Fast Computation of Common Left Multiples of Linear Ordinary Differential Operators*

Alin Bostan, Frédéric Chyzak, Bruno Salvy
Algorithms Project, INRIA (France)
{alin.bostan,frederic.chyzak,bruno.salvy}@inria.fr

Ziming Li
KLMM and AMSS (China)
zmli@mmrc.iss.ac.cn

ABSTRACT
We study tight bounds and fast algorithms for LCLMs of several linear differential operators with polynomial coefficients. We analyse the arithmetic complexity of existing algorithms for LCLMs, as well as the size of their outputs. We propose a new algorithm that recasts the LCLM computation in a linear algebra problem on a polynomial matrix. This algorithm yields sharp bounds on the coefficient degrees of the LCLM, improving by one order of magnitude the best bounds obtained using previous algorithms. The complexity of the new algorithm is almost optimal, in the sense that it nearly matches the arithmetic size of the output.

Categories and Subject Descriptors:
I.1.2 [Computing Methodologies]: Symbolic and Algebraic Manipulations — Algebraic Algorithms
General Terms: Algorithms, Theory.
Keywords: Algorithms, complexity, linear differential operators, common left multiples.

1. INTRODUCTION
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 the cost 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 $\mathbb{K}(x)[\partial]$, 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 [26, 8, 14]. 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 by designing a Euclidean-type algorithm for GCRDs. This was formalised by Ore [27, 28], who set up a common algebraic framework for polynomials and linear differential operators (and other skew polynomials, including difference and $q$-difference operators). Yet, the algorithmic study of linear differential operators is currently much less advanced than in the polynomial case. The cost of product in $\mathbb{K}(x)[\partial]$ has been addressed only recently in [22, 6].

The general aim of this work is to take a step towards a systematic study of the complexity of operations in $\mathbb{K}(x)[\partial]$. We promote the idea that (polynomial) matrix multiplication may well become the common yardstick for measuring complexities in this non-commutative setting. The specific goal of the present article is to illustrate this idea for LCLMs. We focus on LCLMs since several higher level algorithms rely crucially on the efficiency of this basic computational primitive. For instance, algorithms for manipulating D-finite functions represented by annihilating equations use common left multiples for performing addition [34, 32]. LCLMs of several operators are also needed as a basic task in various other higher-level algorithms [2, 24, 12]. 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 folklore that Ore’s non-commutative Euclidean algorithm is computationally expensive; various other algorithms for computing common left multiples of two operators have been proposed [21, 34, 32, 25, 5, 1, 23]. As opposed to Ore’s algorithm, these alternatives have the common feature that they reduce the problem of computing LCLMs to linear algebra. However, few complexity analyses [17, 18, 5, 23] and performance comparisons [25, 1] are available.

Main contributions. As a first contribution, we design a new algorithm for computing the LCLM 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 Heffter’s algorithm [21] to several operators. At the same time, we use modern linear-algebra algorithms [35, 37] to achieve a low arithmetic complexity. Our algorithm is similar in spirit to Grigoriev’s method [20, §5] for computing GCRDs of several operators.

Before stating more precisely our main results, we need the following conventions and notations, that will be used throughout the paper. All algorithms take as input linear differential operators with polynomial coefficients, that is, belonging to $\mathbb{K}[x][\partial]$, instead of rational function coefficients. From the computational viewpoint, this is not too severe a restriction, since the rational coefficients case is easily reduced to that of polynomial coefficients, by normalisation.

*We warmly thank the referees for their very helpful comments.
— This work was supported in part by the MSR–INRIA Joint Centre, and by two NSFC grants (91118001 and 60821002/F02).
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2012 ACM 978-1-4503-1269/12/07 ...$5.00.
For $L_1, \ldots, L_k$ in $\mathbb{K}[x](\partial)$, we write $\text{LCLM}(L_1, \ldots, L_k)$ for the primitive LCLM of $L_1, \ldots, L_k$ in $\mathbb{K}[x](\partial)$. We say that an operator $L \in \mathbb{K}[x](\partial)$ has bidegree at most $(d, r)$ in $(x, \partial)$ if it has order at most $r$ and polynomial coefficients of degree at most $d$. The cost of our algorithms is measured by the number of arithmetic operations that they use in the base field $\mathbb{K}$. The constant $\theta \in [2, 3]$ stands for a feasible exponent for matrix multiplication over $\mathbb{K}$ (see definition in Section 2), and the soft-O notation $\tilde{O}(\cdot)$ indicates that polylogarithmic factors are neglected. Our main result is the following.

**Theorem 1** Let $L_1, \ldots, L_k$ be operators in $\mathbb{K}[x](\partial)$ of bidegrees at most $(d, r)$ in $(x, \partial)$. Then $\text{LCLM}(L_1, \ldots, L_k)$ has order at most $kr$, degrees in $x$ at most $dk(rk + r + 1)$, and it can be computed in $\tilde{O}(k^{28}r^d)$ arithmetic operations in $\mathbb{K}$.

The upper bound $dk(rk + r + 1)$ on coefficient degrees is sharp, in the sense that it is reached on generic inputs. It improves by one order of magnitude the best bound $O(k^{7r^2}d)$ obtained using previous algorithms. Moreover, for fixed $k$, the cost of the new algorithm is almost optimal, in the sense that it nearly matches the arithmetic size of the LCLM.

As a second contribution, we analyse the worst-case arithmetic complexity of existing algorithms for LCLMs, as well as the size of their outputs. For instance, we show that the extension to several operators of the “folklore” algorithm in [34, 32] has complexity $O(k^{10}r^d + 1)$. We call this extension van Hoeij’s algorithm, after the name of the implementer in Maple’s package DEtools of one of its variants. These estimates are in accordance with our experiments showing that our new algorithm performs faster for large order $r$, while van Hoeij’s algorithm is well suited for large $k$.

Using our tight degree bounds, we also show that any algorithm that computes the LCLM of two operators of bidegree $(d, r)$ in $(x, \partial)$ in complexity $O(r^d)$ can be used as the building block of a divide-and-conquer (DAC) algorithm that computes the LCLM of $k$ operators of bidegree $(d, r)$ in complexity $O(k \cdot O(r^d))$. The costs of several algorithms are summarised in Figure 1, where notation $A + DAC$ indicates that algorithm $A$ is used in a DAC scheme.

