Decoding Interleaved Gabidulin Codes using Alekhnovich’s Algorithm

Sven Puchinger\textsuperscript{a}, Sven Müelich\textsuperscript{a}, David Mödinger\textsuperscript{b},
Johan Rosenkilde né Nielsen\textsuperscript{c} and Martin Bossert\textsuperscript{a} \textsuperscript{1}

\textsuperscript{a}Institute of Communications Engineering, Ulm University, Ulm, Germany
\textsuperscript{b}Institute of Distributed Systems, Ulm University, Ulm, Germany
\textsuperscript{c}Department of Applied Mathematics & Computer Science, Technical University of
Denmark, Lyngby, Denmark

\section*{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.

\textbf{Keywords:} Gabidulin Codes, Characteristic Zero, Low-Rank Matrix Recovery

\section{Introduction}
It is shown in \cite{1} that \textit{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 \[2\] matrix into weak Popov form (cf. Section 2)\(^2\):

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

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 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 \(O((\ell n^2)\log(n))\) 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 \(\sigma\) 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 \(\omega\) is the matrix multiplication exponent.

A polynomial \(a\) has length \(\text{len } a\) if \(a_i = 0\) for all \(i = 0, \ldots, \deg a - \text{len } a\) and \(a_{\deg a - \text{len } a + 1} \neq 0\). We can write \(a = \tilde{a} x^{\deg a - \text{len } a + 1}\), where \(\deg \tilde{a} \leq \text{len } a\), and multiply \(a, b \in \mathbb{F}[x, \sigma]\) by \(a \cdot b = [\tilde{a} \cdot \sigma^{\deg a - \text{len } a + 1}(\tilde{b})] x^{\deg a + \deg a - \text{len } a - \text{len } b + 1}\). 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. \(\mathbf{v} = (v_1, \ldots, v_r)\) 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 \(i\)th 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 \(\ell\) 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 $M$ 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 $\text{LP}(v) = \max\{i : \deg v_i = \deg v\}$. The leading coefficient (LC) of a polynomial $a$ is $\text{LC}(a) = a_{\deg a}x^{\deg a}$ and the leading term (LT) of a vector $v$ is $\text{LT}(v) = v_{\text{LP}(v)}$. A matrix $M \in \mathbb{F}[x, \sigma]^{r \times r}$ 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 $\text{LP}(m_1) = 2$ and $\text{LP}(m_2) = 1$

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

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

$$M_{|2} = \begin{bmatrix} x^2 + x & x^2 \\ x^3 & x^3 \end{bmatrix} \text{ and } M_{|1} = \begin{bmatrix} x^2 \\ x^4 & 0 \end{bmatrix}.$$  

We can extend the definition of the length of a polynomial to vectors $v$ as $\text{len}(v) = \max_i (\deg v - \deg v_i + \text{len} v_i)$ and to matrices as $\text{len}(M) = \max_i \{\text{len}(m_i)\}$. With this notation, we have $\text{len}(a_t) \leq t$, $\text{len}(v_t) \leq t$ and $\text{len}(M_{|t}) \leq t$.

## 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 R(M)

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

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

1. $U \leftarrow I$
2. While $\deg M = n$ and $M$ is not in wPf
3. Find $i, j$ such that $\text{LP}(m_i) = \text{LP}(m_j)$ and $\deg m_i \geq \deg m_j$
4. $\delta \leftarrow \deg m_i - \deg m_j$ and $\alpha \leftarrow \text{LC}(\text{LT}(m_i))/\theta^\delta(\text{LC}(\text{LT}(m_j)))$
5. $U \leftarrow (I - \alpha x^\delta E_{i,j}) \cdot U$ and $M \leftarrow (I - \alpha x^\delta E_{i,j}) \cdot M$
6. Return $U$

### Theorem 3.1

Algorithm 1 is correct and if $\text{len}(M) \leq 1$, 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 $M$
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 $\text{LP}(m_i)$ or $\deg m_i$. At some point, $M$ is in wPf, or $\deg m_i$ and likewise $\deg M$ is reduced by one. The matrix $U$ keeps track of the STs, i.e. multiplying $M$ by $(I - \alpha x^dE_{i,j})$ from the left is the same as applying an ST on $M$. At termination, $M = U \cdot M'$, where $M'$ is the input matrix of the algorithm. Since $\sum_i \text{LP}(m_i)$ can be decreased at most $r^2$ times without changing $\deg M$, the algorithm performs at most $r^2$ STs. Multiplying $(I - \alpha x^dE_{i,j})$ by a matrix $V$ consists of scaling a row with $\alpha x^d$ 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. $\blacksquare$

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 $R(M, t) = U \cdot R(U \cdot M)$, 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 (D&C) 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 3.2** 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.** $U$ reduces 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). $\square$

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** $R(M, t) = R(M|_{t}, t)$

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

**Lemma 3.4** $R(M, t)$ 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 $\alpha x^a \cdot \beta x^b = \alpha \sigma^e(\beta) x^{a+b}$ for all $\alpha, \beta \in \mathbb{F}$, $a, b \in \mathbb{N}_0$. $\square$

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

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

**Output:** $U \in \mathbb{F}[x, \sigma]^{r \times r}$: $U \cdot M$ is in wPf or $\deg(U \cdot M) \leq \deg M - t$
1. If \( t = 1 \), then Return \( R(M|_1) \)
2. \( U_1 \leftarrow \hat{R}(M|_t, \lfloor t/2 \rfloor) \) and \( M_1 \leftarrow U_1 \cdot M|_t \)
3. Return \( R(M_1, t - (\deg M|_t - \deg M_1)) \cdot U_1 \)

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

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

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

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 \( r^2 \) times \( r \) multiplications \( M(t) \) and \( r^2 \) times \( r \) additions \( O(t) \) of polynomials of length \( \leq t \), having complexity \( O(r^3M(t)) \). The recursive complexity relation reads \( f(t) = 2 \cdot f(\frac{t}{2}) + O(r^3M(t)) \). By the master theorem, we get \( f(t) \in O(tf(1) + r^3M(t)) \). The base case operation \( R(M|_1) \) with cost \( f(1) \) is called at most \( t \) times since it decreases \( \deg M \) by 1 each time. Since \( \text{len}(M|_1) \leq 1 \), \( f(1) \in O(r^3) \) by Theorem 3.1. Hence, \( f(t) \in O(r^3M(t)) \). \( \square \)

### 4 Implications and Conclusion

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

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

---

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 $\ell$ 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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skew Berlekamp–Massey [5]</td>
<td>$O(\ell n^2)$</td>
</tr>
<tr>
<td>Skew Berlekamp–Massey (D&amp;C) [4]</td>
<td>$O(\ell^K n^{\frac{\omega+1}{2}} \log(n))$, possibly $^6 K = 3$</td>
</tr>
<tr>
<td>Skew Demand–Driven* [1]</td>
<td>$O(\ell n^2)$</td>
</tr>
<tr>
<td>Skew Alekhnovich* (Theorem 3.5)</td>
<td>$O(\ell^3 n^{\frac{\omega}{2}} \log(n)) \subseteq O(\ell^3 n^{1.69} \log(n))$</td>
</tr>
</tbody>
</table>

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


$^6$ In [4], the complexity is given as $O(n^{\frac{\omega+1}{2}} \log(n))$ and $\ell$ 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 $\ell$. We believe the exponent of $\ell$ is really 3 (or possibly $\omega$) but this should be further analyzed.