On the unimodularity testing for operator matrices: the differential input into complexity

S. Abramov

Moscow, Comp. Centre of RAS, MSU

Let K be a field of characteristic 0 with a derivation $\partial = '$.

$\mathrm{Mat}_n(K[\partial])$

Any non-zero operator matrix $L \in \mathrm{Mat}_n(K[\partial])$ can be represented as

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

$A_0, A_1, \ldots, A_d \in \mathrm{Mat}_n(K)$,
the matrix A_d (the *leading matrix* of L) is non-zero.
d is the *order* of L (we write $d = \mathrm{ord}\, L$).

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

Invertible operator matrices are also called *unimodular* matrices.

In M.Miyake's paper, the following example is given ($K = \mathbb{C}$, $\partial = \frac{d}{dx}$):

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

We give a complexity analysis of the unimodularity testing for an operator matrix and of constructing the inverse matrix if it exists.

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

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.

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

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

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.

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

The order of a row of $L \in \mathrm{Mat}_n(K[\partial])$ is the biggest order of operators from $K[\partial]$ belonging to the row.

We use the standard notation

$$M_{i,*}, \quad 1 \leq i \leq n,$$

for the $1 \times n$-matrix which is the i-th row of an $n \times n$-matrix M.

Set $\delta_i(L) = \operatorname{ord} L_{i,*}$.

Let L be represented as $A_d\partial^d + A_{d-1}\partial^{d-1} + \cdots + A_0$.
The matrix $B \in \mathrm{Mat}_n(K)$ such that $B_{i,*} = (A_{\delta_i(L)})_{i,*}$, $i = 1, \ldots, n$, is the *frontal matrix* of L.

Background algorithms: RR (Row-Reduction) and EG (EG-eliminations)

Given $L \in \mathrm{Mat}_n(K[\partial])$ (of full rank), algorithm RR (Row-Reduction) constructs $\check{L} \in \mathrm{Mat}_n(K[\partial])$:

- $\operatorname{ord} \check{L} \leq \operatorname{ord} L$,
- the frontal matrix of $\check{L}$ is invertible,
- $\check{L}$ has the form
$$\check{L} = U_l \dots U_1 L. \tag{3}$$
for some elementary unimodular $U_1, \dots U_l$.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th) having the greatest order.
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j \partial^{\delta_i(L)-\delta_j(L)} L_{j,*}. \tag{4}$$

Continue this process until the frontal matrix becomes nonsingular.

Given $L \in \mathrm{Mat}_n(K[\partial])$ (of full rank), algorithm RR (Row-Reduction) constructs $\check{L} \in \mathrm{Mat}_n(K[\partial])$:

- $\mathrm{ord}\,\check{L} \leq \mathrm{ord}\,L$,
- the frontal matrix of $\check{L}$ is invertible,
- $\check{L}$ has the form
$$\check{L} = U_l \dots U_1 L. \tag{3}$$
for some elementary unimodular $U_1, \dots U_l$.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.

From the rows of L corresponding to nonzero coefficients, select one (i-th) having the greatest order.

Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j \partial^{\delta_i(L)-\delta_j(L)} L_{j,*}. \tag{4}$$

Continue this process until the frontal matrix becomes nonsingular.

Given $L \in \mathrm{Mat}_n(K[\partial])$ (of full rank), algorithm RR (Row-Reduction) constructs $\check{L} \in \mathrm{Mat}_n(K[\partial])$:

- $\mathrm{ord}\, \check{L} \leq \mathrm{ord}\, L$,
- the frontal matrix of $\check{L}$ is invertible,
- $\check{L}$ has the form
$$\check{L} = U_l \dots U_1 L. \tag{3}$$
for some elementary unimodular $U_1, \dots U_l$.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th) having the greatest order.
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j \partial^{\delta_i(L) - \delta_j(L)} L_{j,*}. \tag{4}$$

Continue this process until the frontal matrix becomes nonsingular.

