On Unimodular Matrices of Difference Operators

S. A. Abramov, D. E. Khmelnov

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

Abstract. We consider matrices $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ of scalar difference operators, where K is a difference field of characteristic 0 with an automorphism σ. We discuss approaches to compute the dimension of the space of those solutions of the system of equations $L(y) = 0$ that belong to an adequate extension of K. On the base of one of those approaches, we propose a new algorithm for computing $L^{-1} \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ whenever it exists. We investigate the worst-case complexity of the new algorithm, counting both arithmetic operations in K and shifts of elements of K. This complexity turns out to be smaller than in the earlier proposed algorithms for inverting matrices of difference operators.
Some experiments with our implementation in Maple of the algorithm are reported.

1 Introduction

Matrix calculus has wide application in various branches of science. Testing whether a given matrix over a field or ring is invertible and computing the inverse matrix are classical mathematical problems. Below, we consider these problems for matrices whose entries belong to the ring (non-commutative) of scalar linear difference operators with coefficients from a difference field K of characteristic 0 with an automorphism (shift) σ. We discuss some new algorithms for solving these problems. These problems can be also solved by well-known algorithms proposed originally for more general problems. The new algorithms below have lower complexity.

In the case of matrices of operators, the term "*unimodular* matrix" is usually used instead of the "invertible matrix". This term will be used throughout this paper.

In the differential case when the ground field K is a differential field of characteristic 0 with a derivation $\delta = '$ and when the matrix entries are scalar linear differential operators over K, algorithms for the unimodularity testing of a matrix and computing its inverse were considered in [2]. For a given matrix L, the algorithms discussed below 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 of K associated with L (see [16]). A matrix L of operators, when L is 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 the zero space (see [4]).

There are two significant dissimilarities between the differential and difference cases. One of them gives an advantage to the differential case, the other to the difference case. The differential system $y' = Ay$ has the n-dimensional solution space in the universal differential extension, regardless of the form (singular or non-singular) of the $n \times n$-matrix A [17]. But in the difference case, the non-singularity of

A is required. However, the difference case has the advantage that the automorphism σ has the inverse in $K[\sigma, \sigma^{-1}]$, while the differentiation δ is not invertible in $K[\delta]$.

It is worthy to note that some algorithms for solving the "difference problems" formulated above have been proposed in [3]. The algorithms below have lower complexity (this is the novelty of the results) due to the usage of the EG-eliminations algorithm [1, 6, 7] as an auxiliary tool instead of the algorithm Row-Reduction [11]. The obstacle for such a replacement in the differential case, is the absence of the inverse element for δ in the ring $K[\delta]$.

The problems of unimodularity testing and inverse matrix construction can be solved by applying various other algorithms. For example, the Jacobson and Hermite forms of the given operator matrix can be constructed; their definitions can be found in [13, 15]. The complexity of the algorithms is greater than the complexity of the algorithms in this paper and in [3]. Of course, the algorithms in [13, 15] are intended for more general problems, and the algorithms in this paper and in [3] have advantages only for unimodularity recognition and the construction of an inverse operator matrix.

We use the following notation. 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 $1 \leqslant i \leqslant n$ is the $1 \times n$-matrix equal to the ith row of M. The diagonal $n \times n$-matrix with diagonal elements $r_1, \ldots, r_n$ is denoted by $\mathrm{diag}(r_1, \ldots, r_n)$, and I_n is the $n \times n$ identity matrix.

The proposed algorithms are presented in Section 3. Their implementation in Maple and some experiments are described in Section 5.

2 Preliminaries

2.1 Adequate Difference Extensions

As usual, a *difference ring* K is a commutative ring with identity and an automorphism σ (which will frequently be referred to as a *shift*). If K is additionally a field, then it is called a *difference field*. We will assume that the considered difference fields are of characteristic 0. The *ring of constants* of a difference ring K is $\mathrm{Const}\,(K) = \{c \in K \mid \sigma c = c\}$. If K is a difference field, then $\mathrm{Const}\,(K)$ is a subfield of K. Let K be a difference field with an automorphism σ, and let Λ be a difference ring extension of K (on 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 ring extension of a field K is an *adequate* difference extension of K if $\mathrm{Const}\,(\Lambda)$ is a field and an arbitrary system

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

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

The non singularity of A in this definition is essential: e.g., if the first row of A is zero, then the entry y_1 in any solution of the system (1) is zero as well.

