Fifteenth International Workshop on Algebraic and Combinatorial Coding Theory
June 18-24, 2016, Albena, Bulgaria *pp. 255–260*

Decoding Interleaved Gabidulin Codes using Alekhnovich's Algorithm[1]

Sven Puchinger sven.puchinger@uni-ulm.de
Institute of Communications Engineering, University of Ulm, Germany
Sven Müelich sven.mueelich@uni-ulm.de
Institute of Communications Engineering, University of Ulm, Germany
David Mödinger david.moedinger@uni-ulm.de
Institute of Distributed Systems, University of Ulm, Germany
Johan S. R. Nielsen jsrn@jsrn.dk
Dpt. of App. Mat. & Computer Science, Technical University of Denmark, Denmark
Martin Bossert martin.bossert@uni-ulm.de
Institute of Communications Engineering, University of Ulm, Germany

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.

1 Introduction

It is shown in [1, 2] 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 following *skew polynomial* [3] 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 comprehensive description of Interleaved Gabidulin codes, the mentioned procedure and the resulting decoding radius here and therefore refer to [2, Section 3.1.3]. By adapting row reduction[3] algorithms known for polynomial rings $\mathbb{F}[x]$ to skew polynomial rings, decoding

[1]This work was supported by Deutsche Forschungsgemeinschaft under grant BO 867/29-3.
[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 [4].
[3]By row reduction we mean to transform a matrix into weak Popov form by row operations.

complexities of $O(\ell^2 n^2)$ and $O(\ell n^2)$ can be achieved [2], the latter being as fast as the algorithm in [5]. 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* $\mathbb{F}[x,\sigma]$ [3] 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 [4] to arbitrary skew polynomials that multiplication of 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.

We say that a polynomial a has *length* $\operatorname{len} a$ if $a_i = 0$ for all $i = 0, \ldots, \deg a - \operatorname{len} a$ and $a_{\deg a - \operatorname{len} a + 1} \neq 0$. Thus, it can be written as $a = \tilde{a} x^{\deg a - \operatorname{len} a + 1}$, where $\deg \tilde{a} \leq \operatorname{len} a$ and the multiplication of two polynomials a, b of length $\leq s$ can be accomplished as $a \cdot b = [\tilde{a} \cdot \sigma^{\deg a - \operatorname{len} a + 1}(\tilde{b})] x^{\deg a + \deg a - \operatorname{len} a - \operatorname{len} b + 1}$. It is a reasonable assumption in a that computing $\sigma^i(\alpha)$ with $\alpha \in \mathbb{F}$, $i \in \mathbb{N}$ is in $O(1)$ (cf. [4]). Hence, a and b can be multiplied in $\mathcal{M}(s)$ time, although their degrees might be $\gg s$.

Vectors $\mathbf{v}$ and matrices $\mathbf{M}$ are denoted by bold and small/capital letters. Indices start at 1, e.g. $\mathbf{v} = (v_1, \ldots, v_r)$ 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 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 $\mathrm{LP}(\mathbf{v}) = \max\{i : \deg v_i = \deg \mathbf{v}\}$. We say that 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 weak Popov form 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 $\mathbf{v}|_t = (v_1|_{\min\{0, t - (\deg \mathbf{v} - \deg v_1)\}}, \ldots, v_r|_{\min\{0, t - (\deg \mathbf{v} - \deg v_r)\}})$ and for matrices row-wise, where the degrees of the rows are allowed to be different. 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 weak Popov form. In this section, we show that, with a few modifications, it also works with skew polynomial matrices. 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 $\deg(\mathbf{U}\cdot\mathbf{M}) \leq \deg\mathbf{M} - 1$

$\mathbf{U} \leftarrow \mathbf{I}$
while $\deg\mathbf{M} = n$ *and* $\mathbf{M}$ *is not in weak Popov form* **do**
  Find i, j such that $\mathrm{LP}(\mathbf{m}_i) = \mathrm{LP}(\mathbf{m}_j)$ and $\deg\mathbf{m}_i \geq \deg\mathbf{m}_j$
  $\delta \leftarrow \deg\mathbf{m}_i - \deg\mathbf{m}_j$ and $\alpha \leftarrow \mathrm{LC}(\mathrm{LT}(\mathbf{m}_i))/\theta^\delta(\mathrm{LC}(\mathrm{LT}(\mathbf{m}_j)))$
  $\mathbf{U} \leftarrow (\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j})\cdot\mathbf{U}$ and $\mathbf{M} \leftarrow (\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j})\cdot\mathbf{M}$
return $\mathbf{U}$

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

Proof Inside the while loop, the algorithm performs a so-called *simple transformation.* It is shown in [2] that such a simple transformation on an $\mathbb{F}[x,\sigma]$-matrix $\mathbf{M}$ preserves both its rank and row space (note that 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 weak Popov form (iff no simple transformation is possible anymore), or $\deg\mathbf{m}_i$ and likewise $\deg\mathbf{M}$ is reduced by one. The matrix $\mathbf{U}$ keeps track of the simple transformations, i.e. multiplying $\mathbf{M}$ by $(\mathbf{I} - \alpha x^\delta \mathbf{E}_{i,j})$ from the left is the same as applying a simple transformation 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 simple transformations. 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 simple transformation. The claim follows. ∎

We can decrease a matrix' degree by at least t or transform it into weak Popov form by t recursive calls of Algorithm 1. We can write this operation 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 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 1 *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 $\mathbf{U}$ is a matrix that 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, thus reducing the overall complexity.

