ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2017, Vol. 57, No. 12, pp. 1887–1898. © Pleiades Publishing, Ltd., 2017.
Original Russian Text © S.A. Abramov, 2017, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2017, Vol. 57, No. 12, pp. 1933–1945.

Inverse Linear Difference Operators

S. A. Abramov

Dorodnicyn Computing Center, Federal Research Center "Computer Science and Control", Russian Academy of Sciences, Moscow, 119333 Russia

e-mail: sergeyabramov@mail.ru

Received August 28, 2016; in final form, January 23, 2017

Abstract—For matrices whose elements are scalar linear difference operators, algorithms for checking invertibility (unimodularity) and constructing an inverse matrix (if it exists) are proposed. Their complexity is lower than that of other available algorithms. The differences of these algorithms from their differential analogues are discussed.

Keywords: complexity of algorithms, difference operator, operator matrix, unimodular matrix, unimodularity recognition, inverse matrix construction.

DOI: 10.1134/S0965542517120028

1. INTRODUCTION

Given a matrix over a field or ring, checking whether it is invertible and constructing an inverse matrix (if any) are classical mathematical problems. Below, these problems are considered as applied to operator matrices. In this case, matrix elements are scalar linear difference operators with coefficients from a difference field $\mathbb{K}$ with an automorphism (shift) σ. The field $\mathbb{K}$ is assumed to be of characteristic 0. New algorithms for solving these problems are discussed. Note that they can be solved by well-known algorithms intended originally for more general problems (this matter will be discussed later). The new algorithms proposed have lower complexity.

In the case of operator matrices, the term "unimodular matrix" is usually used instead of "invertible matrix." The former term will be used throughout this paper.

In the differential case when $\mathbb{K}$ is a differential field of characteristic 0 with derivation $\partial = '$ and when the matrix elements are scalar linear differential operators over $\mathbb{K}$, algorithms for checking the unimodularity of a matrix and constructing its inverse were considered in [1]. For a given operator matrix L, both differential and (discussed below) difference algorithms rely on determining the dimension of the solution space V_L of the corresponding system of equations under the assumption that the components of solutions belong to the Picard–Vessiot extension (see [2–4]) of $\mathbb{K}$ associated with L. An operator matrix L of full rank (the rows of L are independent over the ring of scalar linear operators) is unimodular if and only if $\dim V_L = 0$, i.e., V_L is a zero space (see [5]).

The following notation is used below. The ring of $n \times n$ matrices (n is a positive integer) with elements from a ring or field R is denoted by $\mathrm{Mat}_n(R)$. If M is an $n \times n$ matrix, then $M_{i,*}$ with $1 \le i \le n$ is used to denote the $1 \times n$ matrix equal to the ith row of M. A diagonal $n \times n$ matrix with diagonal elements $r_1, \ldots, r_n$ is designated as $\mathrm{diag}(r_1, \ldots, r_n)$, and I_n stands for the $n \times n$ identity matrix.

The results of this paper were preliminarily announced in an extended abstract [6].

2. DIFFERENCE FROM THE DIFFERENTIAL CASE

There are at least three differences from the differential case considered in [1].

1. In the difference case, there is a natural possibility of separately considering the complexities measured as the number of arithmetic operations and as the number of shifts; this is similar to sorting algorithms, for which the number of comparisons and the number of element shifts are considered separately. Given a matrix, computing the number of required arithmetic operations, we can ignore shifts, while, computing the number of shifts, we can ignore arithmetic operations. In the differential case, two com-

plexities are not so easy to consider separately, since the application of the derivation operator ∂ to a scalar differential operator on the left requires additional arithmetic operations (operators are assumed to be represented in the standard form as polynomials in ∂), for example, $\partial(a\partial + b) = a\partial^2 + (a' + b)\partial + b'$. Ignoring derivation operations can lead to misleading estimates for the number of arithmetic operations. In the difference case, we deal with an automorphism σ, which opens up the opportunity to consider the complexities separately.

2. The simple replacement of ∂ by σ in algorithms for the differential case does not yield algorithms for the difference case. Here, additional tricks are required. As was said above, algorithms are based on the determination of $\dim V_L$. The situation is such that the system $\sigma y = Ay$, where A is an $n \times n$ matrix with elements from $\mathbb{K}$, has a solution space of dimension n if and only if A is nonsingular, while, in the differential case $y' = Ay$, this nonsingularity is not required. However, in the difference case, this dimension can also be algorithmically computed for an arbitrary linear difference system (see [5]).

3. Suppose that a difference field $\mathbb{K}$ is the field of rational functions of x and the application of σ to an arbitrary rational function is obtained by substituting $x + 1$ for x, followed by reducing the result to canonical form (for σ^{-1}, $x - 1$ is substituted). Then it is natural to assume that σ has the same complexity as σ^k for any integer k. This point of view can affect the algorithmic complexity in terms of the number of shifts. In the differential case, the identification of the complexities for ∂ and ∂^k would be groundless: it is natural to assume that the application of ∂^k is the k-time application of ∂.

3. ADEQUATE DIFFERENCE EXTENSIONS

Recall that a *difference ring* is a commutative ring $\mathbb{K}$ with identity and an automorphism σ (which will frequently be referred to as a *shift*). If $\mathbb{K}$ is additionally a field, then it is called a *difference field*. The difference fields considered in what follows are always assumed to be fields of characteristic 0.

The *constant ring* of a difference ring $\mathbb{K}$ is $\mathrm{Const}(\mathbb{K}) = \{c \in \mathbb{K} | \sigma c = c\}$. If $\mathbb{K}$ is a difference field, then $\mathrm{Const}(\mathbb{K})$ is a subfield of $\mathbb{K}$ (the *constant field* of $\mathbb{K}$).

Let $\mathbb{K}$ be a difference field with an automorphism σ, and let Λ be a difference ring extension of $\mathbb{K}$ (on $\mathbb{K}$ the corresponding automorphism of Λ coincides with σ; for this automorphism of Λ, we use the same notation σ).

Definition 1. The ring Λ (which is a difference field extension of $\mathbb{K}$) is said to be an *adequate* difference extension of $\mathbb{K}$ if $\mathrm{Const}(\Lambda)$ is a field and an arbitrary system

$$\sigma y = Ay, \quad y = (y_1, \ldots, y_n)^{\mathrm{T}}, \tag{1}$$

with a nonsingular matrix $A \in \mathrm{Mat}_n(\mathbb{K})$ has in Λ^n a linear solution space of dimension n over $\mathrm{Const}(\Lambda)$.

