Computing the sign or the value of the determinant of an integer matrix, a complexity survey[☆]

Erich Kaltofen[a], Gilles Villard[b],[*]

[a]*Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA*
[b]*CNRS, Laboratoire LIP, École Normale Supérieure de Lyon 46, Allée d'Italie, 69364 Lyon Cedex 07, France*

Abstract

Computation of the sign of the determinant of a matrix and the determinant itself is a challenge for both numerical and exact methods. We survey the complexity of existing methods to solve these problems when the input is an $n \times n$ matrix A with integer entries. We study the bit-complexities of the algorithms asymptotically in n and the norm of A. Existing approaches rely on numerical approximate computations, on exact computations, or on both types of arithmetic in combination.
© 2003 Elsevier B.V. All rights reserved.

1. Introduction

Computing the sign or the value of the determinant of an $n \times n$ matrix A is a classical problem. Numerical methods are usually focused on computing the sign via an accurate approximation of the determinant. Among the applications are important problems of computational geometry that can be reduced to the determinant question; the reader may refer to [11,12,9,10,46,45] and to the bibliography therein. In symbolic computation, the problem of computing the exact value of the

determinant is addressed for instance in relation to matrix normal forms problems [42,29,24,50] or in computational number theory [17].

In this paper we survey the known major results for computing the determinant and its sign and give the corresponding references. Our discussion focuses on theoretical computational complexity aspects. For an input matrix $A \in \mathbb{Z}^{n \times n}$ with the infinity matrix norm $\|A\|$, we estimate the worst case bit-complexity in terms of n and $\|A\|$. If $a_{i,j}$ denotes the integer in row i and column j of A, $A = (a_{i,j})_{1 \leqslant i,j \leqslant n}$, then $\|A\| = \max_{1 \leqslant i \leqslant n} \sum_{j=1}^{n} |a_{i,j}|$ and any entry in A has bit-length bounded by

$$\min_{1 \leqslant i,j \leqslant n} \{\beta : |a_{i,j}| < 2^{\beta}, \beta \geqslant 1\} \leqslant 1 + \log(\|A\| + 1).$$

In algebraic complexity—i.e. when counting the number of operations in an abstract domain—we refer to Strassen [52] and Bunch and Hopcroft [13] for the reduction of the problem of computing the determinant to matrix multiplication. Conversely, Strassen [53] and Bunch and Hopcroft [13] reduce matrix multiplication to matrix inversion, and Baur and Strassen reduce matrix inversion to computing the determinant [7]. See also link with matrix powering and the complexity class GapL following Toda, Vinay, Damm and Valiant as explained in [3], for example. Valiant's theorem shows that the determinant is universal for formulas [54].

For integer matrices, computing the sign of the determinant is at least not harder than computing its value. We try to identify the differences between these two problems even if it is not known whether the two complexities are asymptotically different in the worst case. Numerical methods must deal with condition numbers that influence the precision of the computations. Symbolic methods are confronted with intermediate coefficient growth and invariant structure of the matrix that directly influence the costs. We show some techniques for devising algorithms sensitive to these conditions, and we state either the worst case bit-complexities bounds or bounds depending on some additional properties. This implies discussion on algorithms adapted to certain favorable situations, i.e., on classes of input matrices that require much lower running time than for the worst case inputs.

The known upper estimates for the bit-complexity of the sign or the determinant is progressively decreasing. In particular, for the determinant the bit-complexity is known to be below the algebraic complexity times the maximum bit-size of the output (see [34,25,37] and Section 6). This motivates us to focus on the sequential time complexity rather than on other aspects such as memory resources, parallel time or practical considerations. We discuss deterministic and randomized algorithms. The usage of random bits leads to Monte Carlo algorithms where the answer is with controllably high probability correct but not certified (known to be correct); and to Las Vegas algorithms where the answer is always correct and produced quickly with high probability.

The paper is organized as follows. Section 2 recalls classical approximate and exact results about the determinant. Section 3 discusses the sign computation using numerical methods based on floating point numbers. The complexity, because of the precision required for intermediate values, is quite directly driven by the condition number. A typical problem is to have algorithms sensitive to this quantity. Symbolic algorithms on integers frequently rely on Chinese remaindering. In Section 4 exact computation approach with randomization enables complexity estimates sensitive to the size of the determinant. The same approach may also be reduced to constant precision computations for determining the sign. In Sections 5 and 6 we focus on other exact methods. Existing fast algorithms fall into two categories. The first category takes advantage of linear system solving, a problem whose worst case bit-complexity is currently lower than the complexity of the determinant. The second

category relies on Krylov–Lanczos–Wiedemann approaches combined with "baby-steps, giant-steps" strategy to control the integer size growth and hence the cost. In particular, Section 5 deals with the Smith normal form which somehow currently "expresses" the difference between binary system solution and determinant. Section 6 is concerned with improved worst case bounds and presents the known asymptotically fastest algorithms. In Section 7 we briefly consider computations for matrices with sparse high precision numbers under a distinct model of computation. The last section is our conclusion with some discussion of previous results.

