ISSN 0361-7688, Programming and Computer Software, 2017, Vol. 43, No. 2, pp. 67–74. © Pleiades Publishing, Ltd., 2017.
Original Russian Text © S.A. Abramov, A.A. Ryabenko, D.E. Khmelnov, 2017, published in Programmirovanie, 2017, Vol. 43, No. 2.

Revealing Matrices of Linear Differential Systems of Arbitrary Order

S. A. Abramov*, A. A. Ryabenko, and D. E. Khmelnov*****

Dorodnicyn Computing Center, Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, ul. Vavilova 40, Moscow, 119333 Russia

*E-mail: *sergeyabramov@mail.ru, **anna.ryabenko@gmail.com, ***dennis_khmelnov@mail.ru*

Received September 30, 2016

Abstract—If the leading matrix of a linear differential system is nonsingular, then its determinant is known to bear useful information about solutions of the system. Of interest is also the frontal matrix. However, each of these matrices (we call them revealing matrices) may occur singular. In the paper, a brief survey of algorithms for transforming a system of full rank to a system with a nonsingular revealing matrix of a desired type is given. The same transformations can be used to check whether the rank of the original system is full. A Maple implementation of these algorithms (package EGRR) is discussed, and results of comparison of estimates of their complexity with actual operation times on a number of examples are presented.

DOI: 10.1134/S0361768817020025

1. INTRODUCTION

Systems of linear ordinary differential equations arise in many fields of mathematics. One of the goals of computer algebra is the development of new algorithms for searching solutions of systems of differential equations, as well as algorithms used as auxiliary ones in such search or as an integral part of known algorithms of such search. The leading and frontal matrices (these concepts are introduced and discussed in Sections 2.1 and 2.2), if nonsingular, allow one, for exasmple, to determine possible singular points of system solutions, since the set of roots of the determinants of either matrix includes all singular points. However, the determinant of either of these two matrices (we call them *revealing* matrices) may turn equal to zero.

In this paper, we present a brief survey of algorithms for transforming a full-rank system to that with a nonsingular revealing matrix of the desired type. Various algorithms are first compared to each other by analyzing their complexities in terms of the number of operations in the worst case. Note that the differentiation operation is also taken into account (it is shown in [1] that algorithms having identical complexities in terms of the number of arithmetic operations can have different complexities in terms of the total number of operations, which includes differentiation).

Implementation of considered algorithms is discussed, and results of experimental comparison are presented. It is established that, for each algorithm, examples can be found for which this algorithm works faster than others (this does not contradict the conclusions of the complexity analysis, since not all the cases are the "worst" ones in the sense meant in the complexity analysis; moreover, the complexity in terms of the number of operations does not take into account the sizes of the operands, so that comparison of algorithms based on the complexity discussed has preliminary and tentative character). Based on some criteria, the user may select one or another algorithm; such a possibility may be quite useful when a system cannot be solved in a reasonable time or when it is not clear how to manage the system at all.

2. PRELIMINARIES

2.1. Systems and Operators

If R is a ring (in particular, field), then $\mathrm{Mat}_m(R)$ denotes the ring of $m \times m$-matrices with entries from R. M^T denotes a transposed matrix M, and $M_{i,*}$, $1 \le i \le m$, denotes the $1 \times m$-matrix coinciding with the ith row of the $m \times m$-matrix M.

Let $\mathbb{K}$ be a differential field of characteristic 0 with the derivative $\partial = '$. We consider systems of the form

$$A_r\partial^r y + A_{r-1}\partial^{r-1} y + \cdots + A_0 y = 0, \qquad (1)$$

where $y = (y_1, y_2, \ldots, y_m)^T$ is a vector of unknown functions of x. As for the matrices

$$A_0, A_1, \ldots, A_r,$$

we assume that $A_i \in \mathrm{Mat}_m(\mathbb{K})$, $i = 0,1,\ldots,r$, with A_r (the *leading* matrix of the system) being nonzero.

The entries of matrices A_i are called *coefficients of system* (1).

System (1) be written as $L(y) = 0$, where operator L has the form

$$A_r\partial^r + A_{r-1}\partial^{r-1} + \cdots + A_0, \quad (2)$$

and r is the *order* of the operator L (notation $r = \operatorname{ord} L$).

