*ACM Communications in Computer Algebra, Vol. 53, No. 1, Issue 207, March 2019*

# Row reduction process for matrices of scalar operators: storing the intermediate results of row transformation

Sergei A. Abramov
Dorodnicyn Computing Centre
Federal Research Center
“Computer Science and Control”
of the Russian Academy of Sciences;
Vavilova 40, Moscow, 119333
Russia
sergeyabramov@mail.ru

Moulay A. Barkatou
Institut XLIM, DMI,
Université de Limoges;
CNRS
123, Av. A. Thomas,
87060 Limoges
France
moulay.barkatou@unilim.fr

**Abstract**

It is well known that if the leading matrix of a linear ordinary differential or difference system is nonsingular, then the determinant of this matrix contains some useful information on solutions of the system. We investigate a kind of non-arithmetic complexity of known algorithms for transforming a matrix of scalar operators to an equivalent matrix which has non-singular frontal, or, leading matrix. In the algorithms under consideration, the differentiation in the differential case and the shift in the difference case play a significant role. We give some analysis of the complexity measured as the number of differentiations or, resp., shifts in the worst case. We not only offer estimates of the complexity written using the $O$-notation, but we also show that some estimates are sharp and can not be improved.

## 1 Introduction

We will consider a differential, or difference field $K$ of characteristic 0 with a derivation $\partial$ (the differential case) or with an automorphism $\sigma$ (the difference case). To consider both cases simultaneously, we use the common notation $\top$ for $\partial$ and $\sigma$. We denote by $K[\top]$ the ring of Ore polynomials in $\top$ over $K$. For seek of clarity, we will use the symbol $\circ$ to denote the operation of multiplication (composition) in the ring $K[\top]$ : if $f, g \in K[\top]$ then $f \circ g$, the composition of $f$ and $g$, denotes the product of $f$ and $g$ as Ore polynomials. If $a \in K$ then $\top(a)$ denotes the $\top -$ *image* of $a$, i.e., the result of applying the map $\top$ to $a$; one gets $\top(a) \in K$. The multiplication in $K[\top]$ is performed according to known rules, which differ in the cases of $\top = \partial$ and $\top = \sigma$. If, e.g., $f \in K$ then $\top \circ f = \top(f)\top$ when $\top = \sigma$ and $\top \circ f = f\top + \top(f)$ when $\top = \partial$. Elements of $K[\top]$ are called scalar operators (we will also say ‘scalar $\top$-operators’). The order of a nonzero scalar operator $f \in K[\top]$ is equal to the degree of $f$ considered as a polynomial in $\top$. The order of the zero operator is $-\infty$. For a finite collection $F$ of scalar operators (a vector, matrix, row matrix etc), $\operatorname{ord} F$ is defined as the maximum of the orders of its elements. We will denote by $K[\top]^{m\times n}$, the set of $m \times n$-matrices of scalar operators (i.e., $m \times n$-matrices with entries in $K[\top]$). Any nonzero $L \in K[\top]^{n\times n}$ can be represented in the *expanded* form i.e., as a $\top$-operator with coefficients belonging to $K^{n\times n}$:

$$L = A_d\top^d + A_{d-1}\top^{d-1} + \cdots + A_0, \tag{1}$$

where $A_d$ (the *leading matrix* of $L$) is non-zero. The notation $d$ will be used for the order of $L$ (we write $d = \operatorname{ord} L$). We will refer to a matrix $L \in K^{n\times n}$ of the form (1)) as a matrix operator of size $n$ and order $d$. In the sequel, we will associate to such a matrix of $\top$-operators some matrices belonging to $K^{n\times n}$ (like the leading matrix $A_d$ in (1)). To avoid a terminology confusion, matrices of $\top$-operators will be briefly referred to as *operators*.