We assume that multiplying two arbitrary $n \times n$ matrices over a ring R costs $O(n^\omega)$ operations in R. Using standard multiplication gives $\omega = 3$ while asymptotically fast matrix multiplication allows $\omega = 2.376$ [20] and special exponents if the input matrices are rectangular [19,33]. The bit-complexity of multiplying a pair of l-bit integers or floating point numbers is $O(l^2)$ using the straightforward algorithm or $O^\sim(l)$ with a fast algorithm [48]. Here and in the following we will use the soft "O" notation, that is, for any exponent e_1, $O^\sim(n^{e_1})$ denotes $O(n^{e_1}(\log n)^{e_2})$ for some constant exponent e_2. Unless specified otherwise we use the classical cubic complexity algorithm for matrix multiplication and the essentially linear FFT-based one for the numbers. Our model of computation is a random access machine under the logarithmic cost criterion [2, Section 1.3]. The algorithms discussed here can be also implemented on a multi-tape Turing machine, perhaps with a poly-logarithmic slow-down. The worst case bit-cost for computing the sign of the determinant of an $n \times n$ matrix A with infinity norm $\|A\|$ is denoted by $\mathscr{S}_{n,\|A\|}$, the worst case bit-cost for computing the determinant is $\mathscr{D}_{n,\|A\|}$. Hence we have $\mathscr{S}_{n,\|A\|} \leqslant \mathscr{D}_{n,\|A\|}$. For adaptive algorithms (see definition 2) these functions are bounded also in terms of some quantities other than n and $\|A\|$, e.g., the size of the determinant, the condition number, the orthogonal defect or the number of invariant factors, in which case we write the matrix as an argument, namely $\mathscr{S}_{n,\|A\|}(A), \mathscr{D}_{n,\|A\|}(A)$.

2. Classical results on sign and determinant computation

In constant precision computation, the condition number of the determinant plays a central role. Following Higham [32, Problem 13.15], we define this number as follows:

$$\log \operatorname{cond}_{\det} A = \log \max_{i,j} |a_{i,j}(A^{-1})_{i,j}| \leqslant \log(\varDelta(A)\|A\|), \tag{1}$$

where

$$\varDelta(A) = \frac{\prod_{i=1}^n \|a_{i,*}\|_2}{|\det A|} \tag{2}$$

denotes the orthogonality defect of A. Thus the logarithm of the condition number may be as large as $O^\sim(n \log \|A\|)$. For error estimation we can use the numerical rule of thumb [32, p. 10]:

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error}$$

and may also take logarithm on both sides. The consequence is the well-known fact that if one uses a constant precision arithmetic, the output precision for the determinant satisfies:

$$\text{precision} \lesssim \log \operatorname{cond}_{\det} A + \log(\text{backward error}).$$

For accurate computations (with a low relative error for certifying the sign) on badly conditioned matrices (having small determinants compared to $\|A\|$ for instance) this implies that it is potentially

necessary to compute with $O^{\sim}(n \log \|A\|)$-bit numbers. We assume that the logarithm of the backward error—say for computing the determinant from a *LU* or a *QR* decomposition—is in $O(\log^{\alpha} n + \log \|A\|)$ for some α [32, Chapter 9]. With a matrix decomposition using $O(n^3)$ arithmetic operations, the bit-cost for the sign is thus bounded by

$$\mathscr{S}_{n,\|A\|} = O^{\sim}(n^3 \cdot n \log \|A\|) = O^{\sim}(n^4 \log \|A\|). \tag{3}$$

This theoretical formula may be of weak interest numerically. As soon as a family of matrices with a small condition number and an algorithm ensuring a small backward error are considered, the asymptotic bit-cost is in $O^{\sim}(n^3 \log\|A\|)$.

In symbolic computation, most of the difficulties in reducing the bit-complexity are due to the impact of the size of the determinant. We know by Hadamard's inequality [30, Theorem 16.6] that

$$\log |\det A| \leqslant (n/2) \log n + n \log \|A\|,$$

therefore, representing the determinant may require up to $O^{\sim}(n \log \|A\|)$ bits. A detailed analysis of the average accuracy of Hadamard's bound can be found in [1]. Once a bound is found, the determinant can be computed by Gaussian elimination with the sizes of intermediate integers controlled by exact division or by Bareiss's more sophisticated method [6]. Another approach [26,14] is to use matrix arithmetic modulo primes and Chinese remaindering (on this technique see [2, Theorem 8.9] or [8, Problem 4.2]). The cost for the exact computation of the determinant, based on the fast reduction of the matrix entries modulo several primes, is [30, Chapter 5]:

$$\mathscr{D}_{n,\|A\|} = O^{\sim}(n^3 \cdot n \log \|A\|) = O^{\sim}(n^4 \log \|A\|). \tag{4}$$

If fast matrix multiplication is available these estimates can be decreased. Fast multiplication can be plugged into block algorithms, we refer to Demmel and Higham [22] or Higham [32, Chapter 22] for numerical approaches. For algebraic and symbolic aspects we refer to Bini and Pan [8, Chapter 2]. The bit-cost of computing the determinant is

$$\mathscr{D}_{n,\|A\|} = O^{\sim}(n^{\omega+1} \log \|A\|) < O(n^{3.376} \log \|A\|).$$

Remark 1. A sub-problem of the computation of the sign or the determinant is to determine whether a matrix is invertible or not—whether the determinant is nonzero or not. This can be done by testing singularity modulo a randomly chosen prime number p. If p is chosen in a sufficiently large set (large with respect to n and $\log \|A\|$), this leads to a randomized Monte Carlo algorithm (non certified) for testing singularity using $O^{\sim}(n^3 \log \log \|A\| + n^2 \log \|A\|)$ bit-operations. One can choose p in a set of primes having $O(\log n + \log \log \|A\|)$ bits (see [31, Section 3.2]). This technique may also be applied to Monte Carlo rank computations and is related to the randomization of Section 4. A singularity certificate based on system solution will be given in Remark 3.