Note that the q-difference case [10] is covered by the general difference case.

If $\mathrm{Const}\,(K)$ 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 $\mathrm{Const}\,(\Omega) = \mathrm{Const}\,(K)$, which is called the *universal* difference (Picard-Vessiot) ring extension of K. The complete proof of its existence is not easy (see [16, Sect. 1.4]), while the existence of an adequate difference extension for an arbitrary difference field can be rather easily proved (see [5, Sect. 5.1]). However, it should

be emphasized that, for an adequate extension, the equality $\mathrm{Const}\,(\Lambda) = \mathrm{Const}\,(K)$ is not guaranteed; in the general case, $\mathrm{Const}\,(K)$ 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 [12] is the scalar equation over a field with an algebraically closed field of constants. This equation has no nontrivial solutions in any difference extension having an algebraically closed field of constants.

In the sequel, Λ denotes a fixed adequate difference extension of a difference field with an automorphism σ.

2.2 Orders of difference operators

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

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

and the *order* of f is defined as

$$\mathrm{ord}\, f = \overline{\mathrm{ord}}\, f - \underline{\mathrm{ord}}\, f.$$

Set $\overline{\mathrm{ord}}\, 0 = -\infty$, $\underline{\mathrm{ord}}\, 0 = \infty$, and $\mathrm{ord}\, 0 = -\infty$.

For a finite set F of scalar operators (a vector, matrix, matrix row etc), $\overline{\mathrm{ord}}\, F$ is defined as the maximum of the leading orders of its elements; $\underline{\mathrm{ord}}\, F$ is defined as the minimum of the trailing orders of its elements; finally, $\mathrm{ord}\, F$ is defined as $\overline{\mathrm{ord}}\, F - \underline{\mathrm{ord}}\, F$. A matrix of difference operators is a matrix from $\mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$. In the sequel, such a matrix of difference operators is associated with some matrices belonging to $\mathrm{Mat}_n(K)$. To avoid confusion of terminology, matrices of difference operators 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 $K[\sigma, \sigma^{-1}]$. Same-length rows $u_1, \ldots, u_s$ with components belonging to $K[\sigma, \sigma^{-1}]$ are called *linearly dependent* (over $K[\sigma, \sigma^{-1}]$) if there exist $f_1, \ldots, f_s \in K[\sigma, \sigma^{-1}]$ not all zero such that $f_1 u_1 + \cdots + f_s u_s = 0$; otherwise, these rows are called *linearly independent* (over $K[\sigma, \sigma^{-1}]$). If

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

and L is nonzero, then it can be represented in the expanded form as

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

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

Let the leading and trailing 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

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

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

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

If $\alpha_i = -\infty$ (resp. $\beta_i = \infty$) then the i-th row of P (resp. Q) is zero.

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

Definition 2 An operator $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ is *unimodular* or *invertible* if there exists an inverse $L^{-1} \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$: $LL^{-1} = L^{-1}L = I_n$. The group of unimodular $n \times n$-operators is denoted by Υ_n. Two operators are said to be *equivalent* if $L_1 = UL_2$ for some $U \in \Upsilon_n$.

If L has a zero row (in such a case, its frontal and rear matrices have also zero rows) then L is not of full rank, and is not unimodular: suppose, e.g., that the first row of L is zero, then for any $M \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$, the first row of the product LM is also zero, thus, the equality $LM = I_n$ is impossible. Similarly, if $U \in \Upsilon_n$ and UL has a zero row then $L \notin \Upsilon_n$.

Let V_L denote the space of the solutions of the system $L(y) = 0$ that belong to Λ^n (see Section 2.1). For brevity, V_L is sometimes called the solution space of L.

For the difference case, Theorem 1 from [2] can be reformulated as follows.

