Computing the sign or the value of the determinant of an integrer matrix, a complexity survey

Erich Kaltofen, Gilles Villard

Laboratoire de l'Informatique du Parallélisme

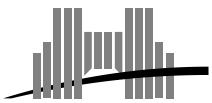
École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON n° 5668

CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

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

Erich Kaltofen and Gilles Villard Janvier 2002

Research Report N° 2002-2



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

INRIA

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

Erich Kaltofen and Gilles Villard

Janvier 2002

Abstract

Certified computation of the sign of the determinant of a matrix or computation of the determinant itself are a challenge for both numerical and exact methods. We survey the complexities 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 in the norm of A. Existing approaches rely either on numerical approximate computations, on exact computations or even on both types of arithmetic in combination.

Keywords: Determinant, bit complexity, integer matrix, approximate computation, exact computation, randomized algorithm.

Résumé

Calculer le signe du déterminant d'une matrice ou calculer le déterminant lui-même est une question importante aussi bien pour les approches numériques que pour les approches exactes. Nous proposons un tour d'horizon des complexités des méthodes pour résoudre ces problèmes quand la matrice en entrée est une matrice $n \times n$ à coefficients entiers. Nous regardons les complexités binaires asymptotiquement en n et en la norme de A. Les approches existantes reposent sur des calculs numériques approchés, sur des calculs exacts ou même sur l'utilisation combinée des deux types d'arithmétiques.

Mots-clés: Déterminant, matrice d'entiers, calcul approché, calcul exact, algorithme probabiliste.

COMPUTING THE SIGN OR THE VALUE OF THE DETERMINANT OF AN INTEGER MATRIX, A COMPLEXITY SURVEY[1]

Erich Kaltofen

Department of Mathematics, North Carolina State University
Raleigh, North Carolina 27695-8205 USA
kaltofen@math.ncsu.edu http://www.kaltofen.net

Gilles Villard

Cnrs, Laboratoire Lip, École Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon Cedex 07, France
Gilles.Villard@ens-lyon.fr http://www.ens-lyon.fr/~gvillard

Abstract. Certified computation of the sign of the determinant of a matrix or computation of the determinant itself are a challenge for both numerical and exact methods. We survey the complexities 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 in the norm of A. Existing approaches rely either on numerical approximate computations, on exact computations or even on both types of arithmetic in combination.

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, 43] and to the bibliographies therein. In symbolic computation, the problem of computing the exact value of the determinant is addressed for instance in relation with matrix normal forms problems [41, 28, 23, 51] or in computational number theory [16].

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

[1] This material is based on work supported in part by the National Science Foundation under grants Nrs. DMS-9977392, CCR-9988177, and CCR-0113121 (Kaltofen) and by the Centre National de la Recherche Scientifique, Actions Incitatives No 5929 et Stic LinBox 2001 (Villard).

aspects. For an input matrix $A \in \mathbb{Z}^{n\times n}$ with infinity matrix norm $\|A\|$, we report worst case bit complexities in terms of n and $\|A\|$. If $a_{i,j}$ denotes the integer in row i and column j of A then $\|A\| = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{i,j}|$ and any entry in A has bit length bounded by

$$\min_{1\le i,j\le n} \{\beta : |a_{i,j}| < 2^{\beta}, \beta \ge 1\} \le 1 + \log\left(\|A\| + 1\right).$$

In algebraic complexity—i.e. when counting the number of operations in an abstract domain D—we refer to Baur and Strassen about the link between matrix multiplication and determinant computation [52, 53, 7]. See also the link with matrix powering and the complexity class GapL following Toda, Vinay, Damm and Valiant as explained in [3], for example. We may also mention Valiant's theorem that the determinant is universal for formulas [54].