3. Numerical computation of the sign

As opposed to using exact arithmetics, specialized algorithms based on floating point operations have been intensively studied for computing the sign of algebraic expressions in general and of the determinant in particular. As seen above, using a small precision may support correct answer for

special classes of matrices or on the average but a high precision is needed in the worst case. An interesting problem is to conceive *adaptive* algorithms that automatically take into account these variations of the precision.

Here is our attempt to define this paradigm of algorithm design.

Definition 2. An algorithm is *adaptive* (*input-sensitive*, *output-sensitive*, *introspective*) if its complexity is asymptotically below its worst case complexity for a nontrivial subset of its inputs.

In the field of algebraic computing, important examples are Lenstra's elliptic curve integer factorization algorithm or Zippel's sparse polynomial interpolation algorithm. Others utilize a so-called "early termination" test. We discuss early termination for Chinese remaindering in Section 4.

One of the first specialized numerical method for the determinant, which adapts the mantissa length of floating point numbers, is due to Clarkson [16] (see also [11,12]). His algorithm works in two steps. From the input matrix A, the first step is to accurately compute a matrix B which columns are "more orthogonal" than those of A. The process iteratively follows the Gram-Schmidt orthogonalization but remains in a lattice and keep the sign of the determinant unchanged. For a better comparison with the exact methods, it is interesting to note that this process uses ideas from the Lenstra, Lenstra and Lovász basis reduction algorithm [39]. Using good properties of B, especially a low orthogonality defect, the second step consists in computing the sign of the determinant by LU decomposition. The first step involves computations with the precision of at most $\log \|A\| + \mathrm{O}(n)$ bits [12]. The arithmetic cost depends on the orthogonality defect $\Delta(A)$ of A (see (2)) which is, similarly to the condition number, in $\mathrm{O}^\sim(n \log \|A\|)$. When A is invertible, the defect bounds the number of iterations of the first step of the algorithm. The overall cost is given by

$$\mathscr{S}_{n, \|A\|}(A) = \mathrm{O}^\sim((n^3 + n^2 \log \Delta(A))(n + \log \|A\|)). \tag{5}$$

We may notice that using Remark 1, the invertibility can be easily tested. In the refined algorithms of Brönnimann and Yvinec [11,12], even for singular matrices the bit-cost satisfies:

$$\mathscr{S}_{n, \|A\|} = \mathrm{O}^\sim(n^4 \log \|A\| + n^3 \log^2 \|A\|). \tag{6}$$

The first step of Clarkson's approach is output sensitive since its cost depends on the magnitude of the determinant. Favorable inputs are matrices with "not too small" determinants, for instance with

$$\log \Delta(A) = \mathrm{O}(n). \tag{7}$$

In these cases the algorithm requires only $\mathrm{O}^\sim(n^4 + n^3 \log \|A\|)$ bit-operations. From (1), this corresponds to matrices such that the condition number satisfies $\log \mathrm{cond}_{\det} A = \mathrm{O}(n + \log \|A\|)$ and not $\Omega(n \log \|A\|)$ as in the worst case. Likewise, the lattice algorithm of Brönnimann et Yvinec [12] generalizes to high dimensions the method of Avnaim et al. [4] for dimensions 2 and 3, to yield a bit-complexity bound similar to (6).

To achieve better complexity bounds for well conditioned matrices, arithmetic filtering has been much studied, especially for algebraic geometry problems (see the introduction). The idea is to rapidly evaluate the sign of the determinant using fast floating point computations and then to certify the sign using an error bound or some other fast certificates [28,38,45]. Existing filters/certificates rely on computed or estimated round-off errors and distances to singular matrices. In particular, evaluations of latter distances with a machine epsilon $\varepsilon = \mathrm{O}(\log n)$ allows the filters in [38,45] to work correctly

for well conditioned matrices. If the condition number is small—say $\log \text{cond}_{\det} A = \text{O}(\log n)$—then the rank is certified using $\text{O}^{\sim}(n^3 \log \|A\|)$ operations. More generally, with a singularity test as in Remark 1 and as suggested by Pan in [45, p. 715], by repeatedly doubling the precision, this leads to the theoretical bound

$$\mathscr{S}_{n, \|A\|}(A) = \text{O}^{\sim}(n^3(\log \text{cond}_{\det} A + \log \|A\|)) = \text{O}^{\sim}(n^4 \log \|A\|). \tag{8}$$

As one could naturally expect, this is highly sensitive to the condition number.

4. Chinese remaindering

Approaches based on computations modulo a collection of primes, together with the reconstruction of integers using Chinese remaindering, are common in symbolic computation. In a way analogous to numerical algorithms that are sensitive to the condition number, Chinese remaindering leads to exact algorithms that are sensitive to the determinant. Here and in subsequent sections the techniques need randomizations. The idea is to compute residues of the determinant modulo primes and to reconstruct the integer value of the determinant "on the fly" (via Newton's method, mixed radix representations). The approach enables acceleration depending on the size of the determinant. Once the reconstructed value remains stable for a relatively small number of consecutive primes then the determinant is correct with constant probability on any input. The corresponding bit-cost is:

$$\mathscr{D}_{n, \|A\|} \leqslant \text{O}^{\sim}(n^3 \log |\det A| \log \log \|A\| + n^2 \log \|A\| + \log^2 |\det(A)|). \tag{9}$$

On this early termination technique the reader is referred to the detailed study in Brönnimann et al. [10] and to Emiris [27] for remarks on success probabilities. Even if the output is not certified (Monte Carlo algorithm), this gives very good results especially for small determinants [10, Tables 2 & 3]. The $\log^2 |\det(A)|$ term in (9) could be reduced by doubling the number of moduli in each Chinese remainder update before checking if the result changes.