For $m = 1$, (2) is a *scalar* operator in the form of a polynomial of ∂. Such operators are added and subtracted as ordinary polynomials; in the multiplication, the commutative rule $\partial a = a\partial + a'$ (or, which is the same, $\partial a = a\partial + \partial(a)$) is used. The product is actually a composition of the operators. To denote the ring of these scalar operators, the standard notation $\mathbb{K}[\partial]$ of the ring of polynomials of ∂ over $\mathbb{K}$ is used. The concept of order is naturally extended to the case of scalar operators, i.e., elements of ring $\mathbb{K}[\partial]$.

Operator (2) can be represented by a single operator matrix belonging to $\mathrm{Mat}_m(\mathbb{K}[\partial])$:

$$\begin{pmatrix} L_{11} & \ldots & L_{1m} \\ \ldots & \ldots & \ldots \\ L_{m1} & \ldots & L_{mm} \end{pmatrix}, \quad (3)$$

$L_{ij} \in \mathbb{K}[\partial]$, $i, j = 1, \ldots, m$, with $\max_{i,j} \operatorname{ord} L_{ij} = r$. The order of the ith row of matrix (operator) (3) is the greatest order of the scalar operators composing this row; i.e.,

$$\operatorname{ord} L_{i,*} = \max_{j=1}^{m} \operatorname{ord} L_{i,j}.$$

We say that the rows of operator (3) with numbers $i_1, \ldots, i_s$, $s \le m$, are *linearly independent* over $\mathbb{K}[\partial]$ if, from the fact that the linear combination of the rows with the left multipliers $f_1, \ldots, f_s \in \mathbb{K}[\partial]$ is equal to zero, i.e., $f_1 L_{i_1,*} + \ldots + f_s L_{i_s,*} = 0$, it follows that $f_1 = \ldots = f_s = 0$.

The system can be defined in an operator form by using one of the two operator representations. In what follows, we select the form of the representation based on convenience considerations.

The matrix A_r is the leading matrix of the system $L(y) = 0$ and of the operator L irrespective of the form of the representation of the system and operator.

If all rows of operator (3) are linearly independent over $\mathbb{K}[\partial]$, then we say that the equations of the corresponding system are independent over $\mathbb{K}[\partial]$. In this case, the operator $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$, as well as the system $L(y) = 0$, has the *full rank*. We will also call them the operator and system *of full rank*. It is these operators and systems that are considered in this paper.

2.2. Dimension of the Solution Space

Let the field of constants $Const(\mathbb{K}) = \{c \in \mathbb{K} | \partial c = 0\}$ of the field $\mathbb{K}$ is algebraically closed. Let Λ denote a fixed *universal Picard—Vessiot extension* for $\mathbb{K}$ (see [2, Sect. 3.2]). This is some differential extension Λ of a field $\mathbb{K}$ such that $Const(\Lambda) = Const(\mathbb{K})$ and any differential system $\partial y = Ay$, $A \in \mathrm{Mat}_m(\mathbb{K}[\partial])$, has a space of solutions (in Λ^m) of dimension m over the field of constants. For an arbitrary operator L of form (3), V_L will denote a linear space of solutions of L (i.e., solutions of the system $L(y) = 0$) over $Const(\Lambda)$ that belong to Λ^m. The dimension of this space is denoted as $\dim V_L$.

Suppose that $Const(\mathbb{K})$ is not algebraically closed. For any differential field $\mathbb{K}$ of characteristic 0, there exists a differential extension that has algebraically closed field of constants: this is, for example, the algebraic closure $\overline{\mathbb{K}}$ with the derivative obtained as a natural extension of the derivative in the field $\mathbb{K}$. In this case, $Const(\overline{\mathbb{K}}) = \overline{Const(\mathbb{K})}$ (see [2, Exercises 1.5, 2: (c), (d); 3, Sect. 3]). Then, V_L is a linear over $Const(\overline{\mathbb{K}})$ space of solutions of L the components of which belong to the universal differential extension of the field $\overline{\mathbb{K}}$.

Let an operator L of full rank have form (2). If $1 \le i \le m$, then we define $\alpha_i(L)$ as the greatest integer k, $1 \le k \le r$, such that $(A_k)_{i,*}$ is a nonzero row. Thus, $\alpha_i(L) = \operatorname{ord} L_{i,*}$.