If $r = (r_1, \dots, r_n) \in K^{1\times n}[\top]$ and $f \in K[\top]$ then $f \circ r = (\top \circ r_1, \dots, \top \circ r_n)$ where as above $f \circ r_i$ is the composition of $f$ and $r_i$. If $r$ is a row matrix belonging to $K^{1\times n}[\top]$ then the composition $\top \circ r$ is called the $\top$ - *portage* of the row $r$.

An operator $L$ is of *full rank* (or is a full-rank operator) if its rows are linearly independent over $K[\top]$.

In the sequel, we shall consider without special mention only operators of full rank. In particular, a full rank operator has no zero row. This is quite enough for our purposes: we will prove that even for those operators the $\top$-complexity of the transformations of interest to us is quite large. The assumption that the operator $L$ is of full rank, concerns, in particular, the definition of the frontal matrix and the description of the algorithm Row Reduction.

**Definition 1** Suppose that $L \in K[\top]^{n\times n}$, $d = \operatorname{ord} L$. Let $\alpha_i = \operatorname{ord} L_{i,*}$ (i.e., $\alpha_i$ is the order of the $i$th row of $L$), $i = 1, \dots, n$. The leading matrix of the product

$$\operatorname{diag}(\top^{d-\alpha_1}, \dots, \top^{d-\alpha_n})L$$

is the *frontal matrix* of $L$ (in [10, 9], this matrix is called the leading coefficient matrix).

The Row Reduction algorithm [10, 9] allows to construct for a given operator $L$ an equivalent operator $L'$ such that its frontal matrix is non-singular ($L' = UL$ for some *unimodular*, i.e. invertible in $K[\top]^{n\times n}$, operator $U$). A description of this algorithm is given in Section 2; the algorithm itself is named RR there.

The output operator $L'$ is constructed in several steps by using elementary transformations of the rows of $L$. This needs to apply a significant amount of $\top$-portages to the rows of the input operator. Therefore, this algorithm requires a significant number of $\top$-images of elements in the ground field $K$. This number is estimated[1] to

$$\Theta(n^3 d^3), \tag{2}$$

in the worst case (see [2, Sect. 3.1, Prop. 2] and Remark **??** below). To reduce this number, one could store the results of the row $\top$-portages. This, of course, will increase the space complexity of the algorithm, but in counterpart it could be hoped that the number of computed $\top-$images (differentiations or shifts) will decrease, in the worst case. We will refer to this number as the $\top -$ *complexity* (we will sometimes say "differential, resp. shift complexity"). The aim of this paper is, however, to show that a significant decreasing in the $\top-$complexity, in the worst case does not occur here.

Concerning the Row Reduction algorithm, the situation that develops here is as follows (recall that $n$ and $d$ represent the size and the order of the input operator):

– If the differentiated or, resp., shifted rows are stored, then, as was shown in [2], the differential complexity admits the estimate $O(n^2d^3)$. Similarly, for the shift complexity, this estimate is valid in the difference case (see below Proposition 1). Note that, in [2], it was not stated that this estimate is sharp.

  In the present paper, we show that

– The exponent 2 for $n$ and the exponent 3 for $d$ cannot be decreased in the estimate for the complexity in question: we prove the estimate $\Omega(n^2d^3)$.

[1] In the rest of the paper, in asymptotic complexity estimates, along with $O$-notation we will use $\Omega$- and $\Theta$-notation (see [15]).

– The estimate $\Omega(n^2d^3)$ and the earlier estimate $O(n^2d^3)$ imply together that for the complexity in question, the estimate $\Theta(n^2d^3)$ holds in the differential and difference cases, and, as a consequence, e.g., the estimate $O(n^2d^2)$ does not hold.

In general, the differentiations and shifts required in the transformation of systems of equations or matrices of scalar operators require time as well as arithmetic operations. Thus, in the complexity estimates associated with algorithms for the transformation of systems of equations and operator matrices, the operations of differentiation and shift should also be taken into account. Incidentally, this also applies to situations in which arithmetic complexity is not considered and instead of it the complexity as the number of bit operations in the worst case or in average is investigated, as, e.g., in [12].