As a third contribution, we prove an upper bound $B \approx 2k(d + r)$ on the total degree in $(x, \partial)$ of nonzero common left multiples (not necessarily of minimal order). This is a new instance of the philosophy, initiated in [7], of relaxing order minimality for linear differential operators, in order to achieve better arithmetic size. While, by Theorem 1, the total arithmetic size of the LCLM is typically $k^3r^d$, there exist common left multiples of total size $4k^2(d + r)^2$ only.

A fourth contribution is a fast Magna implementation that outperforms Maple’s LCLM routine. Experimental results confirm that the practical complexity of the new algorithm behaves as predicted by our theoretical results.

Last, but not least, we have undertaken an extensive bibliographic search, which we now proceed to describe.

**Previous work.** Libri [26] and Brassimme [8] (see also [14]) defined the notions of GCRD and LCLM of linear differential operators, and sketched a Euclidean-type algorithm for GCRDs. Von Escherich [15] defined the related notion of differential resultant of two linear differential operators. Articles [8, 15] contain the embryo of an algorithm for the LCLM based on linear algebra; that algorithm was explicitly stated by Heffter [21], and later rediscovered by Poole in his classical book [31]. The roots of a subresultant theory for differential operators are in Prier’s articles [29, 30]. Blumberg [3] gave one of the first systematic accounts of the algebraic properties of linear differential operators. Building on predecessors’ works, Ore [27, 28] extended the Euclidean-type theory to the more general framework of skew polynomials. He showed [28, Theorem 8, §3] that, while the LCLM is not related to the GCRD by a simple formula as in the commutative case, there nevertheless exists a formula expressing the LCLM in terms of the successive remainders in the Euclidean scheme. Almost simultaneously, Wedderburn [40, §7-8] showed that the LCLM can also be computed by an extended version of the Euclidean-Ore algorithm, that computes Bézout cofactors along the way.

In the computer algebra literature, algorithmic issues for skew polynomials emerged in the 1990s, and were popularised by Bronstein and Petkovšek [9, 10]. Grigoriev [20, §6] designed a fast algorithm for computing the GCRD of a family of linear differential operators; to do so, he proved tight bounds on the degree of the GCRD, by extending von Escherich’s construction of the Sylvester matrix for two differential operators to an arbitrary number of operators. The bound is linear in the number of operators, in their maximal order and in their maximal degree [20, Lemma 5.1]. Giesbrecht analysed the complexity of the LCLM computation for two operators, but only in terms of their order [17, 18]. (Strictly speaking, his method was proposed for a different Ore ring, but it extends to more general settings, including the differential case.) For two operators $L_1, L_2 \in \mathbb{K}(x)(\partial)$ of orders at most $r$, the first (Heffter-style) algorithm [17, Lemma 5] computes LCLM$(L_1, L_2)$ in $O(r^3)$ operations in $\mathbb{K}(x)$, while the second one [18, Lemma 2.1] (based on the extended Euclidean-Ore scheme) uses $O(r^2)$ operations in $\mathbb{K}(x)$. To our knowledge, no algorithm currently exists similar to the Lehmer-Knuth-Schnöie algorithm [16, Chapter 11], using a number of operations in $\mathbb{K}(x)$ that is quasi-linear in $r$. Li [25] pointed out that algorithms for the LCLM computation that have good complexity with respect to the order, such as the naïve Euclidean-Ore algorithm, do not necessarily behave well because of coefficient growth. He developed a generalisation of the classical subresultant theory to Ore polynomials, that provides determinant formulas and degree bounds for the GCRD and the LCLM [25]. He also compared the practical efficiency of Maple implementations of several algorithms.

Giesbrecht and Zhang [19, Theorem 2.1] mention a complexity bound of $O(r^2d^2)$ for the LCLM computation of two operators of bidegree $(d, r)$ in $(x, \partial)$, based on an unpublished 2002 note of Li. Over fields of characteristic zero, Bostan [5, Chapter 10] sketched a general strategy for com-
puting several constructions on differential operators (including LCLMs), based on an evaluation-interpolation approach on power series solutions. He stated, without proofs, several degree bounds and complexity results. For two operators \(L_1, L_2\) of bidegree \((n, n)\) in \((x, \theta)\), he announced that using fast Hermite-Padé approximation for the interpolation step yields an algorithm that computes LCLM\((L_1, L_2)\) in \(O(n^4)\) operations. The approach was enhanced by van der Hoeven [23], who showed that the costs of the basic operations on differential operators can be expressed in terms of the cost of multiplication in \(\mathbb{K}[x]((\theta))\), and proved the complexity bound \(O(n^6)\) stated without proof in [5, §10.5].

2. PRELIMINARIES

Let \((K, \delta)\) be a differential field, that is, a field \(K\) equipped with an additive map \(\delta : K \to K\) satisfying the Leibniz rule \(\delta(xy) = x\delta(y) + \delta(x)y\) for all \(x, y \in K\). We denote by \(K[\delta]\) the ring of linear differential operators over the differential field \((K, \delta)\). A nonzero element \(L\) in \(K[\delta]\) is of the form

\[
L = a_0 + a_1\delta^{-1} + \cdots + a_n
\]

where \(a_0, a_1, \ldots, a_n \in K\) with \(a_n \neq 0\). We call \(r\) the order of \(L\), and denote it by \(\text{ord}(L)\). The noncommutative ring \(K[\delta]\) is a left (and right) principal ideal domain, for which a Euclidean algorithm exists [27, 28].

Let \(L, L_1, \ldots, L_k\) be nonzero elements in \(K[\delta]\). Then \(L\) is called a common left multiple (CLM) of \(L_1, \ldots, L_k\) if \(L = Q_1L_1 + \cdots + Q_kL_k\) for some \(Q_1, \ldots, Q_k \in K[\delta]\). A common left multiple of the least order is called a least common left multiple (LCLM). Two LCLMs of \(L_1, \ldots, L_k\) are \(K\)-linearly dependent.