The matrix $F \in \mathrm{Mat}_m(\mathbb{K})$ such that $F_{i,*} = (A_{\alpha_i(L)})_{i,*}$, $i = 1, \ldots, m$, is referred to as the *frontal matrix* of the operator L.

The group of unimodular operators from $\mathrm{Mat}_m(\mathbb{K}[\partial])$ will be denoted as Υ_m.

Under differentiation of a row $L_{i,*}$ of operator (3) (or, which is the same, application of ∂ to this row), we mean the replacement of $(L_{i1}, \ldots, L_{i,m})$ by the row $(\partial L_{i1}, \ldots, \partial L_{i,m})$, where ∂L_{ij} is a composition (product in $\mathbb{K}[\partial]$) of the scalar operators ∂ and L_{ij}.

The theorem below, which was proved in [1], follows from the assertions proved in [4, 5] (part (iii) of this theorem can also be proved by using results of [6, Theorem III]).

Theorem 1. *Let $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ be of full rank. Then,*

(i) *if L' is the result of differentiation of one of the rows of L, then $\dim V_{L'} = \dim V_L + 1$;*

(ii) *if the frontal matrix for $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ is nonsingular, then*

$$\dim V_L = \sum_{i=1}^{m} \alpha_i(L);$$

(iii) $L \in \Upsilon_m \Leftrightarrow V_L = 0$.

In what follows, we assume that the field $\mathbb{K}$ is constructive; in particular, there exists a procedure for testing whether a given element of the field is equal to zero.

2.3. The EG-algorithm (EG-eliminations)

For a given full-rank operator $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$, the EG algorithm ([4, 7–9]) constructs some *embracing* operator matrix $\hat{L} \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ such that

- $\mathrm{ord}\,\hat{L} = \mathrm{ord}\,L$,
- $\hat{L}$ has a nonsingular leading matrix,
- $\hat{L} = QL$ for $Q \in \mathrm{Mat}_m(\mathbb{K})$, so that $V_L \subseteq V_{\hat{L}}$

If operator L is not of full rank, then the algorithm advises of this.

The algorithm constructs $\hat{L}$ on the place of the operator L; i.e., L is modified step-by-step converting gradually to $\hat{L}$.

Test if the rows of the leading matrix of the operator L are linearly dependent over $\mathbb{K}$. If they are not, then L is not modified and the algorithm stops. Otherwise, perform a series of steps consisting of two stages.

Reduction stage:

Calculate coefficients $p_1, \ldots, p_m \in \mathbb{K}$ of the leading matrix row dependence. Take some i, $1 \le i \le m$ for which $p_i \neq 0$.

Replace row $L_{i,*}$ in (3) by

$$\sum_{k=1}^{m} p_k L_{k,*}. \tag{4}$$

The ith row of the leading matrix of the operator L becomes equal to zero.

Differential shift stage:

Apply $\partial^{r-\alpha_i}$ to the ith row (the value of i was selected on the reduction stage) to get $\alpha_i = r$.

If, at some moment, there appears a zero row in L or the number of "reduction + differential shift" steps is equal to $mr+1$, then the original operator L is not of full rank. In all other cases, we obtain an operator with a nonsingular leading matrix, and the application of the algorithm will require not more than mr "reduction + differential shift" steps.

The above estimate of the number of steps is justified by the fact the reduction stage on each "reduction + differential shift" step does not change the solution space and, according to assertion (i) of Theorem 1, the "differential shift" step increases the dimension of the solution space due to differentiations. At the same time, the dimension of the solution space on any step cannot exceed mr. Thus, the total number of steps of the EG algorithm cannot exceed mr. Each reduction and differential shift step is equivalent to the multiplication of the original operator by some operator from the left; therefore, the resulting operator has the form QL, where $Q \in \mathrm{Mat}_m(\mathbb{K}[\partial])$. (If necessary, construction of matrix Q can be included in the EG algorithm.)

Complexities of the EG algorithm and some other algorithms to be discussed in Sections 2.4 and 2.5 were studied in [1]. Complexity here is meant to be the number of operations in the field $\mathbb{K}$ for fixed m and r in the worst case. The operations taken into account are both arithmetic field operations and the differentiation operation. Complexity $T_{\mathrm{EG}}(m,r)$ of the EG algorithm is shown to be

$$T_{\mathrm{EG}}(m,r) = \Theta(m^{\omega+1}r + m^3 r^2), \tag{5}$$

where ω is the exponent of the matrix multiplication, $2 < \omega \le 3$.