We add that the EG algorithm [1], [6] allows us to obtain $L'$ having an invertible leading matrix and the $\top$ -complexity of EG is $\Theta(n^2d^2)$, that is, less, than the $\top$-complexity of the Row Reduction algorithm. Given that this estimate of complexity was obtained without a predestination for storage intermediate results. However, here $L' = UL$, but in the differential case, the operator $U$ can be not invertible. (In the difference case, $U$ is invertible, though.) While Row Reduction always provides invertibility of the operator $U$. Section 5.1 is devoted to these questions.

It should be noted that very often the costs of differentiations and shifts are considered in terms of the costs of arithmetic operations (see, for example, [11]; [13]) assuming that the field $K$ has a specific form, such as $K = k(t)$ or a finite field. In the present paper, we consider an arbitrary differential or difference field $K$ of characteristic 0 and count the number of differentiations or shifts in the worst case. A research of this kind of complexity is not very typical for computer algebra, although here one can cite, for example, the publication [14] where an upper bound on the number of differentiations of equations in a differential system sufficient for testing its compatibility is established. Factually, such a bound is an estimate for the differential complexity of algorithms for the compatibility test. For other examples see [2, 3, 8]. Such studies are important because of the mentioned reason: the operations of differentiations and shifts are not cheap, at least they are not for free. It is also possible that two algorithms for solving a certain problem have the same arithmetic complexity, but the complexity as the number of derivations or analogously, the shifts for these algorithms is different for the first and the second algorithms. This opens up the possibility of a meaningful choice of one of these algorithms. An example will be discussed in Section 5.1.

## 2 Preliminaries

**Definition 2** We consider $\top$-*complexity* of reduction algorithms as a function of $n$ and $d$. The $\top$-complexity is the number of computed $\top$-images of elements of $K$ (the $\top$-*cost*) in the worst case, when values $n$ and $d$ are fixed (a given $L$ is of size $n \times n$, its order is $d$).

When estimating $\top$-complexity, we assume that the $\top$-cost of computing $\top^m(a)$, $a \in K$, is equal to $m$.

The algorithm Row Reduction which we will call RR for short, allows to construct an operator $L'$ such that its frontal matrix is non-singular, and $L' = UL$ for some *unimodular* (i.e. invertible in $K[\top]^{n\times n}$) operator $U$. This algorithm can be described as follows (we omit the operator $U$ computation).

```
Algorithm: RR (it has been described similarly in [9, Sect 2.1] ).
Input: An operator $L \in K[\top]^{n\times n}$, let $\alpha = (\operatorname{ord} L_{1,*}, \dots, \operatorname{ord} L_{n,*})$ and $d = \max_{i=1}^{m} \alpha_i$.
Output: An operator $L' \in K[\top]^{n\times n}$ such that
            1) $L' = UL$ for some unimodular $U \in K[\top]^{n\times n}$, and
            2) the frontal matrix of $L'$ is non-singular.

$L' := L$; $L'_{\text{fr}}$:= the frontal matrix of $L'$; $\alpha' := \alpha$;
while the rows of $L'_{\text{fr}}$ are linearly dependent over $K$ do
```

```
    compute a vector $v = (v_1, \dots, v_n) \in K^{1\times n}$ in the left nullspace of $L'_{\rm fr}$;
    select an integer $\nu$ such that $\alpha'_\nu = \max_{\substack{1\leqslant i\leqslant n\\ v_i\neq 0}} \alpha'_i$;
    adjust the coefficients $v_1, \dots, v_n$ so that after the adjustment the row
        $\sum_{i=1}^n v_i \top^{\alpha'_\nu-\alpha'_i} \circ L'_{i,*}$ is of order $< \alpha'_\nu$ (see Remark 1 below);
    $L'_{\nu,*} := \sum_{i=1}^n v_i \top^{\alpha'_\nu-\alpha'_i} \circ L'_{i,*}$;
    $\alpha'_\nu :=$ ord $L'_{\nu,*}$; $L'_{\rm fr}:=$ the frontal matrix of $L'$
od
```

