1. INTRODUCTION

Decoding Interleaved Gabidulin Codes using Alekhnovich’s Algorithm

Sven Puchinger  
Sven Müelich  
David Mödinger  
Johan S. R. Nielsen  
Martin Bossert

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 $\ell$-Interleaved Gabidulin codes of length $n$, where $\omega$ 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 matrix into weak Popov form (cf. Section 2):

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

where the skew polynomials $s_1, \ldots, s_\ell, g_1, \ldots, g_\ell$ and the non-negative integers $\gamma_0, \ldots, \gamma_\ell$ 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 algorithms known for polynomial rings $\mathbb{F}[x]$ to skew polynomial rings, decoding

---

1This work was supported by Deutsche Forschungsgemeinschaft under grant BO 867/29-3.
2Afterwards, the corresponding information words are obtained by $\ell$ many divisions of skew polynomials of degree $O(n)$, which can be done in $O(\ell n^{(\omega+1)/2} \log(n))$ time [4].
3By 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 [3]. In this paper, we adapt Alekhnovich’s algorithm [7] for row reduction of $F[x]$ matrices to the skew polynomial case.

## 2 Preliminaries

Let $F$ be a finite field and $\sigma$ an $F$-automorphism. A skew polynomial ring $F[x, \sigma]$ [3] contains polynomials of the form $a = \sum_{i=0}^{\deg a} a_i x^i$, where $a_i \in F$ 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 $M(s) = O(s^{(\omega+1)/2})$ in operations over $F$, where $\omega$ is the matrix multiplication exponent.

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

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

The leading position (LP) of $v$ is the rightmost position of maximal degree $\LP(v) = \max\{i : \deg v_i = \deg v\}$. We say that the leading coefficient (LC) of a polynomial $a$ is $\LT(a) = a_{\deg a} x^{\deg a}$ and the leading term (LT) of a vector $v$ is $\LT(v) = v_{\LP(v)}$. A matrix $M \in F[x, \sigma]^{r \times r}$ 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 $\LP(m_1) = 2$ and $\LP(m_2) = 1$.

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

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

$$M|_2 = \begin{bmatrix} x^2 + x & x^2 \\ x^3 & x^3 \end{bmatrix}$$

and $M|_1 = \begin{bmatrix} x^2 & x^2 \\ x^4 & 0 \end{bmatrix}$. 


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

3 Alekhnovich’s Algorithm over Skew Polynomials

Alekhnovich’s algorithm \cite{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: R(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 \( \text{wPf} \) or \( \deg(\mathbf{U} \cdot \mathbf{M}) \leq \deg \mathbf{M} - 1 \)

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

**Theorem 1** Algorithm \cite{7} is correct and if \( \text{len}(\mathbf{M}) \leq 1 \), it has complexity \( O \left( r^3 \right) \).

**Proof** Inside the while loop, the algorithm performs a so-called *simple transformation*. It is shown in \cite{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 \( \text{LP}(\mathbf{m}_i) \) 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 \( \sum_i \text{LP}(\mathbf{m}_i) \) 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 \( \alpha x^\delta \) 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. \( \blacksquare \)
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 $R(M, t) = U \cdot R(U \cdot M)$, where $U = R(M, t-1)$ for $t > 1$ and $U = I$ 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 $U \cdot M$ 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 $U = R(M, t')$. Then,

$$R(M, t) = R[U \cdot M, t - (\deg M - \deg(U \cdot M))] \cdot U.$$  

Proof $U$ is a matrix that reduces $\deg M$ by at least $t'$ or transforms $M$ into wPf. Multiplication by $R[U \cdot M, t - (\deg M - \deg(U \cdot M))]$ further reduces the degree of this matrix by $t - (\deg M - \deg(U \cdot M)) \geq t - t'$ (or $U \cdot M$ 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 $R(M, t) = R(M|_t, t)$

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

Lemma 3 $R(M, t)$ contains polynomials of length $\leq t$.

Proof The proof works as in the $F[x]$ 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 $\alpha, \beta \in F$, $a, b \in N_0$.

Algorithm 2: $\hat{R}(M, t)$

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

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

1 $M \leftarrow M|_t$
2 if $t = 1$ then
3 \hspace{1cm} return $R(M)$
4 $U_1 \leftarrow \hat{R}(M, \lfloor t/2 \rfloor)$
5 $M_1 \leftarrow U_1 \cdot M$
6 return $\hat{R}(M_1, t - (\deg M - \deg M_1)) \cdot U_1$

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

4. IMPLICATIONS AND CONCLUSION

**Proof** Correctness follows from \( R(M, t) = \hat{R}(M, t) \), which can be proven by induction (for \( t = 1 \), see Theorem 1). Let \( \hat{U} = \hat{R}(M|_t, [\ell/2]) \) and \( U = R(M|_t, [\ell/2]) \).

\[
\hat{R}(M, t) = \hat{R}(\hat{U} \cdot M|_t, t - (\deg M|_t - \deg(\hat{U} \cdot M|_t))) \cdot \hat{U}
\]

\[ \overset{(i)}{=} R(U \cdot M|_t, t - (\deg M|_t - \deg(U \cdot M|_t))) \cdot U \overset{(ii)}{=} R(M|_t, t) \overset{(iii)}{=} R(M, t), \]

where (i) follows from the induction hypothesis, (ii) by Lemma 1 and (iii) by Lemma 2. Algorithm 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 

**4 Implications and Conclusion**

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

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

Table 4 compares the complexities of known decoding algorithms for Interleaved Gabidulin codes. Which algorithm is asymptotically fastest depends on the relative size of \( \ell \) and \( n \). Usually, one considers \( n \gg \ell \), in which case the algorithm of

\[ \text{4The log}(n) \text{ factor is due to the divisions in the decoding algorithm, following the row reduction step (see Footnote 2 on the first page) and can be omitted if log}(n) \in o(\ell^2). \]
this paper provides—to the best of our knowledge—the fastest known algorithm for decoding Interleaved Gabidulin codes.

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

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. Wachtter-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. Wachtter-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.