Given $L \in \mathrm{Mat}_n(K[\partial])$ (of full rank), algorithm RR (Row-Reduction) constructs $\check{L} \in \mathrm{Mat}_n(K[\partial])$:

- $\mathrm{ord}\,\check{L} \leq \mathrm{ord}\,L$,
- the frontal matrix of $\check{L}$ is invertible,
- $\check{L}$ has the form
$$\check{L} = U_l \dots U_1 L. \tag{3}$$
for some elementary unimodular $U_1, \dots U_l$.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th) having the greatest order.
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j \partial^{\delta_i(L)-\delta_j(L)} L_{j,*}. \tag{4}$$

Continue this process until the frontal matrix becomes nonsingular.

Given $L \in \mathrm{Mat}_n(K[\partial])$ (of full rank), algorithm RR (Row-Reduction) constructs $\check{L} \in \mathrm{Mat}_n(K[\partial])$:

- $\mathrm{ord}\, \check{L} \leq \mathrm{ord}\, L$,
- the frontal matrix of $\check{L}$ is invertible,
- $\check{L}$ has the form

$$\check{L} = U_l \dots U_1 L. \tag{3}$$

for some elementary unimodular $U_1, \dots U_l$.

Let the rows of the frontal matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th) having the greatest order.
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j \partial^{\delta_i(L) - \delta_j(L)} L_{j,*}. \tag{4}$$

Continue this process until the frontal matrix becomes nonsingular.

Matrices $U_1, \dots, U_l$ from (3) are of the form

$$\begin{array}{cc} & \\ i: & \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ p_1\partial^{\delta_i-\delta_1} & \dots & p_{i-1}\partial^{\delta_i-\delta_{i-1}} & p_i & p_{i+1}\partial^{\delta_i-\delta_{i+1}} & \dots & p_n\partial^{\delta_i-\delta_n} \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array} \qquad (5)$$

with $1 \leq i \leq n$, $p_1, \dots, p_n \in K$, $p_i \neq 0$.

If L is not of full rank then RR (in its original version) allows to recognize this.

For a given $L \in \mathrm{Mat}_n(K[\partial])$, algorithm EG constructs $\tilde{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\mathrm{ord}\,\tilde{L} \le \mathrm{ord}\,L$,
- the leading matrix of $\tilde{L}$ is invertible,
- each solution of L is a solution of $\tilde{L}$.

Let the rows of the leading matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th).
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j L_{j,*} \tag{6}$$

and differentiate the i-th row.
Continue this process until the leading matrix becomes nonsingular.

If the number of row differentiations becomes on some moment bigger than nd then L is not of full rank and is not invertible.

For a given $L \in \mathrm{Mat}_n(K[\partial])$, algorithm EG constructs $\tilde{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\operatorname{ord} \tilde{L} \leq \operatorname{ord} L$,
- the leading matrix of $\tilde{L}$ is invertible,
- each solution of L is a solution of $\tilde{L}$.

Let the rows of the leading matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.

From the rows of L corresponding to nonzero coefficients, select one (i-th). Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j L_{j,*} \qquad (6)$$

and differentiate the i-th row.

Continue this process until the leading matrix becomes nonsingular.

If the number of row differentiations becomes on some moment bigger than nd then L is not of full rank and is not invertible.

For a given $L \in \mathrm{Mat}_n(K[\partial])$, algorithm EG constructs $\tilde{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\mathrm{ord}\,\tilde{L} \le \mathrm{ord}\,L$,
- the leading matrix of $\tilde{L}$ is invertible,
- each solution of L is a solution of $\tilde{L}$.

Let the rows of the leading matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th).
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j L_{j,*} \qquad (6)$$

and differentiate the i-th row.
Continue this process until the leading matrix becomes nonsingular.

If the number of row differentiations becomes on some moment bigger than nd then L is not of full rank and is not invertible.