The nonsingularity of A in this definition is essential: for example, if the first row of A is all zero, then the component y_1 in any solution of system (1) is zero as well.

Remark 1. The q-difference case (see [7, 8]) is covered by the general difference case.

If $\mathrm{Const}(\mathbb{K})$ is algebraically closed, then there exists a unique (up to a difference isomorphism, i.e., an isomorphism commuting with σ) adequate extension Ω such that $\mathrm{Const}(\Omega) = \mathrm{Const}(\mathbb{K})$, which is called the *universal* difference (Picard–Vessiot) ring extension of $\mathbb{K}$. The complete proof of its existence is not easy (see [4, Section 1.4]), while the existence of an adequate difference extension Λ for an arbitrary difference field can be rather easily proved (see [9, Section 5.1]). However, it should be emphasized that, for an adequate extension, the equality $\mathrm{Const}(\Lambda) = \mathrm{Const}(\mathbb{K})$ is not guaranteed; in the general case, $\mathrm{Const}(\mathbb{K})$ is a proper subfield of $\mathrm{Const}(\Lambda)$.

The assertion that a universal difference extension exists for an arbitrary difference field of characteristic 0 is not true if the extension is understood as a field. Franke's well-known example (see [10]) is the scalar equation $\sigma y = -y$ over a field $\mathbb{K}$ with an algebraically closed constant field. This equation has no nontrivial solutions in any difference extension having an algebraically closed constant field.

In what follows, Λ denotes a fixed adequate difference extension of a difference field $\mathbb{K}$ with an automorphism σ.

4. ORDERS OF OPERATORS AND SOLUTION SPACES

A scalar difference operator is an element of the ring $\mathbb{K}[\sigma, \sigma^{-1}]$. Given a nonzero scalar operator $f = \sum a_i \sigma^i$, its upper and lower orders are defined as

$$\overline{\operatorname{ord}} f = \max\{i | a_i \neq 0\}, \quad \underline{\operatorname{ord}} f = \min\{i | a_i \neq 0\}$$

and the order of f is defined as

$$\operatorname{ord} f = \overline{\operatorname{ord}} f - \underline{\operatorname{ord}} f.$$

Let $\overline{\operatorname{ord}}\, 0 = -\infty$, $\underline{\operatorname{ord}}\, 0 = \infty$, and $\operatorname{ord} 0 = -\infty$.

For a finite set F of scalar operators (a vector, matrix, matrix row), $\overline{\operatorname{ord}}\, F$ is defined as the maximum of the upper orders of its elements; $\underline{\operatorname{ord}} F$ is defined as the minimum of the lower orders of its elements; and, finally, $\operatorname{ord} F$ is defined as $\overline{\operatorname{ord}}\, F - \underline{\operatorname{ord}}\, F$.

A difference operator matrix is a matrix from $\operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$. In the subsequent exposition, such an operator matrix is associated with some matrices belonging to $\operatorname{Mat}_n(\mathbb{K})$. To avoid terminology confusion, operator matrices will be briefly referred to as *operators*. The case of scalar operators will be considered separately.

An operator is of full rank (or is a full-rank operator) if its rows are linearly independent over $\mathbb{K}[\sigma, \sigma^{-1}]$. Recall that same-length rows $u_1, \ldots, u_s$ with elements belonging to $\mathbb{K}[\sigma, \sigma^{-1}]$ are called *linearly dependent* (over $\mathbb{K}[\sigma, \sigma^{-1}]$) if there exist $f_1, \ldots, f_s \in \mathbb{K}[\sigma, \sigma^{-1}]$ not all zero such that $f_1 u_1 + \ldots + f_s u_s = 0$; otherwise, these rows are called *linearly independent* (over $\mathbb{K}[\sigma, \sigma^{-1}]$).

If

$$L \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}]), \quad l = \overline{\operatorname{ord}}\, L, \quad t = \underline{\operatorname{ord}}\, L,$$

and L is a nonzero matrix, then L can be written in expanded form as

$$L = A_l \sigma^l + A_{l-1} \sigma^{l-1} + \cdots + A_t \sigma^t, \tag{2}$$

where $A_l, A_{l-1}, \ldots, A_t \in \operatorname{Mat}_n(\mathbb{K})$ and A_l, A_t (the leading and trailing matrices of the original operator) are nonzero.

Definition 2. Let the upper and lower row orders of an operator L be $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$, respectively. The *frontal* matrix of L is the leading matrix of the operator PL, where

$$P = \operatorname{diag}(\sigma^{l-\alpha_1}, \ldots, \sigma^{l-\alpha_n}), \quad l = \overline{\operatorname{ord}}\, L.$$

Accordingly, the *rear* matrix of L is the trailing matrix of the operator QL, where

$$Q = \operatorname{diag}(\sigma^{t-\beta_1}, \ldots, \sigma^{t-\beta_n}), \quad t = \underline{\operatorname{ord}}\, L.$$

We say that L is *strongly reduced* if its frontal and rear matrices are both nonsingular.

Definition 3. An operator $L \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ is called *unimodular* or *invertible* if there exists an inverse $L^{-1} \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$: $LL^{-1} = L^{-1}L = I_n$. The set of unimodular $n \times n$ operators is denoted by Υ_n. Two operators $L_1, L_2 \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ are said to be *equivalent* if $L_1 = UL_2$ for some $U \in \Upsilon_n$.

Let V_L denote the space of solutions of the system $L(y) = 0$ that belong to Λ^n (see Section 3). For brevity, V_L is sometimes called the solution space of L.

For the difference case, Theorem 1 from [1] can be restated as follows.

Theorem 1. *Let $L \in \operatorname{Mat}_n(K[\sigma, \sigma^{-1}])$ be a full-rank operator. Then the following assertions hold:*

(i) *If L is strongly reduced, then* $\dim V_L = \sum_{i=1}^{n} \operatorname{ord} L_{i,*}$.

(ii) $L \in \Upsilon_n \Leftrightarrow V_L = 0$.

As in the differential case, the proof is based on [5, 9].

