Decoding Interleaved Gabidulin Codes using Alekhnovich's Algorithm

Sven Puchinger[a], Sven Müelich[a], David Mödinger[b], Johan Rosenkilde né Nielsen[c] and Martin Bossert[a 1]

[a] *Institute of Communications Engineering, Ulm University, Ulm, Germany*

[b] *Institute of Distributed Systems, Ulm University, Ulm, Germany*

[c] *Department of Applied Mathematics & Computer Science, Technical University of Denmark, Lyngby, Denmark*

Abstract

We prove that Alekhnovich's algorithm can be used for row reduction of skew polynomial matrices. This yields an $O(\ell^3 n^{(\omega+1)/2} \log(n))$ decoding algorithm for ℓ-Interleaved Gabidulin codes of length n, where ω is the matrix multiplication exponent, improving in the exponent of n compared to previous results.

Keywords: Gabidulin Codes, Characteristic Zero, Low-Rank Matrix Recovery

1 Introduction

It is shown in [1] that *Interleaved Gabidulin codes* of *length* $n \in \mathbb{N}$ and *interleaving degree* $\ell \in \mathbb{N}$ can be error- and erasure-decoded by transforming the

[1] Email: sven.puchinger@uni-ulm.de, sven.mueelich@uni-ulm.de, david.moedinger@uni-ulm.de, jsrn@jsrn.dk, martin.bossert@uni-ulm.de,

following *skew polynomial* [2] matrix into *weak Popov form* (cf. Section 2) [2]:

$$\mathbf{B} = \begin{bmatrix} x^{\gamma_0} & s_1 x^{\gamma_1} & s_2 x^{\gamma_2} & \dots & s_\ell x^{\gamma_\ell} \\ 0 & g_1 x^{\gamma_1} & 0 & \dots & 0 \\ 0 & 0 & g_2 x^{\gamma_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & g_\ell x^{\gamma_\ell} \end{bmatrix}, \tag{1}$$

where the skew polynomials $s_1, \dots, s_\ell, g_1, \dots, g_\ell$ and the non-negative integers $\gamma_0, \dots, \gamma_\ell$ arise from the decoding problem and are known at the receiver. Due to lack of space, we cannot give a description of Interleaved Gabidulin codes, the mentioned procedure and the resulting decoding radius here and therefore refer to [1, Section 3.1.3]. By adapting row reduction [3] algorithms known for polynomial rings $\mathbb{F}[x]$ to skew polynomials, a decoding complexity of $O(\ell n^2)$ can be achieved [1]. In this paper, we adapt Alekhnovich's algorithm [7] for row reduction of $\mathbb{F}[x]$ matrices to the skew polynomial case.

2 Preliminaries

Let $\mathbb{F}$ be a finite field and σ an $\mathbb{F}$-automorphism. A *skew polynomial ring* $\mathbb{F}[x, \sigma]$ [2] contains polynomials of the form $a = \sum_{i=0}^{\deg a} a_i x^i$, where $a_i \in \mathbb{F}$ and $a_{\deg a} \neq 0$ ($\deg a$ is the *degree* of a), which are multiplied according to the rule $x \cdot a = \sigma(a) \cdot x$, extended recursively to arbitrary degrees. This ring is non-commutative in general. All polynomials in this paper are skew polynomials.

It was shown in [6] for linearized polynomials and generalized in [3] to arbitrary skew polynomials that two such polynomials of degrees $\leq s$ can be multiplied with complexity $\mathcal{M}(s) \in O(s^{(\omega+1)/2})$ in operations over $\mathbb{F}$, where ω is the matrix multiplication exponent.

A polynomial a has *length* $\operatorname{len} a$ if $a_i = 0$ for all $i = 0, \dots, \deg a - \operatorname{len} a$ and $a_{\deg a - \operatorname{len} a + 1} \neq 0$. We can write $a = \tilde{a} x^{\deg a - \operatorname{len} a + 1}$, where $\deg \tilde{a} \leq \operatorname{len} a$, and multiply $a, b \in \mathbb{F}[x, \sigma]$ by $a \cdot b = [\tilde{a} \cdot \sigma^{\deg a - \operatorname{len} a + 1}(\tilde{b})] x^{\deg a + \deg a - \operatorname{len} a - \operatorname{len} b + 1}$. Computing $\sigma^i(\alpha)$ with $\alpha \in \mathbb{F}$, $i \in \mathbb{N}$ is in $O(1)$ (cf. [3]). Hence, a and b of length s can be multiplied in $\mathcal{M}(s)$ time, although possibly $\deg a, \deg b \gg s$.