For the computation of the sign only, the authors of [10] also propose an implementation of Chinese remaindering with constant precision numbers such as usual floating point ones (via Lagrange's method). The technique generalizes the one in [5] for integer division. However, in sign computations, the integer reconstruction is not the bottleneck and theoretical costs here remain bounded as in (4).

5. Exact determinant and linear system solution

The first type of fast exact algorithms for computing the determinant exploits Cramer's rule and the relations between solving linear systems and determinant computation. Both under algebraic complexity models and the worst case bit-complexity it remains an open question whether linear system solution is equivalent or strictly easier asymptotically than the determinant computation [7, p. 328]. For solving a linear system exactly over the rationals, we refer to the p-adic approach (Hensel's lifting) proposed by Moenck and Carter [40], then by Dixon [23]. The bit-complexity for solving $Ax = b$ with $b \in \mathbb{Z}^n$ and $\|b\| \leqslant \|A\|$ is bounded by

$$\mathscr{L}_{n, \|A\|} = \text{O}^{\sim}(n^3 \log \|A\|). \tag{10}$$

Hence, using classical matrix multiplication, exact system solution in the worst case has the asymptotic cost of numerical determinant computation for well conditioned matrices (see Section 2). Further, as shown by Mulders and Storjohann [41, Lemma 5.7], fast matrix multiplication techniques can be used and give:

$$\mathscr{L}_{n,\|A\|} = O^{\sim}(n^{f(\omega)} \log \|A\|), \quad \text{with } f(\omega) \approx 2.76. \tag{11}$$

In [51], for an abstract commutative field K, Storjohann shows that if A and b have polynomial entries of degree d in $K[x]$ then the linear system $Ax = b$ may be solved in $O^{\sim}(n^{\omega}d)$ operations in K. An extension of this result to the integer case could achieve $\mathscr{L}_{n,\|A\|} = O^{\sim}(n^{\omega} \log \|A\|)$ (see [51, Section 14] and the discussion at the end of Section 6).

Pan has proposed, in [43, Appendix] and in [44], a way to compute the determinant of A using denominators of solutions to random systems:

$$Ax = b, \quad b \text{ a random vector.} \tag{12}$$

Since the bit-cost of system solution is low, this idea should represent a gain. However, under the influence of the invariant structure of the matrix—the *Smith normal form* [42]—the gain does not appear directly in the worst case. As experimentally studied by Abbott et al. [1] the gain is clear on the average and in some propitious cases. Abbott et al. proceed in two phases. At first one solves several random systems (12) to compute a large divisor σ of the determinant. The second phase finds the missing factor $(\det A)/\sigma$ using classical Chinese remaindering. With (10), the two phases lead to the bit-cost bound

$$\mathscr{D}_{n,\|A\|}(A) = O^{\sim}\left(n^3 \left(\log \frac{\prod_{i=1}^{n} \|a_{i,*}\|}{|\sigma|} + \log \|A\| \right) \right). \tag{13}$$

This is (4) in the worst case. Similarly to the discussion in Section 3, advantageous cases are those of matrices leading to large $|\sigma|$. For random matrices, heuristic arguments in [1, Assumption 1] (see also some related expected values in [25, Section 6]) give

$$\log \frac{\prod_{i=1}^{n} \|a_{i,*}\|}{|\sigma|} = O(n).$$

This may be compared to (7). For such matrices the cost becomes $O^{\sim}(n^4 + n^3 \log^2 \|A\|)$. Using randomization, one can go further on sensitivity aspects. Indeed [1, Section 4], when solution vectors x are vectors of reduced rational fractions then

$$\sigma \,|\, s_n \text{ and } \log \frac{\prod_{i=1}^{n} \|a_{i,*}\|}{|\sigma|} = \log \Delta(A) + \log \frac{|\det A|}{s_n} + \log \frac{s_n}{|\sigma|}, \tag{14}$$

where s_n is the Smith largest invariant factor of A (largest nonzero diagonal entry of the Smith form). The term in $\log(s_n/|\sigma|)$ introduced by (14) in the cost (13) is limited to $O(1)$ [1, Lemma 1]. The term in $\Delta(A)$ can be avoided by the early termination randomized strategy seen in Section 4. This leads to a Monte Carlo algorithm with bit-cost bound:

$$\mathscr{D}_{n,\|A\|}(A) = O^{\sim}\left(n^3 \left(\log \frac{|\det A|}{|s_n|} + \log \|A\| \right) \right). \tag{15}$$

This can be compared to (8): the structural parameter $(\det A)/s_n$ plays a role analogous to $\text{cond}_{\det} A$ in the numerical computations. For random integer matrices with $\log \|A\| > 3 \log n$, where the entries are

uniformly distributed, the expected value of s_n is $\det A$ (by [25, Corollary 6.3] the expected number of nontrivial diagonal entries of the Smith form is one). Thus the average cost for computing the determinant satisfies:

$$E(\mathscr{D}_{n,\|A\|}) = O^{\sim}(n^3 \log \|A\|) \tag{16}$$

using a randomized Monte Carlo algorithm.