2.4. The RR-algorithm

This algorithm relies on the FF algorithm [10]. A simplified version of the FF algorithm is called RowReduction [11]; for brevity, we will use abbreviation RR for it in what follows.

Given an operator $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ of full rank, the RR algorithm constructs an operator $\breve{L} \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ such that

- $\mathrm{ord}\,\breve{L} \le \mathrm{ord}\,L$;
- the frontal matrix of the operator $\breve{L}$ is nonsingular;
- $\breve{L} = UL$ for some $U \in \Upsilon_m$ and, hence, $V_L = V_{\breve{L}}$.

Let us briefly describe this algorithm focusing on the construction of $\breve{L}$. The operator $\breve{L}$ is constructed on the place of the operator L; i.e., L is modified step-by-step converting gradually to $\breve{L}$.

Test if the rows of the frontal matrix of the operator L are linearly dependent over $\mathbb{K}$. If they are not, then $\breve{L} = L$ and the algorithm stops. Otherwise, let $p_1, \ldots, p_m \in \mathbb{K}$ be dependence coefficients. From the rows (3) the coefficients of which in the linear combination are not equal to zero, take the row that has the greatest order (if there are several rows that have this order, take an arbitrary one out of them). Let it be the ith row $L_{i,*}$. The row $L_{i,*}$ of the operator L is replaced by

$$\sum_{j=1}^{m} p_j \partial^{\alpha_i - \alpha_j} L_{j,*}. \tag{6}$$

After a finite number of steps, we obtain an operator with a nonsingular frontal matrix.

If, at some moment of the algorithm execution, there appears a zero row in L, then the rank of L is not full.

The termination of the algorithm is guaranteed by the fact that the sum of orders of rows of the operator L decreases on each step.

It is shown in [1, Prop. 2] that

$$T_{\mathrm{RR}}(m,r) = \Theta(m^{\omega+1}r + m^3r^3). \quad (7)$$

It should be noted that the greatest exponent of r in (5) is less than that in (7). On the other hand, $V_L = V_{\bar{L}}$, whereas, for $\hat{L}$, it may happen that $\dim V_L < \dim V_{\hat{L}}$.

2.5. The ΔEG and ΔRR algorithms

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

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

$1 \le k \le m$, $a \ne 0$. Then, the number k is referred to as the *pin-index* of the ith row of L. If the ith row of the operator L is zero, then its pin-index is considered to be equal to $-\infty$. We, however, consider the case where L has full rank.

If the rows of L have pair-wise different pin-indices, then the frontal matrix is nonsingular: up to the order of the rows, this is a triangular matrix with nonzero diagonal elements. Let rows $r_1 = L_{i,*}$ and $r_2 = L_{j,*}$ have identical pin-indices k, and let $\mathrm{ord}\, L_{i,*} = d_1$, $\mathrm{ord}\, L_{j,*} = d_2$, and $d_1 \le d_2$. Consider $v \in \mathbb{K}$ such that the row

$$r_2 - v\partial^{d_2-d_1}r_1 \quad (8)$$

either has a pin-index greater than k or the order less than d_2. The row r_2 in the operator L, i.e., $L_{j,*}$, is replaced by row (8). If L is of full rank, then, after several steps of the above-described form, the frontal matrix becomes triangular. One can use this[1] instead of searching linear dependence of rows of the frontal matrix (for EG, the leading and frontal matrices coincide at the corresponding moments and $d_2 - d_1 = 0$ in (8)).

This brings us at the new variants of the EG and RR algorithms, which will be further referred to as ΔEG and ΔRR, respectively. It has been shown in [1, Prop. 3] that

$$T_{\Delta\mathrm{EG}}(m,r) = \Theta(m^3r^2) \quad (9)$$

and

$$T_{\Delta\mathrm{RR}}(m,r) = \Theta(m^3r^3). \quad (10)$$

A similar approach (however, without estimates (9) and (10)) is discussed in [10, Sect. 9.1].

[1] For the difference case, this has already been used in [12] in the first version of the EG algorithm. In the discussion on the differential case, A. Storjohann draws attention of the first author to the fact that the complexity of this approach is less than that associated with the solution of linear algebraic systems (see also [13]).

