the sets $M_r := M \cap \langle x_1, \ldots, x_r \rangle_{\mathcal{R}}$ for any $r \in \mathbb{N}$ with $1 \leq r \leq n$. Note that we only show the claim for left modules. The right module case can be shown equivalently.

Base case: We first consider $M_1 = M \cap \langle x_1 \rangle_{\mathcal{R}}$. Since M and $\langle x_1 \rangle_{\mathcal{R}}$ are both left modules over $\mathcal{R}$, M_1 is also a left module over $\mathcal{R}$. Let J be the set of all $a \in \mathcal{R}$ such that $ax_1 \in M$. Then J is a left ideal of $\mathcal{R}$ because for any $a, b \in J$ and $r \in \mathcal{R}$ also $(a+b) \in J$ (because $(a+b)x_1 = ax_1 + bx_1 \in M$) and $(ra) \in J$ (because $(ra)x_1 = r(ax_1) \in M$). J is (left) principal and therefore generated by some $a_1 \in \mathcal{R}$. Hence, M_1 must be of the form (a_1x_1). If $a_1 \neq 0$, $\{a_1x_1\}$ is a basis of M_1. Otherwise, $M_1 = \{0\}$ and $\emptyset$ is a basis of M_1. Thus, M_1 is free and has a basis with ≤ 1 elements.

Induction hypothesis: Now we assume that $M_r = M \cap \langle x_1, \ldots, x_r \rangle_{\mathcal{R}}$ is free and has a basis $\{\tilde{x}_1, \ldots, \tilde{x}_\ell\}$ with $0 \leq \ell \leq r$ for some r with $1 \leq r < n$.

Inductive step: If $M_r = M_{r+1}$, we are done, because by hypothesis M_{r+1} is free with the same basis as M_r which also obviously has $\leq (r+1)$ elements. In the following we therefore assume that $M_{r+1} \setminus M_r \neq \emptyset$.

Let I be the set of all $a \in \mathcal{R}$ such that there exists an element $x^{(a)} \in M$ and $b_1^{(a)}, \ldots, b_r^{(a)} \in \mathcal{R}$ with $ax_{r+1} = x^{(a)} - \sum_{i=1}^{r} b_i^{(a)} x_i$. Then I is a left ideal of $\mathcal{R}$ because if $a, b \in I$ and $r \in \mathcal{R}$, then $a + b \in I$ because $x^{(a+b)} = x^{(a)} + x^{(b)} \in M$ and $b_i^{(a+b)} = b_i^{(a)} + b_i^{(b)} \in \mathcal{R}$ and $ra \in I$ because $x^{(ra)} = rx^{(a)} \in M$ and $b_i^{(ra)} = rb_i^{(a)} \in \mathcal{R}$. Since $\mathcal{R}$ is (left) principal, I is generated by some element $a_{r+1} \in \mathcal{R}$. Since we assume that $M_{r+1} \neq M_r$, a_{r+1} must not be 0. We can choose a $\gamma \in M_{r+1}$ such that $\gamma = x^{(a_{r+1})} \neq 0$.

(Sub-)Claim: $B_{r+1} := \{\tilde{x}_1, \ldots, \tilde{x}_\ell, \gamma\}$ is a basis of M_{r+1}.

(i) B_{r+1} is a generating set of M_{r+1}:

For any $x \in M_{r+1}$, the coefficient of x with respect to x_{r+1} is in I and therefore divisible by a_{r+1}. Hence, there exists a $d \in \mathcal{R}$ such that the coefficient of $x - d\gamma$ with respect to x_{r+1} is 0 and therefore $x - d\gamma \in M_r$. This shows that $\{\tilde{x}_1, \ldots, \tilde{x}_\ell, \gamma\}$ is a generating set of M_{r+1}.

(ii) The elements of B_{r+1} are linearly independent:

Since $M_{r+1} \neq M_r$, there is an $x \in M_{r+1}$ which is not in M_r. Like in the previous statement, there is a $d \in \mathcal{R}$ such that $x - d\gamma \in M_r$ (note that here $d \neq 0$ because $x \notin M_r$). This shows that $\gamma \notin M_r$ because if it was, $d\gamma$ would be in M_r (because M_r is a module) and $x = (x - d\gamma) + d\gamma \in M_r$, which would be a contradiction. Thus, $\gamma \notin M_r = (\tilde{x}_1, \ldots, \tilde{x}_\ell)$ which implies that $\tilde{x}_1, \ldots, \tilde{x}_\ell$ and γ are linearly independent.

Hence, B_{r+1} is a basis of M_{r+1} which has $\ell + 1 \leq r + 1$ elements (by hypothesis $\ell \leq r$).