**Remark 1** This is a comment on the step 'adjust the coefficients $v_1, \dots, v_n$ so that ...'. In the differential case, the coefficients $v_1, \dots, v_n$ do not change (note that only the differential case is discussed in [9, Sect 2.1], and the step under consideration is not involved into the algorithm). In the difference case we should replace, accordingly with Definition 1, $v_i$ by $\sigma^{\alpha'_\nu-d}(v_i)$, $i = 1, \dots, n$. We have $\alpha'_\nu - d \leqslant 0$, and consider $|\ \alpha'_\nu - d\ |$ as the $\top$-cost of applying $\sigma^{\alpha'_\nu-d}$ to an element of $K$.

**Remark 2** It may be that there are multiple choice options for the vector $v$ on the step 'compute a vector $v = (v_1, \dots, v_n) \in K^{1\times n}$ in the left nullspace of $L'_0$'. Any such a choice option may be realized. The algorithm RR does not regulate that. (This is significant for discussing the "worst case".)

## 3 When the transformed rows are stored: Algorithm $\overline{\rm RR}$

One can store the results of applying the transformation $\top$ to the rows of an operator $L$. It was shown in [2, Sect. 3.4, Prop. 6] that in the case of storing the transformed rows, the differential complexity is $O(n^2d^3)$. The same holds for the $\top$-complexity in the difference case.

For the sake of completeness we sketch a quick general proof of this fact for the $\top$-case.

The version of the RR-algorithm that assumes the storage of the rows $\top$-portages will be denoted as $\overline{\rm RR}$.

**Proposition 1** *The number of the row $\top$-portages without repetitions (when the result of each differentiation is stored, i.e., when we collect all such results) executed by the algorithm $\overline{\rm RR}$ is $O(nd^2)$ in the worst case; as a consequence, the number of computing $\top$-images of elements of $K$ is $O(n^2d^3)$.*

**Proof:** Let a row $r$ be changed in the course of RR performance (we have in mind the elimination $L'_{\nu,*} := \sum_{i=1}^n v_i \top^{\alpha'_\nu-\alpha'_i} \circ L'_{i,*}$, with $L'_{\nu,*} = r$). Let the order of $r$ after the changing be $d_0 < d$. In this case, we can compute and store the following $d - d_0$ rows

$$\top \circ r,\ \ \top^2 \circ r,\ \ \dots,\ \ \top^{d-d_0} \circ r. \tag{3}$$

After this, when a row of the form $\top^m \circ r$, $m \leqslant d-d_0$, is needed for following eliminations, pick the needed row from the collection of the stored rows. Thus, we get a modified version of $\overline{\rm RR}$ (this is an auxiliary version that is used only in this proof, we denote it as $\overline{\overline{\rm RR}}$) whose numbers of the row $\top$-portages is not less than the analogous number for $\overline{\rm RR}$. Therefore, it is sufficient to prove for $\overline{\overline{\rm RR}}$ the estimate $O(nd^2)$ for the number of the row $\top$-portages.

Evidently, the total number of elimination steps of each of RR, $\overline{\rm RR}$, $\overline{\overline{\rm RR}}$ does not exceed $nd$. The number of the row $\top$-portages on the accompanying steps of the form (3) is $O(nd \cdot d)$, i.e., $O(nd^2)$.

For $\overline{\overline{\rm RR}}$, besides the elimination steps and the accompanying steps (3) we need also a preliminary step of constructing and storing the $\top$-portages of all the rows of $L$ which are of order less than $d$. This preliminary step will require no more than $nd$ computations of the $\top$-portages.

So, the total number of the row $\top$ -portages is $O(nd^2)$ in the worst case. As a consequence, the number of computing the $\top$-images of the elements of $K$ in the worst case is $O(n^2d^3)$. □

**Remark 3** Concerning $\overline{\mathrm{RR}}$, we suppose that if a row is changed (i.e., replaced by a linear combinations over $K$ of other rows) then the original row as well as all its $\top$-portages are removed from the storage. Note also that Remark 2 is evidently valid for $\overline{\mathrm{RR}}$.

