Solving Shift Register Problems over Skew Polynomial Rings using Module Minimisation

W Li, Johan Sebastian Rosenkilde Nielsen, S Puchinger, V Sidorenko

Solving Shift Register Problems over Skew Polynomial Rings using Module Minimisation

W. Li[1], J.S.R. Nielsen[2], S. Puchinger[1], V. Sidorenko[13]

[1] Institute of Communications Engineering, Ulm University, Germany
{wenhui.li | sven.puchinger | vladimir.sidorenko}@uni-ulm.de
[2] GRACE Project, INRIA Saclay & LIX, École Polytechnique, France
jsrn@jsrn.dk
[3] Institute for Communications Engineering, TU München, Germany

Abstract For many algebraic codes the main part of decoding can be reduced to a shift register synthesis problem. In this paper we present an approach for solving generalised shift register problems over skew polynomial rings which occur in error and erasure decoding of ℓ-Interleaved Gabidulin codes. The algorithm is based on module minimisation and has time complexity $O(\ell\mu^2)$ where μ measures the size of the input problem.

Keywords: Skew Polynomials, Ore Polynomials, Shift Register Synthesis, Module Minimisation, Gabidulin Codes

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, 5, 14], then for Power decoding [19] and also either type of decoder for Hermitian codes [21].

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 [12, 28]. 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 [20].

The main goal of this paper is to extend the module minimisation description to skew polynomial rings and Gabidulin codes, in particular Interleaved Gabidulin codes, with the aim of enjoying similar benefits. 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 [18] to matrices over skew polynomial rings. We analyse its complexity when applied to the shift register problem which arise when decoding Interleaved Gabidulin codes. Finally, we extend the Demand–Driven algorithm for $\mathbb{F}_q[x]$ shift register

problems [19], which is derived from the Mulders–Storjohann, also to the skew polynomial setting.

Gabidulin codes [7, 10, 23] are maximum rank distance codes with various applications like random linear network coding [13, 26] and cryptography [11]. They are the rank-metric analogue of RS codes. An Interleaved Gabidulin code [17,25,26] is a direct sum of several (n, k_i) 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 [24] and its references, the core task of decoding can be reduced to what is known as a multi-sequence skew-feedback shift register synthesis problem [25].

In this paper, we use the introduced module minimisation description to solve a more general form of this problem, which we abbreviate MgLSSR:

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

$$\lambda s_i \equiv \omega_i \mod g_i \tag{1}$$

$$\deg \omega_i + \gamma_i < \deg \lambda + \gamma_0 \tag{2}$$

The original problem of [25] set g_i to powers of x and $\gamma_i = 0$. The above is a natural generalisation, which covers error and erasure decoding of Gabidulin codes [16], as well as an Interleaved extension of the Gao-type decoder for Gabidulin codes ([27, §3.2] combined with the ideas of [25]). For cases where the algorithm of [25] applies, the Demand–Driven algorithm we present has the same complexity. However, the more general perspective of module minimisation gives conceptually simpler proofs, and may prove useful for gaining further insights or faster, more sophisticated algorithms.

Normal form computation of matrices over skew rings and Ore rings has been investigated before, e.g. [1, 3], but the focus has been over rings such as $\mathbb{Z}$ or $K[z]$ for some field K, where coefficient growth is important to control. Since we are inspired mainly by the application to Gabidulin codes, where the skew ring is over a finite field, we count only operations performed in the field; in this measure those previous algorithms are much slower than what is presented here.

We set basic notation in Section 2. Section 3 describes how to solve Problem 1 using module minimisation, and gives the Mulders–Storjohann algorithm for skew polynomial modules to accomplish this. We introduce important concepts for arguing about such modules in Section 4 for performing a complexity analysis. Section 5 describes how to then derive the faster Demand–Driven algorithm. Due to lack of space, a number of proofs are omitted.

2 Notation and Remarks on Generality

Let K be a field. Denote by $\mathcal{R} = K[x; \theta, \delta]$ the noncommutative ring of skew polynomials over K with automorphism θ and derivation δ. Being an Ore extension, $\mathcal{R}$ is both a left and right Euclidean ring. See [22] for more details.

For coding theory we usually take K as a finite field $\mathbb{F} = \mathbb{F}_{q^r}$ for a prime power q 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 [4]. The algorithms in this paper are correct for any field, automorphism and derivation. For complexities, we are counting field operations, and we often assume $\delta = 0$.

