On the Differential and Full Algebraic Complexities of Operator Matrices Transformations

S. A. Abramov*

Dorodnitsyn Computing Centre,
Federal Research Center Computer Science and Control
of Russian Academy of Sciences,
Vavilova, 40, Moscow 119333, Russia
sergeyabramov@mail.ru

Abstract. We consider $n \times n$-matrices whose entries are scalar ordinary differential operators of order $\leqslant d$ over a constructive differential field K. We show that to choose an algorithm to solve a problem related to such matrices it is reasonable to take into account the complexity measured as the number not only of arithmetic operations in K in the worst case but of all operations including differentiation. The algorithms that have the same complexity in terms of the number of arithmetic operations can though differ in the context of the full algebraic complexity that includes the necessary differentiations. Following this, we give a complexity analysis, first, of finding a superset of the set of singular points for solutions of a system of linear ordinary differential equations, and, second, of the unimodularity testing for an operator matrix and of constructing the inverse matrix if it exists.

1 Introduction

In this paper, we discuss some algorithms which use operations on elements of a differential field. A complexity analysis of such algorithms is based sometimes on considering the complexity as the number of arithmetic operations in K in the worst case (the *arithmetic complexity*). This approach is not always productive. First, the differentiations are not for free, and there is no reason to believe that, e.g., a differentiation is much cheaper than an addition or a multiplication when the differential field is $\mathbb{Q}(x)$ with the standard differentiation by x. Second, we may face a situation where two algorithms have the same arithmetic complexity. However, it may be that the complexities of those two algorithms are different, if we compute the number of differentiations in the worst case (the *differential complexity*). Therefore, consideration of not only arithmetic but also differential complexity seems reasonable. This is similar to the situation with sorting algorithms, when we consider separately the complexity as the number of comparisons and, resp., the number of swaps. (For example, in [19], an upper bound on the number of differentiations of equations in a differential system sufficient for testing its compatibility is established. Actually, such a bound is an estimate for the complexity of algorithms for the compatibility testing.)

We will also consider the *full* complexity as the total number of all operations in the basic differential field in the worst case, when differentiations are included. This complexity will be considered in the context of *algebraic complexity theory*: the complexity is measured as the number of operation in K in the worst case without

* Supported in part by the Russian Foundation for Basic Research, project no. 16-01-00174.

taking into account the possible growth of "sizes" of the elements computed by algorithms (similarly, say, to the complexity $\Theta(n^{\log_2 7})$ of Strassen's algorithm [27] for multiplying square $n \times n$-matrices).

Below, we discuss two well known algorithms for transforming operator matrices, i.e., square matrices whose entries are scalar differential operators with coefficients in the basic differential field. Earlier, only arithmetic complexity of those algorithms was investigated, and it was established that asymptotically, their arithmetic complexities agree. However, we show that their differential complexities are different. Note that the functionalities of the algorithms are also slightly different.

In Section 4, we discuss two computational problems, for which the above-mentioned algorithms for transforming operator matrices are useful. The first problem is related to finding singular points of solutions of the corresponding system of linear ordinary differential equations. The second one is the problem of testing unimodularity (i.e., invertibility) of an operator matrix and of constructing the inverse matrix if it exists. Invertibility testing is a classical mathematical problem, whose specifics depend on a field or a ring containing the matrix entries. The question of unimodularity of such operator matrices arises, in particular, in connection with the existence problems for solutions of differential systems ([23]). The "binary" testing (with the output *yes* or *no*, without constructing the inverse when it exists) can also be considered as a problem which is of independent interest. A careful complexity analysis allows one to make an informed choice of a transformation algorithm as an adequate auxiliary tool for solving each of these problems.

2 Preliminaries

2.1 Operator Matrices

Let K be a differential field of characteristic 0 with a derivation $\partial = '$. For a non-negative integer n, the ring of $n \times n$-matrices with entries belonging to a ring R is denoted by $\mathrm{Mat}_n(R)$. The ring of scalar differential operators with coefficients in K is denoted by $K[\partial]$; the order of an operator $l \in K[\partial]$ which is denoted by $\operatorname{ord} l$ is equal to the degree of the corresponding non-commutative polynomial from $K[\partial]$. Any non-zero operator matrix $L \in \mathrm{Mat}_n(K[\partial])$ can be represented as a differential operator with matrix coefficients in $\mathrm{Mat}_n(K)$:

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

where $A_0, A_1, \ldots, A_d \in \mathrm{Mat}_n(K)$, and the matrix A_d (the *leading matrix* of L) is non-zero. The number d is the *order* of L; we write $d = \operatorname{ord} L$. The order of a row of L is the biggest order of operators from $K[\partial]$ belonging to the row. Thus, the order of an operator matrix coincides with the biggest order of all rows of the operator matrix. A matrix $L \in \mathrm{Mat}_n(K[\partial])$ is of *full rank* if its rows are linearly independent over $K[\partial]$.