## 4 Complexity Analysis

### 4.1 Operator Having Four Rows

Let $n, d$ be positive integers and $a_1, a_2, a_3, a_4$ row matrices from $K^{1\times n}$ which are linearly independent over $K$. Let us define the function $\varkappa : \mathbb{Z} \to \{1, 2\}$:

$$\varkappa(m) = \begin{cases} 2, & \text{if } 2 \mid m, \\ 1, & \text{otherwise,} \end{cases}$$

and the sequence $P_0, P_1, \dots$ of rows belonging to $K[\top]^{1\times n}$:

$$P_0 = 0, \ \ P_i = \sum_{s=1}^{i} \top^s \circ a_{\varkappa(s)}, \ \ i = 1, 2, \dots$$

(i.e., $P_0 = 0$, $P_1 = \top \circ a_1$, $P_2 = \top^2 \circ a_2 + \top \circ a_1$, $P_3 = \top^3 \circ a_1 + \top^2 \circ a_2 + \top \circ a_1$, and so on). Evidently $\operatorname{ord} P_i = i$ when $i > 0$. Set

$$d' = \begin{cases} \lfloor \frac{d}{2} \rfloor, & \text{if } \varkappa\left(d - \lfloor \frac{d}{2} \rfloor\right) = 2, \\ \lfloor \frac{d}{2} \rfloor - 1, & \text{otherwise,} \end{cases}$$

and $d'' = d - d'$.

We construct an operator $Q \in K[\top]^{4\times n}$, $\operatorname{ord} Q = d$, as follows. Set

$$Q_{1,*} = a_1, \ \ Q_{2,*} = a_2,$$

$$Q_{3,*} = P_{d'} + a_3, \quad Q_{4,*} = \left( \sum_{i=0}^{d'} \top^{d''} \circ (P_{d'-i} + a_3) \right) + a_4.$$

**Example 1** *For $d = 4$ we have*
$Q_{*,1} = a_1,$
$Q_{*,2} = a_2,$
$Q_{*,3} = \top^2 \circ a_2 + \top \circ a_1 + a_3,$
$Q_{*,4} = \top^2 \circ (\top^2 \circ a_2 + \top \circ a_1 + a_3) + \top^2 \circ (\top \circ a_1 + a_3) + \top^2 \circ a_3 + a_4.$

**Proposition 2** *In the worst case, the amount of $\top$-images of elements of $K$ computed during the process of applying algorithm $\overline{\mathrm{RR}}$ to $Q$ is $\Omega(nd^3)$.*

**Proof:** According to Remarks 2, 3, the applying of the algorithm $\overline{\mathrm{RR}}$ to $Q$, can be equivalent to the following transformation (the symbol • marks a checkpoint; we will also consider (a), (b) not only as names of sub-steps of the loop `for ... do ... od` but also as checkpoints of the algorithm):

```
for $k = d'$ downto 0 do
    (a) $Q_{4,*} := Q_{4,*} - \top^{d''} \circ Q_{3,*}$ ;
    (b) $Q_{3,*} := Q_{3,*} - \top^k \circ Q_{\varkappa(k),*}$
    •
od;
```

indeed, when passing through the point (a), the orders of the rows $Q_{4,*}$ and $\top^{d''}Q_{3,*}$ are equal, and the order of the difference of those rows is less by 1 than the order of each of them; similarly for (b): the orders of the rows $Q_{3,*}$ and $\top^{k}Q_{\varkappa(k),*}$ are equal and the order of their difference is less by 1 than the order of each of those rows.