Theorem 1 *Let $L \in Mat_n(K[\sigma, \sigma^{-1}])$ be of full rank. Then*

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

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

The proof is based on [4, 5].

2.3 Complexity

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

Thus, we will consider two complexities. 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 in the worst case.

We can also consider the *full algebraic complexity* as *the total number of all operations* in the worst case.

Supposing that $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$, $\mathrm{ord}\, L = d$, each of the mentioned complexities is a function of n and d.

In asymptotic complexity estimates, along with the O notation we use the Θ notation (see [14]): the relation $f(n, d) = \Theta(g(n, d))$ is equivalent to

$$f(n, d) = O(g(n, d)) \text{ and } g(n, d) = O(f(n, d)).$$

Note that the full complexity of an algorithm counting operations of two different types in the worst case is not, in general, equal to the sum of two complexities, counting operations of the first and, resp. second type. We can claim only that the full complexity does not exceed that sum. If for the first and second complexities we have asymptotic estimates $\Theta(f(n, d))$ and $\Theta(g(n, d))$ then for the full complexity we have the estimate $O(f(n, d) + g(n, d))$. To this we can add that if for the first and second complexities we have estimates $O(f(n, d))$ and $O(g(n, d))$ then we have the estimate $O(f(n, d) + g(n, d))$ for the full complexity.

2.4 EG-eliminations (Family of EG-algorithms)

Definition 3 Let the ith row of the frontal matrix of $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ be non-zero and have the form

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

$0 \leqslant k \leqslant n$, $a \neq 0$. In this case, k is the *indent* of the ith row of L.

The algorithm $\mathtt{EG}_\sigma$ (the version published in [1]) is as follows:

`Algorithm:` $\mathtt{EG}_\sigma$
`Input:` An operator $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ whose leading matrix has no zero row.
`Output:` An equivalent operator having an upper triangle leading matrix (that operator is also denoted by L) or the message "is not of full rank".
`while` L has rows with equal indents `do`

(Reduction) Let some rows r_1, r_2 of L have the same indent k. Then compute $v \in K$ such that the indent of the row

$$r = r_1 - vr_2 \tag{3}$$

is greater than k or $\underline{\mathrm{ord}}\, r < \underline{\mathrm{ord}}\, L$ (the computation of v uses one arithmetic operation); `if` r is zero row of L `then` STOP with the message "is not of full rank"`fi`; The row from r_1, r_2 which has the smaller trailing order, must be replaced by r (if $\underline{\mathrm{ord}}\, r_1 = \underline{\mathrm{ord}}\, r_2$ then any of r_1, r_2 can be taken for the replacement);

(Shift) If $\overline{\mathrm{ord}}\, r < \overline{\mathrm{ord}}\, L$ then apply $\sigma^{\overline{\mathrm{ord}}\, L - \overline{\mathrm{ord}}\, r}$ to r in L
`od`;
Return L. □

Thus, each step of the algorithm EG_σ is a combination "reduction + shift". All the steps are unimodular since the operator σ^{-1} is the inverse for σ.

Example 1

$$L = \begin{pmatrix} 1 & -\frac{1}{x}\sigma \\ \frac{x^2}{2} & -\frac{x}{2}\sigma + 1 \end{pmatrix} = \begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & -\frac{x}{2} \end{pmatrix} \sigma + \begin{pmatrix} 1 & 0 \\ \frac{x^2}{2} & 1 \end{pmatrix}. \tag{4}$$

By applying the algorithm $\mathtt{EG}_\sigma$, the operator L is transformed as follows:

$$\begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & -\frac{x}{2} \end{pmatrix} \sigma + \begin{pmatrix} 1 & 0 \\ \frac{x^2}{2} & 1 \end{pmatrix} \xrightarrow{1} \begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & 0 \end{pmatrix} \sigma + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \xrightarrow{2}$$
$$\begin{pmatrix} 0 & -\frac{1}{x} \\ 0 & 1 \end{pmatrix} \sigma + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{3} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \sigma + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{4} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \sigma + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Here

$$\begin{aligned} &1.\ L_{2,*} := \frac{-x^2}{2} L_{1,*} + L_{2,*}, \\ &2.\ L_{2,*} := \sigma L_{2,*}, \\ &3.\ L_{1,*} := L_{1,*} + \frac{1}{x} L_{2,*}, \\ &4.\ L_{1,*} := \sigma L_{1,*}. \end{aligned} \tag{5}$$