An operator matrix L is invertible in $\mathrm{Mat}_n(K[\partial])$ and M is its inverse, if $LM = ML = I_n$ where I_n is the identity $n \times n$-matrix. We write L^{-1} for M. Invertible operator matrices are also called *unimodular* matrices.

In [23], the following example of a unimodular matrix and the inverse is given ($K = \mathbb{Q}(x)$, $\partial = \frac{d}{dx}$):

$$\begin{pmatrix} x^2/2 & -(x/2)\partial + 1 \\ -x\partial - 3 & \partial^2 \end{pmatrix}^{-1} = \begin{pmatrix} \partial^2 & (x/2)\partial \\ x\partial + 1 & x^2/2 \end{pmatrix}. \tag{2}$$

2.2 The Dimension of the Solutions Space

Let the constant field $Const(K) = \{c \in K \mid \partial c = 0\}$ of K be algebraically closed. We denote by Λ a fixed *universal Picard–Vessiot differential extension field* of K (see [25, Sect. 3.2]). This is a differential extension Λ of K with $Const(\Lambda) = Const(K)$ such that any differential system $\partial y = Ay$ with $A \in \mathrm{Mat}_n(K[\partial])$ has a solution space of dimension n over the constants. For arbitrary operator matrix L of the form (1), we denote by V_L the linear space over $Const(\Lambda)$ of solutions of L (i.e., solutions of the equation $L(y) = 0$) belonging to Λ^n. Its dimension will be denoted by $\dim V_L$.

Suppose that $\mathrm{Const}\,(K)$ is not algebraically closed. It is not difficult to see that for any differential field K of characteristic 0 there exists a differential extension whose constant field is algebraically closed. Indeed, this is the algebraic closure $\bar{K}$ with the derivation obtained by extending the derivation of K in the natural way. In this case, $\mathrm{Const}\,(\bar{K}) = \overline{\mathrm{Const}\,(K)}$ (see [25, Exercises 1.5, 2:(c),(d)], [26, Sect. 3]). In this case, V_L is the linear space over $Const(\bar{K})$ of solutions of L whose components belong to the the universal differential extension of $\bar{K}$.

We use the notation $M_{i,*}$, $1 \leqslant i \leqslant n$, for the $1 \times n$-matrix which is the ith row of an $n \times n$-matrix M. Let a full rank operator matrix L be of the form (1). If $1 \leqslant i \leqslant n$ then define $\alpha_i(L)$ as the maximal integer k, $1 \leqslant k \leqslant d$, such that $(A_k)_{i,*}$ is a nonzero row. So, $\alpha_i(L) = \mathrm{ord}\, L_{i,*}$.

The matrix $F \in \mathrm{Mat}_n(K)$ such that $F_{i,*} = (A_{\alpha_i(L)})_{i,*}$, $i = 1, \ldots, n$, is the *frontal matrix* of L.

The group of unimodular matrices from $\mathrm{Mat}_n(K[\partial])$ will be denoted by Υ_n.

We formulate a theorem which is a consequence of statements proven in [2, 3] (the equivalence (iii) can also be proven using [23, Thm III]).

Theorem 1 *Let $L \in Mat_n(K[\partial])$ be of full rank. In this case*

(i) If L' is the result of differentiating of a row of L then $\dim V_{L'} = \dim V_L + 1$.

(ii) If the frontal matrix of $L \in Mat_n(K[\partial])$ is invertible then

$$\dim V_L = \sum_{i=1}^{n} \alpha_i(L).$$

(iii) $L \in \Upsilon_n \Longleftrightarrow V_L = 0$.

We suppose in the sequel that the field K is constructive, in particular that there exists a procedure for recognizing whether a given element of K is equal to 0.

2.3 Algorithm EG (EG-Eliminations)

Given $L \in \mathrm{Mat}_n(K[\partial])$ of full rank, algorithm EG ([4–6, 2]) constructs an *embracing* operator $\widehat{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\mathrm{ord}\, \widehat{L} = \mathrm{ord}\, L$,
- the leading matrix of $\widehat{L}$ is invertible,
- $\widehat{L} = QL$ for some $Q \in \mathrm{Mat}_n(K[\partial])$, thus $V_L \subseteq V_{\widehat{L}}$.

If L is not of full rank then this algorithm reports this.

This algorithm is based on alternation of reductions and differentiations. First, explain how the reduction works. It is checked whether the rows of the leading matrix are linearly dependent over K. If they are, coefficients of the dependence $p_1, \ldots, p_n \in K$ are found. From the rows of L corresponding to nonzero coefficients, we select one. Let it be the ith row. This row is replaced by the linear combination of the rows of L with the coefficients $p_1, \ldots, p_n$. As a result, the ith row of the leading matrix vanishes. This operation is called *reduction*.