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_{ij} 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 $\operatorname{maxdeg} V := \max_i\{\deg \boldsymbol{v}_i\} = \max_{i,j}\{\deg v_{ij}\}$.
- The *leading position* of a vector $\boldsymbol{v}$ is $\operatorname{LP}(\boldsymbol{v}) := \max\{i \,:\, \deg v_i = \deg \boldsymbol{v}\}$. Furthermore $\operatorname{LT}(\boldsymbol{v}) := v_{\operatorname{LP}(\boldsymbol{v})}$ and $\operatorname{LC}(\boldsymbol{v})$ is the leading coefficient of $\operatorname{LT}(\boldsymbol{v})$.

3 Finding a Solution using Module Minimisation

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

$$\mathcal{M} := \left\{(\lambda, \omega_1, \ldots, \omega_\ell) \in \mathcal{R}^{\ell+1} \mid \lambda s_i \equiv \omega_i \mod g_i \; \forall i = 1, \ldots, \ell\right\}. \tag{3}$$

Lemma 1. *$\mathcal{M}$ 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 $\mathcal{M}$:*

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

The above gives a simple description of all solutions of the congruence relations (1). To solve Problem 1, we therefore need an element in the $\mathcal{M}$ which satisfies the degree condition (2) and has minimal degree. For this purpose, define

$$\Phi : \mathcal{R}^{\ell+1} \to \mathcal{R}^{\ell+1}, \quad \boldsymbol{u} = (u_0, \ldots, u_\ell) \mapsto (u_0 x^{\gamma_0}, \ldots, u_\ell x^{\gamma_\ell}). \tag{4}$$

We can extend the domain of Φ to matrices over $\mathcal{R}$ by applying it row-wise. It is easy to see that $\Phi(\mathcal{M})$ is also a left $\mathcal{R}$-module and that Φ is a module isomorphism. Using this notation, we can restate how to solve Problem 1:

Lemma 2. *A vector $\boldsymbol{v} \in \mathcal{M}^*$ is a solution to Problem 1 if and only if $LP(\Phi(\boldsymbol{v})) = 0$ and for all $\boldsymbol{u} \in \mathcal{M}^*$ with $LP(\Phi(\boldsymbol{u})) = 0$ it holds that $\deg \Phi(\boldsymbol{v}) \leq \deg \Phi(\boldsymbol{u})$.*

Proof: $\boldsymbol{v} \in \mathcal{M}^*$ is a solution to Problem 1 iff it satisfies (2) and v_0 has minimum possible degree. That $\boldsymbol{v}$ satisfies (2) means $\deg v_0 + \gamma_0 > \deg v_i + \gamma_i$ and so $\deg(v_0 x^{\gamma_0}) > \deg(v_i x^{\gamma_i})$ i.e. $\mathrm{LP}(\Phi(\boldsymbol{v})) = 0$. The reverse direction is similar. ■

So we should find a vector $\boldsymbol{v} \in \Phi(\mathcal{M})$ with minimum-degree leading term among vectors with leading position zero. We do this by finding a basis of $\Phi(\mathcal{M})$ of a specific form. This extends similar ideas for matrices over $K[x]$ [18, 19].

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.*

The following *value function* for $\mathcal{R}$ vectors will prove useful: $\psi : \mathcal{R}^{\ell+1} \to \mathbb{N}_0$,

$$\psi(\boldsymbol{v}) = (\ell + 1) \deg \boldsymbol{v} + \mathrm{LP}(\boldsymbol{v}) + 1 \qquad \text{for } \boldsymbol{v} \neq \boldsymbol{0} \text{ and} \qquad \psi(\boldsymbol{0}) = 0.$$

Lemma 3. *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 $\boldsymbol{u}_i$ all have different leading position, so the $a_i \boldsymbol{v}_i$ must as well for those $a_i \neq 0$, which in turn means that the their $\psi(a_i \boldsymbol{v}_i)$ are all different. Notice that for any two $\boldsymbol{u}_1, \boldsymbol{u}_2$ with $\psi(u_1) \neq \psi(u_2)$, then $\psi(\boldsymbol{u}_1 + \boldsymbol{u}_2)$ either equals $\psi(\boldsymbol{u}_1)$ or $\psi(\boldsymbol{u}_2)$. Applied inductively, that 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 $\deg \boldsymbol{u} = \deg a_i + \deg \boldsymbol{v}_i$. ■

Lemma 2 and Lemma 3 imply that a basis of $\mathcal{M}$ in weak Popov form gives a solution to Problem 1 as one of its rows. The following definition leads to a remarkably simple algorithm for computing such a basis: Algorithm 1, an $\mathcal{R}$ variant of the Mulders–Storjohann algorithm [18], originally described for $K[x]$.