Remark 3. System solution also provides a certificate for matrix singularity. Following Remark 1 we work with a random prime p. Without loss of generality, assume that the input matrix A has rank r modulo p and that its leading $r \times r$ principal minor A_r is nonzero modulo p. With high probability, r is also the rank of A over $\mathbb{Q}$ and if $r < n$ then the solution vector u to the linear system

$$A_r u = A_{(1,\ldots,r),r+1} \tag{17}$$

should be a vector in the nullspace of A. The singularity certificate computes r modulo p, solves system (17) over $\mathbb{Q}$ and check whether $Au = 0$.

6. Exact determinant: better worst case bounds

All previously shown algorithms have bit-costs bounded like

$$\text{bit-cost} \lesssim \text{arithmetic cost} \times \text{output maximum size} \tag{18}$$

with approximate equality always attained in the worst case. We are going to see two different ideas that actually lead to much smaller worst case complexity estimates. Even based on the straightforward cubic matrix multiplication algorithm those new algorithms bring the exponent of n below 4.

The first approach is, again, to take advantage of solving linear systems and to look at the Smith normal form. Using arguments similar to those of previous section and from [25, Section 2], several solutions of linear systems with random right-hand side vectors are sufficient to compute the largest entry s_n of the Smith normal form of A. The use of system solution can be extended to computing the determinant by applying the same technique iteratively to perturbations of A. This approach—initially proposed in [56] for computing the characteristic polynomial of a sparse matrix—is developed in the integer case by Eberly et al. [25]. The resulting randomized Monte Carlo algorithm is sensitive to the size of the determinant and to a parameter $\phi(A)$, the number of distinct invariant factors, which characterizes the Smith form. The number of distinct invariant factors satisfies

$$\phi(A) = O(\sqrt{|\det A|}) \leqslant O^{\sim}(\sqrt{n \log \|A\|}).$$

Together with (25), the corresponding cost (see [25]) is bounded by

$$\mathscr{D}_{n,\|A\|}(A) = O^{\sim}(\phi(A)\, n^3 \log \|A\|) = O^{\sim}(\sqrt{|\det A|}\, n^3 \log \|A\|)$$

$$= O^{\sim}(n^{3.5} \log^{1.5}\|A\|). \tag{19}$$

In fact, the algorithm in [25] computes both the determinant and the Smith normal form at the same bit-cost bound. Another variant based on system solution has been designed to take advantage of fast

matrix multiplication [25, Section 5]. The determinant is computed as the product of large invariant factors—using denominators of system solutions—and of smaller invariant factors—using a direct algorithm for the Smith form [49].

Since $\phi(A)$ is small on the average [25, Corollary 6.3]:

$$E(\phi) = O(\log n), \tag{20}$$

which shows that (16) was already established using (19).

To yield strict inequality in the bit-cost bound (18), the Smith form approach has focused on the parameter $\phi(A)$. Another strategy has been applied earlier on polynomial matrices by Kaltofen [34] and can be carried over to the integer matrix case. The idea is to perform a large amount of precomputation with shorter integers by applying Shanks's "baby-steps, giant-steps" principle to Wiedemann's determinant algorithm [57]. The number of arithmetic operations on integers of length $O^\sim(n \log \|A\|)$ is sufficiently reduced, and one obtains a Las Vegas (certified) randomized algorithm with

$$\mathscr{D}_{n,\|A\|} = O^\sim(\sqrt{n}\,(n^3 \log \|A\|)) = O^\sim(n^{3.5} \log \|A\|) \tag{21}$$

bit-complexity [34,37]. Unlike in Kaltofen's 1992 paper, the integer matrix case requires randomization. The algorithm has a Chinese-remainder based implementation and can be made sensitive to $|\det A|$. For instance, if $\log |\det A| = O(n^{1-\eta} \log \|A\|)$, where $0 \leqslant \eta \leqslant 1$, the Monte Carlo running time [36] in bit-operations using standard matrix multiplication is

$$\mathscr{D}_{n,\|A\|}(A) = O^\sim(\sqrt{\log |\det A| \log \|A\|}\, n^3) = O^\sim(n^{3+1/2-\eta/2} \log \|A\|). \tag{22}$$

With asymptotically fast rectangular matrix product procedures, the cost of the algorithm becomes [34]:

$$\mathscr{D}_{n,\|A\|} < O^\sim(n^{3.03} \log \|A\|). \tag{23}$$

As initially conceived, the approach also leads to similar bounds for the division-free complexity of the determinant over an abstract commutative ring R. The determinant of a matrix in $R^{n \times n}$ can be computed in

$$\mathscr{D}_{n,R} = O^\sim(n^{3.5}) \tag{24}$$

additions, substractions and multiplications in R (without divisions) or in $O(n^{3.03})$ ring operations if a fast matrix product is employed. The previously known division-free determinant complexity was using Strassen's technique for division removal [53]. Similarly to (4) or (18), the best known cost bound in R had been the product $O^\sim(n^{\omega+1})$ of arithmetic cost times the size (degree of the determinant of a degree one matrix polynomial).