2.6. The Case Where the Derivatives of Rows Are Stored

All results of differentiation of rows can be stored. For this case, some upper estimates of the total number of differentiations were obtained in [1].

Proposition 1 [1, Proposition 6]. The numbers of differentiations without repetitions (when the result of each differentiation is stored) performed by the RR and ΔRR algorithms are $O(mr^2)$ and $O(m^2r^2)$ in the worst case, respectively; as a result, the numbers of differentiations of elements of field $\mathbb{K}$ are $O(m^2r^3)$ and $O(m^3r^3)$, respectively.

However, the estimates $O(m^2r^3)$ and $O(m^3r^3)$ for the number of differentiations do not allow us to reduce the exponent for r in (7) and (10). Based on Proposition 1, we cannot conclude that storing results of all differentiations considerably reduces complexities of the RR and ΔRR algorithms. However, the space complexity grows considerably when we store all differentiation results.

Note that we have no arguments in favor of the conclusion that, for example, the upper estimate $O(mr^2)$ of the number of differentiations of rows by the RR algorithm is exact in one or another sense. An interesting question is whether the estimate $O(mr)$ is valid?

It is worth noting that the space complexity (the use of additional memory in the worst case) grows considerably when we store all differentiation results.

3. IMPLEMENTATION

The algorithms considered above were implemented in the computer algebra system `Maple` ([14]) as procedures of the new package `EGRR`[2] for systems whose coefficients are rational functions of one variable over a field of rational numbers.

The package includes the following procedures:

- `EG`, implements the EG algorithm,
- `RR`, implements the RR algorithm,
- `TriangleEG`, implements the ΔEG algorithm,
- `TriangleRR`, implements the ΔRR algorithm.

3.1. Input Parameters and Values Returned by the Procedures

A differential system (1) is specified at the input of the procedures of package `EGRR` as an *explicit matrix of system* $(A_r|A_{r-1}|\ldots|A_0)$ of size $m \times m(r+1)$. In turn, this system matrix is specified by means of the standard object `Matrix`. The entries of the explicit matrix are

[2] The package and a `Maple` session with examples of use of the procedures described are available at the address http://www.ccas.ru/ca/egrr.

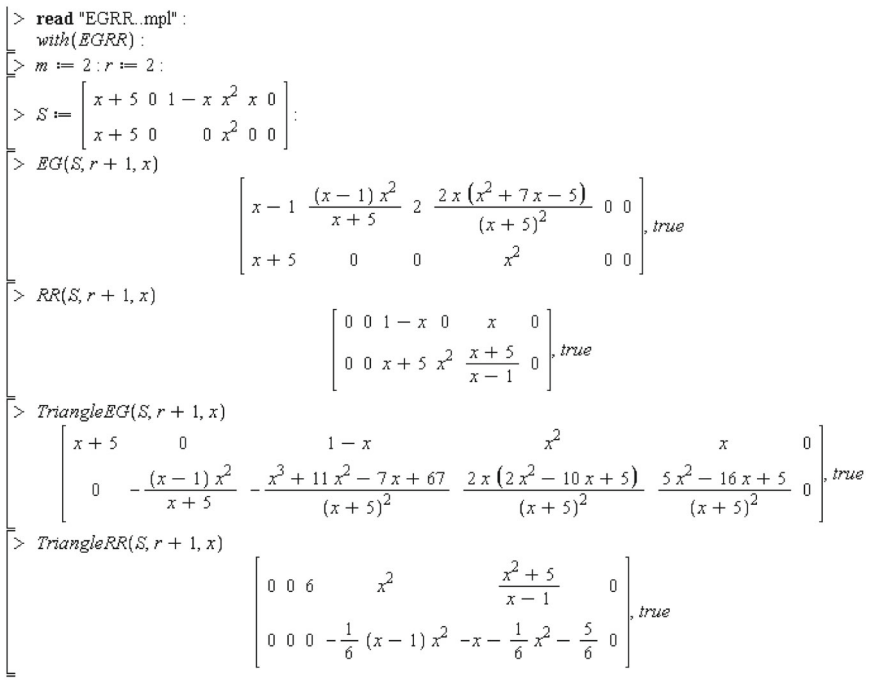

Fig. 1.

rational functions of one variable, which are also specified in a standard way accepted in `Maple`.