5. REDUCTION OF OPERATORS

In this section and below, we assume without explicit mention that n and d are integers such that $n > 0$ and $d \geq 0$ and the original operator L is such that

$$L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}]), \quad l = \overline{\mathrm{ord}}\, L, \quad t = \underline{\mathrm{ord}}\, L, \quad d = \mathrm{ord}\, L. \tag{3}$$

Algorithms for solving the considered matrix problems are characterized by two complexities that are functions of n and d:

• arithmetic complexity, i.e., the number of arithmetic operations in $\mathbb{K}$ in the worst case for fixed n and d;

• shift complexity, i.e., the number of operations σ, σ^{-1} applied to elements of $\mathbb{K}$ in the worst case for fixed n and d.

In deriving asymptotic complexity bounds, along with big O notation, we use Θ notation (see [11]): the relation $f(n,d) = \Theta(g(n,d))$ is equivalent to $f(n,d) = O(g(n,d))$, where $g(n,d) = O(f(n,d))$, i.e., $f(n,d)$ and $g(n,d)$ are quantities of the same order as $n, d \to \infty$.

Concerning the concept of shift complexity, we note that the computation of $\sigma^k a$ for $k \in \mathbb{Z}$ and $a \in \mathbb{K}$ requires, by assumption, that the operation σ or σ^{-1} be applied $|k|$ times.

5.1. EG Algorithms

5.1.1. Algorithm EG_σ. If L is a full-rank operator, then the algorithm EG_σ (see [12–14]) constructs an equivalent operator $\hat{L}_+ \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ with $\overline{\mathrm{ord}}\, \hat{L}_+ = \overline{\mathrm{ord}}\, L$ and $\underline{\mathrm{ord}}\, \hat{L}_+ \geq \underline{\mathrm{ord}}\, L$ that has a nonsingular leading matrix. If L is not of full rank, the algorithm returns a corresponding message.

The algorithm EG_σ constructs $\hat{L}_+$ by transforming L, i.e., L changes step by step, gradually turning into $\hat{L}_+$:

Check whether the rows of the leading matrix of L are linearly independent over $\mathbb{K}$. If yes, then L does not change and the algorithm terminates. Otherwise, a series of same-type steps are executed, each consisting of the following operations:

• Calculate the dependence coefficients $p_1, \ldots, p_n \in \mathbb{K}$ for the rows of the leading matrix.

• Among the rows of the operator corresponding to nonzero coefficients, choose one with the least lower order (if there are several such rows, take any one of them). Let i be the index of the chosen row.

• Replace the row $L_{i,*}$ by $\sum_{k=1}^{n} p_k L_{k,*}$. If the original operator L is of full rank, then the resulting ith row is nonzero. Let the upper order of the ith row be α. Apply $\sigma^{l-\alpha}$ to this row.

If L is of full rank, then, after at most nd steps, the algorithm produces $\hat{L}_+$ with a nonsingular leading matrix. If L is not of full rank, then, after at most nd steps, a zero row appears in the operator. Indeed, the sum of the lower orders of all rows of L increases at every step. However, this sum is initially at least tn and cannot become greater than ln at any step. We have $ln - tn = dn$. If there is no zero row after executing $nd - 1$ steps, then, at the ndth step, the sum of the lower orders of all rows of the operator becomes equal to ln and the operator itself is a matrix from $\mathrm{Mat}_n(\mathbb{K})$. This step is the last: either the resulting matrix is nonsingular or its rows are linearly dependent and, hence, a zero row appears in the matrix at this step.

Remark 2. If the rear matrix of L is nonsingular, it remains nonsingular after the application of EG_σ. This is ensured by the fact that the rows with the least lower orders are chosen to be eliminated. Note additionally that, if the trailing matrix of L is nonsingular, then this matrix is simultaneously the rear (nonsingular) matrix of L.

The fact that $\hat{L}_+$ is equivalent to L and, hence, $V_L = V_{\hat{L}_+}$, is derived from the existence of an inverse σ^{-1} of the automorphism σ.

5.1.2. Algorithm $\mathrm{EG}_{\sigma^{-1}}$. By analogy with EG_σ, we can propose an algorithm $\mathrm{EG}_{\sigma^{-1}}$ in which the trailing matrix of the operator is considered instead of its leading matrix, the row with the highest upper order is replaced by a linear combination of rows, and σ raised to the corresponding negative power is applied to the resulting linear combination. If L is of full rank, then, after at most nd steps, the algorithm yields an

equivalent operator $\hat{L}_-$ with a nonsingular trailing matrix; moreover, $\underline{\operatorname{ord}}\,\hat{L}_- = \underline{\operatorname{ord}}\,L$ and $\overline{\operatorname{ord}}\,\hat{L}_- \le \overline{\operatorname{ord}}\,L$. If L is not of full rank, then, after at most nd steps, a zero row appears in the operator. If the frontal matrix of L is nonsingular, the algorithm produces $\hat{L}_-$ that also has a nonsingular frontal matrix, because the rows with the highest upper orders are chosen to be eliminated. By analogy with Remark 2, we note that, if the leading matrix of L is nonsingular, then this matrix is simultaneously the frontal (nonsingular) matrix of L.

Remark 3. Thus, if L is of full rank, then the sequential application of $\mathrm{EG}_{\sigma^{-1}}$ and EG_σ to the original operator yields an equivalent operator in the strongly reduced form. By Theorem 1(i), we can calculate $\dim V_L$.

5.2. Complexity of the EG Algorithms

Proposition 1. *The arithmetic complexity of each of the algorithms* EG_σ *and* $\mathrm{EG}_{\sigma^{-1}}$ *is*

$$\Theta(n^{\omega+1}d + n^3d^2), \tag{4}$$

where ω *is the matrix multiplication exponent*, $2 < \omega \leqslant 3$, w*hile the shift complexity is*

$$\Theta(n^2d^2). \tag{5}$$

Proof. The search for a linear dependence of rows in the leading or trailing matrices is reduced to solving a homogeneous system of n linear algebraic equations with n unknowns. The complexity of this search is $\Theta(n^\omega)$. The arithmetic complexity of reducing the order of a row by one in the worst case is $\Theta(n^\omega + n^2d)$, which yields estimate (4). Let the orders of all rows in L be equal to d. If every step of the algorithm reduces the order of a single row by one and finally the algorithm produces $\hat{L}$ of order 0 (which corresponds to the worst case), then each of the rows requires $nd + n(d-1) + \ldots + n\Theta(nd^2)$ shifts, while all rows require the number of shifts given by (5). The proposition is proved.