Let the ith row of the leading matrix be zero. Then we differentiate the ith row of the operator matrix, i.e., replace each entry $L_{ij} \in K[\partial]$ by the composition of ∂ and L_{ij}, $j = 1, \ldots, n$ (this operation is called *row differentiation*).

A version of algorithm EG is as follows.

If the rows of the leading matrix of L are linearly dependent over K then the reduction is performed. Suppose that this makes the ith row of the leading matrix zero. Then, we perform the differentiation of the ith row of the operator matrix and continue the process of alternated reductions and differentiations until the leading matrix becomes nonsingular, and, therefore, we get the output matrix $\widehat{L}$.

If at some moment a zero row appears in the operator matrix or the number of row differentiations becomes bigger than nd then L is not of full rank.

2.4 Algorithm RR

This algorithm is based originally on the algorithm FF [11]. A simplified version of FF is RowReduction [9]; for short, we will use the abbreviation RR in the sequel.

For a given $L \in \mathrm{Mat}_n(K[\partial])$ of full rank, algorithm RR constructs $\breve{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\mathrm{ord}\,\breve{L} \leqslant \mathrm{ord}\,L$,
- the frontal matrix of $\breve{L}$ is invertible,
- $\breve{L} = UL$ for some $U \in \Upsilon_n$, thus $V_L = V_{\breve{L}}$.

A comment on the matrix U will be given later in this section. A version of algorithm RR is as follows.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence be $p_1, \ldots, p_n \in K$. From the rows of L corresponding to nonzero coefficients, we select one having the highest order (if there are few such rows then take any of them). Let it be the ith row. Then we replace $L_{i,*}$ of the operator matrix by

$$\sum_{j=1}^{n} p_j \partial^{\alpha_i(L)-\alpha_j(L)} L_{j,*} \tag{3}$$

and continue this process until the frontal matrix becomes nonsingular, and, therefore, we get the output matrix $\breve{L}$.

If at some moment, a zero row appears in the operator matrix then L is not of full rank.

Remark 1 *In addition, algorithm RR allows (if it is required) to construct such operator matrices $U_1, \ldots, U_l \in Mat_m(K[\partial])$, that $U_1, \ldots, U_l$ are unimodular and*

$$\breve{L} = U_l \ldots U_1 L. \tag{4}$$

Each U_j is of the form

$$\begin{array}{c} \\ \\ \\ i: \\ \\ \\ \\ \end{array} \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ p_1\partial^{\alpha_i-\alpha_1} & \ldots & p_{i-1}\partial^{\alpha_i-\alpha_{i-1}} & p_i & p_{i+1}\partial^{\alpha_i-\alpha_{i+1}} & \ldots & p_n\partial^{\alpha_i-\alpha_n} \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \end{pmatrix} \tag{5}$$

with $1 \leqslant i \leqslant n$, $p_1, \ldots, p_n \in K$, $p_i \neq 0$ *(this matrix corresponds to the replacement* $L_{i,*}$ *by (3)).*

A matrix of the form (5) will be called *elementary*. Each an elementary matrix is unimodular: to obtain the inverse, one can replace in (5) its ith row by

$$\left(-\tfrac{p_1}{p_i}\partial^{\alpha_i-\alpha_1} \ \ldots\ -\tfrac{p_{i-1}}{p_i}\partial^{\alpha_i-\alpha_{i-1}} \ \ \tfrac{1}{p_i} \ \ -\tfrac{p_{i+1}}{p_i}\partial^{\alpha_i-\alpha_{i+1}} \ \ldots\ -\tfrac{p_n}{p_i}\partial^{\alpha_i-\alpha_n}\right).$$

The list

$$U_1, \ldots, U_l \tag{6}$$

of the elementary matrices involved into (4) can be constructed in the course of executing RR with no extra cost.

3 Differential and Full Complexities of Algorithms for Operator Matrices Transformation

3.1 Diversity of Algebraic Complexities

Besides the complexity as the number of arithmetic operations (the *arithmetic complexity*) one can consider the number of differentiations in the worst case (the *differential complexity*).

We will also discuss the *full complexity* as the total number of all operations (the differentiations are included) in the field K in the worst case. In [9, 3], when the complexity of algorithms EG and RR was considered, the differentiations were ignored (the same concerning algorithm which we will denote by ExtRR and will discuss in Section 3.3). We will denote such a kind of complexity by $F_{\mathrm{XX}}(n, d)$, where xx is the name of an algorithm under consideration, for example, xx $\in$ {RR, EG}. It is worthy to note that if we ignore the differentiations then the arithmetic complexity can be in wrong values, since differentiating a scalar differential operator requires to execute also arithmetic operations (additions). Therefore, $F_{\mathrm{XX}}(n, d)$ is only a visible (apparent) complexity (F = "at First sight"). We will consider also the following functions of n, d:

- $B_{\mathrm{XX}}(n, d)$ — the number of row differentiations in the worst case,
- $\tilde{T}_{\mathrm{XX}}(n, d)$ — the number of differentiations of elements of K in the worst case (the differential complexity),
- $T_{\mathrm{XX}}(n, d)$ — the number of all operations in K in the worst case (the full complexity).