Each procedure of package `EGRR` has three input parameters:

- `M` is an explicit matrix of the original system;
- `d` is the number of *blocks* of the explicit matrix, $d = r + 1$;
- `x` is the independent variable of the system.

The values returned are

- `res`, the system obtained as a result of the transformations performed by the algorithm implemented in the procedure;
- `full_ranktrue` if, in the course of the algorithm operation, it was determined that the system has a full rank and `false` otherwise.

The application of the package procedures to a system of full rank is illustrated in the figure.

3.2. Determination of Dependence of Matrix Rows

The EG and RR algorithms rely on the determination of linear dependence of matrix rows, which is solved by procedure `NullSpace` from the `LinearAlgebra` package in `Maple`.

The idea of searching linear dependences of rows of the leading matrix without transforming the entire operator L was put forward in [8], where it was also noted that these dependences could be sought by procedures from the `LinearAlgebra` package, some of which are very efficient. The use of these procedures can save time when employing the version of the EG algorithm described in Section 2.3 (see the description of the reduction stage presented there). It was emphasized in [8] that the coefficients of a linear dependence are solutions of some system of linear homogeneous algebraic equations and all elements of any basis of the space of solutions to this system can be used (after simple transformations) on the subsequent reduction stages in the EG algorithm (a similar possibility for the RR algorithm was noted in [5]).

4. EXPERIMENTS

4.1. Random Selection of Systems

For each pair m, r, where $m = 4, 8, 10$ and $r = 3, 9, 27, 12$ systems have been generated. The coefficients of all systems were random polynomials (the standard Maple command `randpoly` was used to select the coefficients of the polynomials from the interval [–99, 99], and the degrees of the polynomials did not exceed 8). We generated sparse systems, with the probability that an element is nonzero being equal to $\frac{1}{[m/2]+1}$, $\frac{1}{[m/3]+1}$, or $\frac{1}{[m/4]+1}$, where [...] is the integer part of the number. For each variant, four (out of 12) systems have been generated. In the course of the experiments, we recorded the execution time, the number of the differentiation operations of the system matrix entries, and the number of arithmetic operations.

Table 1

		$r = 3$			$r = 9$			$r = 27$		
		$m = 4$	$m = 8$	$m = 16$	$m = 4$	$m = 8$	$m = 16$	$m = 4$	$m = 8$	$m = 16$
EG	WT	0.024	0.294	5.360	0.140	1.733	1.763	0.300	3.590	40.747
	NOWT	608	4920	32992	1624	8592	15664	3252	27600	121776
	MNO	608	4920	36640	1624	8592	50896	4712	27600	123856
	TMNO	0.024	0.294	1.574	0.140	1.733	1.743	0.110	3.590	3.800
RR	WT	0.034	0.336	5.386	0.150	1.653	2.140	0.424	5.507	57.41
	NOWT	408	4912	21856	1524	7704	40880	3276	25864	102224
	MNO	408	4912	27728	1524	7704	40880	3276	25864	102224
	TMNO	0.034	0.336	1.790	0.150	1.653	2.140	0.424	5.507	57.41
ΔEG	WT	0.027	0.310	3.503	0.107	1.130	4.650	0.497	4.307	104.534
	NOWT	320	2176	13760	1080	6400	27680	3696	29568	90496
	MNO	320	2176	13760	1160	6400	27680	3696	29568	90496
	TMNO	0.027	0.310	3.503	0.087	1.130	4.650	0.497	4.307	104.534
ΔRR	WT	0.037	0.540	3.703	0.180	2.017	16.970	0.890	17.510	226.590
	NOWT	384	2208	14016	1240	6400	24160	3472	30912	103936
	MNO	384	2336	14016	1400	6880	27680	4480	30912	103936
	TMNO	0.037	0.410	3.703	0.120	1.287	4.600	0.294	17.510	226.590

WT is the worst time of algorithm execution for random systems with fixed r, m; NOWT is the number of operations for systems with the worst time; MNO is the maximum number of operations for fixed r, m; TMNO is the algorithm execution time for system requiring the maximum number of operations.

Results of the experiments are presented in Table 1, which contains information on the maximum time[3] and the maximum number of operations performed by the algorithm.