For a given $L \in \mathrm{Mat}_n(K[\partial])$, algorithm EG constructs $\tilde{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\operatorname{ord} \tilde{L} \leq \operatorname{ord} L$,
- the leading matrix of $\tilde{L}$ is invertible,
- each solution of L is a solution of $\tilde{L}$.

Let the rows of the leading matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \dots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th).
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j L_{j,*} \tag{6}$$

and differentiate the i-th row.
Continue this process until the leading matrix becomes nonsingular.

If the number of row differentiations becomes on some moment bigger than *nd* then L is not of full rank and is not invertible.

For a given $L \in \mathrm{Mat}_n(K[\partial])$, algorithm EG constructs $\tilde{L} \in \mathrm{Mat}_n(K[\partial])$ such that

- $\mathrm{ord}\,\tilde{L} \leq \mathrm{ord}\,L$,
- the leading matrix of $\tilde{L}$ is invertible,
- each solution of L is a solution of $\tilde{L}$.

Let the rows of the leading matrix of L be linearly dependent over K and coefficients of the dependence are $p_1, \ldots, p_n \in K$.
From the rows of L corresponding to nonzero coefficients, select one (i-th).
Replace the i-th row $L_{i,*}$ by

$$\sum_{j=1}^{m} p_j L_{j,*} \qquad (6)$$

and differentiate the i-th row.
Continue this process until the leading matrix becomes nonsingular.

If the number of row differentiations becomes on some moment bigger than nd then L is not of full rank and is not invertible.

RR — no extra solutions (+), many differentiations (-)

EG — not so many differentiations (+), extra ("parasitic") solutions can appear (-)

each of this algorithms allows "triangular" version: ΔRR, ΔEG. Using ΔRR, ΔEG we get the frontal (resp., leading) matrix in the triangular form.

Let the i-th row r of $L \in \mathrm{Mat}_n(K)$ be such that the corresponding row of the frontal (the case RR) or the leading (the case EG) matrix has the form

$$(a_1, \dots, a_k, 0, \dots, 0), \tag{7}$$

$a_k \neq 0$. Then k is the *pivot index* of the i-th row of L.

If all rows of L have distinct pivot indices then the frontal (leading) matrix of L is nonsingular.

Suppose that two rows r_1, r_2 of L have the same pivot index k.
Set $d_1 = \mathrm{ord}\, r_1$, $d_2 = \mathrm{ord}\, r_2$. Let $d_1 \leq d_2$.
There exists a v in K such that the difference

$$r_2 - v\partial^{d_2 - d_1} r_1 \tag{8}$$

either has the pivot index which is less than k or has the order which is less than d_2.

This can be used instead of a search for a linear dependency of the rows of the frontal (leading) matrix of L.

Let the i-th row r of $L \in \mathrm{Mat}_n(K)$ be such that the corresponding row of the frontal (the case RR) or the leading (the case EG) matrix has the form

$$(a_1, \dots, a_k, 0, \dots, 0), \tag{7}$$

$a_k \neq 0$. Then k is the *pivot index* of the i-th row of L.

If all rows of L have distinct pivot indices then the frontal (leading) matrix of L is nonsingular.

Suppose that two rows r_1, r_2 of L have the same pivot index k.
Set $d_1 = \mathrm{ord}\, r_1$, $d_2 = \mathrm{ord}\, r_2$. Let $d_1 \leq d_2$.
There exists a v in K such that the difference

$$r_2 - v\partial^{\,d_2 - d_1} r_1 \tag{8}$$

either has the pivot index which is less than k or has the order which is less than d_2.

This can be used instead of a search for a linear dependency of the rows of the frontal (leading) matrix of L.

For the difference case this was used in first versions of EG-eliminations, and can still be found in Maple:
`LinearFunctionalSystems[MatrixTriangularization]`.

A.Storjohann drew my attention to the fact that the complexity of this approach is less than of one which uses solving linear algebraic systems.