As the result of this transformation, we obtain the operator $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \sigma$, i.e.,

$$\begin{pmatrix} \sigma & 0 \\ 0 & \sigma \end{pmatrix}. \tag{6}$$

By analogy with $\mathtt{EG}_\sigma$ we can propose an algorithm $\mathtt{EG}_{\sigma^{-1}}$ in which the trailing matrix of the operator is considered instead of its leading matrix.

Proposition 1 *The arithmetic complexity of the algorithms* $\mathtt{EG}_\sigma$, $\mathtt{EG}_{\sigma^{-1}}$ *is*

$$\Theta(n^3d^2), \tag{7}$$

the shift complexity is

$$\Theta(n^2d^2). \tag{8}$$

Correspondingly, the full algebraic complexity is

$$O(n^3d^2). \tag{9}$$

See [3, Sect. 5.4] for the proof.

3 Unimodularity Testing, Computing Inverse Operator

3.1 Unimodularity Testing

Proposition 2 *Let the rear matrix of an operator* $L \in Mat_n(K[\sigma,\sigma^{-1})$ *be non-singular. Then applying* EG_σ *to* L *gives an operator having a non-singular rear matrix.*

Proof. Let us prove that one step of EG_σ does not change the determinant of the rear matrix of L. Indeed, let the reduction stage of this step change a row r_1 of L and before this step, we have $\underline{\mathrm{ord}}\, r_1 = \beta$. The row r_1 is replaced by a sum of r_1 and another row r_2, multiplied by $v \in K$: $r_1 := r_1 + vr_2$. The inequality $\underline{\mathrm{ord}}\, r_2 \geqslant \beta$ holds. If $\underline{\mathrm{ord}}\, r_2 > \beta$ then the rear matrix gets no change. If $\underline{\mathrm{ord}}\, r_2 = \beta$ then the determinant of the rear matrix gets no change since the shift stage does not change the rear matrix.

The following algorithm can be verified by means of Theorem 1 and Proposition 2:

`Algorithm: Unimodularity testing` (this algorithm has been described in [3])
`Input:` An operator $L \in \mathrm{Mat}_n(K)$.
`Output:` "is unimodular" or "is not unimodular" depending on whether L is unimodular or not.
`if` $\mathtt{EG}_{\sigma^{-1}}$ did not find that L is not of full rank and $\underline{\mathrm{ord}}\, r = \overline{\mathrm{ord}}\, r$ for each row r of $\mathtt{EG}_\sigma(\mathtt{EG}_{\sigma^{-1}}(\mathtt{L}))$ `then` Return "is unimodular" `otherwise` Return "is not unimodular" `fi`. □

Example 2 Let L be again as in Example 1, i.e., of the form (4). The rear matrix coincides with the trailing one, and is nonsingular. By applying the algorithm $\mathtt{EG}_\sigma$, the operator L is transformed to $\tilde{L}$ of the form (6). We have $\dim V_{\tilde{L}} = 0$. Thus, the original operator L is unimodular.

Proposition 3 *The arithmetic, shift and full algebraic complexities of the algorithm* `Unimodularity testing` *are, resp. (7), (8), and (9).*

Proof. This follows from Proposition 1 and the fact that the values of n, d are not increased after applying $\mathtt{EG}_{\sigma^{-1}}$.

3.2 Inverse Operator

`Algorithm: ExtEG`$_\sigma$
`Input:` Operators $J, L \in \mathrm{Mat}_n(K)$.
`Output:` The operator $M = UJ$, where U is such that $\mathtt{EG}_\sigma(L) = UL$.
Apply $\mathtt{EG}_\sigma$ to L, and repeat in parallel the application of all the operations to J.□

Note that in the case when we use I_n as J, we obtain M which is equal to U.