By applying $\mathrm{EG}_{\sigma^{-1}}$ to L and, in the case of a full-rank operator, subsequently applying EG_σ, we can find $\dim V_L$ (see Remark 3). By Theorem 1(ii), the operator L is unimodular if and only if $\dim V_L = 0$, and testing for unimodularity can be performed with arithmetic complexity (4) and shift complexity (5). It will be shown in Subsection 6.1 that the arithmetic complexity of this testing can be reduced.

5.3. RR Algorithms

5.3.1. Algorithms RR_σ and $\mathrm{RR}_{\sigma^{-1}}$. If L is a full-rank operator, then the algorithm RR_σ (see [16–18]) constructs an equivalent operator $\breve{L}_+ \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ with $\overline{\operatorname{ord}}\,\breve{L}_+ \le \overline{\operatorname{ord}}\,L$ and $\underline{\operatorname{ord}}\,\breve{L}_+ \ge \underline{\operatorname{ord}}\,L$ that has a nonsingular frontal matrix. If L is not of full rank, then the algorithm returns a corresponding message.

The algorithm RR_σ constructs $\breve{L}_+$ by transforming L, i.e., L changes step by step, gradually turning into $\breve{L}_+$:

Check whether the rows of the frontal matrix of L are linearly independent over $\mathbb{K}$. If yes, then L does not change and the algorithm terminates. Otherwise, a series of same-type steps are executed, each consisting of the following operations:

(a) Calculate the dependence coefficients $p_1, \ldots, p_n \in \mathbb{K}$ for the rows of the frontal matrix.

(b) Among the rows corresponding to nonzero coefficients, choose one with the highest upper order (if there are several such rows, take any one of them). Let i be the index of the chosen row.

(c) Replace the row $L_{i,*}$ by $\sum_{k=1}^{n}(\sigma^{\alpha_i - l}p_k)(\sigma^{\alpha_i - \alpha_k}L_{k,*})$.

If L is of full rank, then, after at most nd steps, the algorithm yields $\breve{L}_+$ with a nonsingular frontal matrix. If L is not of full rank, then, after at most nd steps, a zero row appears in the operator.

By analogy with $\mathrm{EG}_{\sigma^{-1}}$, we can propose an algorithm $\mathrm{RR}_{\sigma^{-1}}$ that constructs an equivalent operator $\breve{L}_-$ with a nonsingular rear matrix.

5.3.2. Extended RR_σ and $\mathrm{RR}_{\sigma^{-1}}$. The extended algorithm RR_σ, which is denoted by ExtRR_σ, produces, along with $\breve{L}_+$, a unimodular operator $U \in \Upsilon_n$ such that $\breve{L}_+ = UL$.

In fact, the input to ExtRR_σ is L and an arbitrary operator $W \in \mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$ and, if L is of full rank, then ExtRR_σ yields $\breve{L}_+$ and UW, where U is the above-mentioned unimodular operator (if $W = I_n$, then ExtRR_σ produces $\breve{L}_+$ and U):

Apply RR_σ to L and apply all operations executed over L to the operator initially equal to W.

Proposition 2. *Let* $L,W \in \mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, L *be of full rank*, $\operatorname{ord} L = d$, *and* $\operatorname{ord} W = d_1$. *Then* $\operatorname{ord} W = O(nd + d_1)$ *at every step of the algorithm* ExtRR_σ.

Proof. This result is a consequence of [16, Proposition 1; 22, Theorem 4.9]; here, the difference between the differential and difference cases is of no matter.

What was said above remains valid for the algorithm $\mathrm{RR}_{\sigma^{-1}}$ extended in a similar manner. This extended version is denoted by $\mathrm{ExtRR}_{\sigma^{-1}}$.

Remark 4. For the differential case, it was proved in [1, Proposition 5] that a unimodular $n\times n$ operator L of order d satisfies the inequality

$$\operatorname{ord} L^{-1} \le (n-1)d, \tag{6}$$

and, for any n, d, there exists an operator L such that $\operatorname{ord} L^{-1} = (n-1)d$. The proof is easy to modify for the difference case.

Example 1. For $n = 2$, any unimodular operator L satisfies $\operatorname{ord} L = \operatorname{ord} L^{-1}$; for example,

$$\begin{pmatrix} 1 & -\frac{1}{x}\sigma \\ \frac{x^2}{2} & -\frac{x}{2}\sigma + 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 - \frac{(x+1)^2}{2x}\sigma & \frac{1}{x}\sigma \\ -\frac{x^2}{2} & 1 \end{pmatrix}. \tag{7}$$

In this case,

$$\mathbb{K} = \mathbb{Q}(x), \quad \sigma x = x+1, \quad \operatorname{ord} L = \operatorname{ord} L^{-1} = 1.$$

5.3.3. Complexity of RR algorithms. The following result is true.

Proposition 3. *The arithmetic complexity of each of the algorithms* RR_σ *and* $\mathrm{RR}_{\sigma^{-1}}$ *is*

$$\Theta(n^{\omega+1}d + n^3d^2), \tag{8}$$

while the shift complexity is

$$\Theta(n^3d^3). \tag{9}$$

Proof. Using the same argument as in the proof of Proposition 1, we can see that the arithmetic complexity of each of the algorithms RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ coincides, in the order of growth, with the arithmetic complexity (4) of the algorithms EG_σ and $\mathrm{EG}_{\sigma^{-1}}$ described in Subsection 5.3.1.

Direct verification shows that the shift complexity of RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ is $O(n^3d^3)$. Now, to prove (9), it is sufficient to associate each pair n and d with an operator $L_{(n,d)}$ such that, for any n,d and a certain positive constant c common for all n and d, the number of shifts required by any of the algorithms RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ is greater than cn^3d^3. For RR_σ, for example, $L_{(n,d)}$ can be chosen from operators in which the upper orders of the first $[n/2]$ rows are equal to d, the upper orders of the other rows are equal to $[d/2]$, and the upper orders of all rows after the application of RR are equal to $[d/2]$. Additionally, it is required that the sum of the upper orders of the rows be reduced precisely by one at every step of the algorithm and that every elimination involve all rows of order $[d/2]$. The same is true for $\mathrm{RR}_{\sigma^{-1}}$, but the upper orders are replaced by lower ones.