For integer matrices, computing the sign of the determinant is—a priori—an easier problem than computing its value. We will 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 conditionedness that influences the precision of the computations. Symbolic methods are confronted with intermediate coefficient growth and with the invariant structure of the matrix that directly influence the costs. We will see which techniques can be used to obtain algorithms sensitive to these conditions. Consequently, the bit complexities we give are either worst case bounds or bounds depending on some additional properties. This will imply discussion on algorithms that adapt to certain favorable situations, i.e., on classes of input matrices that require much lower running time than the worst case inputs.

The lowest known exponent of n in the bit complexity of the sign or of the determinant is decreasing. In particular, for the determinant this bit complexity is known to be below the algebraic complexity times the maximum bit size of the output (see [33, 24, 36] and section 6). This has motivated this survey to focus on the sequential time complexity rather on other aspects such as memory resources, parallel time or practical considerations. We will 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. We will see in section 4 that this first exact approach with randomization allows to be 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 will focus on other exact methods. Existing fast algorithms fall into two categories. The first category takes advantage of linear system solution, a problem whose bit worst case 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 will deal 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. Section 7 will then briefly consider computations on sparse numbers with a different model of computation. The last section includes a conclusion along with a discussion of previous results.

We will use that the cost of 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 $\omega = 2.376$ [19] and special exponents if the input matrices are rectangular [18, 32]. The bit complexity of multiplying two l-bit integers or floating point numbers will be $O(l^2)$ using the straightforward algorithm or $O^\sim(l)$ with a fast algorithm [48]. Here and in the following, 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 we will use the classical cubic algorithm for the 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 should be also implementable 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 $\|A\|$ will be denoted by $\mathcal{S}_{n,\|A\|}$, the worst case bit cost for computing the determinant will be $\mathcal{D}_{n,\|A\|}$. Hence we have $\mathcal{S}_{n,\|A\|} \leq \mathcal{D}_{n,\|A\|}$. For adaptive algorithms (see definition 3.1) these functions will be bounded by 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 shall write the matrix as an argument, namely $\mathcal{S}_{n,\|A\|}(A), \mathcal{D}_{n,\|A\|}(A)$.

2 Classical results on sign and determinant computation

In constant precision computation, the condition number of the determinant plays a central role. Following Higham [31, Problem. 13.15], for such a number we may take:

$$\begin{aligned} \log \operatorname{cond}_{\det} A &= \log \max_{i,j} |a_{i,j}(A^{-1})_{i,j}| \\ &\leq \log\left(\frac{\prod_{i=1}^{n} \|a_{i,*}\|_2}{|\det A|} \cdot \|A\|\right), \end{aligned} \tag{1}$$

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 [31, p. 10]:

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

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

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

For accurate computations (with low relative error for certifying the sign) on badly conditioned matrices (having small determinants 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 [31, Chapter 9]. With a matrix decomposition using $O(n^3)$ arithmetic operations the bit cost for the sign is thus bounded as

$$\mathcal{S}_{n,\|A\|} \leq O^\sim(n^3 \cdot n \log \|A\|) = O^\sim(n^4 \log \|A\|). \tag{2}$$

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 say in $O^\sim(n^3 \log \|A\|)$.

In symbolic computation, most of the difficulties in reducing the complexity are governed by the size of the determinant. We know by Hadamard's inequality [29, Theorem 16.6] that

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

therefore, the determinant may have up to $O^{\sim}(n\log\|A\|)$ digits. 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 a Gaussian elimination with the sizes of the intermediate integers controlled by exact division or more sophisticatedly by Bareiss's method [6]. Another approach [25, 13] is to use matrix arithmetic modulo primes and Chinese remaindering (on this technique see [2, Theorem 8.9] or [8, Problem 4.2]). The classical associated cost for the exact computation of the determinant, including a fast reduction of the matrix entries modulo the different primes, is [29, Chapter 5]:

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

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 [21] or Higham [31, Chapter 22] for numerical approaches. For algebraic and symbolic aspects we refer to Bini and Pan [8, Chapter 2]. The bit cost for computing the determinant is

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

Remark 2.1 *A sub-problem of the computation of the sign or of 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, e.g., [30, 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 5.1.* □

3 Numerical computation of the sign

As opposed to using exact arithmetics, specialized algorithms based on floating point operations have been intensively studied to compute the sign of algebraic expressions in general and of the determinant in particular. As seen above, a small precision may give a 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 of *adaptive* algorithms that automatically take into account these variations of the precision.