Vectors $\mathbf{v}$ and matrices $\mathbf{M}$ are denoted by bold and small/capital letters. Indices start at 1, e.g. $\mathbf{v} = (v_1, \dots, v_r)$ for $r \in \mathbb{N}$. $\mathbf{E}_{i,j}$ is the matrix containing only one non-zero entry $= 1$ at position (i, j) and $\mathbf{I}$ is the identity matrix. We denote the ith row of a matrix $\mathbf{M}$ by $\mathbf{m}_i$. The degree of a vector $\mathbf{v} \in \mathbb{F}[x, \sigma]^r$ is the maximum of the degrees of its components $\deg \mathbf{v} = \max_i \{\deg v_i\}$ and

[2] Afterwards, the corresponding information words are obtained by ℓ many divisions of skew polynomials of degree $O(n)$, which can be done in $O(\ell n^{(\omega+1)/2} \log(n))$ time [3].

[3] By row reduction we mean to transform a matrix into weak Popov form by row operations.

the degree of a matrix $\mathbf{M}$ is the sum of its rows' degrees $\deg \mathbf{M} = \sum_i \deg \mathbf{m}_i$.

The *leading position* (LP) of $\mathbf{v}$ is the rightmost position of maximal degree $\mathrm{LP}(\mathbf{v}) = \max\{i : \deg v_i = \deg \mathbf{v}\}$. The *leading coefficient* (LC) of a polynomial a is $\mathrm{LT}(a) = a_{\deg a} x^{\deg a}$ and the *leading term* (LT) of a vector $\mathbf{v}$ is $\mathrm{LT}(\mathbf{v}) = v_{\mathrm{LP}(\mathbf{v})}$. A matrix $\mathbf{M} \in \mathbb{F}[x,\sigma]^{r \times r}$ is in *weak Popov form* (wPf) if the leading positions of its rows are pairwise distinct. E.g., the following matrix is in wPf since $\mathrm{LP}(\mathbf{m}_1) = 2$ and $\mathrm{LP}(\mathbf{m}_2) = 1$

$$\mathbf{M} = \begin{bmatrix} x^2 + x & x^2 + 1 \\ x^4 & x^3 + x^2 + x + 1 \end{bmatrix}.$$

Similar to [7], we define an *accuracy approximation to depth* $t \in \mathbb{N}_0$ of skew polynomials as $a|_t = \sum_{i=\deg a - t + 1}^{\deg a} a_i x^i$. For vectors, it is defined as $\mathbf{v}|_t = (v_1|_{\min\{0, t-(\deg \mathbf{v} - \deg v_1)\}}, \dots, v_r|_{\min\{0, t-(\deg \mathbf{v} - \deg v_r)\}})$ and for matrices row-wise. E.g., with $\mathbf{M}$ as above,

$$\mathbf{M}|_2 = \begin{bmatrix} x^2 + x & x^2 \\ x^4 & x^3 \end{bmatrix} \text{ and } \mathbf{M}|_1 = \begin{bmatrix} x^2 & x^2 \\ x^4 & 0 \end{bmatrix}.$$

We can extend the definition of the length of a polynomial to vectors $\mathbf{v}$ as $\operatorname{len} \mathbf{v} = \max_i\{\deg \mathbf{v} - \deg v_i + \operatorname{len} v_i\}$ and to matrices as $\operatorname{len} \mathbf{M} = \max_i\{\operatorname{len} \mathbf{m}_i\}$. With this notation, we have $\operatorname{len}(a|_t) \leq t$, $\operatorname{len}(\mathbf{v}|_t) \leq t$ and $\operatorname{len}(\mathbf{M}|_t) \leq t$.

3 Alekhnovich's Algorithm over Skew Polynomials

Alekhnovich's algorithm [7] was proposed for transforming matrices over ordinary polynomials $\mathbb{F}[x]$ into wPf. Here, we show that, with a few modifications, it also works with skew polynomials. As in the original paper, we prove the correctness of Algorithm 2 (main algorithm) using the auxiliary Algorithm 1.