By analogy with $\mathtt{ExtEG}_\sigma$ we can propose an algorithm $\mathtt{ExtEG}_{\sigma^{-1}}$ in which the trailing matrix of the operator is considered instead of its leading matrix.

Proposition 4 *We have* $\operatorname{ord} U \leqslant nd$ *on each step of applying of* $\mathtt{ExtEG}_\sigma$ *to* $L \in Mat_n(K[\sigma, \sigma^{-1}])$, $\operatorname{ord} L = d$.

Proof. If in a step of the algorithm the shift σ^k of a row r was performed, then the order of U will be increased by no more than $|k|$, while the order of the shifted row is decreased by $|k|$. This implies that $\operatorname{ord} U$ after any step of ExtEG_σ does not exceed the sum of the orders of all rows of L. The order of each row does not exceed d and the sum of the orders of all rows of L does not exceed nd.

Proposition 5 *Both arithmetic and shift complexities of each of the algorithms* $\mathtt{ExtEG}_\sigma$, $\mathtt{ExtEG}_{\sigma^{-1}}$ *can be estimated by* $O(n^4d^2)$. *The full complexity is* $O(n^4d^2)$ *as well.*

Proof. When one applies $\mathtt{EG}_\sigma$ or $\mathtt{EG}_{\sigma^{-1}}$ to $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$, $\operatorname{ord} L = d$, then the operation (3) is performed at most $n \cdot nd$ times. By Proposition 4, when we compute U, each operation (3) uses at most $O(n \cdot nd)$ arithmetic operations, i.e., $O(n^2d)$ arithmetic operations. Totally, the number of arithmetic operations is $O(n^2d \cdot n^2d)$, i.e. $O(n^4d^2)$.

The shift complexity of each of $\mathtt{EG}_\sigma$, $\mathtt{EG}_{\sigma^{-1}}$ is $O(n^2d^2)$. When we substitute nd for d (by Proposition 4) we obtain $O(n^4d^2)$.

The estimate $O(n^4d^2)$ for the full complexity follows from the obtained estimates for the arithmetic and shift complexities.

`Algorithm: Inverse operator`
`Input:` An operator $L \in \mathrm{Mat}_n(K)$.
`Output:` The inverse of L or the message "is not unimodular".
$U := I_n$; $(U, L) := \mathtt{ExtEG}_{\sigma^{-1}}(U, L)$; $(U, L) := \mathtt{ExtEG}_\sigma(U, L)$;
`if` $\underline{\operatorname{ord}}\, r \neq \overline{\operatorname{ord}}\, r$ for at least one row r of L `then` STOP with the message "is not unimodular"
`fi`;
Let $\beta_1, \ldots, \beta_n$ be the trailing orders of rows of L,
thus $L = MD$ with $M \in \mathrm{Mat}_n(K)$, $D = \operatorname{diag}(\sigma^{\beta_1}, \ldots, \sigma^{\beta_n})$;
$L^{-1} := \operatorname{diag}(\sigma^{-\beta_1}, \ldots, \sigma^{-\beta_n})M^{-1}U$. □

Example 3 Consider again operator (4). To find L^{-1} after getting the operator $\tilde{L}$ (as it was shown in Example 2), we, first, apply (5) to I_2. We get

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \xrightarrow{1} \begin{pmatrix} 1 & 0 \\ -\frac{x^2}{2} & 1 \end{pmatrix} \xrightarrow{2} \begin{pmatrix} 1 & 0 \\ -\frac{(x+1)^2}{2}\sigma & \sigma \end{pmatrix} \xrightarrow{3}$$

$$\begin{pmatrix} 1 - \frac{(x+1)^2}{2x}\sigma & \frac{1}{x}\sigma \\ -\frac{(x+1)^2}{2}\sigma & \sigma \end{pmatrix} \xrightarrow{4} \begin{pmatrix} \sigma - \frac{(x+2)^2}{2(x+1)}\sigma^2 & \frac{1}{x+1}\sigma^2 \\ -\frac{(x+1)^2}{2}\sigma & \sigma \end{pmatrix}.$$