We shall attempt a definition of this algorithm design paradigm.

Definition 3.1 *An algorithm is* adaptive *(*input-sensitive, output-sensitive, introspective*) if its complexity is asymptotically below its worst case complexity for a non-trivial subset of its inputs.* □

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 will 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 [15] (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 [38]. Using good properties of B, especially a low orthogonality defect (see (4)), the second step then computes the sign of the determinant by LU decomposition. The first step asks to compute on numbers with at most $\log \|A\| + O(n)$ bits [12]. The arithmetic cost depends on the orthogonality defect of A defined by

$$\Delta(A) = \frac{\prod_{i=1}^{n} \| a_{i,*} \|_2}{|\det A|}. \tag{4}$$

Similarly to the condition number, the defect is in $O^{\sim}(n \log \|A\|)$. When A is invertible, the defect bounds the number of iterations of the first step of the algorithm. The overall cost is given by

$$\mathcal{S}_{n,\|A\|}(A) \leq O^{\sim}((n^3 + n^2 \log \Delta(A)) \cdot (n + \log \|A\|)). \tag{5}$$

We may notice that using remark 2.1, the invertibility can be easily tested. Using the generalization of Brönnimann et Yvinec [11, 12], even for singular matrices the bit cost satisfies:

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

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

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

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

To have a better complexity 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 certificate [27, 44, 37, 43]. 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 $\epsilon = O(\log n)$ allows the filters in [37, 43] to work correctly for well conditioned matrices. If the condition number is small—say $\log \mathrm{cond}_{\det} A = O(\log n)$—then the rank is certified using $O^{\sim}(n^3 \log \|A\|)$ operations. More generally, with a singularity test as in remark 2.1 and as suggested by Pan in [43, p. 715], by repeatedly doubling the precision this leads to the theoretical bound

$$\mathcal{S}_{n,\|A\|}(A) \leq O^{\sim}\left(n^3 \cdot (\log \mathrm{cond}_{\det} A + \log \|A\|)\right) \leq O^{\sim}(n^4 \log \|A\|). \tag{8}$$

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 size of the determinant. Here and in subsequent sections the techniques need randomizations. The idea to ensure sensitivity 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). 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:

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

About this early termination technique the reader may refer to the detailed study of Brönnimann et al. [10] and to that of Emiris [26] for remarks on success probabilities. Even if the output is not certified (Monte Carlo algorithm), this will give 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 (3).

5 Exact determinant and linear system solution

The first type of fast exact algorithms for computing the determinant tries to exploit Cramer rules and the relations between system solution and determinant computation. Either using an algebraic model or for worst case bit complexities it remains an open question whether linear system solution is asymptotically a strictly easier problem than determinant computation [7, p. 328]. At this time, the known worst case cost for solving a linear system exactly over the rationals is strictly smaller than the one for computing the determinant. We refer to the p-adic system solution proposed by Moenck and Carter [39] then by Dixon [22] and improved by Mulders and Storjohann [40, Section. 5.1.2]. The bit complexity for solving $Ax = b$ with $b \in \mathbb{Z}^n$ and $\|b\| \leq \|A\|$ is bounded by

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

Further, as shown by Storjohann [49], fast matrix multiplication techniques can be used and give:

$$\mathcal{L}_{n,\|A\|} \leq O^{\sim}(n^{\omega} \log\|A\|). \tag{11}$$

Hence exact system solution in the worst case has the asymptotic cost of numerical determinant computation for well conditioned matrices (see section 2). Pan has proposed, in [42, Appendix] and in [45], 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 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* [41]—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. The 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