Thus, as $n,d \to \infty$, the shift complexity of RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ has a higher order of growth than that of EG_σ and $\mathrm{EG}_{\sigma^{-1}}$.

For the differential case, it was proved in [1, Proposition 6] that, if all results of row differentiation are stored (so that differentiations are not repeated), then the total number of row differentiations in the dif-

ferential version of RR in the worst case is $O(nd^2)$. This estimate was not stated to be tight in some sense. This estimate is also valid for the number of row shifts in the difference case (in fact, the proof remains the same). Due to the storage of all results of row shifts, the space complexity is increased significantly, but the shift complexity of RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ is reduced: instead of (9), we obtain

$$O(n^2d^3). \tag{10}$$

The complexity of ExtRR_σ and $\mathrm{ExtRR}_{\sigma^{-1}}$ can be estimated by applying Proposition 2. Assuming as before that $\operatorname{ord} W = d_1$, we have $\Theta(nd(n^\omega + n^2 \cdot (d + d_1)) = \Theta(n^{\omega+1}d + n^3d^2 + n^3dd_1)$ for arithmetic complexity and $\Theta(nd \cdot n^2 \cdot (d + d_1)^2) = \Theta(n^3d^3 + n^3d^2d_1 + n^3dd_1^2)$ for shift complexity. If all shift results are stored, then the shift complexity is estimated by $O(nd \cdot n \cdot (d + d_1)^2) = O(n^2d^3 + n^2d^2d_1 + n^2dd_1^2)$.

Combining these relations with Remark 4 yields the following result.

Proposition 4. *For* $\operatorname{ord} W = \Theta(nd)$, *without storing all row shift results, the arithmetic and shift complexities of the algorithms* ExtRR_σ *and* $\mathrm{ExtRR}_{\sigma^{-1}}$ *are*

$$\Theta(n^4d^2), \quad \Theta(n^5d^3). \tag{11}$$

If all shift results are stored, then the complexities are

$$\Theta(n^4d^2), \quad O(n^4d^3). \tag{12}$$

5.4. Triangular Matrices

Definition 4. Suppose that the ith row of the frontal operator matrix of L has the form

$$(\underbrace{0, \ldots, 0}_{k-1}, a, \ldots, b), \quad 1 \le k \le n, \quad a \ne 0.$$

Then the number k is called the *pin index* of the ith row of L. If the ith row of L is zero, then its pin index is equal to $-\infty$.

If the rows of L have pairwise distinct positive pin indices, then the frontal matrix is nonsingular: up to the ordering of rows, it is a triangular matrix with nonzero diagonal elements. Assume that the rows $r_1 = L_{i,*}$ and $r_2 = L_{j,*}$ have the same positive pin index k and, moreover, $\overline{\operatorname{ord}}\, L_{i,*} = d_1$ and $\overline{\operatorname{ord}}\, L_{j,*} = d_2$, where $d_1 \le d_2$. Executing one arithmetic operation in $\mathbb{K}$, we can find $v \in \mathbb{K}$ such that the row

$$r_2 - v\sigma^{d_2-d_1}r_1 \tag{13}$$

has a pin index greater than k or an upper order lower than d_2 (or both). Of the rows r_1, r_2 in L, the one having a higher order is replaced by (13). If they have the same orders, any of them is replaced by (13). If L is of full rank, then, the frontal matrix becomes triangular after at most $n \cdot nd$ such unimodular transformations (each being equivalent to multiplication on the left by a unimodular operator). This technique can be used instead of the search for a linear dependence of the rows in the frontal matrix.

Remark 5. Under the assumption that $\mathbb{K}$ is the field of rational functions of x with the automorphism $x \to x + 1$, this technique was used in [12] in the first version of EG. In a discussion of the differential case, A. Storjohann attracted my attention to the fact that this approach has a lower complexity than the approach associated with solving linear algebraic systems (see also [20]).

Moreover, the algorithm $\Delta\mathrm{RR}_{\sigma^{-1}}$ can be proposed for obtaining an equivalent operator with a nonsingular rear matrix.

For EG_σ the leading and the frontal matrices coincide at the moment of comparing the pin indices; moreover, $d_2 - d_1 = 0$ in (13). Of the rows r_1 and r_2, (13) replaces the one with a smaller lower order, which is equivalent to a higher order row chosen for this replacement (if the rows have the same order, any of them is taken).

The algorithm is supplemented with a preliminary step at which the upper orders of all rows are equalized with the help of σ. (This transformation is unimodular, since σ and σ^{-1} are mutually inverse.) Following this approach, we obtain the algorithm $\Delta\mathrm{EG}_\sigma$.

Remark 6. Applying $\Delta\mathrm{EG}_\sigma$ to an operator with a nonsingular rear matrix yields an operator with both leading and rear matrices being nonsingular (the same is true of EG_σ, see Remark 2). This is a consequence of the replacement rule that "out of the rows r_1, r_2 in L, the one with a higher order is replaced by (13); if these rows have the same order, any of them is replaced." Following an analogy with the differential case [1], the replacement rule is that "the row r_2 is replaced in L by (13)," but the nonsingularity of the rear matrix may then not be conserved (in the differential case, this conservation is of no interest).

The algorithm $\Delta\mathrm{EG}_{\sigma^{-1}}$ is obtained in a similar manner. It finds an operator that is equivalent to the original one and has a nonsingular trailing matrix.

Proposition 5. *For the* ΔEG *algorithms, the arithmetic complexity is*

$$\Theta(n^3d^2) \tag{14}$$

and the shift complexity is

$$\Theta(n^2d^2). \tag{15}$$

Proof. Obviously, for the ΔEG family, the arithmetic complexity is $\Theta(nd\cdot n\cdot nd)$, i.e., (14). It is also easy to see that, for the ΔEG family, the number of shifts in the worst case is the same as for EG, i.e., $\Theta(n^2d^2)$.