We get

$$L^{-1} = \mathrm{diag}(\sigma^{-1}, \sigma^{-1})\, I_2^{-1} \begin{pmatrix} \sigma - \frac{(x+2)^2}{2(x+1)}\sigma^2 & \frac{1}{x+1}\sigma^2 \\ -\frac{(x+1)^2}{2}\sigma & \sigma \end{pmatrix} = \begin{pmatrix} 1 - \frac{(x+1)^2}{2x}\sigma & \frac{1}{x}\sigma \\ -\frac{x^2}{2} & 1 \end{pmatrix}.$$

Proposition 6 *The arithmetic, shift, and full complexities of the algorithm* `Inverse operator` *can be estimated by* $O(n^4d^2)$.

Proof. The statement follows from Proposition 5.

4 Other Versions of `EG` and `Inverse operator`

The algorithm `Inverse operator` proposed in this paper is based on the version [1] of the EG-eliminations algorithm as an auxiliary tool. Another variant of the algorithm for constructing the inverse operator has been proposed in [3], it is based on a version (named `RR` in [3]) of the Row-Reduction algorithm [11] as an auxiliary tool. For a given operator, the algorithm $\mathtt{RR}_\sigma$ constructs an equivalent operator that has a nonsingular frontal matrix. Similarly, the algorithm $\mathtt{RR}_{\sigma^{-1}}$ constructs an equivalent operator that has a nonsingular rear matrix. The arithmetic complexity of the algorithms presented in this paper and, resp. in [3], is the same, however, the shift complexity (and, hence, the full algebraic complexity) of the new algorithm is lower: $O(n^4d^2)$ instead of $\Theta(n^4d^3)$.

Some other versions of the algorithms belonging to the EG-eliminations family ([6, 7]), whose full complexity does not differ much from the full complexity of the above considered version, can be to some extent more convenient for implementation. This question has been discussed in [8]. In our Maple-implementation of the `Inverse operator` algorithm represented below, we use elements of various variants of EG-eliminations. (It is well known that an algorithm that looks the best in terms of complexity theory is not necessarily the best in computational practice.)

5 Implementation and Experiments

The implementation[1] is performed in Maple [18]. The existing implementation of the algorithm `EG` described in [9] is taken as a starting point. The procedure is adjusted to the difference case and to provide extended versions, both $\mathtt{ExtEG}_\sigma$ and $\mathtt{ExtEG}_{\sigma^{-1}}$. On top of the procedure for $\mathtt{ExtEG}_\sigma$ and $\mathtt{ExtEG}_{\sigma^{-1}}$, the procedure `IsUnimodular` to test the unimodularity of an operator and to compute its inverse is implemented.

An operator $L = A_l\sigma^l + A_{l-1}\sigma^{l-1} + \cdots + A_t\sigma^t$ is specified at the input of the procedures as the list

$$[A, l, t], \tag{10}$$

where A is an *explicit matrix*

$$A = (A_l|A_{l-1}|\ldots|A_t) \tag{11}$$

of size $n \times n(l-t+1)$. The explicit matrix A is defined by means of the standard Maple object `Matrix`. The entries of the explicit matrix are rational functions of one variable, which are also specified in a standard way accepted in Maple. If $t = 0$ then the input may be given alternatively just by the explicit matrix A.

The procedure `IsUnimodular` returns `true` or `false` as the result of checking the unimodularity of the given operator, its inverse operator is returned additionally being assigned to a given variable name (an optional input parameter of the

[1] Available at http://www.ccas.ru/ca/egrrext

procedure). The inverse operator is also represented by the list of its explicit matrix and its leading and trailing orders. If the optional variable name is not given, then the procedure uses the algorithm `Unimodularity Testing` from the section 3.1, otherwise the algorithm `Inverse Operator` from the section 3.2 is used.