Our main focus is on the particular case \(K = \mathbb{K}(x)\), the field of rational functions with coefficients in \(K\), and \(\delta = \frac{d}{dx}\), the usual derivation with respect to \(x\). In this case, we use the notation \(\mathbb{K}(x)((\theta))\) for \(K[\delta]\), and LCLM\((L_1, \ldots, L_k)\) for the primitive LCLM of \(L_1, \ldots, L_k\) in \(\mathbb{K}(x)((\theta))\), that is the LCLM of \(L_1, \ldots, L_k\) computed in \(\mathbb{K}(x)((\theta))\) with trivial content. However, in order to keep the mathematical exposition as independent as possible of any particular case, we stick to the more general setting \(K[\delta]\) whenever we discuss mathematical properties and bird’s-eye view descriptions of algorithms.

Polynomial and matrix arithmetic. The cost of our algorithms will be measured by the number of field operations in \(K\) they use. To simplify the presentation, we assume that polynomials in \(\mathbb{K}[x]_{\leq n}\) (i.e., of degree less than \(n\) in \(x\)) can be multiplied within \(O(n \log(n) \log \log(n)) = \tilde{O}(n)\) operations in \(K\), using the FFT-based algorithms in [33, 31]. Most basic polynomial operations in \(\mathbb{K}[x]_{\leq n}\) (division, extended gcd, interpolation, etc.) have cost \(\tilde{O}(n)\) [16]. We suppose that \(\theta\) is a feasible exponent for matrix multiplication over \(K\), that is, a real constant \(2 \leq \theta \leq 3\), such that two \(n \times n\) matrices with coefficients in \(K\) can be multiplied in time \(O(n^\theta)\). The current tightest upper bound is \(\theta < 2.3727\) [39], following work of Coppersmith and Winograd [13], and Stothers [38].

The following result, due to Storjohann and Villard [35, 37], will be helpful to estimate complexities for solving linear systems arising from LCLM computations. Note that this is currently the best complexity result on polynomial linear algebra. The probabilistic aspects of the algoriths described in this article are entirely inherited from it.

**Theorem 2** [35, 37] Let \(M\) be an \(m \times n\) matrix with entries in \(\mathbb{K}[x]_{<d}\). The rank \(\rho\) of \(M\) can be computed together with \(m - \rho\) linearly independent polynomial elements in the left kernel of \(M\) within \(\tilde{O}(mn^\rho d^2)\) operations in \(K\) by a (certified) randomised Las Vegas algorithm.

Moreover, if \(m = n\), then the determinant of \(M\) can be computed using \(\tilde{O}(n^d)\) operations in \(\mathbb{K}\).

3. LINEAR FORMULATION FOR COMMON LEFT MULTIPLES

In order to connect the computation of common left multiples with linear algebra, we introduce some more notation. For a nonnegative integer \(n\), we denote by \(K[\delta]^\leq n\) the \(K\)-linear subspace of \(K[\delta]\) consisting of all linear differential operators whose orders are at most \(n\). Moreover, we define a \(K\)-linear bijection

\[
\phi_n : K[\delta]^{\leq n} \rightarrow K^{n+1}
\]

\[
\sum_{i=0}^{n} a_i \delta^i \mapsto \left( a_0, a_1, \ldots, a_n \right).
\]

For a nonzero element \(P \in K[\delta]\) of order \(m\), and for an integer \(n\) with \(n \geq m\), we define the Sylvester-type matrix

\[
S_n(P) := \begin{pmatrix}
\phi_n (\partial^{m-n} P) \\
\phi_n (\partial^{m-n-1} P) \\
\vdots \\
\phi_n (P)
\end{pmatrix}
\]

The matrix \(S_n(P)\) has \(n - m + 1\) rows and \(n + 1\) columns. In particular, \(S_n(1)\) is the identity matrix of size \(n + 1\). This matrix enables one to express multiplication by \(P\) in \(K[\delta]\) as a vector-matrix product. Precisely, for \(Q \in K[\delta]^{\leq n-m}\),

\[
\phi_n (PQ) = \phi_{n-m} (Q) S_n (P).
\]

Let \(L_1, \ldots, L_k\) be nonzero elements in \(K[\delta]\). For an integer \(n \geq \max_{1 \leq i \leq k} \text{ord}(L_i)\), the matrix

\[
M_n := \begin{pmatrix}
S_n (L_1) \\
S_n (L_2) \\
\vdots \\
S_n (L_k)
\end{pmatrix}
\]

has \((k+1)(n+1) - \sum_{i=1}^{k} \text{ord}(L_i)\) rows and \(k(n+1)\) columns.

The following theorem is the main result of this section.

**Theorem 3** Let \(L_1, \ldots, L_k\) be elements in \(K[\delta] \setminus \{0\}\) of orders \(r_1, \ldots, r_k\), and let \(n \geq \text{ord}(\text{LCLM}(L_1, \ldots, L_k))\).

(i) If \(L\) is a common left multiple of \(L_1, \ldots, L_k\) such that \(\text{ord}(L) \leq n\) and \(L = Q_1 L_1 + \cdots + Q_k L_k\), then the vector \(\phi_n (\partial_{-r_1} (Q_1), \ldots, \partial_{-r_k} (Q_k), \phi_n (L))\) belongs to the left kernel of the matrix \(M_n\) defined in (2).

(ii) If the vector \((u_1, \ldots, u_k, u)\) is a nonzero vector in the left kernel of \(M_n\), where \(u_i \in K^{n+1-i}\) for \(i = 1, \ldots, k\) and \(u \in K^{n+1}\), then \(u\) is nonzero, and \(\phi_n^{-1} (u)\) is a common left multiple of \(L_1, \ldots, L_k\) with respective left cofactors \(\phi_{n-r_1} (u_1), \ldots, \phi_{n-r_k} (u_k)\).

(iii) If \(\rho\) is the rank of \(M_n\), then \(\text{ord}(\text{LCLM}(L_1, \ldots, L_k)) = \rho + \sum_{i=1}^{k} r_i - k(n+1)\).
Proof. Suppose that \( L = QL_i \) for \( 1 \leq i \leq k \). By (1),
\[
\phi_{n-r_i}(Q_i)S_n(L_i) = \phi_n(L),
\]
which is equivalent to \( \phi_{n-r_i}(Q_i)S_n(L_i) + \phi_n(L)^{-1} = 0 \). Therefore the vector \((\phi_{n-r_i}(Q_1), \ldots, \phi_{n-r_k}(Q_k), \phi_n(L))\) belongs to the left kernel of \( M_n \). The first assertion is proved.