Conclusion: At this point we found a basis B_n for M_n which has $\leq n$ elements. Since $\{x_1, \ldots, x_n\}$ is a basis of F and $M \subset F$, $M_n = M \cap \langle x_1, \ldots, x_n \rangle_{\mathcal{R}} = M \cap F = M$ and therefore B_n is also a basis of M. Note that the existence of a basis directly implies that M is free. ■

An important corollary of Theorem 12 is that for $n \in \mathbb{N}$, any submodule of $\mathcal{R}^n$ is free.

The following theorem proves that $\mathcal{R}$ has the *Invariant Basis Number* property. Its statement and proof combines ideas from [47, Theorem 18] and [47, Proposition 16]. Note that $\mathcal{R}$ is left (right) Noetherian because it is left (right) principal. Hence, any finitely generated $\mathcal{R}$-module is Noetherian which implies that any submodule of a finitely generated $\mathcal{R}$-module is finitely generated.

Theorem 13 *Any two finite bases of a left (right) $\mathcal{R}$-module have the same number of elements.*

Proof: Again we show the claim only for left modules, the right case can be shown equivalently. Assume we have two bases B_1 and B_2 of the same $\mathcal{R}$-module with $|B_1| =: n \in \mathbb{N}$ and $|B_2| =: m \in \mathbb{N}$. We want to show that $n = m$.

We first note that from any $\mathcal{R}$-module with a basis $B = \{b_1, \ldots, b_k\}$ of cardinality k, there is a natural module isomorphism $\varphi : \langle B \rangle_{\mathcal{R}} \to \mathcal{R}^k$, $\varphi(x) = (x_1, \ldots, x_k)$ which is defined by the unique representation $x = \sum_{i=1}^{k} x_i b_i$ for any $x \in (B)$.

This means that $\mathcal{R}^n \cong \langle B_1 \rangle_{\mathcal{R}} = \langle B_2 \rangle_{\mathcal{R}} \cong R^n$, so $\mathcal{R}^n \cong R^m$, or in other words there is an isomorphism $\alpha : \mathcal{R}^n \to \mathcal{R}^m$.

We assume $n > m$. This means that α gives us an embedding $\alpha : \mathcal{R}^n = \mathcal{R}^m \bigoplus \mathcal{R}^{n-m} \hookrightarrow \mathcal{R}^m$ with $\mathcal{R}^{n-m} \neq \{0\}$ because $n > m$ and $\mathcal{R} \neq \{0\}$. Therefore, $A_0 := \mathcal{R}^m$ must have a submodule of the form $A_1 \bigoplus B_1$ such that $A_1 \cong \mathcal{R}^m$ and $B_1 \cong \mathcal{R}^{n-m} \neq \{0\}$. Constructing $A_i \bigoplus B_i$ as a submodule of A_{i-1} inductively with the same approach, we obtain a submodule $\bigoplus_{i=1}^{\infty} B_i$ of $\mathcal{R}^m$ which is an infinite direct sum of non-zero modules and can therefore not be finitely generated. This contradicts the fact that $\mathcal{R}^m$ is a (left) Noetherian module, which means that all its submodules are finitely generated. Hence, $n \leq m$. By the same argument we get that $m \leq n$ and therefore $n = m$. ∎

We continue with some remarks on matrices over $\mathcal{R}$ and the module their rows span. An elementary row operation on an $\mathcal{R}$ matrix can be the following two actions:

(i) Interchange two rows.
(ii) Add an $\mathcal{R}$ multiple of a row $\boldsymbol{v}_i$ to another row $\boldsymbol{v}_j$.

Theorem 14 *An elementary row operation on a matrix neither changes its row space nor its rank.*

Proof: The statement is obvious for type (i) row operations because $(\mathcal{R}, +)$ is an abelian group. Suppose that in a type (ii) operation on a matrix V, resulting in V', $\boldsymbol{v}_j$ is replaced by $\boldsymbol{v}'_j \leftarrow a\boldsymbol{v}_i + \boldsymbol{v}_j$, where $a \in \mathcal{R}$. Clearly we then have $\langle V' \rangle_{\mathcal{R}} \subset \langle V \rangle_{\mathcal{R}}$. But since $\boldsymbol{v}_i = \boldsymbol{v}'_i$ and $\boldsymbol{v}_j = \boldsymbol{v}'_j - a\boldsymbol{v}'_i$ then also $\boldsymbol{v}_j \in \langle V' \rangle_{\mathcal{R}}$ and so $\langle V' \rangle_{\mathcal{R}} \subset \langle V \rangle_{\mathcal{R}}$. So the two modules spanned by the rows are the same. Therefore the rank can also not change due to Theorem 13. ∎

Note that a simple transformation (cf. Section 3) is an elementary row operation (type (ii)). A type (i) row operation is used in Section 5.4 (swapping of rows within the Demand-Driven algorithm).