When passing through the point • on each of steps of the loop, we have

$$Q_{3,*} = P_k + a_3, \;\; Q_{4,*} = \left( \sum_{i=d''}^{d''+k-1} \top^{d''} \circ (P_i + a_3) \right) + a_4,$$

$$\operatorname{ord} Q_{3,*} = k, \;\; \operatorname{ord} Q_{4,*} = d'' + k - 1. \tag{4}$$

At the last passage through this point (with $k = 0$) we have

$$Q_{3,*} = P_0 + a_3 = a_3, \;\; Q_{4,*} = \left( \sum_{i=d''}^{d''-1} \top^{d''} \circ (P_i + a_3) \right) + a_4 = a_4.$$

By Remark 3, if some compositions $\top^{d''} \circ Q_{3,*}$ were stored then after the sub-step (b), they would be removed from the storage, and the next performance of the sub-step (a) requires the computation $\top^{d''} \circ Q_{3,*}$. (On the other hand, all the compositions $\top^{k} \circ Q_{1,*}$ and $\top^{k} \circ Q_{2,*}$ which are used in (b) can be stored till the end of the computation.)

According to (4), the amount of $\top$-images of elements of $K$ computed on the step (a) of the loop is not less than

$$\sum_{k=0}^{d'} k d'' n = \left( \frac{d'^2 d''}{2} + O(d'd'') \right) n$$

By $d' \sim d'' \sim \frac{d}{2}$ we have

$$\left( \frac{d'^2 d''}{2} + O(d'd'') \right) n \sim \frac{d^3}{16} n\,. \tag{5}$$

The claim follows. □

## 4.2 Operator Having $n$ Rows

Let $A \in K^{n \times n}$, $\det A \neq 0$. We can pick $\lfloor n/4 \rfloor$ quadruples (sets of four) from the set of rows of $A$. For each of them we can construct four rows of an operator $L$ as it was explained in Section 4.1. The remaining $n - \lfloor n/4 \rfloor \cdot 4$ rows of $L$ coincide with the corresponding rows of $A$. By Remarks 2, 3, the algorithm $\overline{\mathrm{RR}}$ can operate with the constructed sets of four rows independently. For low estimating the total $\top$-cost $C_{n,d}$ of those transformations, we multiply (5) by $\frac{n}{4}$ and get

$$C_{n,d} \sim \frac{1}{64} n^2 d^3. \tag{6}$$

**Theorem 1** *The $\top$-complexity of $\overline{\mathrm{RR}}$ is $\Theta(n^2 d^3)$.*

**Proof:** Relation (6) indicates that the $\top$-complexity of $\overline{\mathrm{RR}}$ is $\Omega(n^2 d^3)$. We know that this complexity is $O(n^2 d^3)$, as it was mentioned in Section 1. The claim follows. □

## 5 Specifics of the difference case

### 5.1 Another approach for desingularization of the leading matrix

A visible dissimilarity between the difference and differential cases is that the automorphism $\sigma$ has an inverse in $K[\sigma, \sigma^{-1}]$, while in the differential case the derivative $\partial$ is not invertible in $K[\partial]$. The RR-algorithm (Section 2) computes $L'$ of the form $UL$ where $L'$ has a non-singular frontal matrix, and $U$ is invertible in $K[\top]^{n\times n}$ (the version of RR under consideration does not compute $U$, but the existence of such $U$ is guaranteed) and both $U, U^{-1}$ do not contain $\top^{-1}$. In the difference case, it is acceptable though that $U^{-1}$ contains $\sigma^{-1}$. The computation of such $U$ can be cheaper, i.e. can have a smaller $\top$-complexity. It can be achieved by using EG-algorithm [1, 5, 6]. The version which is presented in [1] computes in the difference case an equivalent operator (i.e. an operator $L'$ which can be represented as $\tilde{U}L$, where $\tilde{U}$ is invertible in $K[\sigma, \sigma^{-1}]^{n\times n}$) having a triangular leading matrix: an analogous idea was independently used in [[16]] for constructing the so called weak Popov form of a polynomial matrix. The $\top$-complexity of the EG-algorithm is $O(n^2d^2)$ (see, e.g. [3, 8]).