Conversely, suppose that \((u_1, \ldots, u_k, u)\) is a nonzero vector in the left kernel of \( M_n \). Then \( u_iS_n(L_i) + uS_n(-1) = 0 \) for all \( i \) with \( 1 \leq i \leq k \). It follows from (1) that
\[
\phi_{n-r_i}(u_i) = \phi_n^{-1}(u).
\]
Thus, \( u \) is nonzero, for otherwise, since \( \phi_n \) is an isomorphism, all the \( u_i \) would be equal to zero. The second assertion follows.

To prove the last assertion, we set \( L = \text{LCLM}(L_1, \ldots, L_k) \) and \( \ell = \text{ord}(L) \). Assume further that \( L = QL_i \) for all \( i \) with \( 1 \leq i \leq k \). Then, \( L, \partial L, \ldots, \partial^{\ell-1}L \) are common left multiples of \( L_1, \ldots, L_k \) of orders at most \( n \), and such that
\[
\partial^i L = (\partial^i Q_i)L_i \text{ for all } 1 \leq i \leq k \text{ and } 0 \leq j \leq n - \ell.
\]
By the first assertion, for \( 0 \leq j \leq n - \ell \), the vector
\[
v_j = \left( \phi_{n-r_1}(\partial^j Q_1), \ldots, \phi_{n-r_k}(\partial^j Q_k), \phi_n(\partial^j L) \right)
\]
beings to the left kernel of \( M_n \). These vectors are \( K \)-linearly independent because \( L, \partial L, \ldots, \partial^{\ell-1}L \) are. On the other hand, if \((u_1, \ldots, u_k, u)\) is a nonzero vector in the left kernel of \( M_n \), where \( u_i \in K^{n-r_i+1}, \ldots, u_k \in K^{n-1} \), and \( u \in K^n \), then \( \phi_n^{-1}(u) \) is a common left multiple of \( L_1, \ldots, L_k \) with order no greater than \( n \) by the second assertion. Hence, \( \phi_n^{-1}(u) \) is a \( K \)-linear combination of \( \partial^{\ell-1}L \), \( \partial^{\ell-2}L \), \ldots, \( \partial L \), because it is a left multiple of \( L \). Hence, there exist \( c_{n-r_1}, c_{n-r_2}, \ldots, c_{n-k} \) in \( K \) such that
\[
\phi_n^{-1}(u) = c_{n-r_1}\partial^{\ell-1}L + c_{n-\ell}\partial^{\ell-2}L + \cdots + c_0L
\]
which implies that the last \( n - 1 \) coordinates of the vector \((u_1, \ldots, u_k, u) = \sum c_jv_j \) are all equal to zero. Since this vector belongs to the left kernel of \( M_n \), all its coordinates are zero by the second assertion. We conclude that \( \{v_0, \ldots, v_{n-1}\} \) is a \( K \)-basis of the left kernel of \( M_n \), and thus \( n - \ell + 1 \) is its dimension. Then (iii) follows from the rank-nullity theorem, because \( M_n \) has \( (k+1)(n+1) - \sum_{i=1}^{r_1} \) rows.

Since the rank of \( M_n \) is at most \( k(n+1) \), a direct consequence of Theorem 3 (iii) is the following classical result.

Corollary 4 For \( L_1, \ldots, L_k \in K[\partial; \delta] \setminus \{0\} \),
\[
\text{ord}(\text{LCLM}(L_1, \ldots, L_k)) \leq \text{ord}(L_1) + \cdots + \text{ord}(L_k).
\]

4. COMPUTING LCLMS

In this section, we review a few known methods for computing LCLMs and present a new one based on Theorem 3.

4.1 Computing an LCLM of two operators

Given two nonzero elements \( L_1 \) and \( L_2 \) of respective orders \( r_1 \) and \( r_2 \), we consider various methods for computing their LCLMs. The first methods compute left cofactor(s) of the given operator(s) first, and find an LCLM by multiplication in \( K[\partial; \delta] \). The last method is specific to \( K = K(x) \).

4.1.1 Heffter’s algorithm

The first method can be traced back to Brassinin [8], von Escherich [15] and Heffter [21]. The sequence:
\[
\partial^2 L_1, \ldots, \partial L_1, L_1, \partial^{r_1} L_2, \ldots, \partial L_2, L_2
\]
has \( r_1 + r_2 + 2 \) elements, each of which is of order at most \( r_1 + r_2 \). Thus, these elements are \( K \)-linearly dependent. To compute \( \text{LCLM}(L_1, L_2) \), the strategy is to find the maximal integer \( m \) and corresponding elements \( a_1, \ldots, a_{r_1}, \ldots, a_{r_2}, m, a_{2,0}, \ldots, a_{2, r_1 - m} \in K \) with \( a_{1, r_2 - m}a_{2, r_1 - m} \neq 0 \) such that
\[
\sum_{i=1}^{r_1} a_{1, i} \partial^{r_1 - i} L_1 + \sum_{j=m}^{r_2} a_{2, j} \partial^{r_2 - j} L_2 = 0.
\]
Set \( A_1 = \sum_{i=1}^{r_1} a_{1, i} \partial^{r_1 - i} L_1 \) and \( A_2 = \sum_{j=m}^{r_2} a_{2, j} \partial^{r_2 - j} L_2 = 0 \). Then \( A_1 L_1 + A_2 L_2 = 0 \). Therefore, the product \( A_1 L_1 \) is an LCLM of \( L_1 \) and \( L_2 \) due to the maximality of \( m \).

This method can be reformulated using the notation introduced in Section 3. For a vector \( v \neq 0 \) represented by
\[
\left( 0, \ldots, 0, v_{k+1}, \ldots \right), \text{ where } v_{k+1} \neq 0,
\]
in a finite-dimensional \( K \)-vector space equipped with the standard basis \( (1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, \ldots, 0, 1) \), we define \( \mathcal{N}(v) \) to be \( k \). For \( n \geq \max(r_1, r_2) \), define
\[
U_n = \left( S_n(L_1), S_n(L_2) \right).
\]
Then, Heffer’s method consists in computing a vector \( v \neq 0 \) in the left kernel of \( U_{r_1+2} \) such that \( \mathcal{N}(v) \) is maximal.