By preconditioning the input matrix (in an algebraic sense [57,15]), Wiedemann's algorithm first reduces the problem of computing the determinant to the problem of computing the minimum polynomial. Then the latter polynomial is computed à la Krylov-Lanczos. Kaltofen and Villard improve bounds (21) and (23) by exploiting block projections during the Krylov-Lanczos step. See [18,35,55] on some earlier applications of block Krylov-Lanczos and Wiedemann algorithm to other computational problems. Blocking further reduces the operation count on large numbers and achieves the bit-cost

$$\mathscr{D}_{n,\|A\|} = O^\sim(n^{3+1/3} \log \|A\|)$$

with straightforward arithmetics or, using fast polynomial arithmetic including the half GCD algorithm on matrix polynomials [37],

$$\mathcal{D}_{n,\|A\|} = \mathrm{O}^{\sim}(n^{3+1/5} \log \|A\|). \tag{25}$$

The same asymptotic bounds in n work for the Smith normal form and the division-free determinant complexity. Asymptotically fast square and rectangular matrix multiplication can also be exploited and give

$$\mathcal{D}_{n,\|A\|} \leqslant \mathrm{O}(n^{2.698} \log \|A\|)$$

for the worst case bit-complexity of the Las Vegas randomized computation of the determinant. As noticed by Pan, this asymptotically fast approach also allows for extension of Kaltofen's adaptative techniques (22), see [36, p. 143].

In addition to these improvements of bound (18) on the complexity of computing the determinant we may also mention the work of Storjohann for polynomial matrices over an abstract commutative field K [51]. Using Hensel's lifting (Section 5) for the solution of linear systems $AX = B$ with B an $n \times m$ matrix, Storjohann computes the determinant and the Smith normal form of an $n \times n$ matrix of degree d over $K[x]$ in $\mathrm{O}^{\sim}(n^{\omega}d)$ operations in K. The algorithm proceeds by repeatedly solving linear systems with matrix right-hand sides of increasing column dimension m, each system leading to a group of invariant factors of decreasing sizes. Even if the integer case is quite different than the polynomial case one may think that this approach and the corresponding complexity will be extended to integer matrices [51, Section 14].

7. Matrices of sparse numbers

In numerical computation, rather than studying the complexity with respect to $\log \|A\|$, one may model the size of the entries of A using a mantissa size s_x and an exponent size e_x. Following Priest [47] and using *sparse high precision numbers*, in the course of the algorithms we represent the numbers as list of pairs (mantissa, exponent). The length of such lists may be arbitrarily large, and the cost of an arithmetic operation $+, -, \times$ in this set of numbers is polynomial in the size of the operands. Under this model, the problem of the determinant is addressed by Demmel and Koev in [21]. The complexity classes are different from those of the "classical" model in the previous sections. Indeed, under this model the algorithms we have seen so far all require exponential time. Assuming, for instance, $e_x = \log \log \|A\|$, all the cost functions we have seen have the form $n^k (2^{e_x})^l$ for some integers k and l. Furthermore, the straightforward method which computes the determinant using recursive minor expansions performs at a cost polynomial in s_x and e_x but exponential in n. Hence the question is left open whether it is possible to accurately compute the determinant—and thus its sign—in time polynomial in s_x, e_x and n [21, Section 12].

The general answer is not known but the answer is "yes" for a class $\mathscr{F}$ of matrices whose determinant (viewed as a polynomial in the entries of A) admits a special factorization (see [21, Theorem 3]). This class includes a significant range of structured matrices. In terms of the bit-complexity model, the study proves that the cost of accurate computations on those matrices is related to $\log \log \|A\|$

rather than to $\log \|A\|$. In particular one has

$$\tilde{\mathscr{S}}_{n,\|A\|} = \mathrm{polynomial}(n, \log \log \mathrm{cond}\, A) = \mathrm{polynomial}(n, \log \log \|A\|),$$

where $\tilde{\mathscr{S}}_{n,\|A\|}$ is the sign computation complexity for input matrices in $\mathscr{F}$.

8. Discussion

Focusing on the exponents of n, we recapitulate the different complexity estimates in Table 1. Concerning the worst case exponent of n, the record value has been progressing from 4 to $3 + \frac{1}{5}$ (with classical matrix multiplication). It is natural to hope for further evolutions independently of the choice of the underlying arithmetic. As noticed at the end of Section 6, one may hope that the estimate $O^{\sim}(n^{\omega} \log \|A\|)$ will be achieved for the bit-complexity of the sign and the determinant.

Apart from the worst case situations, the heuristic arguments in [1] and the probability analysis in [25] show that Pan's linear system based approach is the symbolic companion piece to numerical results. Indeed, the numerical sign estimate $O^{\sim}(n^3 \log \|A\|)$ for well conditioned matrices somehow corresponds to the symbolic determinant estimate $O^{\sim}(n^3 \log \|A\|)$ for small values of $\phi(A)$. However, one can also possibly identify here a difference between sign and determinant computation. A small condition number does not seem to imply a small number $\phi(A)$ of distinct invariant factors and vice versa. Another advantageous situation for exact computations is the case of small determinants where Chinese remaindering performs very well. One wonders if eventually no bad, i.e., supercubic, worst case inputs are left.

Missing aspects in this paper concern memory complexity, practical costs (log factors are hidden in our soft-O notation) and discussions for particular classes of matrices such as structured or sparse ones. We have seen that computing the determinant of an integer matrix has strong links with computing the Smith normal form. For matrix polynomials, this shows that further studies may also

Table 1
Estimates for the bit-complexity of the sign and the determinant. Exponents of n in $O^{\sim}$ functions for $A \in \mathbb{Z}^{n \times n}$ with $\omega = 3$ and $b = \log \|A\|$