The results presented demonstrate that the variation of the maximum number of operations required for solving one system and the worst (maximum) time of the algorithm execution for one system behave differently when r and m vary: the worst time grows faster than the maximum number of operations as r and m grow. Note that the worst time does not always correspond to the maximum number of operations. This is explained by the fact that the time spent on one operation is not constant. In particular, the results of the experiments show that the expenditures on one operation for the ΔEG and ΔRR algorithms grow faster with the growth of r and m than those for the EG and RR algorithm. As a result, for large m and r, the worst time for ΔEG and ΔRR is considerably greater than that for EG and RR, in spite of the fact that the maximum number of operations for ΔEG and ΔRR is less than that for EG and RR. This is explained by "cumbersome" form of entries of the explicit matrix in the course of calculation, which is discussed in more detail in Section 4.3.

[3] In seconds. Computations were carried out in Maple 2016, Ubuntu 8.04.4 LTS, AMD Athlon(tm) 64 Processor 3700+, 3GB RAM.

4.2. Storing Differentiated Rows

As noted in Section 2.6, the estimates we have obtained for the number of differentiations without repetitions do not allow us to make the exponent of r in the complexity estimates (7) and (10) less than 3, so that it remains unclear whether this exponent can be reduced to 2. Our experiments showed that the use of option `remember` for the procedure of differential row shift almost does not improve the operation time of the RR and ΔRR algorithms and even worsens it sometimes, whereas the number of differentiations can be reduced significantly. Table 2 shows results of operation of the RR and ΔRR algorithms (the computation time and the number of differentiations) on randomly generated systems with the number of equations $m = 6$ such that a half of the system equations has order $r = 3, 2, 4, 5, 6, 7, 8, 9$, whereas the other half has order equal to 1 and the application of the algorithm yields a system of order 1. For such systems, rows are repeatedly differentiated many times; however, storing of these differentiations improves the operation time insignificantly.

4.3. Comparison of EG, RR and ΔEG, ΔRR

Estimates (5), (7), (9), and (10) describe complexity in terms of the number of operations in the field $\mathbb{K}$ and do not take into account the size of the operands. In this sense, the bit complexity is more informative;

Table 2. Comparison of execution times of the algorithms with and without option remember

$m = 6$		$r = 3$	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$	$r = 9$
RR	T	0.690	1.303	2.597	8.950	37.480	50.950	296.633
	NOD	480	1170	2160	3654	5760	8370	11760
RR remember	T	0.657	1.257	2.593	8.970	37.213	51.550	309.010
	NOD	192	360	576	840	1152	1512	1920
ΔRR	T	0.614	1.137	2.330	4.414	8.264	14.810	24.747
	NOD	384	1170	2520	4746	7824	12150	17640
ΔRR remember	T	0.596	1.067	2.164	4.144	7.583	13.747	23.933
	NOD	192	630	1224	1974	2880	3942	5160

T is the algorithm execution time; NOD is the number of operations of differentiation.

however, its evaluation is associated with certain difficulties (although there are some studies of bit complexity for similar problems; see, for example, [15]). Clearly, in the process of operation of the EG, RR, ΔEG, and ΔRR algorithms, we encounter "cumbersome" form of elements of the operator L. The EG and RR algorithms, as described in Sections 2.3 and 2.4, have certain advantages in this regard: the replacement of row $L_{i,*}$ by row (4) or (6) results in the "cumbersome" form of elements in only one row. At the same time, when the revealing matrices are transformed to a triangular form (Section 2.5), the "cumbersome" form affects a greater number of elements, and the operations on elements become more expensive. Therefore, in spite of the attractiveness of estimates (9) and (10), algorithms EG and RR, as a rule, work faster than ΔEG and ΔRR, which is substantiated by the experiments (see Table 1).

Of course, it should be taken into account that algorithms ΔEG and ΔRR perform additional work compared to EG and RR, namely, convert the revealing matrix to a triangular form. Sometimes, this may occur preferable. In some cases, the ΔEG and ΔRR algorithms had advantages in terms of time either. For example, the advantage of ΔRR over RR can be seen from Table 2. The advantage of ΔEG over EG manifested itself in the experiments with sparse low-order systems of many equations. Table 3 shows the operation times of the algorithms for randomly generated systems of order $r = 1$ with the number of equations $mP = 50$; the part of the nonzero coefficients is 7–11%, and the coefficients themselves are polynomials of the degree 8 or less.