The next lemma connects the order of \( L = \text{LCLM}(L_1, L_2) \) with the rank of \( U_{r_1+2} \). It easily follows from the observation that a maximal subset of \( K \)-linearly independent elements in \( \{\partial^2 L_1, \ldots, L_1, \partial^{r_1} L_2, \ldots, L_2\} \) consists of \( \partial^2 L_1, \ldots, L_1, \partial^{r_1} L_2, \ldots, L_2 \), where \( \ell = \text{ord}(L) \).

Lemma 5 Let \( L_1, L_2 \) be two nonzero elements in \( K[\partial; \delta] \) of orders \( r_1, r_2 \). Then \( \text{ord}(\text{LCLM}(L_1, L_2)) = \text{rank}(U_{r_1+2}) - 1 \).

4.1.2 Euclidean algorithms

The second family of methods is based on the Euclidean-Ore algorithm for differential operators [28].

Ore’s algorithm. Assume that \( r_1 \geq r_2 \). Setting \( R_1 = L_1, R_2 = L_2 \), one can compute the Euclidean (right) divisions
\[
R_1 = R_2 - Q_1 R_{r_1-1},
\]
for quotients \( Q_i \in K[\partial; \delta] \), and remainders \( R_i \neq 0 \) satisfying \( \text{ord}(R_i) \leq \text{ord}(R_{i-1}) \) for \( i = 3, \ldots, m, \) and \( R_{m+1} = 0 \). Then, as in the commutative case, \( R_m \) is shown to be the GCRD of \( L_1 \) and \( L_2 \). Ore [28, §2] proved that the following product
\[
R_{m-1} R_{m-2} \cdots R_{m-2} \cdots R_4 R_3 R_2 R_1 R_1
\]
is an LCLM of \( L_1 \) and \( L_2 \). (Here \( AB^{-1} \) denotes the exact left quotient of \( A \) and \( B \), that is \( Q \) such that \( A = Q B \).

Extended Euclidean-Ore algorithm. Wedderburn [40, §7-8] observed (see also [9]) that the computation of (4) can be avoided, if one replaces the Euclidean algorithm by its extended version. Precisely, letting \( C_1 = 1, C_2 = 0, \) and
\[
C_i = C_{i-2} - Q_i C_{i-1}, \text{ for } i = 3, \ldots, m,
\]
the product \( (C_{m-1} - Q_{m-1} C_m)R_1 \) is an LCLM of \( L_1 \) and \( L_2 \).
Li's determinantal expression. As in the commutative case, a more efficient version of the extended Euclidean-Ore algorithm is based on subresultants [25, §5]. To avoid technicalities, we present an alternative, efficient, variant of the subresultant algorithm, based on a determinantal formulation [25, Proposition 6.1]. This method assumes that the order of the CCRD of $L_1$ and $L_2$ is already known. Then, one constructs a square matrix $L$ of size $r_1+r_2-2g+2$ whose first $r_1+r_2-2g+1$ columns are the first $r_1+r_2-2g+1$ columns of the matrix \( \begin{pmatrix} S_{r_1+r_2-g}(L_1) \\ S_{r_1+r_2-g}(L_2) \end{pmatrix} \), and whose last column is the transpose of the vector \((\partial^{2-g}, \ldots, \partial, 0, 0, \ldots, 0)\).

If $\det(L)$ is denoted $U$, then $UL_1$ is an LCLM of $L_1$ and $L_2$.

4.1.3 Van der Hoeij's algorithm

The algorithm that we very briefly mention now is specific to the case $K = \mathbb{K}(x)$, where the base field $\mathbb{K}$ has characteristic zero. It works by evaluation-interpolation. The idea, originating from [5], is to perform operations on differential operators by working on their fundamental systems of solutions. Due to space limitations, and in view of its complexity analysis, we do not give more details here, and refer the reader to the article [23].

4.2 Computing an LCLM of several operators

Given several nonzero operators $L_1, L_2, \ldots, L_k \in \mathbb{K}[\partial; \delta]$ we describe various ways to compute $\text{LCLM}(L_1, \ldots, L_k)$.

4.2.1 Iterative LCLMs

An obvious method is to compute an LCLM of $k$ operators iteratively, that is,

$$L = \text{LCLM}(L_1, \text{LCLM}(L_2, \ldots, \text{LCLM}(L_{k-1}, L_k)))$$

(5)

A computationally more efficient (though mathematically equivalent) method is by a divide-and-conquer algorithm, based on the repeated use of the formula

$$L = \text{LCLM}(\text{LCLM}(L_1, \ldots, L_{\lfloor k/2 \rfloor}), \text{LCLM}(L_{\lceil k/2 \rceil+1}, \ldots, L_k))$$

(6)

Of course, the efficiency of an iterative algorithm depends on that of the algorithm used for the LCLM of two operators. This is quantified precisely in Section 5.6.

4.2.2 Van Hoeij's algorithm

Another algorithm for computing the LCLM of $k$ linear differential operators was implemented by van Hoeij as Maple's DEtools[\text{LCLM}] command; it seemingly was never published. For $k = 2$, the method is folklore; it is implicit, for instance, in the proof of [34, Theorem 2.3]. A variant of it is also implemented by the 'diffeq\text{diffeq}' command in Salvy and Zimmermann's \text{gfan} package for Maple [32].

Informally speaking, the method consists in considering a generic solution $h_0$ of $L_j$ for $1 \leq j \leq k$, then in finding the first linear dependency between the row vectors $\partial^i(h_1, \ldots, h_k) = (\partial^i \cdot h_1, \ldots, \partial^i \cdot h_k)$. In order to perform actual computations, these vectors are represented by the canonical forms \(\text{rem}(\partial^i, L_1), \ldots, \text{rem}(\partial^i, L_k)\), for $i = 0, 1, \ldots, \text{rem}(A, B)$ denotes the remainder of the right Euclidean division of $A$ by $B$. Let

$$L = a_0 a^n + a_{n-1} \partial^{n-1} + \cdots + a_0,$$

where $a_0, a_1, \ldots, a_n$ are undetermined coefficients in $K$. For all $i$ with $1 \leq i \leq k$, let $R_i$ be the right remainder of $L$ in the division by $L_i$. Then $L \equiv R_i \mod L_i$. Since $L$ has generic coefficients, ord$(R_i)$ is equal to $r_i - 1$, e.g., by [25, Lemma 2.3]. Note that $a_0, a_1, \ldots, a_n$ depend linearly on the coefficients of the $R_i$'s. There are $s = r_1 + \cdots + r_k$ coefficients in $R_1, \ldots, R_k$. Equating $R_i = 0$, for $i = 1, \ldots, k$, we obtain a linear system