Example 4 We apply the procedure `IsUnimodular` to the operator matrix (4) considered in Examples 1–3. The explicit matrix for the operator is

$$\begin{pmatrix} 0 & -\frac{1}{x} & 1 & 0 \\ 0 & -\frac{x}{2} & \frac{x^2}{2} & 1 \end{pmatrix},$$

with $l = 1$ and $t = 0$. The procedure is applied twice: first time just for checking the unimodularity, and the second time, for computing the inverse operator as well. One can see that the result of the application coincides with the result presented in Example 3 (the computation time is also presented):

```
> L := Matrix([[0, -1/x, 1, 0], [0, -x/2, x^2/2, 1]]);
```

$$L := \begin{bmatrix} 0 & -\frac{1}{x} & 1 & 0 \\ 0 & -\frac{x}{2} & \frac{x^2}{2} & 1 \end{bmatrix}$$

```
> st:=time(): IsUnimodular(L, x); time()-st;
```

$$true$$

$$0.032$$

```
> st:=time(): IsUnimodular(L, x, 'InvL'); time()-st;
```

$$true$$

$$0.063$$

```
> InvL;
```

$$\left[\begin{bmatrix} -\frac{(x+1)^2}{2x} & \frac{1}{x} & 1 & 0 \\ 0 & 0 & -\frac{x^2}{2} & 1 \end{bmatrix}, 1, 0 \right]$$

Example 5 Consider the operator

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

The explicit matrix for the operator is

$$\begin{pmatrix} 0 & 0 & 0 & -\frac{1}{x-1} & 1 & 0 \\ 0 & -\frac{x}{2} & \frac{x^2}{2} & 1 & 0 & 0 \end{pmatrix}$$

with $l = 1$ and $t = -1$. The procedure `IsUnimodular` is applied twice again: first time just for checking the unimodularity, and the second time, for computing the inverse operator as well. The computation time is also presented.

```
> L:= Matrix([[0, 0, 0,-1/(x-1), 1, 0], [0, -x/2, x^2/2, 1, 0, 0]]);
```

$$L := \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{x-1} & 1 & 0 \\ 0 & -\frac{x}{2} & \frac{x^2}{2} & 1 & 0 & 0 \end{bmatrix}$$

```
> st:=time(): IsUnimodular([L, 1, -1], x); time()-st;
```

$$true$$

$$0.078$$

```
> st:=time(): IsUnimodular([L, 1, -1], x, 'InvL'); time()-st;
```

$$true$$

$$0.109$$

```
> InvL;
```

$$\left[\begin{bmatrix} -\frac{(x+1)^2}{2x} & 0 & 1 & \frac{1}{x} & 0 & 0 \\ 0 & 0 & -\frac{x^2}{2} & 0 & 0 & 1 \end{bmatrix}, 2, 0 \right]$$

It means that

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

In addition, a series of experiments has been executed.

Example 6 Consider the following $n \times n$-operator with $n = 2k$

$$M = \begin{pmatrix} I_k & A \\ 0_k & I_k \end{pmatrix}, \tag{12}$$

where 0_k is the zero $k \times k$-matrix, $A \in \mathrm{Mat}_k(K[\sigma, \sigma^{-1})$ is an arbitrary operator. The operator (12) is unimodular for any A, its inverse operator is

$$M^{-1} = \begin{pmatrix} I_k & -A \\ 0_k & I_k \end{pmatrix}. \tag{13}$$

For each experiment, we have generated an operator A whose entries are scalar difference operators having random rational function coefficients with the numerators and denominators of the degree up to 2. We compute the inverse for M of the form (12). The order of A, and hence, the order of M varies as $d = 1, 2, 4, 6, 8, 10$ and the number of rows of M varies as $n = 4, 6, 8, 10$ (hence, the number of rows of A varies as $k = 2, 3, 4, 5$). The inverse of M is calculated in each experiment by `IsUnimodular`. The results are presented in Table 1.

The table shows that the computation time in general corresponds to the complexity estimates (it should not be exact since the estimates are for the worst case and asymptotical). However, the computing time starts to increase faster than expected with the growth of n and d. It is again caused by the significant growth of the size of the elements of the matrix in the course of the computation. The size of the elements in M^{-1} is equal to the size of the elements in M in these experiments, so the coefficients of the elements are rational functions with the numerators and denominators of the degree up to 2. But in the course of the computation, the elements of the matrix have coefficients with the numerators and denominators of the degree up to several dozens for the smaller n and d, and up to several hundreds and even more than a thousand for the greater n and d.