Table 3. Algorithm execution times for sparse systems with $r = 1$ and $m = 50$

$r = 1$, $m = 50$	7%	8%	9%	10%	11%
EG	0.994	36.067	71.540	97.503	177.897
RR	0.010	31.014	22.420	12.717	19.757
ΔEG	0.020	52.880	43.467	56.773	88.563
ΔRR	0.010	54.090	45.410	73.053	190.403

As has already been noted, the unquestionable advantage of RR and ΔRR is that the resulting system is equivalent to the original one. In a number of cases, RR is more efficient in terms of time than EG (Table 3). However, in some cases, EG and ΔEG are more efficient (Table 1).

The use of standard linear algebra procedures for reducing the revealing matrix to a triangular form (similar to that in Section 3.2) in the ΔEG and ΔRR algorithms will change the order of growth of the complexities of the ΔEG and ΔRR algorithms: instead of m^3 in (9) and (10), we will get $m^{\omega+1}$, which deprives the ΔEG and ΔRR algorithms of their theoretical advantage (see Section 2.5) over the EG and RR algorithms.

5. CONCLUSIONS

As can be seen, the EG and RR algorithms admit modifications that, based on general considerations and complexity estimates, may result in more efficient versions of these algorithms. However, our experiments show that these improvements are not of systematic character. Moreover, on the whole, this happens quite rarely. Therefore, the main recommendation is to use procedures `EG` and `RR` (depending on what revealing matrix—leading or frontal one—is to be made nonsingular). If the operator or system cannot be transformed as required in an acceptable time, then one may try to apply other procedures from the `EGRR` package, which, possibly, will yield the desired result.

ACKNOWLEDGMENTS

This work was supported in part by the Russian Foundation for Basic Research, project no. 16-01-00174-a.

REFERENCES

1. Abramov, S.A., On the differential and full algebraic complexities of operator matrices transformations, *Proc. of CASC 2016,* 2016, pp. 1—14.
2. van der Put, M. and Singer, M.F., *Galois Theory of Linear Differential Equations*, Grundlehren der mathematischen Wissenschaften 328, Heidelberg: Springer, 2003.
3. Rosenlicht, M., Integration in finite terms, *Am. Math. Monthly,* 1972, vol. 79, no. 9, pp. 963–972.
4. Abramov, S.A. and Barkatou, M.A., On the dimension of solution spaces of full rank linear differential systems, *Proc. of CASC 2013,* 2013, pp. 1–9.
5. Abramov, S. and Barkatou, M., On solution spaces of products of linear differential or difference operators, *ACM Commun. Comput. Algebra,* 2014, vol. 48, no. 4, pp. 155–165.
6. Miyake, M., Remarks on the formulation of the Cauchy problem for general system of ordinary differential equations, *Tohoku Math. J.,* 1980, vol. 32, no. 1, pp. 79–89.
7. Abramov, S. and Bronstein, M., On solutions of linear functional systems, *Proc. of ISSAC 2001,* 2001, pp. 1–6.
8. Abramov, S.A. and Bronstein, M., Linear algebra for skew-polynomial matrices, *Rapport de Recherche INRIA,* RR-4420, March 2002. http://www.inria.fr/RRRT/RR-4420.html.
9. Abramov, S.A. and Khmelnov, D.E., On singular points of solutions of linear differential systems with polynomial coefficients, *J. Math. Sci.,* 2012, vol. 185, no. 3, pp. 347—359.
10. Beckermann, B., Cheng, H., and Labahn, G., Fraction-free row reduction of matrices of Ore polynomials, *J. Symbolic Computation,* 2006, vol. 41, no. 5, pp. 513–543.
11. Barkatou, M.A., El Bacha, C., Labahn, G.,and Pflugel, E., On simultaneous row and column reduction of higher-order linear differential systems, *J. Symbolic Computation,* 2013, vol. 49, pp. 45–64.
12. Abramov, S., EG-eliminations, *J. Difference Equations Appl.,* 1999, vol. 5, nos. 4–5, pp. 393–433.
13. Mulders, T. and Storjohann, A., On lattice reduction for polynomial matrices, *J. Symbolic Computation,* 2003, vol. 35, no. 4, pp. 377–401.
14. Maple online help. http: //www.maplesoft.com/support/help/
15. Giesbrecht, M. and Sub Kim, M., Computing the Hermite form of a matrix of Ore polynomials, *J. Algebra,* 2013, vol. 376, pp. 341–362.

Translated by A. Pesterev