Definition 2. *Applying a simple transformation on a matrix V means finding non-zero rows $\boldsymbol{v}_i, \boldsymbol{v}_j$, $i \neq j$ such that $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(\boldsymbol{v}_j)$ and $\deg \boldsymbol{v}_i \leq \deg \boldsymbol{v}_j$, and replace $\boldsymbol{v}_j$ by $\boldsymbol{v}_j - \alpha x^\beta \boldsymbol{v}_i$, where $\beta = \deg v_j - \deg v_i$ and $\alpha = \mathrm{LC}(\boldsymbol{v}_j) / \theta^\beta(\mathrm{LC}(\boldsymbol{v}_i))$.*

Remark 1. Note that a simple transformations cancels the leading term of the polynomial $\mathrm{LT}(v_j)$. Also elementary row operations keep the module spanned by the matrix' rows unchanged, see e.g. [3], so the same is true for any sequence of simple transformations.

Lemma 4. *If $\boldsymbol{v}'$ replaces $\boldsymbol{v}$ in a simple transformation, then $\psi(\boldsymbol{v}') < \psi(\boldsymbol{v})$.*

Proof: The operations used in a simple transformation ensure that $\deg \boldsymbol{v}' \leq \deg \boldsymbol{v}$. If $\deg \boldsymbol{v}' < \deg \boldsymbol{v}$, we are done because $\mathrm{LP}(\boldsymbol{v}') < \ell + 1$. If $\deg \boldsymbol{v}' = \deg \boldsymbol{v}$, then $\mathrm{LP}(\boldsymbol{v}') < \mathrm{LP}(\boldsymbol{v})$: by the definition of the leading position, all terms to the right of $\mathrm{LP}(\boldsymbol{v})$ in $\boldsymbol{v}$ and $\alpha x^\beta \boldsymbol{v}_i$, and therefore also in $\boldsymbol{v}'$, have degree less than $\deg \boldsymbol{v}$. Furthermore $\deg \boldsymbol{v}'_{\mathrm{LP}(\boldsymbol{v})} < \deg \boldsymbol{v}$ by the definition of a simple transformation. ■

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

Input: A square matrix V over $\mathcal{R}$, whose rows span the module $\mathcal{V}$
Output: A basis of $\mathcal{V}$ in weak Popov form.

1 Apply simple transformations on the rows of V until no longer possible.
2 **return** V.

Theorem 1. *Algorithm 1 is correct.*

Proof: By Lemma 4, the value of one row of V decreases for each simple transformation. The sum of the values of the rows must at all times be non-negative so the algorithm must terminate. Finally, when the algorithm terminates there are no simple transformations possible on V anymore, i.e. there are no $i \neq j$ such that $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(\boldsymbol{v}_j)$. That is to say, V is in weak Popov form. ∎

This gives an algorithm to solve Problem 1. The above proof could also easily lead to a rough complexity estimate. To obtain a more fine-grained one, we will in the next section restrict ourselves to matrices which are square and full rank.

4 Complexity Analysis

Lenstra [15] introduced the notion of orthogonality defect of square, full rank $K[x]$ matrices, and in [19], it was shown it can describe the complexity of the Mulders–Storjohann and Alekhnovich [2] algorithms for such matrices more fine-grained than originally, and that this improves the asymptotic estimate 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 [8] 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 [6, Chapter 1]. We know that $\mathcal{R}$ is a principal left ideal domain which implies that it is left Ore and 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 $s^{-1}r \sim s'^{-1}r'$ if $\exists u, u' \in \mathcal{R}^*$ such that $ur = u'r'$ and $us = u's'$. The degree map on $\mathcal{R}$ can be naturally extended to $\mathcal{Q}$ by defining

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

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

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

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 [8] and can also be found in [9].

Lemma 5. *There is a function* $\det :\ \mathcal{Q}^{n\times n} \to \mathcal{Q}^{\mathrm{ab}}$ *s.t. for all* $A \in \mathcal{Q}^{n\times n}$, $k \in \mathcal{Q}$*:*

(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 3. *A function* det *with the properties of Lemma 5 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 to be any given Dieudonné determinant.