Table 1. Results of the experiments, in seconds

	d=1	d=2	d=4	d=6	d=8	d=10
n=4	0.125	0.188	0.500	0.969	2.078	2.906
n=6	0.282	0.593	1.734	6.563	79.375	92.562
n=8	0.516	1.500	37.938	94.813	427.375	1836.547
n=10	0.703	5.562	910.218	1006.797	7576.063	13372.172

6 Conclusion

In this paper, we have presented some new algorithms for solving problems for matrices whose entries belong to the non-commutative ring of scalar linear difference operators with coefficients from a difference field K of characteristic 0 with an automorphism σ. Some algorithms for solving the difference problems formulated in the paper had been proposed in [3]. The algorithms in the present paper have lower complexity due to the usage of the EG-eliminations algorithm as an auxiliary tool instead of Row-Reduction algorithm. The implementation of the algorithm in Maple was done and some experiments were reported. The experimental results show that the computation time corresponds in general to the complexity estimates from the proposed theory.

From our work, new questions arise (they were earlier formulated in [3]). For example, it is not clear, whether the problem of inverting can be reduced to the matrix multiplication problem (similarly to the "commutative" case)?

One more question: whether there exists an algorithm for such $n \times n$-matrices inverting with the full complexity $O(n^a d^b)$, with $a < 3$? It is possible to prove by the usual way that the matrix multiplication can be reduced to the problem of the matrix inverting (we have in mind the difference matrices). However, it is not so easy to prove that the problem of the matrix inverting can be reduced to the problem of the the matrix multiplication.

We will continue to investigate this line of enquiry.

Acknowledgments

The authors are thankful to anonymous referees for useful comments.

References

1. Abramov, S.: EG–Eliminations. J. Difference Equations Appl., **5**, 393–433 (1999)
2. Abramov, S.: On the Differential and Full Algebraic Complexities of Operator Matrices Transformations. In: Gerdt, V.P. et al. (eds.) CASC 2016, LNCS, vol. 9890, pp. 1–14. Springer, Heidelberg (2016)
3. Abramov, S.: Inverse linear difference operators. Comput. Math. Math. Phys. **57**, 1887–1898 (2017)
4. 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)
5. Abramov, S., Barkatou, M.: On solution spaces of products of linear differential or difference operators. ACM Comm. in Computer Algebra **4**, 155–165 (2014)
6. Abramov, S., Bronstein, M.: On solutions of linear functional systems. In: ISSAC'2001 Proc., pp. 1–6 (2001)
7. 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` (2002)

8. Abramov, S.A., Glotov, P.E., Khmelnov, D.E.: A scheme of eliminations in linear recurrent systems and its applications. Transactions of French-Russian Lyapunov Institute **3**, 78–89 (2001)
9. 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)
10. Andrews, G.E.: q-Series: their development and application in analysis, number theory, combinatorics, physics, and computer algebra. Pennsylvania: CBMS Regional Conference Series, AMS, R.I., vol. 66 (1986)
11. Beckermann, B., Cheng, H., Labahn, G.: Fraction-free row reduction of matrices of Ore polynomials. J. Symb. Comput. **41**, 513–543 (2006)
12. Franke, C.H.: Picard–Vessiot theory of linear homogeneous difference equations. Trans. Amer. Math. Soc. **108**, 491–515 (1986)
13. Giesbrecht, M., Sub Kim, M.: Computation of the Hermite form of a matrix of Ore Polynomials. J. Algebra **376**, 341–362 (2013)
14. Knuth, D.E.: Big omicron and big omega and big theta. ACM SIGACT News **8**(2), 18–23 (1976)
15. 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)
16. van der Put, M., Singer, M.F.: Galois Theory of Difference Equations. LNM, vol. 1666. Heidelberg: Springer (1997)
17. van der Put, M., Singer, M.F.: Galois Theory of Linear Differential Equations. Grundlehren der mathematischen Wissenschaften, 328. Springer, Heidelberg (2003)
18. Maple online help: https://www.maplesoft.com/support/help/