Method	Worst case	Propitious case
Class. numerical—(3)	$n^4 b$	$n^3 b$
Class. exact—(4)	$n^4 b$	—
Certified sign	$(n^3 + n^2 \log \Delta(A))(n+b)$	
$\mathscr{S}_{n,\|A\|}(A)$—(5)	$n^4 b + n^3 b^2$	$n^4 + n^3 b$
Filters	$n^3 (\log \mathrm{cond}_{\det} A + b)$	
$\mathscr{S}_{n,\|A\|}(A)$—(8)	$n^4 b$	$n^3 b$
Chinese remainders	$n^3 \log |\det A| \log b + n^2 b$	
$\mathscr{D}_{n,\|A\|}$—(9)	$n^4 b$	$n^3 \log b + n^2 b$
Linear systems	$n^3 (\log(|\det A|/|s_n|) + b)$	
$\mathscr{D}_{n,\|A\|}$—(15)	$n^4 b$	$n^3 b$
Smith form	$\phi(A) n^3 b \leqslant \sqrt{\log |\det A|}\, n^3 b$	
$\mathscr{D}_{n,\|A\|}$—(19), (20)	$n^{3+1/2} b^{1+1/2}$	$E(\mathscr{D}_{n,\|A\|}) \leqslant n^3 b$
Division-free—(25), (22)	$n^{3+1/5} b$	$\sqrt{b \log |\det A|}\, n^3$

involve links with eigenvalues problems such as the characteristic polynomial and the Frobenius normal form computation.

We conclude that in the case of the determinant, speedup can be achieved by exploiting the interplay of the algebraic structure with the bits of the intermediately computed integers. Such could be the case when computing the values of other polynomials, for instant, resultants.

References

[1] J. Abbott, M. Bronstein, T. Mulders, Fast deterministic computation of determinants of dense matrices, in: Procceedings of the International Symposium on Symbolic and Algebraic Computation, Vancouver, BC, Canada, ACM Press, New York, July 1999, pp. 197–204.

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

[3] E. Allender, R. Beals, M. Ogihara, The complexity of matrix rank and feasible systems of linear equations, Comput. Complexity 8 (2) (1999) 99–126.

[4] F. Avnaim, J.D. Boissonnat, O. Devillers, F. Preparata, M. Yvinec, Evaluating signs of determinants using single-precision arithmetic, Algorithmica 17 (1997) 111–132.

[5] J.C. Bajard, L.S. Didier, J.M. Muller, A new Euclidean division algorithm for residue number systems, J. VLSI Signal Process. 19 (1998) 167–178.

[6] E.H. Bareiss, Computational solution of matrix problems over an integral domain, J. Inst. Math. Appl. 10 (1972) 68–104.

[7] W. Baur, V. Strassen, The complexity of partial derivatives, Theor. Comput. Sci. 22 (1982) 317–330.

[8] D. Bini, V.Y. Pan, Polynomial and Matrix Computations, Birkhäuser, Boston, 1994.

[9] H. Brönnimann, I.Z. Emiris, V.Y. Pan, S. Pion, Computing exact geometric predicates using modular arithmetic, in: Procroceedings of the 13th Annual ACM Symposium on Computational Geometry, 1997, pp. 174–182.

[10] H. Brönnimann, I.Z. Emiris, V.Y. Pan, S. Pion, Sign determination in residue number systems, Theoret. Comput. Sci. 210 (1) (1999) 173–197.

[11] H. Brönnimann, M. Yvinec, A complete analysis of Clarkson's algorithm for safe determinant evaluation, Rapport de Recherche, Institut National de Recherche en Informatique et en Automatique, INRIA-2051, France, November 1996.

[12] H. Brönnimann, M. Yvinec, Efficient exact evaluation of signs of determinant, Algorithmica 27 (2000) 21–56.

[13] J. Bunch, J. Hopcroft, Triangular factorization and inversion by fast matrix multiplication, Math. Comp. 28 (1974) 231–236.

[14] S. Cabay, T.P.L. Lam, Congruence techniques for the exact solution of integer systems of linear equations, ACM Trans. Math. Software 3 (4) (1977) 386–397.

[15] L. Chen, W. Eberly, E. Kaltofen, B.D. Saunders, W.J. Turner, G. Villard, Efficient matrix preconditioners for black box linear algebra, Linear Algebra and its Applications 343–344 (2002) 119–146.

[16] K.L. Clarkson, Safe and effective determinant evaluation, in: Proceedings of the 33rd Annual IEEE Symposium Foundations of Computer Science, Pittsburg, USA, 1992, pp. 387–395.

[17] H. Cohen, A Course in Computational Number Theory, Springer, Berlin, 1996.

[18] D. Coppersmith, Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm, Math. Comp. 62 (205) (1994) 333–350.

[19] D. Coppersmith, Rectangular matrix multiplication revisited, J. Complexity 13 (1997) 42–49.

[20] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, in: Proceedings of the 19th Annual ACM Symposium on Theory and Computation, 1987, pp. 1–6.

[21] J. Demmel, P. Koev, Necessary and sufficient conditions for accurate and efficient rational function evaluation and factorizations of rational matrices, in: V. Olshevsky (Ed.), Structured Matrices in Mathematics, Computer Science and Engineering II, Vol. 281, Contemporary Mathematics Series, American Mathematical Society, Providence, Rhode Island, 2001.

[22] J.W. Demmel, N. Higham, Stability of block algorithms with fast level-3 blas, ACM Trans. Math. Software 18 (3) (1992) 274–291.

[23] J.D. Dixon, Exact solution of linear equations using p-adic expansions, Numer. Math. 40 (1982) 137–141.

[24] P.D. Domich, R. Kannan, L.E. Trotter Jr., Hermite normal form computation using modulo determinant arithmetic, Math. Oper. Res. 12 (1) (1987) 50–59.