Lemma 6. *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

$$\det A \overset{\text{Lemma 5 (iii)}}{=} \left[\prod_{i=0}^{n-1} \phi(d_i)\right] \cdot \det A' \overset{\text{Lemma 5 (ii)}}{=} \left[\prod_{i=0}^{n-1} \phi(d_i)\right] \cdot \det I \overset{\text{Lemma 5 (i)}}{=} \prod_{i=0}^{n-1} \phi(d_i).$$

■

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 [19].

Definition 4. *The orthogonality defect of* V *is* $\Delta(V) := \deg V - \deg \det V$.

Lemma 7. *If* $V \in \mathrm{GL}_n(\mathcal{Q})$ *is in weak Popov form, then* $\Delta(V) = 0$.

proof sketch: We can assume that $\mathrm{LP}(v_i) = i$ for all i because if not, we can change the order of the rows of V and obtain a matrix with the same determinant and degree. We can then apply elementary row operations to bring the matrix to upper triangular form. After these row operations, the property $\mathrm{LP}(v_i) = i$ is preserved and $\deg v_{ii}$ is equal to $\deg v_{ii}$ of the start matrix for all i. By Lemma 6 the degree of the determinant equals the sum of the degree of the diagonal elements, and hence $\deg V = \deg \det V$. ■

4.2 Complexity of Mulders–Storjohann

We can now bound the complexity of Algorithm 1 using arguments similar to those in [19]. These are in turn, the original arguments of [18] but finer grained by using the orthogonality defect. In the following, let $\mu := \max_i\{\gamma_i + \deg g_i\}$. We can assume that $\gamma_0 < \mu$ since otherwise $(1, s_1, \ldots, s_\ell)$ is the minimal solution to the MgLSSR.

Lemma 8. $\Delta(\Phi(M)) \leq \mu - \gamma_0$.

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

Proof: Every simple transformation reduces the value ψ of one row with at least 1. So the number of possible simple transformations is upper bounded by the difference of the sums of the values of the input matrix $\Phi(M)$ and the output matrix V, i.e.:

$$\begin{aligned}
&\textstyle\sum_{i=0}^{\ell}[(\ell+1)\deg\Phi(\boldsymbol{m}_i)+\mathrm{LP}(\Phi(\boldsymbol{m}_i))-\big((\ell+1)\deg\Phi(\boldsymbol{v}_i)+\mathrm{LP}(\boldsymbol{v}_i)\big)]\\
&= \mathrm{LP}(\Phi(\boldsymbol{m}_0)) + (\ell+1)\textstyle\sum_{i=0}^{\ell}[\deg\Phi(\boldsymbol{m}_i)-\deg\boldsymbol{v}_i]\\
&\leq (\ell+1)[\deg\Phi(M)-\deg V+1] = (\ell+1)[\Delta(\Phi(M))+1],
\end{aligned}$$

where the last equality follows from $\deg V = \deg\det V = \deg\det M$.

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 $\deg v_j \leq \mu$ this costs $O(\ell\mu)$. ∎

5 Demand-Driven Algorithm

It was observed in [19] that the Mulders–Storjohann algorithm over $K[x]$ admits a "demand–driven" variant when applied to matrices coming from shift register problems, where coefficients of the working matrix are computed only when they are needed. 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; in fact, both the algorithm and the proof are almost line-for-line the same for $\mathcal{R}$ as for $K[x]$. We therefore focus on the idea of the algorithm, and the original proofs can be found in [19] (extended version).

The central observation is that due to the special form of M of Lemma 1, only the first column is needed during the Mulders–Storjohann algorithm in order to construct the rest. That is formalised in the following lemma:

Lemma 9. *Consider Algorithm 1 with input $\Phi(M)$. Consider a variant where, when replacing $\boldsymbol{v}_j$ with $\boldsymbol{v}'_j$ in a simple transformation, 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 correctness of the algorithm or the upper bound on the number of simple transformations performed.*

The Demand–Driven algorithm, Algorithm 2, therefore calculates just the first element of a vector whenever doing a simple transformation, being essentially enough information. To retain speed it is important, however, that the algorithm can also figure out which simple transformation it can next apply, without having to recompute the whole matrix. For this, we cache for each row its degree η_j and the leading coefficient of its leading position α_j. The following observations then lead to Algorithm 2:

1. In $\Phi(M)$ there is at most one possible choice of the first simple transformation, due to the matrix' shape. This is true throughout the algorithm, making it deterministic.