Algorithm 1 $\mathrm{R}(\mathbf{M})$

Input: Module basis $\mathbf{M} \in \mathbb{F}[x,\sigma]^{r \times r}$ *with* $\deg \mathbf{M} = n$

Output: $\mathbf{U} \in \mathbb{F}[x,\sigma]^{r \times r}$*:* $\mathbf{U} \cdot \mathbf{M}$ *is in wPf or* $\deg(\mathbf{U} \cdot \mathbf{M}) \leq \deg \mathbf{M} - 1$

1. $\mathbf{U} \leftarrow \mathbf{I}$

2. While $\deg \mathbf{M} = n$ *and* $\mathbf{M}$ *is not in wPf*

3. *Find* i, j *such that* $\mathrm{LP}(\mathbf{m}_i) = \mathrm{LP}(\mathbf{m}_j)$ *and* $\deg \mathbf{m}_i \geq \deg \mathbf{m}_j$

4. $\delta \leftarrow \deg \mathbf{m}_i - \deg \mathbf{m}_j$ *and* $\alpha \leftarrow \mathrm{LC}(\mathrm{LT}(\mathbf{m}_i)) / \theta^\delta(\mathrm{LC}(\mathrm{LT}(\mathbf{m}_j)))$

5. $\mathbf{U} \leftarrow (\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j}) \cdot \mathbf{U}$ *and* $\mathbf{M} \leftarrow (\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j}) \cdot \mathbf{M}$

6. Return $\mathbf{U}$

Theorem 3.1 *Algorithm 1 is correct and if* $\operatorname{len}(\mathbf{M}) \leq 1$*, it is in* $O(r^3)$.

Proof. Inside the while loop, the algorithm performs a so-called *simple transformation* (ST). It is shown in [1] that such an ST on an $\mathbb{F}[x,\sigma]$-matrix $\mathbf{M}$

preserves both its rank and row space (this does not trivially follow from the $\mathbb{F}[x]$ case due to non-commutativity) and reduces either $\mathrm{LP}(\mathbf{m}_i)$ or $\deg \mathbf{m}_i$. At some point, $\mathbf{M}$ is in wPf, or $\deg \mathbf{m}_i$ and likewise $\deg \mathbf{M}$ is reduced by one. The matrix $\mathbf{U}$ keeps track of the STs, i.e. multiplying $\mathbf{M}$ by $(\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j})$ from the left is the same as applying an ST on $\mathbf{M}$. At termination, $\mathbf{M} = \mathbf{U} \cdot \mathbf{M}'$, where $\mathbf{M}'$ is the input matrix of the algorithm. Since $\sum_i \mathrm{LP}(\mathbf{m}_i)$ can be decreased at most r^2 times without changing $\deg \mathbf{M}$, the algorithm performs at most r^2 STs. Multiplying $(\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j})$ by a matrix $\mathbf{V}$ consists of scaling a row with αx^δ and adding it to another (target) row. Due to the accuracy approximation, all monomials of the non-zero polynomials in the scaled and the target row have the same power, implying a cost of r for each ST. The claim follows. □

We can decrease a matrix' degree by at least t or transform it into wPf by t recursive calls of Algorithm 1. We can write this as $\mathrm{R}(\mathbf{M}, t) = \mathbf{U} \cdot \mathrm{R}(\mathbf{U} \cdot \mathbf{M})$, where $\mathbf{U} = \mathrm{R}(\mathbf{M}, t-1)$ for $t > 1$ and $\mathbf{U} = \mathbf{I}$ if $t = 1$. As in [7], we speed this method up by two modifications. The first one is a divide-&-conquer (D&C) trick, where instead of reducing the degree of a "$(t-1)$-reduced" matrix $\mathbf{U} \cdot \mathbf{M}$ by 1 as above, we reduce a "t'-reduced" matrix by another $t - t'$ for an arbitrary t'. For $t' \approx t/2$, the recursion tree has a balanced workload.

Lemma 3.2 *Let $t' < t$ and $\mathbf{U} = \mathrm{R}(\mathbf{M}, t')$. Then,*

$$\mathrm{R}(\mathbf{M}, t) = \mathrm{R}\big[\mathbf{U} \cdot \mathbf{M}, t - (\deg \mathbf{M} - \deg(\mathbf{U} \cdot \mathbf{M}))\big] \cdot \mathbf{U}.$$

Proof. $\mathbf{U}$ reduces reduces $\deg \mathbf{M}$ by at least t' or transforms $\mathbf{M}$ into wPf. Multiplication by $\mathrm{R}[\mathbf{U} \cdot \mathbf{M}, t - (\deg \mathbf{M} - \deg(\mathbf{U} \cdot \mathbf{M}))]$ further reduces the degree of this matrix by $t - (\deg \mathbf{M} - \deg(\mathbf{U} \cdot \mathbf{M})) \geq t - t'$ (or $\mathbf{U} \cdot \mathbf{M}$ in wPf).□

The second lemma allows to compute only on the top coefficients of the input matrix inside the divide-&-conquer tree, reducing the overall complexity.

Lemma 3.3 $\mathrm{R}(\mathbf{M}, t) = \mathrm{R}(\mathbf{M}|_t, t)$

Proof. Arguments completely analogous to the $\mathbb{F}[x]$ case of [7, Lemma 2.7] hold. □

Lemma 3.4 $\mathrm{R}(\mathbf{M}, t)$ *contains polynomials of length* $\leq t$.