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

This is (3) 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 [24, 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

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

where s_n is the 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 5. This leads to a Monte Carlo algorithm with cost:

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

This may now be directly compared to the cost bound (8), the structural parameter $(\det A)/s_n$ plays a role analogous to $\mathrm{cond}_{\det} A$ 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 [24, Corollary 6.3] the expected of the number of nontrivial diagonal entries of the Smith form is one) thus the average cost for computing the determinant satisfies:

$$E(\mathcal{D}_{n,\|A\|}) \leq O^{\sim}\left(n^3 \cdot \log \|A\|\right). \tag{16}$$

using a randomized Monte Carlo algorithm.

Remark 5.1 *System solution also provides a certificate for matrix singularity. Following remark 2.1 we work with a random prime p. Without loss of generality we 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 vector u solution to*

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

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

6 Exact determinant: better worst case bounds

All previously seen 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 lower worst case complexities. Even by plugging into the straightforward cubic matrix multiplication algorithm those new algorithms bring the exponent of n below 4.

A first solution is, again, to take advantage of linear system solution and to look at the Smith normal form. Using arguments similar to those of previous section and from [24, Section 2], several system solutions with random right 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 generalized to computing the whole determinant by applying the same technique iteratively to perturbations of A [56]. This approach—initially proposed for computing the characteristic polynomial of a sparse matrix—is developed in the integer case by Eberly et al. [24]. 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

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

Together with (10), the corresponding cost is (see [24]):

$$\begin{aligned} \mathcal{D}_{n,\|A\|}(A) \leq O^{\sim}(\phi(A) \cdot n^3 \log \|A\|) &\leq O^{\sim}(\sqrt{|\det A|} \cdot n^3 \log \|A\|) \\ &\leq O^{\sim}(n^{3.5} \log^{1.5} \|A\|). \end{aligned} \tag{19}$$

We may notice that the same bound is valid for computing both the determinant and the Smith normal form. It may not be so surprising that the bit complexity of computing the latter form is similar to the complexity of computing the determinant. Another variant based on system solution has been design for taking advantage of fast matrix multiplication [24]. 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 [50]). Using (11) for the bit cost of system solution the methods of [24, Section 5] lead to:

$$\mathcal{D}_{n,\|A\|} \leq O^{\sim}(\sqrt{n} \cdot n^{\omega} \log^{1.5} \|A\|) < O^{\sim}(n^{2.88} \log^{1.5} \|A\|).$$

Since $\phi(A)$ is small on the average [24, Corollary 6.3]:

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

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

To overcome the product (18), the Smith form approach has focused on the parameter $\phi(A)$. Another strategy has been applied earlier on polynomial matrices by Kaltofen [33] and can be carried over in the integer matrix case. The idea is to perform a large amount of precomputation with shorter integers by an application of 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

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

bit complexity [33, 36]. 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 $0 \leq \eta \leq 1$, the Monte Carlo running time in bit operations is [35]

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

With asymptotically fast rectangular matrix product procedures, the cost of the algorithm becomes [33]:

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

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

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

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 (3) or (18), the best known cost over R had been the product $O^{\sim}(n^{\omega+1})$ of an arithmetic cost times a size (degree of the determinant of a degree one matrix polynomial).

By preconditioning the input matrix (in an algebraic sense [57, 14]), 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 obtain improvements on (21) and (23) by introducing block projections during the Krylov-Lanczos step (see [17, 34, 55] on these aspects). Blocking further reduces the operation count on large numbers and leads to the cost

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

with straightforward arithmetics or, using fast polynomial arithmetic including the half GCD algorithm on matrix polynomials, to [36]:

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

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

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

for the worst case bit complexity of the Las Vegas randomized computation of the determinant.