Algorithm 2 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 $\mathcal{M}$ whose Φ image is in 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 $\alpha \leftarrow$ coefficient to x^η in $(\lambda_0 \tilde{s}_h \mod \tilde{g}_h)$
7 **if** $\alpha \neq 0$ **then**
8 **if** $\eta < \eta_h$ **then** swap $(\lambda_0, \alpha, \eta)$ and $(\lambda_h, \alpha_h, \eta_h)$
9 $\lambda_0 \leftarrow \lambda_0 - \alpha/\theta^{\eta-\eta_h}(\alpha_h)x^{\eta-\eta_h}\lambda_h$
10 $(\eta, h) \leftarrow (\eta, h-1)$ **if** $h > 1$ **else** $(\eta - 1, \ell)$
11 **return** $\left(\lambda_0 x^{-\eta_0}, \ldots, \lambda_\ell x^{-\eta_0}\right)$

2. To begin with, if there is a possible simple transformation, row 0 is involved. Just before doing a simple transformation, we possibly swap the two rows involved such that the row changed is always row 0. That means row 0 is always involved if there is a possible simple transformation, and that the algorithm terminates when row 0 has leading position 0.
3. To begin with row i has leading position i for $i > 0$. The above swap ensures that this will keep being true.
4. After doing a simple transformation, we need to update the degree, leading position and leading coefficient of only row 0; the rest remains unchanged. We do this by going through each possible degree and leading position in decreasing order of value ψ. This is correct since we know that the simple transformation must decrease the value of row 0.

To express the complexity, by $\mathrm{supp}(f), f \in \mathcal{R}$, we mean the set of degrees such that f has a non-zero coefficient for this degree. By $\deg_2 f$ we mean the degree of the second largest coefficient. Let again $\mu := \max_i\{\gamma_i + \deg g_i\}$.

Theorem 3. *Algorithm 2 is correct. Over $\mathcal{R}$ with derivation zero, it has computational complexity $O(\ell\mu^2\rho)$, where*

$$\rho = \begin{cases} \max_i\{\#\mathrm{supp}(g_i)\} & \text{if } \deg_2 g_i < \frac{1}{2}\deg g_i \text{ for all } i \\ \mu & \text{otherwise} \end{cases}$$

It has memory complexity $O(\ell\mu)$.

proof sketch: We only prove the complexity statement. 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 *estimate* 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 2, the loop will iterate at most $O(\ell\mu)$ times. Each execution of Line 9 costs $O(\mu)$ since the λ_j all have degree at most μ.

For Line 6, we can compute the needed coefficient α in complexity $O(\mu\rho)$: if $\deg_2 g_h > \frac{1}{2}\deg g_h$, we simply compute the entire polynomial $\lambda_0 \tilde{s}_h \mod \tilde{g}_h$ in time $O(\mu^2)$. Otherwise, an easy argument shows that at most $\#\mathrm{supp}(g_h)+1$ coefficients of $\lambda_j \tilde{s}_h$ affects the computation of α. Each of these can be computed by convolution in time $O(\mu)$. ∎

For generic g_i, Algorithm 2 will have complexity $O(\ell\mu^3)$ which is usually worse than $O(\ell^2\mu^2)$ of Algorithm 1. However, for decoding of Interleaved Gabidulin codes, two important cases are $g_i = x^k$ (syndrome decoding [25]) and $g_i = x^{q^m} - 1$ (Gao-type decoding [27, §3.2]), and here Algorithm 2 runs in complexity $O(\ell\mu^2)$.

Remark 2. Algorithm 2 bears a striking similarity to the Berlekamp–Massey variant for multiple shift registers [25] where all g_i are powers of x, and has the same running time in this case. However, 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. For instance, using known properties for the weak Popov form, it is trivial to prove that Algorithm 2 can be modified to return a basis for *all* solutions to the shift register problem, as well as decompose any given solution as an $\mathcal{R}$-linear combination of this basis.

6 Conclusion

In this paper, we have given two module-based methods for solving generalised shift register problems over skew polynomial rings. For ordinary polynomial rings, module minimisation 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 and other skew polynomial-based codes.