$F_{\mathrm{XX}}(n,d), \quad B_{\mathrm{XX}}(n,d),$

$F_{\mathrm{XX}}(n,d)$ — "visible" (or "apparent") arithmetic complexity of xx
($F =$ "at First sight"),
$B_{\mathrm{XX}}(n,d)$ — the sharp upper bound for the number of row differentiations

We have

$$F_{\mathrm{RR}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad B_{\mathrm{RR}}(n,d) = \Theta(n^2d^2)$$

$$F_{\mathrm{EG}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad B_{\mathrm{RR}}(n,d) = \Theta(nd^2)$$

$$F_{\Delta\mathrm{RR}}(n,d) = \Theta(n^3d^2), \quad B_{\Delta\mathrm{RR}}(n,d) = \Theta(n^2d^2)$$

$$F_{\Delta\mathrm{EG}}(n,d) = \Theta(n^3d^2), \quad B_{\Delta\mathrm{RR}}(n,d) = \Theta(nd^2).$$

$F_{\mathrm{XX}}(n,d), \quad B_{\mathrm{XX}}(n,d),$

$F_{\mathrm{XX}}(n,d)$ — "visible" (or "apparent") arithmetic complexity of xx
(F = "at First sight"),
$B_{\mathrm{XX}}(n,d)$ — the sharp upper bound for the number of row differentiations

We have

$$F_{\mathrm{RR}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad B_{\mathrm{RR}}(n,d) = \Theta(n^2d^2)$$

$$F_{\mathrm{EG}}(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad B_{\mathrm{RR}}(n,d) = \Theta(nd^2)$$

$$F_{\triangle\mathrm{RR}}(n,d) = \Theta(n^3d^2), \quad B_{\triangle\mathrm{RR}}(n,d) = \Theta(n^2d^2)$$

$$F_{\triangle\mathrm{EG}}(n,d) = \Theta(n^3d^2), \quad B_{\triangle\mathrm{RR}}(n,d) = \Theta(nd^2).$$

Let $\mathrm{xx} \in \{\mathrm{RR},\ \ \mathrm{EG},\ \ \Delta\mathrm{RR},\ \ \Delta\mathrm{EG}\}$.

It is possible to show that

$$\tilde{T}_{\mathrm{XX}}(n,d) = \Theta(B_{\mathrm{XX}}(n,d)nd) \tag{9}$$

for the the differential complexity $\tilde{T}$, and

$$T_{\mathrm{XX}}(n,d) = \Theta(F_{\mathrm{XX}}(n,d) + B_{\mathrm{XX}}(n,d)nd) \tag{10}$$

for the arithmetic complexity T.

Concerning (10) note that each differentiation of a row of L uses besides nd of differentiations of elements of K also the same number of arithmetic operations (additions) in K. By this reason, (10) holds for both total and arithmetic complexities.

The space complexity:

$S_{\mathrm{RR}}(n,d) = \Theta(n^2 + nd), \quad S_{\mathrm{EG}}(n,d) = \Theta(n^2 + nd)$

$S_{\Delta\mathrm{RR}}(n,d) = \Theta(nd), \quad S_{\Delta\mathrm{EG}}(n,d) = \Theta(nd)$

Let xx $\in \{$RR, EG, ΔRR, ΔEG$\}$.

It is possible to show that

$$\tilde{T}_{\mathrm{XX}}(n,d) = \Theta(B_{\mathrm{XX}}(n,d)nd) \tag{9}$$

for the the differential complexity $\tilde{T}$, and

$$T_{\mathrm{XX}}(n,d) = \Theta(F_{\mathrm{XX}}(n,d) + B_{\mathrm{XX}}(n,d)nd) \tag{10}$$

for the arithmetic complexity T.

Concerning (10) note that each differentiation of a row of L uses besides nd of differentiations of elements of K also the same number of arithmetic operations (additions) in K. By this reason, (10) holds for both total and arithmetic complexities.