6. TESTING FOR UNIMODULARITY AND INVERTING OPERATORS

6.1. Testing for Unimodularity

An algorithm for recognizing unimodularity with arithmetic complexity (4) and shift complexity (5) was mentioned at the end of Section 2. On the basis of $\Delta\mathrm{EG}_{\sigma^{-1}}$ and $\Delta\mathrm{EG}_{\sigma-1}$, we can propose an algorithm with the same shift complexity but with a lower arithmetic complexity. Specifically, $\Delta\mathrm{EG}_{\sigma^{-1}}$ is used to transform the original operator L into an equivalent one with a nonsingular trailing matrix. Then $\Delta\mathrm{EG}_\sigma$ is applied to the resulting operator. If the original operator is of full rank (which is determined in the course of applying $\Delta\mathrm{EG}_{\sigma^{-1}}$ and $\Delta\mathrm{EG}_\sigma$), then, according to Remark 6, this application yields the value of $\dim V_L$. By Theorem 1(ii), $L\in\Upsilon_n \Leftrightarrow \dim V_L = 0$.

Taking into account estimates (14) and (15), we conclude that the arithmetic and shift complexities of the described algorithm are

$$\Theta(n^3d^2),\quad \Theta(n^2d^2), \tag{16}$$

respectively.

6.2. Construction of an Inverse Operator

The inversion algorithm relies on the algorithms described in Subsection 5.3.2. It is represented as consisting of four steps. If the operator is not of full rank, this is uncovered at the first step and the algorithm terminates.

Step 1. Apply $\mathrm{ExtRR}_{\sigma^{-1}}$ to L and I_n. Let the result be $L_1\in \mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, $W_1\in\Upsilon_n$.

Step 2. Apply ExtRR_σ to L_1 and W_1. Let the result be $L_2\in \mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, $W_2\in\Upsilon_n$.

Step 3. If $\operatorname{ord} L_2 > 0$, then $L\notin\Upsilon_n$.

Step 4. Let $\operatorname{ord} L_2 = 0$ and $\beta_1, \ldots, \beta_n$ be the lower orders of the rows of L_2. Then $L_2 = MD$, where $M\in\mathrm{Mat}_n(\mathbb{K})$ and $D = \operatorname{diag}(\sigma^{\beta_1},\ldots,\sigma^{\beta_n})$. From this, $L^{-1} = \operatorname{diag}(\sigma^{-\beta_1},\ldots,\sigma^{-\beta_n})M^{-1}W_2$.

After performing Step 2, the frontal and rear matrices of L_2 are nonsingular. The total arithmetic and shift complexities of Steps 1–3 is given by (11). The arithmetic complexity of computing $M^{-1}W_2$ (Step 4) is $O(n^2\cdot nd) = O(n^3d)$. This complexity grows more slowly than n^4d^2. The first estimate in (11) and (12) remains unchanged. The values $\beta_1,\ldots,\beta_n$ obtained at Step 4 satisfy $0\le\beta_i\le d$, $i = 1,\ldots,n$. Moreover, $\operatorname{ord} M^{-1}W_2 = \operatorname{ord} W_2 \le (n-1)\operatorname{ord} L = (n-1)d$. Therefore, the shift complexity of multiplying

$\operatorname{diag}(\sigma^{-\beta_1}, \ldots, \sigma^{-\beta_n})$ by $M^{-1}W_2$ in the worst case is $O(n^2d^2)$, while the second estimates in (11) and (12) remain unchanged.

Finally, we have proved the following result.

Theorem 2. (i) *To test an arbitrary operator L for unimodularity, there exists an algorithm with arithmetic and shift complexities equal to $\Theta(n^3d^2)$ and $\Theta(n^2d^3)$. The algorithm returns "yes" (the operator is unimodular) or "no" (the operator is not unimodular).*

(ii) *To test an arbitrary operator L for unimodularity and, if L is unimodular, to construct L^{-1}, there exists an algorithm with arithmetic and shift complexities*

$$\Theta(n^4d^2), \quad O(n^4d^3), \tag{17}$$

respectively. The algorithm produces an operator L^{-1} or says that L is not unimodular. (If the results of all performed shifts are not stored, then the arithmetic and shift complexities are $\Theta(n^4d^2)$ and $\Theta(n^5d^3)$, but the space complexity increases, i.e., more storage space is required in the worst case.)

Example 2. The operator on the left-hand side of (7) is written in expanded form:

$$L = \begin{pmatrix} 1 & -\frac{1}{x}\sigma \\ \frac{x^2}{2} & -\frac{x}{2}\sigma + 1 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & -\frac{x}{2} \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ \frac{x^2}{2} & 1 \end{pmatrix}.$$

The rear matrix coincides with the trailing one, and the latter is nonsingular. By applying $\operatorname{ExtRR}_\sigma$, L is transformed as follows:

$$\begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & -\frac{x}{2} \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ \frac{x^2}{2} & 1 \end{pmatrix} \to \begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & 0 \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The last matrix in the chain is I_2. It has a full rank, and its order is zero. We have $\beta_1 = \beta_2 = 0$. It follows that the original operator L is unimodular. By applying $\operatorname{ExtRR}_\sigma$, the operator W initially equal to I_2 is transformed as

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 \\ -\frac{x^2}{2} & 1 \end{pmatrix} \to \begin{pmatrix} 1 - \frac{(x+1)^2}{2x}\sigma & \frac{1}{x}\sigma \\ -\frac{x^2}{2} & 1 \end{pmatrix}. \tag{18}$$

According to the algorithm, the last operator in the chain is multiplied on the left by the inverse of I_2. This multiplication does not change the last operator in (18), and the same is true of multiplication by $\operatorname{diag}(\sigma^0, \sigma^0)$. Thus, the last operator in (18) is the inverse of L. It coincides with the operator on the right-hand side of (7).

Consider the case of 1×1 matrices. According to the above algorithms, every 1×1 matrix is unimodular if and only if its only element a is an operator of order 0, i.e., $a \in \mathrm{K}\backslash\{0\}$. In this case, the inverse matrix consists of the single element a^{-1}.