$$(a_n, a_{n-1}, \ldots, a_0) H_n = (0, 0, \ldots, 0),$$

where $H_n$ is an $(n + 1) \times s$ matrix over $K$. Thus, computing LCLM($L_1, \ldots, L_k$) amounts to computing a nontrivial vector $v$ in the left kernel of $H_n$ with $\mathcal{N}(v)$ being maximal. The rank of $H_n$ is equal to the order of $\text{LCLM}(L_1, \ldots, L_k)$, e.g., by [1, Proposition 4.3]. Note that the original version of van Hoeij's algorithm does not make use of this last fact, and potentially needs to solve more linear systems, thus being less efficient when the LCLM is not of maximal order.

4.2.3 The new algorithm

As a straightforward consequence of Theorem 3, the LCLM($L_1, \ldots, L_k$) can be computed by determining a nontrivial vector $v$ in the left kernel of $M_k$ given in equation (2), with $\mathcal{N}(v)$ being maximal. This method computes not only the LCLM, but also its left cofactors $Q_1, Q_2, \ldots, Q_k$, while van Hoeij's algorithm does not compute any cofactor.

5. ALGORITHMS, BOUNDS, COMPLEXITY

In this section, we let $K = \mathbb{K}(x)$ be the field of rational functions with coefficients in a field $\mathbb{K}$, and $\delta = \frac{d}{dx}$ be the usual derivation with respect to $x$. Recall that in this case we use the notation $\mathbb{K}(x)\langle\partial\rangle$ for $K[\partial; \delta]$, and $\text{LCLM}(L_1, \ldots, L_k)$ for the primitive LCLM of $L_1, \ldots, L_k$ in $\mathbb{K}[x]\langle\partial\rangle$.

All algorithms analysed below are specialisations of the algorithms reviewed in the previous section to $K = \mathbb{K}(x)$.

Moreover, we make the non-restrictive assumption that all algorithms take as input linear differential operators with polynomial coefficients, that is, belonging to $\mathbb{K}[x]\langle\partial\rangle$.

The degree of a nonzero operator $L \in \mathbb{K}[x]\langle\partial\rangle$, denoted deg$_L(L)$, is defined as the maximal degree of its coefficients.

As in the case of usual commutative polynomials, deg$_L(AB) = \text{deg}_L(A)+\text{deg}_L(B)$ for all $A, B \in \mathbb{K}[x]\langle\partial\rangle \setminus \{0\}$.

5.1 Tight degree bounds for the LCLM

First, we give a sharp degree bound for LCLMs. As we show later, this bound improves upon the bound that can be derived from van Hoeij's algorithm.

Theorem 6 Let $L_1, \ldots, L_k$ be operators in $\mathbb{K}[x]\langle\partial\rangle \setminus \{0\}$. Let $s = \text{ord}(L_1) + \cdots + \text{ord}(L_k)$, and $d = \max_{i=1}^{k} \text{deg}_L(L_i)$. If $L = \text{LCLM}(L_1, \ldots, L_k)$, then deg$_L(L) \leq d(k(s + 1) - s)$.

Proof. By Corollary 4, ord$(L) \leq s$. It follows from Theorem 3 and Cramer’s rule that every nonzero coefficient of $L$ is a quotient of two minors of $M_k$. Note that every square submatrix of $M_k$ has size at most $k(s + 1)$, since $M_k$ has $k(s + 1) + 1$ rows and $k(s + 1)$ columns. Thus, the degree of the determinant of such a submatrix is bounded by $d(k(s + 1) - s)$, because every entry of $M_k$ is of degree at most $d$, and the last $s + 1$ rows of $M_k$ are free of $x$.}$
5.2 LCLMs of two operators

The following result encapsulates complexity analyses of LCLMs for two operators. Heffter’s, van Hoeij’s and our new algorithm are summarised in Figure 2.

**Theorem 7** Let $L_1, L_2 \in \mathbb{K}[x][\partial]$ be operators of bidegrees at most $(d,r)$ in $(x,\partial)$. Then it is possible to compute the LCLM of $L_1$ and $L_2$ in complexity

(a) $O(\min(r^2d^2, r^3d))$ by Heffter’s and van der Hoeven’s algorithms,
(b) $O(r^{d+1}d)$ by Li’s and by van Hoeij’s algorithms,
(c) $O(r^d d)$ by the new algorithm.

**Proof.** By [23, Theorems 5, 8 & 23], and using bounds from Theorem 6, the complexity of van der Hoeven’s algorithm is $O(\min((rd)^2, rd^2, r^3d))$. The most costly parts of Heffter’s algorithm are Steps 2, 4 and 6. Since the matrix $U_{1+r,2}$ has size $O(r)$ and polynomial coefficients of degree at most $d$, the rank and kernel computations involved in Steps 2 and 4 can be performed using $O(r^d d)$ operations, by Theorem 2. Step 6 consists in multiplying two operators in $\mathbb{K}[x][\partial]$ of bidegrees at most $(r−1)d, r)$ and $(d, r)$ in $(x,\partial)$. This can be done using $O(\min((rd)^2, rd^2, r^3d))$. This proves (a).

The dominant parts of Li’s algorithm are the computation of $g = \gcd(\gcd(L_1, L_2))$, and the expansion of $O(r)$ minors of a polynomial matrix of size $O(r)$ and degree at most $d$. By using [20, Lemma 5.1] and Theorem 2, $g$ can be computed using $O(r^d d)$ operations in $\mathbb{K}$, and the minors can be expanded in $O(r^{d+1}d)$. The dominant parts of van Hoeij’s algorithm are Steps 2 and 4. Since $k = 2$, matrix $H_r$ has size $O(r)$. By an easy induction, its $(r+j)$th row has polynomial coefficients of degree at most $2jd$, thus $H_r$ has degree $O(dr)$. By Theorem 2, the rank and kernel computations have complexity $O(r^{d+1}d)$. This proves (b).