The space complexity:

$S_{\mathrm{RR}}(n,d) = \Theta(n^2 + nd), \quad S_{\mathrm{EG}}(n,d) = \Theta(n^2 + nd)$

$S_{\Delta\mathrm{RR}}(n,d) = \Theta(nd), \quad S_{\Delta\mathrm{EG}}(n,d) = \Theta(nd)$

Let xx $\in \{\mathrm{RR}, \ \mathrm{EG}, \ \Delta\mathrm{RR}, \ \Delta\mathrm{EG}\}$.

It is possible to show that

$$\tilde{T}_{\mathrm{XX}}(n,d) = \Theta(B_{\mathrm{XX}}(n,d)nd) \tag{9}$$

for the the differential complexity $\tilde{T}$, and

$$T_{\mathrm{XX}}(n,d) = \Theta(F_{\mathrm{XX}}(n,d) + B_{\mathrm{XX}}(n,d)nd) \tag{10}$$

for the arithmetic complexity T.

Concerning (10) note that each differentiation of a row of L uses besides nd of differentiations of elements of K also the same number of arithmetic operations (additions) in K. By this reason, (10) holds for both total and arithmetic complexities.

The space complexity:

$S_{\mathrm{RR}}(n,d) = \Theta(n^2 + nd), \quad S_{\mathrm{EG}}(n,d) = \Theta(n^2 + nd)$

$S_{\Delta\mathrm{RR}}(n,d) = \Theta(nd), \quad S_{\Delta\mathrm{EG}}(n,d) = \Theta(nd)$

Unimodularity testing

Alg 1 — RR — ("yes" + a sequence of elementary matrices) or "no"

Alg 2 — ΔRR — ("yes" + a sequence of elementary matrices) or "no"

Alg 3 — ΔEG — "yes" or "no" only

$$F_1(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad \tilde{T}_1(n,d) = \Theta(n^3d^3)$$
$$T_1(n,d) = \Theta(n^{\omega+1}d + n^3d^3),$$

$$F_2(n,d) = \Theta(n^3d^2), \quad \tilde{T}_2(n,d) = \Theta(n^3d^3),$$
$$T_2(n,d) = \Theta(n^3d^3),$$

$$F_3(n,d) = \Theta(n^3d^2), \quad \tilde{T}_3(n,d) = \Theta(n^2d^3),$$
$$T_3(n,d) = \Theta(n^3d^2 + n^2d^3),$$

Note that for all the three algorithms the full complexity T grows faster than the "visual" complexity F.

Alg 1 — RR — ("yes" + a sequence of elementary matrices) or "no"

Alg 2 — ΔRR — ("yes" + a sequence of elementary matrices) or "no"

Alg 3 — ΔEG — "yes" or "no" only

$$F_1(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad \tilde{T}_1(n,d) = \Theta(n^3d^3)$$
$$T_1(n,d) = \Theta(n^{\omega+1}d + n^3d^3),$$

$$F_2(n,d) = \Theta(n^3d^2), \quad \tilde{T}_2(n,d) = \Theta(n^3d^3),$$
$$T_2(n,d) = \Theta(n^3d^3),$$

$$F_3(n,d) = \Theta(n^3d^2), \quad \tilde{T}_3(n,d) = \Theta(n^2d^3),$$
$$T_3(n,d) = \Theta(n^3d^2 + n^2d^3),$$

Note that for all the three algorithms the full complexity T grows faster than the "visual" complexity F.

Explicit inverse matrix

Proposition 1

(Barkatou, El Bacha, Labahn, Pfluegel) Let algorithm RR compute step-by-step the unimodular matrices of the form (5) for $L \in \mathrm{Mat}_n(K[\partial])$, $\operatorname{ord} L = d$. Then $\operatorname{ord}(U_k \dots U_1) = O(nd)$ for all $k = 1, \dots, l$.

Proposition 2

Let L be unimodular. Then

$$\operatorname{ord} L^{-1} \le (n-1)d \qquad (11)$$

and (11) is the sharp bound.

The bound (11) follows from some estimates related to computing the Hermite form given by Giesbrecht and Kim. We prove that this bound is sharp.