[25] W. Eberly, M. Giesbrecht, G. Villard, Computing the determinant and Smith form of an integer matrix, in: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA, IEEE Computer Society Press, Silver Spring, MD, November 2000, pp. 675–685.

[26] J. Edmonds, Systems of distinct representatives and linear algebra, J. Res. Nat. Bur. Standards—B 71 (4) (1967) 241–245.

[27] I.Z. Emiris, A complete implementation for computing general dimensional convex hulls, Inter. J. Comput. Geom. Appl. 8 (2) (1998) 223–253.

[28] S. Fortune, C.J. Van Wyk, Efficient exact arithmetic for computational geometry, in: Proceedings of the Ninth Annual ACM Symposium on Computational Geometry, 1993, pp. 163–172.

[29] M.A. Frumkin, Polynomial time algorithms in the theory of linear diophantine equations, in: Fundamentals of Computation Theory, Lecture Notes in Computer Science, Vol. 56, Springer, Berlin, 1977, pp. 386–392.

[30] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, Cambridge University Press, Cambridge, 1999.

[31] M. Giesbrecht, Fast computation of the Smith form of a sparse integer matrix, Comput. Complexity 10 (2001) 41–69.

[32] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.

[33] X. Huang, V.Y. Pan, Fast rectangular matrix multiplication and applications, J. Complexity 14 (1998) 257–299.

[34] E. Kaltofen, On computing determinants without divisions, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Berkeley, CA, USA, ACM Press, New York, July 1992, pp. 342–349.

[35] E. Kaltofen, Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems, Math. Comp. 64 (210) (1995) 777–806.

[36] E. Kaltofen, An output-sensitive variant of the baby steps/giant steps determinant algorithm, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Lille, France, ACM Press, New York, July 2002, pp. 138–144.

[37] E. Kaltofen, G. Villard, On the complexity of computing determinants, in: K. Shirayanagi, K. Yokoyama (Eds.), Proceedings of the Fifth Asian Symposium on Computer Mathematics (ASCM 2001), Vol. 9, Lecture Notes Series on Computing, World Scientific, Singapore, 2001, pp. 13–27. Maple 6 complexity worksheet http://www.math.ncsu.edu/∼kaltofen/bibliography/01/KaVi01.mws.

[38] J. Keyser, T. Culver, D. Manocha, S. Krishnan, MAPC: a library for efficient and exact manipulation of algebraic points and curves, in: Proceedings of the 15th Annual ACM Symposium on Computational Geometry, 1999, pp. 360–369.

[39] A.K. Lenstra, H.W. Lenstra, L. Lovász, Factoring polynomials with rational coefficients, Math. Ann. 261 (1982) 515–534.

[40] R.T. Moenck, J.H. Carter, Approximate algorithms to derive exact solutions to systems of linear equations, in: Proceedings of the EUROSAM, Lecture Notes in Computer Science, Vol. 72, Springer, Berlin, 1979, pp. 63–73

[41] T. Mulders, A. Storjohann, Certified dense linear system solving, Technical Report 355, Institute of Computer Systems, ETH Zürich, January 2000, J. Symbolic Comput., to appear.

[42] M. Newman, Integral Matrices, Academic Press, New York, 1972.

[43] V.Y. Pan, Complexity of parallel matrix computations, Theoret. Comput. Sci. 54 (1987) 65–85.

[44] V.Y. Pan, Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations, Inform. Process. Lett. 28 (1988) 71–75.

[45] V.Y. Pan, Y. Yu, Certification of numerical computation of the sign of the determinant of a matrix, Algorithmica 30 (2001) 708–724.

[46] S. Pion, De la géométrie algorithmique au calcul géométrique, Thèse de Doctorat, Université de Nice Sophia-Antipolis, France, November 1999.

[47] D. Priest, Algorithms for arbitrary precision floating point arithmetic, in: Proceedings of the Tenth Symposium on Computer Arithmetic, IEEE Computer Society Press, Silver Spring, MD, June 1991, pp. 132–145.

[48] A. Schönhage, V. Strassen, Schnelle Multiplikation grosser Zahlen, Computing 7 (1971) 281–292.

[49] A. Storjohann, Near optimal algorithms for computing Smith normal forms of integer matrices, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Zurich, Switzerland, ACM Press, New York, July 1996, pp. 267–274.

[50] A. Storjohann, Algorithms for matrix canonical forms, Ph.D. Thesis, Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zurich, Switzerland, November 2000.

[51] A. Storjohann, High-order lifting (Extended Abstract), in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Lille, France, ACM Press, New York, July 2002, pp. 246–254.

[52] V. Strassen, Gaussian elimination is not optimal, Numer. Math. 13 (1969) 354–356.

[53] V. Strassen, Vermeidung von Divisionen, J. Reine Angew. Math. 264 (1973) 182–202.

[54] L.G. Valiant, Completeness classes in algebra, in: Proceedings of the 11th Annual ACM Symposium on Theory Computation, ACM Press, New York, 1979, pp. 249–261.

[55] G. Villard, Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems, in: Proceedings of the International Symposium on Symbolic and Algebraic Computation, Maui, Hawaii, USA, ACM Press, New York, July 1997, pp. 32–39.

[56] G. Villard, Computing the Frobenius form of a sparse matrix, in: Proceedings of the Third International Workshop on Computer Algebra in Scientific Computing, Samarkand, Uzbekistan, Springer, Berlin, October 2000, pp. 395–407.

[57] D. Wiedemann, Solving sparse linear equations over finite fields, IEEE Trans. Inform. Theory IT-32 (1986) 54–62.