Along with O-notation we use the Θ-notation which is very common in complexity theory ([21]). Recall that $f(n, d) = \Theta(g(n, d))$ is equivalent to

$$f(n, d) = O(g(n, d)) \ \&\ g(n, d) = O(f(n, d)).$$

If $f(n, d) = \Theta(g(n, d))$ then we call $\Theta(g(n, d))$ a *sharp* bound for $f(n, d)$.

If xx $\in$ {RR, EG} then

$$\tilde{T}_{\mathrm{XX}}(n, d) = \Theta(B_{\mathrm{XX}}(n, d)nd) \tag{7}$$

for the the differential complexity $\tilde{T}$. We have also

$$T_{\mathrm{XX}}(n, d) = \Theta(F_{\mathrm{XX}}(n, d) + B_{\mathrm{XX}}(n, d)nd) \tag{8}$$

for the full complexity T, since the case of a big number of differentiations is concurrently the case when the number of arithmetic operations is big and vice versa.

Asymptotic relation similar to (8) holds for the arithmetic complexity since each differentiation of a row of L uses in the worst case besides nd of differentiations of elements of K also the same number of arithmetic operations (additions) in K.

Searching for coefficients $p_1, \ldots, p_n$ of a linear dependence for rows of a matrix from $\mathrm{Mat}_n(K)$ is equivalent to solving a homogeneous system of linear algebraic equations with coefficients in K. The complexity of solving such a system is $\Theta(n^\omega)$, where ω is the matrix multiplication exponent, $2 < \omega \leqslant 3$. We have $F_{\mathrm{EG}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2)$, $B_{\mathrm{EG}}(n,d) = \Theta(nd)$.

Proposition 1 *The differential and the full complexity of EG allow the asymptotic estimates*

$$\tilde{T}_{EG}(n,d) = \Theta(n^2d^2), \quad T_{EG}(n,d) = \Theta(n^{\omega+1}d + n^3d^2). \tag{9}$$

Proof. We have mentioned that by Theorem 1(i, ii) each differentiation of a row increases the dimension of the solutions space of an operator matrix by 1, and in our case that dimension does not exceed nd. This implies the first estimate from (9).

Each reduction step uses in the worst case $\Theta(n^\omega + n^2d)$ arithmetic operations. The number of such steps is nd in the worst case. Together with the first estimate from (9) this gives the second estimate. (The differentiation operations do not affect significantly the full complexity of EG.)

Proposition 2 *The differential and the full complexity of RR admit the asymptotic estimates*

$$\tilde{T}_{RR}(n,d) = \Theta(n^3d^3), \quad T_{RR}(n,d) = \Theta(n^{\omega+1}d + n^3d^3). \tag{10}$$

Proof. We have $F_{\mathrm{RR}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2)$ and $B_{\mathrm{RR}}(n,d) = \Theta(n^2d^2)$. This implies the claim.

We see that $B_{\mathrm{RR}}(n,d)$ grows faster than $B_{\mathrm{EG}}(n,d)$. In the case of RR, the differentiating operations increase the full complexity.

3.2 Algorithms △EG and △RR

Let the ith row r of the frontal matrix of $L \in \mathrm{Mat}_n(K[\partial])$ have the form

$$(\underbrace{0, \ldots, 0}_{k-1}, a, \ldots, b),$$

$1 \leqslant k \leqslant n$, $a \neq 0$. Then k is the *pin index* of the ith row of L.

If all rows of L have distinct pin indices then the frontal matrix of L is nonsingular. Suppose that two rows r_1, r_2 of L have the same pin index k. Set $d_1 = \operatorname{ord} r_1$, $d_2 = \operatorname{ord} r_2$. Let $d_1 \leqslant d_2$. There exists a v in K such that the difference

$$r_2 - v\partial^{\,d_2-d_1} r_1 \tag{11}$$

either has the pin index which is bigger than k or has the order which is less than d_2. This can be used[1] instead of a search for a linear dependency of the rows of the frontal matrix of L (in the case of EG, on key moments, the leading and the frontal matrices coincide, and $d_2 - d_1 = 0$ in (11)). If L is of full rank then the frontal matrix after the transformation is in triangular form.

This leads to modified versions of EG and RR. We will denote them as △EG and, resp., △RR.

[1] For the difference case, this was used in first versions [1] of algorithm EG. In a discussion related to the differential case, A. Storjohann drew the author's attention to the fact that the complexity of this approach is less than of one which uses solving of linear algebraic systems (see also [24]).