Acknowledgements Sven Puchinger (grant BO 867/29-3), Wenhui Li and Vladimir Sidorenko (both grant BO 867/34-1) were supported by the German Research Foundation "Deutsche Forschungsgemeinschaft" (DFG). Johan S. R. Nielsen gratefully acknowledges the support of the Digiteo foundation, project IdealCodes.

References

1. S. A. Abramov and M. Bronstein. On solutions of linear functional systems. In *Proc. of ISSAC*, pages 1–6, New York, NY, USA, 2001.
2. M. Alekhnovich. Linear Diophantine equations over polynomials and soft decoding of Reed–Solomon codes. *IEEE Trans. Inf. Theory*, 51(7):2257–2265, July 2005.
3. B. Beckermann, H. Cheng, and G. Labahn. Fraction-free row reduction of matrices of Ore polynomials. *J. Symb. Comp.*, 41(5):513–543, 2006.
4. D. Boucher and F. Ulmer. Linear codes using skew polynomials with automorphisms and derivations. *Designs, Codes and Cryptography*, 70(3):405–431, 2014.

5. H. Cohn and N. Heninger. Ideal forms of Coppersmith's theorem and Guruswami–Sudan list decoding. *arXiv*, 1008.1284, 2010.
6. P. M. Cohn. *Skew Field Constructions*, volume 27. Cambridge Univ. Press, 1977.
7. P. Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Comb. Th.*, 25(3):226–241, 1978.
8. J. Dieudonné. Les déterminants sur un corps non commutatif. *Bull. Soc. Math. France*, 71:27–45, 1943.
9. P. K. Draxl. *Skew Fields*, volume 81. Cambridge Univ. Press, 1983.
10. E. M. Gabidulin. Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1):3–16, 1985.
11. E. M. Gabidulin, A. Paramonov, and O. Tretjakov. Ideals over a non-commutative ring and their application in cryptology. In *Eurocrypt*, pages 482–489, 1991.
12. P. Giorgi, C. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *Proc. of ISSAC*, pages 135–142, 2003.
13. R. Koetter and F. R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Trans. Inf. Theory*, 54(8):3579–3591, 2008.
14. K. Lee and M. E. O'Sullivan. List decoding of Reed–Solomon codes from a Gröbner basis perspective. *J. Symb. Comp.*, 43(9):645 – 658, 2008.
15. A. Lenstra. Factoring multivariate polynomials over finite fields. *J. Comp. Syst. Sc.*, 30(2):235–246, 1985.
16. W. Li, V. Sidorenko, and D. Silva. On transform-domain error and erasure correction by Gabidulin codes. *Designs, Codes and Cryptography*, 73(2):571–586, 2014.
17. P. Loidreau and R. Overbeck. Decoding rank errors beyond the error correcting capability. In *Proc. of ACCT*, pages 186–190, 2006.
18. T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *J. Symb. Comp.*, 35(4):377–401, 2003.
19. J. S. R. Nielsen. Generalised multi-sequence shift-register synthesis using module minimisation. In *Proc. of IEEE ISIT*, pages 882–886, 2013. Extended version at http://arxiv.org/abs/1301.6529.
20. J. S. R. Nielsen. Power decoding Reed–Solomon codes up to the Johnson radius. In *Proc. of ACCT*, 2014.
21. J. S. R. Nielsen and P. Beelen. Sub-quadratic decoding of one-point Hermitian codes. *arXiv*, 1405.6008, May 2014. Submitted to IEEE Trans. Inf. Theory.
22. O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics*, 34(3):480–508, July 1933.
23. R. M. Roth. Maximum-rank array codes and their application to crisscross error correction. *IEEE Trans. Inf. Theory*, 37(2):328–336, 1991.
24. G. Schmidt, V. R. Sidorenko, and M. Bossert. Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs. *IEEE Trans. Inf. Theory*, 55(7):2991–3012, 2009.
25. V. Sidorenko, L. Jiang, and M. Bossert. Skew-feedback shift-register synthesis and decoding interleaved Gabidulin codes. *IEEE Trans. Inf. Theory*, 57(2):621–632, 2011.
26. D. Silva, F. R. Kschischang, and R. Koetter. A rank-metric approach to error control in random network coding. *IEEE Trans. Inf. Theory*, 54(9):3951–3967, 2008.
27. A. Wachter-Zeh and A. Zeh. Interpolation-based decoding of interleaved Gabidulin codes. In *Proc. of WCC*, pages 528–538, 2013.
28. W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symb. Comp.*, 47(7):793–819, July 2012.