Proposition 1

(Barkatou, El Bacha, Labahn, Pfluegel) Let algorithm RR compute step-by-step the unimodular matrices of the form (5) for $L \in \mathrm{Mat}_n(K[\partial])$, $\mathrm{ord}\, L = d$. *Then* $\mathrm{ord}(U_k \dots U_1) = O(nd)$ *for all* $k = 1, \dots, l$.

Proposition 2

Let L be unimodular. Then

$$\mathrm{ord}\, L^{-1} \leq (n-1)d \tag{11}$$

and (11) is the sharp bound.

The bound (11) follows from some estimates related to computing the Hermite form given by Giesbrecht and Kim. We prove that this bound is sharp.

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

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

Btw, as a consequence of Proposition 2 we get:

If $L \in \mathrm{Mat}_2(K[\partial])$ *is unimodular then* $\mathrm{ord}\, L^{-1} = \mathrm{ord}\, L$*.*

(Miyake's example confirms this.)

4:

Output: The result of the unimodularity testing of L, and L^{-1} if L is unimodular; in the last case L^{-1} is represented as a matrix from $\mathrm{Mat}_n(K[\partial])$:

The operations which algorithm 1 performs on the operator matrix which is originally equal to L are doubled on the matrix which is originally equal to the unit matrix I_n. This results the product $U = U_l \dots U_1$ in the explicit form. If L was transformed by algorithm 1 into $\breve{L}$ of order 0 then the operator matrix L is unimodular and $L^{-1} = \breve{L}^{-1} U$.

The algorithm was presented by Barkatou, El Bacha, Labahn, Pfluegel.

$F_4(n,d) = \Theta(n^4 d^2), \quad \tilde{T}_4(n,d) = \Theta(n^4 d^3), \quad T_4(n,d) = \Theta(n^4 d^3)$

$S_4(n,d) = \Theta(n^2 d)$

4:

Output: The result of the unimodularity testing of L, and L^{-1} if L is unimodular; in the last case L^{-1} is represented as a matrix from $\mathrm{Mat}_n(K[\partial])$:

The operations which algorithm 1 performs on the operator matrix which is originally equal to L are doubled on the matrix which is originally equal to the unit matrix I_n. This results the product $U = U_l \dots U_1$ in the explicit form. If L was transformed by algorithm 1 into $\check{L}$ of order 0 then the operator matrix L is unimodular and $L^{-1} = \check{L}^{-1}U$.

The algorithm was presented by Barkatou, El Bacha, Labahn, Pfluegel.

$F_4(n,d) = \Theta(n^4 d^2), \quad \tilde{T}_4(n,d) = \Theta(n^4 d^3), \quad T_4(n,d) = \Theta(n^4 d^3)$

$S_4(n,d) = \Theta(n^2 d)$

4:

Output: The result of the unimodularity testing of L, and L^{-1} if L is unimodular; in the last case L^{-1} is represented as a matrix from $\mathrm{Mat}_n(K[\partial])$:

The operations which algorithm 1 performs on the operator matrix which is originally equal to L are doubled on the matrix which is originally equal to the unit matrix I_n. This results the product $U = U_l \dots U_1$ in the explicit form. If L was transformed by algorithm 1 into $\check{L}$ of order 0 then the operator matrix L is unimodular and $L^{-1} = \check{L}^{-1} U$.

The algorithm was presented by Barkatou, El Bacha, Labahn, Pfluegel.

$$F_4(n,d) = \Theta(n^4 d^2), \quad \tilde{T}_4(n,d) = \Theta(n^4 d^3), \quad T_4(n,d) = \Theta(n^4 d^3)$$

$$S_4(n,d) = \Theta(n^2 d)$$

When all differentiated rows are stored

Proposition 3

The number of row differentiations without repetitions (when the result of each differentiation is stored, i.e., when we collect all such results) executed by the algorithm RR is $O(nd^2)$; as a consequence, the number of differentiations of elements of K is $O(n^2d^3)$.