Proof. The proof works as in the $\mathbb{F}[x]$ case, cf. [7, Lemma 2.8], by taking care of the fact that $\alpha x^a \cdot \beta x^b = \alpha \sigma^c(\beta) x^{a+b}$ for all $\alpha, \beta \in \mathbb{F}$, $a, b \in \mathbb{N}_0$. □

Algorithm 2 $\hat{\mathrm{R}}(\mathbf{M}, t)$
Input: *Module basis* $\mathbf{M} \in \mathbb{F}[x, \sigma]^{r \times r}$ *with* $\deg \mathbf{M} = n$
Output: $\mathbf{U} \in \mathbb{F}[x, \sigma]^{r \times r}$*:* $\mathbf{U} \cdot \mathbf{M}$ *is in wPf or* $\deg(\mathbf{U} \cdot \mathbf{M}) \leq \deg \mathbf{M} - t$

1. If $t=1$*, then Return* $\mathrm{R}(\mathbf{M}|_1)$
2. $\mathbf{U}_1 \leftarrow \hat{\mathrm{R}}(\mathbf{M}|_t, \lfloor t/2 \rfloor)$ *and* $\mathbf{M}_1 \leftarrow \mathbf{U}_1 \cdot \mathbf{M}|_t$
3. Return $\hat{\mathrm{R}}(\mathbf{M}_1, t-(\deg \mathbf{M}|_t - \deg \mathbf{M}_1)) \cdot \mathbf{U}_1$

Theorem 3.5 *Algorithm 2 is correct and has complexity* $O(r^3\mathcal{M}(t))$*.*

Proof. Correctness follows from $\mathrm{R}(\mathbf{M},t) = \hat{\mathrm{R}}(\mathbf{M},t)$ by induction (for $t=1$, see Theorem 3.1). Let $\hat{\mathbf{U}} = \hat{\mathrm{R}}(\mathbf{M}|_t, \lfloor \frac{t}{2} \rfloor)$ and $\mathbf{U} = \mathrm{R}(\mathbf{M}|_t, \lfloor \frac{t}{2} \rfloor)$. Then,

$$\begin{aligned}\hat{\mathrm{R}}(\mathbf{M},t) &= \hat{\mathrm{R}}(\hat{\mathbf{U}} \cdot \mathbf{M}|_t, t-(\deg \mathbf{M}|_t - \deg(\hat{\mathbf{U}} \cdot \mathbf{M}|_t))) \cdot \hat{\mathbf{U}} \\ &\overset{\text{(i)}}{=} \mathrm{R}(\mathbf{U} \cdot \mathbf{M}|_t, t-(\deg \mathbf{M}|_t - \deg(\mathbf{U} \cdot \mathbf{M}|_t))) \cdot \mathbf{U} \overset{\text{(ii)}}{=} \mathrm{R}(\mathbf{M}|_t, t) \overset{\text{(iii)}}{=} \mathrm{R}(\mathbf{M},t),\end{aligned}$$

where (i) follows from the induction hypothesis, (ii) by Lemma 3.2, and (iii) by Lemma 3.3. Algorithm 2 calls itself twice on inputs of sizes $\approx \frac{t}{2}$. The only other costly operations are the matrix multiplications in Lines 2 and 3 of matrices containing only polynomials of length $\leq t$ (cf. Lemma 3.4). This costs[4] r^2 times r multiplications $\mathcal{M}(t)$ and r^2 times r additions $O(t)$ of polynomials of length $\leq t$, having complexity $O(r^3\mathcal{M}(t))$. The recursive complexity relation reads $f(t) = 2 \cdot f(\frac{t}{2}) + O(r^3\mathcal{M}(t))$. By the master theorem, we get $f(t) \in O(tf(1) + r^3\mathcal{M}(t))$. The base case operation $\mathrm{R}(\mathbf{M}|_1)$ with cost $f(1)$ is called at most t times since it decreases $\deg \mathbf{M}$ by 1 each time. Since $\mathrm{len}(\mathbf{M}|_1) \leq 1$, $f(1) \in O(r^3)$ by Theorem 3.1. Hence, $f(t) \in O(r^3\mathcal{M}(t))$. □