It is easy to note that in algorithm of an inversion of an operator we did not use the approaches based on deriving of triangular matrices and discussed in item 5.4, although this approach has allowed to reduce arithmetic complexity of algorithm of verification of a unimodularity. By analogy with $\operatorname{ExtRR}_\sigma$, $\operatorname{ExtRR}_{\sigma^{-1}}$ we can describe algorithms $\operatorname{Ext}\Delta\operatorname{RR}_\sigma$, $\operatorname{Ext}\Delta\operatorname{RR}_{\sigma^{-1}}$, but remains unclear, for example, whether on all steps of these new algorithms will be satisfied of Proposition 2.

7. FIELD OF RATIONAL FUNCTIONS AS AN INITIAL DIFFERENCE FIELD

If the elements of the field $\mathbb{K}$ are represented by expressions involving the variable x and if the shift σ is the replacement of x by $x+1$, followed by a possible simplification of the expression, then, for any

$k \in \mathbb{Z}$, the costs of applying σ^k and σ can be considered identical. As a result, the shift complexity of some of the above algorithms is reduced. Let us examine these complexity changes, assuming that $\mathbb{K}$ is the field $K(x)$ of rational functions over a field K of characteristic 0 and $\sigma x = x + 1$.

Note that, if the application of σ^k to any $a \in \mathbb{K}$ requires finding $\sigma a, \ldots, \sigma^k a$, then these quantities can be stored for further use. At the same time, the computation of $\sigma^k a$ as discussed above does not yield $\sigma a, \ldots, \sigma^{k-1} a$ and, if we need, for example, $\sigma^{k-1} a$ later, this element has to be computed. Therefore, the complexity of an algorithm with the above-discussed treatment of the cost of applying σ^k is not necessarily lower than that in the case of the shift complexity treatment described in the preceding sections. However, in our situation, the shift complexity is indeed reduced slightly.

It is easy to see that estimate (9) for RR_σ and $\mathrm{RR}_{\sigma^{-1}}$ is transformed into $\Theta(n^3 d^2)$: each of nd steps in the worst case requires $n-1$ row shifts. For ExtRR_σ and $\mathrm{ExtRR}_{\sigma^{-1}}$, the second (shift complexity) estimate in (10) is transformed into $\Theta(n^4 d^2)$.

For the families $\Delta\mathrm{EG}$ and $\Delta\mathrm{RR}$, estimate (15) remains unchanged, while the shift complexity estimate for algorithms of the family $\Delta\mathrm{RR}$ becomes the same as for $\Delta\mathrm{EG}$: with the new treatment of σ^k, every step in the worst case requires n shifts of nd-element rows.

The shift complexity estimate for a unimodularity recognition algorithm (the second estimate in (16)) remains unchanged, but, in the corresponding shift complexity estimate for the construction of an inverse operator, d^3 is replaced by d^2 because of similar changes in the shift complexity of ExtRR_σ and $\mathrm{ExtRR}_{\sigma^{-1}}$. Thus, estimates (17) are transformed into $\Theta(n^4 d^2)$ and $O(n^4 d^2)$.

The following result has been proved.

Theorem 3. *Let $\mathbb{K} = K(x)$, where K is a field of characteristic 0 and x* is a variable. Let $\sigma R(x) = R(x+1)$ *for any rational function $R(x) \in K(x)$. Then there exists an algorithm for recognizing the unimodularity of an arbitrary operator and for constructing an inverse operator* (*if it exists*) *such that its arithmetic and shift complexities are both $O(n^4 d^2)$.*

A similar result can be obtained for the q-difference case. In its simplest version, $\mathbb{K} = K(q, x)$, where q is another variable, and σ is the automorphism defined as $x \to qx$.

8. OTHER APPROACHES

The problems of unimodularity recognition and inverse matrix construction can be solved by applying various algorithms. For example, the Jacobson and Hermite forms of the given operator matrix can be constructed; their definitions can be found in [21, 22]. In [21] an algorithm for constructing the Jacobson form was proposed and its complexity was treated as a function of three variables, two of which are the above n and d (other notation was used in [21]). The value of the third variable in the worst case is equal to nd, and the complexity regarded as a function of n, d can be estimated as $\Theta(n^9 d^9)$. The Hermite form of a unimodular matrix is an identity matrix, and, in this case, the transformation matrix U is the inverse of the original one. In our notation, the complexity estimate presented in [22] is $O(n^7 d^3 \log(nd))$. It seems that this estimate is tight. (Of course, the algorithms in [21, 22] are intended for more general problems, and the algorithms described above have advantages only for unimodularity recognition and the construction of an inverse operator matrix.) The algorithms in [21, 22] are described in terms of rings of noncommutative Ore polynomials (see [23, 24]). As a result, those algorithms are applicable to the differential and difference cases.

Note that we are discussing algorithms in terms of complexity. An algorithm that looks the best in this sense is not necessarily the best in computational practice.

9. OPEN QUESTIONS AND HYPOTHESES

It is unclear whether there exists a unimodularity recognition algorithm with complexity equal to $O(n^\alpha d^\beta)$, where α, β are real numbers and $\alpha < 3$. For matrices whose elements are usual commutative polynomials from $K[x]$, there is an algorithm (see [25]) for constructing an inverse matrix with complexity

$O(n^3\rho)$, where ρ is the maximum degree of elements of given matrices (strictly speaking, the algorithm in [25] is intended for inverting only matrices in general position). It is also unclear whether the problem of constructing an inverse operator matrix can be reduced to matrix multiplication by analogy with reducibility in the case when the matrix elements belong to a field (see [26, Section. 16.4; 27, Chapter 6]). Here, reducibility is understood in the sense that, if there exists an algorithm for multiplying operator matrices with complexity (arithmetic or shift) $T(n,d)$, then there exists a matrix inversion algorithm with complexity $O(T(n,d))$. The hypothesized reducibility is doubtful, but reducibility in the opposite direction is proved in the same manner as for matrices over a field.

Returning to matrices with polynomial elements, we note that there exists a matrix multiplication algorithm with complexity $O(n^\omega\rho f(\log n\log\rho))$, where ω is, as before, the matrix multiplication exponent, $2 < \omega \leqslant 3$, ρ is again denotes the maximum degree of elements of these matrices, and f is a polynomial (see [28]). However, most likely, a matrix inversion algorithm with such complexity does not exist.

