Fast computation of common left multiples of linear ordinary differential operators

Alin Bostan[1], Frédéric Chyzak[1], Ziming Li[2], and Bruno Salvy[1]
[1]Algorithms project, INRIA Paris-Rocquencourt, France.
[2]Key Lab of Mathematics Mechanization, AMSS, Beijing, China.

April 29, 2011

The design of efficient algorithms for basic operations on algebraic objects like integers, polynomials and matrices is a classical topic in computer algebra, as testified by Knuth's book [7]. This topic recently gained a renewed interest [4], thanks to the spread of personal computers able to tackle computational problems of very big sizes. On the one hand, asymptotic complexity analyses allow reliable predictions about timings of calculations, on the other hand, fast algorithms based on Fast Fourier Transform techniques become profitable in practice. The complexity of operations in the polynomial ring $\mathbb{K}[x]$ over a field $\mathbb{K}$ has been intensively studied in the computer-algebra literature. It is well established that polynomial multiplication is a *commutative complexity yardstick*, in the sense that the complexity of operations in $\mathbb{K}[x]$ can be expressed in terms of that of multiplication, and for most of them, in a quasi-linear way.

Linear differential operators in the derivation $\partial = \frac{\partial}{\partial x}$ and with coefficients in $\mathbb{K}(x)$ form a non-commutative ring, denoted $\mathbb{K}(x)\langle\partial\rangle$, that shares many algebraic properties with the commutative ring $\mathbb{K}[x]$. The structural analogy between polynomials and linear differential equations was discovered long ago by Libri and Brassinne [3]. They introduced the bases of a non-commutative elimination theory, by defining the notions of greatest common right divisor (GCRD) and least common left multiple (LCLM) for differential operators, and designing an Euclidean-type algorithm for computing GCRDs and LCLMs. This was formalized by Ore [9, 10], who set up a common algebraic framework for polynomials and differential operators. Yet the algorithmic study of linear differential operators is currently much less advanced than in the polynomial case. The complexity of the product in $\mathbb{K}(x)\langle\partial\rangle$ has been addressed only recently in [6, 1].

The aim of this work is to take a first step towards a systematic study of the complexity of operations in $\mathbb{K}(x)\langle\partial\rangle$. We promote the idea that (polynomial) matrix multiplication may well become the common yardstick for measuring complexities in this non-commutative setting. The goal of the present work is to obtain fast algorithms and implementations for LCLMs. We focus on LCLMs since several higher level algorithms rely crucially on the efficiency of this basic computational primitive. Our approach is based on using complexity analysis as a tool for algorithmic design, and on producing tight size bounds on the various objects involved in the algorithms.

It is known that Ore's non-commutative Euclidean algorithm is computationally expensive; various other algorithms for computing common left multiples of two operators were proposed [11, 13, 12, 8]. As opposed to Ore's approach, all these alternative algorithms reduce the problem of computing LCLMs to linear algebra. However, very few complexity analyses and performance comparisons are available.

As a first contribution in this poster, we present a new algorithm for computing LCLMs of several operators. It reduces the LCLM computation to a linear algebra problem on a polynomial matrix. The new algorithm can be viewed as an adaptation of Poole's algorithm [11, Chap. 3, §9] to several operators. At the same time, we use modern linear-algebra algorithms [14, 15] to achieve a lower arithmetic complexity. Our algorithm is similar in spirit to Grigoriev's algorithm [5, §5] for computing GCRDs of several operators.

In what follows, ω denotes the exponent of matrix multiplication over $\mathbb{K}$, and the soft-O notation $\tilde{\mathcal{O}}(\,)$ indicates that polylogarithmic factors are neglected.

Theorem 1. *Let $L_1, \ldots, L_k$ be operators in $\mathbb{K}[x]\langle\partial\rangle$, of orders at most r, with polynomial coefficients of degree at most d. Let L denote the LCLM of $L_1, \ldots, L_k$ computed in $\mathbb{K}(x)\langle\partial\rangle$ and normalized in $\mathbb{K}[x]\langle\partial\rangle$ with no content. Then L has order at most kr, degrees in x at most k^2rd, and it can be computed using $\tilde{\mathcal{O}}(k^{2\omega}r^{\omega}d)$ arithmetic operations in $\mathbb{K}$.*

The bound k^2rd on the coefficient degrees of the LCLM is tight and improves by one order of magnitude the previously known bound k^2r^2d. Moreover, for fixed k, the complexity of the new algorithm is almost optimal, in the sense that it nearly matches the arithmetic size of the output.

As a second contribution, we prove an upper bound $B \approx 2k(d+r)$ on the total degree in (x, ∂) in which (non-minimal) common left multiples exist. This is a new instance of the philosophy, promoted in [2], of relaxing order minimality for linear differential operators, in order to achieve better total arithmetic size. While the total arithmetic size of the LCLM is at most k^3r^2d, there exist common left multiples of total size $4k^2(d+r)^2$ only.

As a third contribution, we analyze the worst-case arithmetic complexity of existing algorithms for computing LCLMs, as well as the size of their outputs. For instance, we show that the extension of the algorithm in [13, 12] to several operators (which is implemented in Maple's package `DEtools`) has complexity $\tilde{\mathcal{O}}(k^{\omega+1}r^{\omega+1}d)$. These estimates are in accordance with our experiments showing that the new algorithm performs faster for large r, while the other algorithm is well suited for large k.

A fourth contribution is fast Maple and Magma implementations. Preliminary experimental results indicate that our implementations can outperform Maple's and Magma's library routines.

References

[1] A. Bostan, F. Chyzak, and N. Le Roux. Products of ordinary differential operators by evaluation and interpolation. In *Proc. ISSAC'08*, pages 23–30. ACM, 2008.

[2] A. Bostan, F. Chyzak, G. Lecerf, B. Salvy, and É. Schost. Differential equations for algebraic functions. In *Proc. ISSAC'07*, pages 25–32. ACM, 2007.

[3] S. S. Demidov. On the history of the theory of linear differential equations. *Arch. Hist. Exact Sci.*, 28(4):369–387, 1983.

[4] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2nd edition, 2003.

[5] D. Y. Grigoriev. Complexity of factoring and calculating the GCD of linear ordinary differential operators. *J. Symb. Comp.*, 10(1):7–37, 1990.

[6] J. Van der Hoeven. FFT-like multiplication of linear differential operators. *J. Symb. Comp.*, 33(1):123–127, 2002.

[7] D. E. Knuth. *The art of computer programming. Vol. 2: Seminumerical algorithms*. Addison-Wesley, 1969.

[8] Z. Li. A subresultant theory for Ore polynomials with applications. In *ISSAC'98*, pages 132–139. ACM, 1998.

[9] O. Ore. Formale Theorie der linearen Differentialgleichungen. *J. Reine Angew. Math.*, 167:221–234, 1932.

[10] O. Ore. Theory of non-commutative polynomials. *Ann. of Math.*, 34(3):480–508, 1933.

[11] E. G. C. Poole. *Introduction to the theory of linear differential equations*. Dover Publications Inc., NY, 1960.

[12] B. Salvy and P. Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Software*, 20(2):163–177, 1994.

[13] R. P. Stanley. Differentiably finite power series. *European J. Combin.*, 1(2):175–188, 1980.

[14] A. Storjohann. High-order lifting and integrality certification. *J. Symb. Comp.*, 36(3-4):613–648, 2003.

[15] A. Storjohann and G. Villard. Computing the rank and a small nullspace basis of a polynomial matrix. In *ISSAC'05*, pages 309–316. ACM, New York, 2005.