Proposition 3 *The differential and the full complexities of $\triangle EG$ and $\triangle RR$ admit the asymptotic estimates*

$$\tilde{T}_{\triangle EG}(n,d) = \Theta(n^2d^2),\ \ T_{\triangle EG}(n,d) = \Theta(n^3d^2) \tag{12}$$

and

$$\tilde{T}_{\triangle RR}(n,d) = \Theta(n^3d^3),\ \ T_{\triangle RR}(n,d) = \Theta(n^3d^3). \tag{13}$$

Proof. The replacement of r_2 by (11) is a unimodular operation on L. This operation has the complexity $\Theta(nd)$. A row can have its pin index increased at most n times before the order of the row is decreased. Thus, $F_{\mathrm{XX}}(n,d) = \Theta(nd\cdot n\cdot nd) = \Theta(n^3d^2)$ for $\mathrm{xx} \in \{\triangle\mathrm{EG}, \triangle\mathrm{RR}\}$. Concerning the differential complexity, for $\triangle$EG and, resp. $\triangle$RR it is the same as for EG and RR.

We emphasize that the difference between two estimates $T_{\triangle\mathrm{RR}} = \Theta(n^3d^3)$ and $T_{\triangle\mathrm{EG}}(n,d) = \Theta(n^3d^2)$ is due to the differential component: if we ignored the operation of differentiation, then we would have the estimate $\Theta(n^3d^2)$ for both complexities.

3.3 Extended RR: Computing U Along with $\breve{L}$

To compute along with $\breve{L}$ the multiplier U such that $\breve{L} = UL$, one can apply the following algorithm (we call it ExtRR) to L:

Apply RR to L (this gives $\breve{L}$), and repeat in parallel all the operations for the matrix which is originally equal to the identity matrix I_n (this gives U).

The algorithm was presented in [9, Sect.4]. Evidently, $B_{\mathrm{ExtRR}}(n,d) = 2B_{\mathrm{RR}}(n,d)$. The following proposition is useful for estimating $F_{\mathrm{ExtRR}}(n,d)$, $\tilde{T}_{\mathrm{ExtRR}}(n,d)$ and $T_{\mathrm{ExtRR}}(n,d)$:

Proposition 4 *Let algorithm RR compute step-by-step the unimodular matrices of the form (5) for $L \in Mat_n(K[\partial])$, $\operatorname{ord} L = d$ (see Remark 1). Then $\operatorname{ord}(U_k \dots U_1) = O(nd)$ for all $k = 1, \dots, l$.*

Proof. It follows from [9, Prop. 1], [18, Thm 4.9].

Proposition 4, estimates (10) and the equality $nd\cdot(n^2\cdot nd) = n^4d^2$ imply the estimates $F_{\mathrm{ExtRR}}(n,d) = O(n^{\omega+1}d + n^4d^2) = O(n^4d^2)$. Thus

$$\tilde{T}_{\mathrm{ExtRR}}(n,d) = O(n^4d^3),\ \ T_{\mathrm{ExtRR}}(n,d) = O(n^4d^2 + n^4d^3) = O(n^4d^3). \tag{14}$$

These estimates can be sharpened.

Proposition 5 *Let L be unimodular. Then*

$$\operatorname{ord} L^{-1} \leqslant (n-1)d \tag{15}$$

and (15) is the tight bound: for all integer n, d such that $d \geqslant 0$, $n \geqslant 2$ there exists $L \in Mat_n(K[\partial])$ such that $\operatorname{ord} L^{-1} = (n-1)d$.

Proof. The bound (15) follows from some estimates related to computing the Hermite form given in [18, Thm 4.9]. (The Hermite form of a unimodular matrix is the

identity, the transformation matrix U is the inverse.) Let us prove that this bound is tight. Indeed, the $n \times n$ operator matrix of order d

$$\begin{pmatrix} 1 & \partial^d & 0 & 0 & \dots & \dots & 0 & 0 \\ 0 & 1 & \partial^d & 0 & \dots & \dots & 0 & 0 \\ & & \ddots & & & & & \\ & & & \ddots & & & & \\ & & & & \ddots & & & \\ & & & & & & & \\ 0 & 0 & 0 & 0 & \dots & \dots & 1 & \partial^d \\ 0 & 0 & 0 & 0 & \dots & \dots & 0 & 1 \end{pmatrix} \tag{16}$$

has the inverse of order $(n-1)d$:

$$\begin{pmatrix} 1 & -\partial^d & \partial^{2d} & -\partial^{3d} & \dots & (-1)^{n-2}\partial^{(n-2)d} & (-1)^{n-1}\partial^{(n-1)d} \\ 0 & 1 & -\partial^d & \partial^{2d} & \dots & (-1)^{n-3}\partial^{(n-3)d} & (-1)^{n-2}\partial^{(n-2)d} \\ & & \ddots & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & & \\ 0 & 0 & 0 & 0 & \dots & 1 & -\partial^d \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}. \tag{17}$$