However, what was said above is only a hypothesis. To the author's knowledge, there are no algorithms, for example, for unimodularity recognition with complexity lower than that in Theorem 2(i). At least, a search through the literature has not revealed such an algorithm, but, of course, its existence is not excluded. For example, on the basis of the ideas underlying algorithms for fast matrix multiplication over a field (see [29–31]) and the idea underlying algorithms for fast multiplication of scalar linear operators (see [19, 32, 33]), an algorithm for fast multiplication of operator matrices can be proposed and then a corresponding algorithm for unimodularity recognition can be obtained.

Numerous recent works (see, e.g., [18, 22]) have been aimed to determine the growth of coefficients belonging to $\mathbb{K}$when, for example, $\mathbb{K} = \mathbb{Q}(x)$. It would be of interest to investigate the bit complexity of unimodularity recognition algorithms. Another approach is to treat complexity as a function of three variables: n, d, and ρ, where ρ is such that all polynomials involved in L as numerators and denominators of elements' coefficients have degrees at most ρ; moreover, for fixed n, d, and ρ, the complexity in the worse-case is the number of operations in $\mathbb{Q}$.

ACKNOWLEDGMENTS

The author is grateful to the reviewer for advice and remarks concerning the first version of this paper.

This work was supported by the Russian Foundation for Basic Research, project no. 16-01-001174.

REFERENCES

1. S. A. Abramov, "On the differential and full algebraic complexities of operator matrices transformations," Lect. Notes Comput. Sci. **9890**, 1–14 (2016).
2. M. van der Put and M. F. Singer, *Galois Theory of Differential Equations* (Springer, Heidelberg, 2003).
3. A. G. Khovanskii, *Topological Galois Theory: Solvability and Unsolvability of Equations in Finite Terms* (MTsNMO, Moscow, 2008; Springer, Berlin, 2014).
4. M. van der Put and M. F. Singer, *Galois Theory of Difference Equations* (Springer, Heidelberg, 1997).
5. S. Abramov and M. Barkatou, "On the dimension of solution spaces of full rank linear differential systems," Lect. Notes Comput. Sci. **8136**, 1–9 (2016).
6. S. A. Abramov, "On inversion of difference operator matrices," in *Differential Equations and Related Mathematical Issues: Proceedings of 8th Oka Scientific Conference, Konstantinovo, Kolomna, June 10–11, 2016* (2016), pp. 4–12.
7. V. G. Kats and P. Chen, *Quantum Analysis* (MTsHMO, Moscow, 2005) [in Russian].
8. G. E. Andrews, *q-Series: Their Development and Application in Analysis, Number Theory, Combinatorics, Physics, and Computer Algebra* (Pennsylvania State Univ., University Park, PA, 1986).
9. S. Abramov and M. Barkatou, "On solution spaces of products of linear differential or difference operators," ACM Commun. Comput. Alg. **4**, 155–165 (2014).
10. C. H. Franke, "Picard–Vessiot theory of linear homogeneous difference equations," Trans. Am. Math. Soc. **108**, 491–515 (1963).
11. D. E. Knuth, "Big omicron and big omega and big theta," ACM SIGACT News **8** (2), 18–23 (1976).
12. S. Abramov, "EG-Eliminations," J. Differ. Equations Appl. **5**, 393–433 (1999).
13. S. Abramov and M. Bronstein, "Linear algebra for skew-polynomial matrices," Rapport de Recherche INRIA, March 2002, RR-4420. http://www.inria.fr/RRRT/RR-4420.html.

14. S. Abramov and M. Bronstein, "On solutions of linear functional systems," in *ISSAC'2001 Proceedings* (2001), pp. 1–6.
15. S. A. Abramov and D. E. Khmelnov, "Linear differential and difference systems: EG_δ- and EG_σ-eliminations," Program. Comput. Software **39** (2), 91–109 (2013).
16. M. A. Barkatou, C. El Bacha, G. Labahn, and E. Pflugel, "On simultaneous row and column reduction of higher-order linear differential systems," J. Symbol. Comput. **49** (1), 45–64 (2013).
17. B. Beckermann and G. Labahn, "Fraction-free computation of matrix rational interpolants and matrix GCDs," SIAM J. Matrix Anal. Appl. **77** (1), 114–144 (2000).
18. H. Cheng and G. Labahn, "Fraction-free row reduction of matrices of Ore polynomials," J. Symbol. Comput. **41**, 513–543 (2006).
19. A. Benoit, A. Bostan, and J. van der Hoeven, "Quasi-optimal multiplication of linear differential operators," *Proceedings of FOCS'12* (IEEE Comput. Soc., New Brunswick, 2012), pp. 524–530.
20. T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," J. Symbol. Comput. **37** (4), 485–510 (2004).
21. J. Middeke, "A polynomial-time algorithm for the Jacobson form for matrices of differential operators," Tech. Report in RISC Report Series, No. 08-13 (2008).
22. M. Giesbrecht and KimM. Sub, "Computation of the Hermite form of a matrix of Ore polynomials," J. Algebra **376**, 341–362 (2013).
23. O. Ore, "Theory of non-commutative polynomials," Ann. Math. **34**, 480–508 (1933).
24. M. Bronstein and M. Petkovsek, "An introduction to pseudo-linear algebra," Theor. Comput. Sci. **157** (1), 3–33 (1996).
25. C.-P. Jeannerod and G. Villard, "Essentially optimal computation of the inverse of generic polynomial matrices," J. Complexity **21** (1), 72–86 (2005).
26. P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory* (Springer, Heidelberg, 1997).
27. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, Mass., 1976; Mir, Moscow, 1979).
28. A. Bostan and E. Schost, "Polynomial evaluation and interpolation on special sets of points," J. Complexity **21** (4), 420–446 (2005).
29. V. Strassen, "Gaussian elimination is not optimal," Numer. Math. **13**, 354–356 (1969).
30. D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," J. Symbol. Comput. **9** (3), 251–280 (1990).
31. V. Williams, "Multiplying matrices faster than Coppersmith–Winograd," *STOC'2012 Proceedings* (2012), pp. 887–898.
32. J. van der Hoeven, "FFT-like multiplication of linear differential operators," J. Symbol. Comput. **33**, 123–127 (2002).
33. A. Bostan, F. Chyzak, and N. Le Roix, "Products of ordinary differential operators by evaluating and interpolation," *Proceedings of ISSAC'2008* (2008), pp. 23–30.

Translated by I. Ruzanova

SPELL: OK