The dominant parts of the new algorithm are Steps 2 and 4. Since $k = 2$, the polynomial matrix $M_r$ has size $O(r)$ and degree at most $d$. By Theorem 2 again, the rank and kernel computations have cost $O(r^d d)$. This completes the proof. □

Quite surprisingly, the costs of Heffter’s and of van der Hoeven’s algorithms are penalised by the complexity of multiplication of operators, which is not well-understood yet for general bidegrees. Precisely, it is an open problem whether two operators of bidegree $(d,r)$ in $(x,\partial)$ can be performed using $O(r^d d)$, improving the corresponding entries in Figure 1.

5.3 LCLMs of several operators

We analyse three algorithms for LCLMs of several operators: DAC, van Hoeij’s and our new algorithm.

5.3.1 LCLMs by divide-and-conquer

**Theorem 8** Suppose that we are given an algorithm computing the LCLM of two differential operators which, on input $L_1, L_2 \in \mathbb{K}[x][\partial]$ of bidegree at most $(D,R)$ in $(x,\partial)$, computes LCLM($L_1, L_2$) in complexity $O(R^d D^\beta)$ for some constants $\alpha \geq 2$ and $\beta \geq 1$ independent of $D$ and $R$.

There exists an algorithm which, on input $L_1, \ldots, L_k \in \mathbb{K}[x][\partial]$ of bidegrees at most $(d,r)$ in $(x,\partial)$, computes $L = \text{LCLM}(L_1, \ldots, L_k)$ using $O(k^{\alpha+2}\alpha^\alpha \beta^\beta d^\beta)$ operations in $\mathbb{K}$.

**Proof.** Suppose without loss of generality that $k = 2^t$ is a power of 2. To compute $L$, we use a strategy based on [6], similar to that of the subproduct tree [16, §10.1]: we partition the family $(L_1, \ldots, L_k)$ into pairs, compute the LCLM of each pair using algorithm $\text{A}$ available for two operators, remove the polynomial content, then compute LCLMs of pairs, and so on. Let $L_{i\ldots j}$ denote the LCLM of $L_i, \ldots, L_j$ with the content removed. At level 1, the algorithm computes the $k/2$ operators $L_{1,2}, \ldots, L_{k-1,k}$, at level 2 the $k/4$ operators $L_{1,4}$, \ldots, $L_{k-4,k}$, and so on, the last computation at level $\ell$ being that of $L$ as the LCLM of $L_{i\ldots j}$ and $L_{j/2+1\ldots i}$. Let $C(R, D) = O(R^\alpha D^\beta)$ define the complexity of algorithm A on inputs of bidegrees at most $(D, R)$. By Theorem 6, the operators computed at level $1 \leq \ell \leq \ell'$ have bidegree at most $(2^{\ell}r, 2^\ell d((2^\ell−1)r+1))$, plus the cost of the content removal, which is negligible. Up to polylogarithmic factors, the cost is bounded by

$$\sum_{j=0}^{\ell−1} \frac{k}{2^{\ell+1}} (2^{\ell}r)^{\alpha} (4^\ell d)^{\beta} = \frac{k}{2} r^{\alpha+\beta} d^{\beta} \sum_{j=0}^{\ell−1} (2^j)^{\alpha+2\beta−1},$$

which is $O(k^{\alpha+2\beta}r^{\alpha+\beta} d^\beta)$. This completes the proof. □
The cost of the algorithm is essentially that of its last step; this is a typical feature of DAG algorithms. A similar analysis shows that the iterative algorithm based on formula (5) is less efficient, and has complexity $O(k^{r+2})$.

As a corollary of Theorems 7 and 8, we get a proof of the complexity estimates in the first three entries of Figure 1.

5.3.2 Van Hoeij’s and the new algorithm

Theorem 9 Let $L_1, \ldots, L_k \in \mathbb{K}[x](\partial)$ have bidegrees at most $(d, r)$ in $(x, \partial)$. One can compute LCLM($L_1, \ldots, L_k$) in $O(k^{r+1}d)$ operations by van Hoeij’s algorithm,

(a) in $O(k^{r+1}d)$ operations by van Hoeij’s algorithm.

(b) in $O(k^{r+1}d)$ operations by the new algorithm.

Proof. The proof is similar to that of Theorem 7(b) and (c). The most costly parts of van Hoeij’s algorithm are Steps 2 and 4. Matrix $H_1$ has size $O(kr)$ and polynomial coefficients of degree $O(kr^d)$. By Theorem 2, the rank and kernel computations have complexity $O((kr)^d r^d)$.

As a corollary of Theorem 9, we get a proof of the complexity estimates in the last two entries of Figure 1. Note that Cramer’s rule applied to the matrix $H_1$ analyzed in the previous proof yields the bound $O(k^{r+1}d)$ on the coefficient degrees of the LCLM. This bound is improved by Theorem 1.

6. SMALLER COMMON LEFT MULTIPLES

Our approach to computing more common left multiples (CLMs), that are generally not of minimal order, but smaller in total arithmetic size than the LCLM, is similar to the linear-algebraic approach used in Section 3. However, instead of considering a matrix encoding the $\partial^i L_j$, with polynomial coefficients, we turn our attention to a matrix encoding the $x^i \partial^j L_j$, with constant coefficients.

Existence of smaller CLMs. The new building block to consider is, for an operator $P$ in $\mathbb{K}[x](\partial)$ of total degree $\Delta$ in $x$ and $\partial$, and an integer $N \geq \Delta$, the $(N+4) \times (N+2)$ matrix $C_N(P)$ with scalar coefficients whose rows represent the operators of the form $x^i \partial^j P$ for $0 \leq i + j \leq N$, in any fixed order, and whose columns are indexed by the monomials of total degree at most $N$, in any fixed order.