If algorithm ExtRR is applied to matrix (16) then this yields two matrices U and I_n where U is equal to (17). Note in addition to (14) that applying RR to matrix (16) yields $U_1, \dots, U_{n-1}$ (see (4), where $l = n-1$ in this case) such that $\operatorname{ord}(U_k \dots U_1) = kd$, $k = 1, \dots, n-1$. Since $\sum_{k=1}^{n-1} k^3 d^3 = \Theta(n^4 d^3)$, we obtain

$$\tilde{T}_{\mathrm{ExtRR}}(n,d) = \Theta(n^4 d^3), \;\; T_{\mathrm{ExtRR}}(n,d) = \Theta(n^4 d^2 + n^3 d^3). \tag{18}$$

Remark 2 *As a consequence of (15) we obtain that if $L \in Mat_2(K[\partial])$ is unimodular then* $\operatorname{ord} L^{-1} = \operatorname{ord} L$. *(Miyake's example (2) illustrates this.)*

Note finally that to show the correctness of using $\triangle$RR instead of *RR* for ExtRR we have to prove an analog of the statement of Proposition 4. It is not clear whether such a statement holds. In addition, the replacement of RR by $\triangle$RR does not improve estimate (18) due to the term $n^4 d^3$, which replaces $n^3 d^3$ in $T_{\triangle\mathrm{RR}}(n,d)$ since the number of elements in a row is now $(n-1)nd$.

3.4 When Differentiated Rows are Stored

One can store all the results of row differentiations. In this case, some upper estimates for the total number of differentiations can easily be obtained.

Proposition 6 *The number of row differentiations without repetitions (when the result of each differentiation is stored, i.e., when we collect all such results) executed by algorithms RR and $\triangle$RR is $O(nd^2)$ and, resp., $O(n^2 d^2)$ in the worst case; as a consequence, the number of differentiations of elements of K is $O(n^2 d^3)$ and, resp., $O(n^3 d^3)$.*

Proof. Let a row r be changed in the course of RR or $\triangle$RR performance. Let the order of r after the changing be $d_0 < d$. In this case, one can compute and store $d - d_0$ rows

$$\partial r, \;\; \partial(\partial r), \;\; \dots, \;\; \underbrace{\partial(\partial \dots (\partial r) \dots)}_{d-d_0 \text{ differentiations}}.$$

After this, when a row of the form $\partial^m r$, $m \leqslant d - d_0$ is needed for following eliminations, pick the needed row from the collection of the stored rows. Thus, we get modified versions of RR and $\triangle$RR whose numbers of row differentiations are not less than the analogous numbers for the original versions. It is easy to see that for the modified version of RR, this number is not bigger than nd^2, and not bigger than n^2d^2 for $\triangle$RR. The claim follows.

However, the estimates $O(n^2d^3)$ and $O(n^3d^3)$ for the number of differentiations do not allow to decrease the exponent of d in (10), (13). Based on Proposition 6, we cannot draw the conclusion that the storage of the results of all the differentiations decreases significantly the full complexity of RR and $\triangle$RR. Similarly, using the upper bound $O(n^3d^3)$ we cannot decrease the exponent of d in (18). But the space complexity will go up when we store all the results of differentiating.

Remark 3 *It is not clear whether, say, the upper bound $O(nd^2)$ for the number of row differentiations by RR is sharp. If for this number, the estimate $O(nd)$ holds then we would have the estimate $O(n^4d^2)$ for the full complexity of the version of ExtRR such that if a row r is differentiated m times then the rows $\partial r, \ldots, \partial^m r$ are stored for potential later uses.*

4 Two Computational Problems

4.1 Singularities of Systems

If, for example, $K = \mathbb{Q}(x)$, $\partial = \frac{d}{dx}$ and we are interested in singular points of solutions of a system $L(y) = 0$, $L \in \mathrm{Mat}_n(K)$, $\mathrm{ord}\, L = d$, then by each of algorithms EG, RR, $\triangle$EG, *DRR* we can find a polynomial whose roots form a finite superset of the set of such points [6–8]. The basic idea is that if the leading matrix of L is invertible in $\mathrm{Mat}_n(K)$ then we can take the (square-free factorized) determinant of the leading matrix.

Similarly, we can use the frontal matrix, if it is invertible. The fact is that if $\alpha_1, \ldots, \alpha_n$ are the row orders of $\breve{L}$, $d = \mathrm{ord}\, \breve{L} = \max\{\alpha_1, \ldots, \alpha_n\}$ and $D = \mathrm{diag}(\partial^{d-\alpha_1}, \ldots, \partial^{d-\alpha_n})$ then the leading matrix of $D\breve{L}$ coincides with the frontal matrix of $\breve{L}$ (we do not need to compute $D\breve{L}$).

Therefore, for example, algorithms $\triangle$EG, $\triangle$RR can be used to compute the desirable polynomial. The distinction between complexities (12) and (13) shows that at least when n and d are large enough algorithm $\triangle$EG is probably better. The full complexity (in the meaning of this paper) is $\Theta(n^3d^2)$.