Lemma 2 $\mathrm{R}(\mathbf{M}, t) = \mathrm{R}(\mathbf{M}|_t, t)$

Proof Elementary row operations as in Algorithm 1 behave exactly as their $\mathbb{F}[x]$ equivalent, cf. [2]. Hence, the arguments of [7, Lemma 2.7] hold. ∎

Lemma 3 $\mathrm{R}(\mathbf{M}, t)$ *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 $\deg(\mathbf{U} \cdot \mathbf{M}) \leq \deg \mathbf{M} - t$

```
M ← M|_t
if t = 1 then
    return R(M)
U_1 ← R̂(M, ⌊t/2⌋)
M_1 ← U_1 · M
return R̂(M_1, t − (deg M − deg M_1)) · U_1
```

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

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

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

where (i) follows from the induction hypothesis, (ii) by Lemma 1, and (iii) by Lemma 2. Algorithm 2 calls itself twice on inputs of sizes $\approx \frac{t}{2}$. The only other costly operations are the matrix multiplications in Lines 5 and 6 of matrices containing only polynomials of length $\leq t$ (cf. Lemma 3). In order to control the size of the polynomial operations within the matrix multiplication, sophisticated matrix multiplication algorithms are not suitable in this case. E.g., in divide-&-conquer methods like Strassen's algorithm the length of polynomials in intermediate computations might be much larger than t. Using the definition of matrix multiplication, we will have 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 $f(t) = 2 \cdot f(\frac{t}{2}) + O(r^3\mathcal{M}(t))$. 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. With the master theorem, we obtain $f(t) \in O(tf(1) + r^3\mathcal{M}(t))$. $\mathrm{R}(\mathbf{M}|_1)$ calls Algorithm 1 on input matrices of length 1, implying $f(1) \in O(r^3)$ (cf. Theorem 1). Hence, $f(t) \in O(r^3\mathcal{M}(t))$. ∎

4 Implications and Conclusion

The *orthogonality defect* [2] of a square, full-rank, skew polynomial matrix $\mathbf{M}$ is $\Delta(\mathbf{M}) = \deg \mathbf{M} - \deg \det \mathbf{M}$, where det is any Dieudonné determinant; see [2] why $\Delta(\mathbf{M})$ does not depend on the choice of det. It can be shown that $\deg \det \mathbf{M}$ is invariant under row operations and a matrix $\mathbf{M}$ in weak Popov form has $\Delta(\mathbf{M}) = 0$. Thus, if $\mathbf{V}$ is in wPf and obtained from $\mathbf{M}$ by simple transformations, then $\deg \mathbf{V} = \Delta(\mathbf{V}) + \deg \det \mathbf{V} = 0 + \deg \det \mathbf{M} = \deg \mathbf{M} - \Delta(\mathbf{M})$. In combination with $\Delta(\mathbf{M}) \geq 0$, this implies that $\hat{\mathrm{R}}(\mathbf{M}, \Delta(\mathbf{M})) \cdot \mathbf{M}$ is always in weak Popov form. It was shown in [2] that $\mathbf{B}$ from Equation (1) has orthogonality defect $\Delta(\mathbf{B}) \in O(n)$, which implies the following theorem.

Theorem 3 (Main Statement) $\hat{\mathrm{R}}(\mathbf{B}, \Delta(\mathbf{B})) \cdot \mathbf{B}$ *is in weak Popov form. This implies that we can decode Interleaved Gabidulin codes in*[4] $O(\ell^3 n^{(\omega+1)/2} \log(n))$.

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 algorithm of

[4] The $\log(n)$ factor is due to the divisions in the decoding algorithm, following the row reduction step (see Footnote 2 on the first page) and can be omitted if $\log(n) \in o(\ell^2)$.

this paper provides—to the best of our knowledge—the fastest known algorithm for decoding Interleaved Gabidulin codes.

Algorithm	Complexity
Generalized Berlekamp–Massey [5]	$O(\ell n^2)$
Mulders–Storjohann* [2]	$O(\ell^2 n^2)$
Demand–Driven* [2]	$O(\ell n^2)$
Alekhnovich* (Theorem 2)	$O(\ell^3 n^{\frac{\omega+1}{2}} \log(n))$ $\subseteq \begin{cases} O(\ell^3 n^{1.91} \log(n)), & \omega \approx 2.81, \\ O(\ell^3 n^{1.69} \log(n)), & \omega \approx 2.37. \end{cases}$

Table 1: Comparison of decoding algorithms for Interleaved Gabidulin codes. Algorithms marked with * are based on the row reduction problem of [2].

Note that in the case of non-interleaved Gabidulin codes ($\ell = 1$), we obtain an alternative to the *Linearized Extended Euclidean* algorithm from [6] of almost the same complexity. In fact, the two algorithms are equivalent except for the implementation of a simple transformation.

References

[1] W. Li, J. S. R. Nielsen, S. Puchinger, and V. Sidorenko, "Solving Shift Register Problems over Skew Polynomial Rings using Module Minimisation," in *International Workshop on Coding and Cryptography*, Paris, 2015.

[2] S. Puchinger, J. S. R. Nielsen, W. Li, and V. Sidorenko, "Row Reduction Applied to Rank Metric and Subspace Codes," *Submitted to Designs, Codes and Cryptography*, 2015, arXiv preprint 1510.04728.

[3] O. Ore, "Theory of Non-commutative Polynomials," *Annals of mathematics*, pp. 480–508, 1933.

[4] 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.

[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.