Let $L_1, \ldots, L_k$ be elements of $\mathbb{K}[x](\partial)$, with respective total degrees $\Delta_1, \ldots, \Delta_k$. For $N \geq \max\{\Delta_1, \ldots, \Delta_k\}$, let $M_N(L_1, \ldots, L_k)$ be the matrix

$$M_N = M_N(L_1, \ldots, L_k) = \begin{pmatrix} C_N(L_1) & \cdots & C_N(L_k) \\ -I_{(N+2)} & \cdots & -I_{(N+2)} \end{pmatrix}.$$ 

This matrix has $m(N)$ rows and $n(N)$ columns, where

$$m(N) = \left(\frac{N + 2}{2}\right) + \sum_{j=1}^{k} \left(\frac{N - \Delta_j + 2}{2}\right), \quad n(N) = k \left(\frac{N + 2}{2}\right).$$

Assuming all $\Delta_j$ equal to a same value $\Delta$, the matrix $M’_N$ certainly has a nontrivial left kernel when $m(N) > n(N)$, that is when $k(\frac{N - \Delta + 2}{2}) > (k-1)(\frac{N + 2}{2})$, which happens when

$$N \geq B \text{ for } B = \left[ \frac{k\Delta + \sqrt{4k(k-1)\Delta^2 + 1 - 3}}{2} \right] \leq 2k\Delta,$$

where the approximation holds for large values of $k$ or $\Delta$.

Using $\Delta \leq d + r$ yields the main result of this section.

Theorem 10 Let $L_1, \ldots, L_k$ be elements in $\mathbb{K}[x](\partial) \setminus \{0\}$ of orders at most $r$, and with coefficients of degrees at most $d$.

There exist nonzero common left multiples of total degree $\leq 2k(d + r)$ in $(x, \partial)$, and total arithmetic size $O(k^{r+1}(d + r)^2)$.

Algorithms for CLMs. A simple algorithm for computing common left multiples of total degree $B \approx 2k(d + r)$ in $(x, \partial)$ is based on the left kernel computation of the scalar matrix $M_N(L_1, \ldots, L_k)$. This matrix has sizes of order $kB^2/2 \approx 2k(d + r)^2$. The cost of the procedure is $O(k^{r+1}(d + r)^2)$; it is dominated by the kernel computation.

To simplify the discussion, we assume in the remaining of the section that $L_1, \ldots, L_k$ have bidegrees at most $(d, r) = (n, n)$. Then, Theorem 10 implies that, while the LCLM has order at most $kn$ and degrees at most $k^2n^2$, there exist common left multiples of order and degree at most $4kn$.

However, computing such a small multiple by the previous algorithm of complexity $O(k^{r+1}n)$ is more costly than computing the LCLM by the last two algorithms in Figure 1.

Here we briefly sketch a faster algorithm for computing a common left multiple of order and degree at most $4kn$, based on Hermite-Padé approximation [5, Chapter 10]. One determines series solutions of the $L_i$ at order $O(k^{r+2})$, takes a random linear combination $f$ of them, computes its first $4kn$ derivatives, and outputs a Hermite-Padé approximant of $f', \ldots, f(4kn)$ of type $(4kn, \ldots, 4kn)$. The dominant complexity is that of the Hermite-Padé step, $O(k^{r+1}n^{3/4})$ [36].

A Fast LCLM Heuristic. As an interesting consequence of this fast CLM computation, we deduce a very efficient heuristic for LCLMs, asymptotically faster than all algorithms in Figure 1. It proceeds in 3 steps: (i) compute $O(1)$ CLMs of order and degree at most $4kn$; (ii) take two random linear combinations with coefficients in $\mathbb{K}[x]$; (iii) return their GCRD. The dominating steps are (i) and (iii). By using Hermite-Padé approximants for step (i) and Grigoriev’s algorithm [20] combined with Theorem 2 for step (iii), the total complexity is $O(k^{r+1}n^{3/4})$. This is nearly optimal, in view of the LCLM size $k^{n^3}$. However, we are not yet able to turn this heuristic into a fully proved algorithm.

7. EXPERIMENTS

We implemented\(^1\) two variants of our new algorithm in Magma V2.16-7 [4] and compared them with Magma’s built-in LCLM routine (command `LeastCommonLeftMultiple`).

Some experimental results are summarised in Table 1. We take as input $k = 2$ random operators in $\mathbb{F}[x](\partial)$, each of bidegree $(d, r) = (n, n)$ in $(x, \partial)$, where $p$ is a medium-sized prime and $n$ is of the form $2^i/7$, for $2 \leq j \leq 11$. Column `New gives timings` for the first variant of the new algorithm, that uses Magma’s built-in polynomial linear algebra solver (the `Kernel` routine), while column `New+5` gives timings for the second variant, based on our own high-level implementation of Storjohann’s high-order lifting algorithm [35]. Column $(N, D)$ displays the size $N$ and the degree $D$ of the polynomial matrix dealt with by algorithms `New` and `New+5`. The dominating part of these algorithms is the left kernel computation for a polynomial matrix of size $(N+1) \times N$ and

\(^1\)All computer calculations were performed on a Quad-Core Intel Xeon X5160 processor at 3GHz, with 8GB of RAM.
degree $D$. The most consuming part of New+S consists in $O((\log N)^2)$ polynomial matrix multiplications of size $N$ and degree $D$. To facilitate comparisons, column MM(N,D) shows the total time taken by 10 products of random polynomial matrices of size $N$ and degree $D$ over $\mathbb{F}_p$. Finally, column output size displays the total arithmetic size of the computed LCLM, that is, its number of coefficients in $\mathbb{F}_p$.

Several conclusions can be drawn from Table 1. First, Magma’s LCLM tool exhibits an exponential arithmetic complexity behaviour (when passing from bidegree $(n, n)$ to $(n+1, n+1)$, timings are multiplied by a factor close to 1.5), but it is relatively efficient for small input sizes. Both variants of the new algorithm are faster for $n \geq 10$, and New+S gains a factor 65 for $n = 23$, and almost 1300 for $n = 46$.

Second, timings in column New exhibit a practical complexity proportional to $n^8$, which is inherited from Magma’s linear algebra solver on polynomial matrices. In contrast, New+S has a practical complexity proportional to $n^{3.5}$ (but with a higher proportionality factor). This good behaviour, closer to the theoretical complexity $O(\alpha^{3.1})$ predicted by Theorem 1, is inherited from Magma’s very efficient polynomial matrix multiplication, through Storjohann’s algorithm.

Finally, timings in column New+S grow nearly linearly in the corresponding output sizes given in the last column, and these sizes match exactly the sharp bounds in Theorem 1. This experimentally confirms that size bounds and worst-case complexity analyses predicted by our theoretical results are reached in generic cases.

**REFERENCES**