4.2 The Unimodularity Testing

Applying algorithm ExtRR to L we transform L into $\breve{L}$. Theorem 1(ii, iii) implies that L is unimodular if and only if $\breve{L}$ is invertible in $\mathrm{Mat}_m(K)$. In this case, $(\breve{L})^{-1}UL = I_n$, where U is unimodular. Therefore, $(\breve{L})^{-1}U$ is the inverse for L.

The matrices U and $\breve{L}$ can be constructed by ExtRR. By (14) the full complexity of the computation of the inverse is

$$\Theta(n^4d^3). \tag{19}$$

The multiplication of $\breve{L}^{-1}$ and U does not change this estimate (recall that $\mathrm{ord}\, \breve{L}^{-1} \leqslant (n-1)d$).

In the case when we want only to test whether L is unimodular without constructing L^{-1}, then we can use $\triangle$EG; as we have mentioned, L is unimodular if and only if the number of differentiations has to be exactly equal to nd, and the

leading matrix is invertible in K after those differentiations. Such testing has the complexity

$$\Theta(n^3d^2). \tag{20}$$

The formulated computational problems related to unimodularity can be solved by different algorithms. For example, algorithms to construct the Jacobson and Hermite forms of a given operator matrix can be used. A polynomial-time deterministic algorithm for constructing the Jacobson form of L was proposed in [22]. Its complexity is considered in [22] as a function of three variables, and two of them are our n, d (in [22], another notation is used). The value of the third variable is in the worst case nd, and for the complexity as a function of the variables n, d one can derive the estimate $\Theta(n^9d^9)$. As we have mentioned in the proof of Proposition 5, the Hermite form of a unimodular matrix is the identity, the transformation matrix U is the inverse. The complexity estimate for the algorithm given in [18, Thm 5.5] is $O(n^7d^3\log(nd))$ (in our notation). It looks like this estimate is tight. (Of course, the algorithms from [22, 18] solve more general problems, and the algorithm given above in this section has some advantages only for recognizing invertibility of an operator matrix and computing the inverse matrix.)

The author is unaware of the algorithms which solve the testing unimodularity problem with a complexity which is less than (20). Search in the literature gave no positive result, but of course it is possible that such algorithms exist. The author makes no attempt to offer a champion algorithm for solving this problem. Perhaps, for example, using the ideas of the fast matrix multiplication over a field [27, 16, 28], as well as the fast multiplication algorithm for scalar differential operators [20, 12, 13], one can propose an algorithm for fast multiplication operator matrices and then get out of it the appropriate algorithm for solving the unimodularity testing problem.

5 Conclusions

The author's goal is to show that to choose an algorithm to solve a problem over a differential field K it is reasonable to take into account the complexity measured as not only the number of arithmetic operation in K but all operations including the operation of differentiation. The algorithms that have the equivalent complexity as the number of arithmetic operations in the worst case can differ in the context of the full algebraic complexity that includes needed differentiations.

It is worthy to note that some versions of algorithms EG and RR are used quite successfully, for example, for finding singular points of differential systems, and we mentioned this in Section 4.1. We can expect that by these algorithms, the unimodularity testing will be performed in practice in a reasonable time.

From the current work, new questions arise.

First, the question formulated in Section 3.4: suppose that we store all the results of differentiations; does it allow to decrease the complexities (13), (14), (19) (it would be desirable to get d^2 instead of d^3)?

Second. It is unclear whether there exists an algorithm for unimodularity testing whose complexity is $O(n^\alpha d^\beta)$, where α, β are real numbers and $\alpha < 3$. For matrices whose entries are commutative polynomials from $K[x]$, there is an algorithm [17] for constructing the inverse matrix with complexity $O(n^3\rho)$, where ρ is the maximal degree of entries of given matrices (strictly speaking, algorithm from [17] is for the case of "generic matrix inversion" only). It is unclear whether the problem of constructing the inverse operator matrix is reducible to the problem of the operator matrix multiplication, similarly to the case when entries of matrices belong to a field [15, Sect. 16.4]. Going back to matrices with polynomial entries, note that there exists a matrix multiplication algorithm [14] with complexity $O(n^\omega\rho\, f(\log n \log\rho))$, where

f is a polynomial. However, an algorithm with a similar complexity for the matrix inversion does not probably exists. It looks like that the problem of constructing the inverse matrix is not reducible to the problem of the operator matrix multiplication neither for polynomial matrices nor for operator matrices.

Third, much recent work has focused on properly dealing with the growth in the size of coefficients from K, for example, when $K = \mathbb{Q}(x)$ ([11], [18] etc). It would be valuable to investigate the bit complexity of the unimodularity testing algorithms. Another way is to consider the complexity as a function of three variables: n, d and ρ, where ρ is such that all the polynomials involved into L as numerators and denominators of coefficients of entries of L are of degree which is ρ at most. The complexity itself is then the number of operations in $\mathbb{Q}$ in the worst case. The algorithms should allow control over coefficients growth. It is reasonable to continue to investigate this line of enquiry.