4 Implications and Conclusion

The *orthogonality defect* [1] of a square, full-rank, skew polynomial matrix $\mathbf{M}$ is $\Delta(\mathbf{M}) = \deg \mathbf{M} - \deg\det \mathbf{M}$, where $\deg\det$ is the "determinant degree" function, see [1]. A matrix $\mathbf{M}$ in wPf has $\Delta(\mathbf{M}) = 0$ and $\deg\det \mathbf{M}$ is invariant under row operations. Thus, if $\mathbf{V}$ is in wPf and obtained from $\mathbf{M}$ by simple transformations, then $\deg \mathbf{V} = \Delta(\mathbf{V}) + \deg\det \mathbf{V} = \deg \mathbf{M} - \Delta(\mathbf{M})$. With $\Delta(\mathbf{M}) \geq 0$, this implies that $\hat{\mathrm{R}}(\mathbf{M}, \Delta(\mathbf{M})) \cdot \mathbf{M}$ is always in wPf. It was shown in [1] that $\mathbf{B}$ from Equation (1) has orthogonality defect $\Delta(\mathbf{B}) \in O(n)$, which implies the following theorem.

Theorem 4.1 (Main Statement) $\hat{\mathrm{R}}(\mathbf{B}, \Delta(\mathbf{B})) \cdot \mathbf{B}$ *is in wPf. This implies that we can decode Interleaved Gabidulin codes in*[5] $O(\ell^3 n^{(\omega+1)/2} \log(n))$*.*

---

[4] In D&C matrix multiplication algorithms, the length of polynomials in intermediate computations might be much larger than t. Thus, we have to compute it naively in cubic time.
[5] The $\log(n)$ factor is due to the divisions in the decoding algorithm, following the row reduction step (see Footnote 2) and can be omitted if $\log(n) \in o(\ell^2)$.

Table 1 compares the complexities of known decoding algorithms for Interleaved Gabidulin codes. Which algorithm is asymptotically fastest depends on the relative size of ℓ and n. Usually, one considers $n \gg \ell$, in which case the algorithms in this paper and in [4] provide—to the best of our knowledge—the fastest known algorithms for decoding Interleaved Gabidulin codes.

Algorithm	Complexity
Skew Berlekamp–Massey [5]	$O(\ell n^2)$
Skew Berlekamp–Massey (D&C) [4]	$O(\ell^K n^{\frac{\omega+1}{2}} \log(n))$, possibly [6] $K = 3$
Skew Demand–Driven* [1]	$O(\ell n^2)$
Skew Alekhnovich* (Theorem 3.5)	$O(\ell^3 n^{\frac{\omega+1}{2}} \log(n)) \subseteq^\dagger O(\ell^3 n^{1.69} \log(n))$

Table 1

Comparison of decoding algorithms for Interleaved Gabidulin codes. Algorithms marked with * are based on the row reduction problem of [1]. †Example $\omega \approx 2.37$.

In the case of Gabidulin codes ($\ell = 1$), we obtain an alternative to the *Linearized Extended Euclidean* algorithm from [6] of the same complexity. The algorithms are equivalent up to the implementation of a simple transformation.

References

[1] S. Puchinger, J. Rosenkilde né Nielsen, W. Li, and V. Sidorenko, "Row Reduction Applied to Rank-Metric and Subspace Codes," *Des. Codes Cryptogr.*, DOI: 10.1007/s10623-016-0257-9, arXiv preprint 1510.04728, 2016.

[2] O. Ore, "Theory of Non-commutative Polynomials," *Ann. Math.*, pp. 480–508, 1933.

[3] S. Puchinger and A. Wachter-Zeh, "Fast Operations on Linearized Polynomials and their Applications in Coding Theory," *Submitted to: Journal of Symbolic Computation*, 2016, arXiv preprint 1512.06520.

[4] V. Sidorenko and M. Bossert, "Fast Skew-Feedback Shift-Register Synthesis," *Des. Codes Cryptogr.*, vol. 70, no. 1-2, pp. 55-67, 2014.

[5] V. Sidorenko, L. Jiang, and M. Bossert, "Skew-Feedback Shift-Register Synthesis and Decoding Interleaved Gabidulin Codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 621–632, 2011.

[6] A. Wachter-Zeh, "Decoding of Block and Convolutional Codes in Rank Metric," Ph.D. dissertation, Ulm University and University of Rennes, 2013.

[7] M. Alekhnovich, "Linear Diophantine Equations over Polynomials and Soft Decoding of Reed–Solomon Codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2257–2265, 2005.

[6] In [4], the complexity is given as $O(n^{\frac{\omega+1}{2}} \log(n))$ and ℓ is considered to be constant. By a rough estimate, the complexity becomes $O(\ell^{O(1)} n^{\frac{\omega+1}{2}} \log(n))$ when including ℓ. We believe the exponent of ℓ is really 3 (or possibly ω) but this should be further analyzed.