Conjecture 1

The estimates $O(nd^2)$ and, resp., $O(n^2d^3)$ given by Proposition 3 can be replaced by $\Theta(nd)$ and, resp., $\Theta(n^2d^2)$.

Proposition 3

The number of row differentiations without repetitions (when the result of each differentiation is stored, i.e., when we collect all such results) executed by the algorithm RR is $O(nd^2)$; as a consequence, the number of differentiations of elements of K is $O(n^2d^3)$.

Conjecture 1

The estimates $O(nd^2)$ and, resp., $O(n^2d^3)$ given by Proposition 3 can be replaced by $\Theta(nd)$ and, resp., $\Theta(n^2d^2)$.

Algorithms $1'$, $2'$, $4'$

$F_{1'}(n,d) = \Theta(n^{\omega+1}d + n^3d^2)$, $\tilde{T}_{1'}(n,d) = O(n^2d^3)$
$T_{1'}(n,d) = O(n^{\omega+1}d + n^3d^2 + n^2d^3)$,
$S_{1'}(n,d) = O(n^2d^3)$

$F_{2'}(n,d) = \Theta(n^3d^2)$, $\tilde{T}_{2'}(n,d) = O(n^2d^3)$,
$T_{2'}(n,d) = O(n^3d^2 + n^2d^3)$,
$S_{2'}(n,d) = O(n^2d^3)$

$F_{4'}(n,d) = \Theta(n^4d^2)$, $\tilde{T}_{4'}(n,d) = O(n^3d^3)$,
$T_{4'}(n,d) = O(n^4d^3)$.
$S_{4'}(n,d) = O(n^3d^3)$

If Conjecture 1 is correct then

$F_{2'}(n,d) = \Theta(n^3d^2)$, $\tilde{T}_{2'}(n,d) = \Theta(n^2d^2)$,
$T_{2'}(n,d) = \Theta(n^3d^2)$,
$S_{2'}(n,d) = \Theta(n^2d^2)$

$F_{4'}(n,d) = \Theta(n^4d^2)$, $\tilde{T}_{4'}(n,d) = \Theta(n^3d^2)$,
$T_{4'}(n,d) = \Theta(n^4d^2)$.
$S_{4'}(n,d) = \Theta(n^3d^2)$

Algorithms $1'$, $2'$, $4'$

$$F_{1'}(n,d) = \Theta(n^{\omega+1}d + n^3d^2), \quad \tilde{T}_{1'}(n,d) = O(n^2d^3)$$
$$T_{1'}(n,d) = O(n^{\omega+1}d + n^3d^2 + n^2d^3),$$
$$S_{1'}(n,d) = O(n^2d^3)$$

$$F_{2'}(n,d) = \Theta(n^3d^2), \quad \tilde{T}_{2'}(n,d) = O(n^2d^3),$$
$$T_{2'}(n,d) = O(n^3d^2 + n^2d^3),$$
$$S_{2'}(n,d) = O(n^2d^3)$$

$$F_{4'}(n,d) = \Theta(n^4d^2), \quad \tilde{T}_{4'}(n,d) = O(n^3d^3),$$
$$T_{4'}(n,d) = O(n^4d^3).$$
$$S_{4'}(n,d) = O(n^3d^3)$$

If Conjecture 1 is correct then

$$F_{2'}(n,d) = \Theta(n^3d^2), \quad \tilde{T}_{2'}(n,d) = \Theta(n^2d^2),$$
$$T_{2'}(n,d) = \Theta(n^3d^2),$$
$$S_{2'}(n,d) = \Theta(n^2d^2)$$

$$F_{4'}(n,d) = \Theta(n^4d^2), \quad \tilde{T}_{4'}(n,d) = \Theta(n^3d^2),$$
$$T_{4'}(n,d) = \Theta(n^4d^2).$$
$$S_{4'}(n,d) = \Theta(n^3d^2)$$