Acknowledgments

The author is thankful to M. Barkatou and A. Storjohann for interesting discussions, and to anonymous referees for useful comments.

References

1. Abramov, S: EG–Eliminations. J. Difference Equations Appl. 5, 393–433 (1999)
2. Abramov, S., Barkatou, M.: On the dimension of solution spaces of full rank linear differential systems. In: Gerdt, V.P. et al. (eds.) CASC 2013. LNCS, vol. 8136, pp. 1–9. Springer, Heidelberg (2013)
3. Abramov, S., Barkatou, M.: On solution spaces of products of linear differential or difference operators. ACM Comm. in Computer Algebra 4, 155–165 (2014)
4. Abramov, S., Bronstein, M.: On solutions of linear functional systems. In: ISSAC'2001 Proceedings, pp. 1–6 (2001)
5. Abramov, S., Bronstein, M.: Linear algebra for skew-polynomial matrices. Rapport de Recherche INRIA RR-4420, March 2002 `http://www.inria.fr/RRRT/RR-4420.html`
6. Abramov, S., Khmelnov, D.: On singular points of solutions of linear differential systems with polynomial coefficients. J. Math. Sciences 185(3), 347–359 (2012) (Translated from: Fundamentalnaya i prikladnaya matematika 17(1), 3–21 (2011/12))
7. Abramov, S., Khmelnov, D., Ryabenko, A.: Procedures for searching local solutions of linear differential systems with infinite power series in the role of coefficients. Programming Comput. Software 42(2), 55–64 (2016)
8. Barkatou, M., Cluzeau, T., El Bacha, C.: Simple form of higher-order linear differential systems and their application in computing regular solutions. J. Symb. Comput. 46(6), 633–658 (2011)
9. Barkatou, M., El Bacha, C., Labahn, G., Pflügel, E.: On simultaneous row and column reduction of higher-order linear differential systems J. Symb. Comput. 49(1), 45–64 (2013)
10. Beckermann, B., Labahn, G.: Fraction-free computation of matrix rational interpolants and matrix GCDs. SIAM J. Matrix Anal. Appl. 77(1), 114–144 (2000)
11. Beckermann, B., Cheng, H., Labahn, G.: Fraction-free row reduction of matrices of Ore polynomials. J. Symb. Comput. 41, 513–543 (2006)
12. Benoit, A., Bostan, A., van der Hoeven, J.: Quasi-optimal multiplication of linear differential operators. In: FOCS'2012 Proceedings, pp. 524–530 (2012)
13. Bostan, A., Chyzak, F., Le Roix, N.: Products of ordinary differential operators by evaluating and interpolation. In: ISSAC'2008 Proceedings, pp. 23–30 (2008)
14. Bostan, A., Schost, E.: Polynomial evaluation and interpolation on special sets of points. J. Complexity. 21(4), 420–446 (2005)
15. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Grundlehren der mathematischen Wissenschaften, 315 Springer, Heidelberg (1997)

16. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. J. Symb. Comput. 9(3), 251–280 (1990).
17. Jeannerod, C.-P., Villard, G.: Essentially optimal computation of the inverse of generic polynomial matrices. J. Complexity 21(1), 72–86 (2005)
18. Giesbrecht, M., Sub Kim, M.: Computation of the Hermite form of a Matrix of Ore Polynomials. J. Algebra 376, 341–362 (2013)
19. Gustavson, R., Kondratieva, M., Ovchinnikov, A.: New effective differential Nullstellensatz. Advances in Mathematics 290, 1138–1158 (2016)
20. van der Hoeven, J.: FFT-like multiplication of linear differential operators. J. Symb. Comput. 33, 123–127 (2002)
21. Knuth, D.E.: Big omicron and big omega and big theta, ACM SIGACT News 8(2), 18–23 (1976)
22. Middeke, J.: A polynomial-time algorithm for the Jacobson form for matrices of differential operators. Tech. Report No. 08-13 in RISC Report Series (2008)
23. Miyake, M.: Remarks on the formulation of the Cauchy problem for general system of ordinary differential equations. Tôhoku Math. J. 32(2), 79–89 (1980)
24. Mulders, T., Storjohann, A.: On Lattice Reduction for Polynomial Matrices. J. Symb. Comput. 37(4), 485–510 (2004)
25. van der Put, M., Singer, M.F.: Galois Theory of Linear Differential Equations. Grundlehren der mathematischen Wissenschaften, 328 Springer, Heidelberg (2003)
26. Rosenlicht, M.: Integration in finite terms. Amer. Math. Monthly 79(9), 963–972 (1972)
27. Strassen, V.: Gaussian elimination is not optimal. Numer. Math. 13, 354–356 (1969)
28. Vassilevska Williams, V.: Multiplying matrices faster than Coppersmith–Winograd. In: STOC2012 Proceedings, pp. 887–898 (2012)