Note that it does not hold in the differential case: if we use EG-algorithm, then $\tilde{U}$ is not, in general, invertible in $K[\partial]^{n\times n}$, but the leading matrix of the output operator $L'$ is invertible in both differential and difference cases.

### 5.2 On arithmetic complexity

The arithmetic complexity of RR is $\Theta(n^{\omega+1}d^2)$, where $\omega$ is the matrix multiplication exponent, $\omega > 2$. As for the version of EG which is mentioned in Section 5.1, its arithmetic complexity is $O(n^3d^2)$. However it is possible to use for RR the same complexity decreasing approach as for EG (see, for example, [3]), getting an operator with a triangular frontal matrix. The arithmetic complexity is then $O(n^3d^2)$, as for EG.

## Acknowledgments

The authors would like to thank an anonymous referee for his valuable remarks to improve the paper. The first author is supported in part by the Russian Foundation for Basic Research, project no. 19-01-00032. The first author thanks also Department of Mathematics and Informatics of XLIM Institute of Limoges University for the hospitality during his visits.

## References

[1] S.A. Abramov. EG–eliminations. *J. of Difference Equations and Applications*, 5(4-5): 393–433, 1999.

[2] S.A. Abramov. On the Differential and Full Algebraic Complexities of Operator Matrices Transformations. *CASC 2016, LNCS 9890*, Springer, Heidelberg, 1–14, 2016.

[3] S.A. Abramov. Inverse Linear Difference Operators. *Computational Mathematics and Mathematical Physics*, 57, 1887–1898, 2017.

[4] S.A. Abramov, M.A. Barkatou. On the dimension of solution spaces of full-rank linear differential systems. *CASC 2013, LNCS 8136*, Springer, Heidelberg, 1–9, 2013.

[5] S.A. Abramov, M. Bronstein. On solutions of linear functional systems. *Proc. ISSAC 2001*: 1–6, 2001.

[6] S.A. Abramov, M. Bronstein. Linear algebra for skew-polynomial matrices. Rapport de Recherche INRIA RR-4420, March 2002 `http://www.inria.fr/RRRT/RR-4420.html`

[7] S.A. Abramov, D.E. Khmelnov. Linear differential and difference systems: EG$_\delta$- and EG$_\sigma$-eliminations. *Programming and Computer Software*, 39(2): 91–109, 2013.

[8] S.A. Abramov, D.E. Khmelnov. On unimodular matrices of difference operators. *Proc. CASC 2018, LNCS 11077*, Springer, Heidelberg, 18–31, 2018.

[9] M.A. Barkatou, C. El Bacha, G. Labahn, E. Pflügel. On simultaneously row and column reduction of higher-order linear differential systems. *J. of Symbolic Comput.*, 49(1): 45–64, 2013.

[10] B. Beckermann, H. Cheng, G. Labahn. Fraction-free row reduction of matrices of Ore polymomials. *J. of Symbolic Comput.*, 41(5): 513–543, 2006.

[11] M. Giesbrecht. Factoring in skew polynomial rings over finite fields. *J. of Symbolic Comput.*, 26(4): 463–486, 1998.

[12] M. Giesbrecht, M. Sub Kim. Computation of the Hermite form of a Matrix of Ore Polynomials. *J. of Algebra*, 376: 341–362, 2013.

[13] M. Giesbrecht, Y. Zhang, Factoring and Decomposing Ore Polynomials over $F_q(t)$. *Proc. ISSAC 2003*: 127–134, 2003.

[14] R. Gustavson, M. Kondratieva, A. Ovchinnikov. New effective differential Nullstellensatz. *Advances in Mathematics*, 290: 1138–1158, 2016.

[15] D.E. Knuth. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2): 18–23, 1976.

[16] T. Mulders, A. Storjohann: On Lattice Reduction for Polynomial Matrices. *J. Symb. Comput.*, 37(4): 485–510, 2004.