7 Matrices of sparse numbers

Especially in numerical computation, rather than studying the complexity with respect to $\log\|A\|$, on may consider for modelling the size of the entries of A, 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 the numbers are represented as list of pairs (mantissa, exponent). The length of such lists may be

arbitrary 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 [20]. The complexity classes are different than those of the "classical" model we have considered in previous sections. Indeed, the algorithms we have seen so far all require exponential time. Taking 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. Also notice that the straightforward method which computes the determinant using recursive minor expansions would have a cost polynomial in s_x and e_x but exponential in n. Hence the question if left open to know whether it is possible to accurately compute the determinant—and thus its sign—in time polynomial in s_x, e_x and n [20, Section 12].

The general answer is not known but the answer is yes for a class $\mathcal{F}$ of matrices whose determinant (viewed as a polynomial in the entries of A) admits a special factorization (see [20, 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

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

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

8 Discussion

Focusing on the exponents of n, we recapitulate the different complexities in Table 1 below. 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. Do the estimates (10) and (11) obtained for the bit complexity of system solution apply to the complexity of the sign or of the determinant?

Apart from worst case situations, the heuristic arguments of [1] and the analysis of [24] show that Pan's linear system based approach is the symbolic companion piece to numerical results. Indeed, to the numerical sign estimate $O^{\sim}(n^3\log\|A\|)$ for well conditioned matrices somehow corresponds 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 involve links with eigenvalues problems such as the characteristic polynomial and the Frobenius normal form.

Table 1. Bit complexities of the sign and of 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 – (2)	$n^4 b$	$n^3 b$
Class. exact – (3)	$n^4 b$	–
Certified sign $\mathcal{S}_{n,\|A\|}(A)$ – (5)	$(n^3 + n^2 \log \Delta(A)) \cdot (n + b)$ $n^4 b + n^3 b^2$	$n^4 + n^3 b$
Filters $\mathcal{S}_{n,\|A\|}(A)$ – (8)	$n^3 \cdot (\log \mathrm{cond}_{\det} A + b)$ $n^4 b$	$n^3 b$
Chinese remainders $\mathcal{D}_{n,\|A\|}$ – (9)	$n^3 \cdot \log \lvert\det A\rvert \cdot \log b + n^2 b$ $n^4 b$	$n^3 \log b + n^2 b$
Linear systems $\mathcal{D}_{n,\|A\|}$ – (15)	$n^3 \cdot (\log(\lvert\det A\rvert / \lvert s_n \rvert) + b)$ $n^4 b$	$n^3 b$
Smith form $\mathcal{D}_{n,\|A\|}$ – (19), (20)	$\phi(A) \cdot n^3 b \leq \sqrt{\log \lvert\det A\rvert} \cdot n^3 b$ $n^{3.5} b^{1.5}$	$E(\mathcal{D}_{n,\|A\|}) \leq n^3 b$
Division-free – (25), (22)	$n^{3+1/5} b$	$\sqrt{b \log \lvert\det A\rvert} \cdot n^3$

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, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In *International Symposium on Symbolic and Algebraic Computation, Vancouver, BC, Canada*, pages 197–204. ACM Press, Jul 1999.

[2] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

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

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

[5] J.C. Bajard, L.S. Didier, and J.M. Muller. A new Euclidean division algorithm for residue number systems. *Journal of VLSI Signal Processing*, 19:167–178, 1998.

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

[7] W. Baur and V. Strassen. The complexity of partial derivatives. *Theor. Comp. Sc.*, 22:317–330, 1982.

[8] D. Bini and V. Pan. *Polynomial and matrix computations.* Birkhäuser, 1994.

[9] H. Brönnimann, I.Z. Emiris, V.Y. Pan, and S. Pion. Computing exact geometric predicates using modular arithmetic. In *Proc. 13th Annual ACM Symp. Comput. Geom.*, pages 174–182, 1997.

[10] H. Brönnimann, I.Z. Emiris, V.Y. Pan, and S. Pion. Sign determination in residue number systems. *Theoret. Comput. Sci.*, 210(1):173–197, 1999. Special Issue on Real Numbers and Computers.

[11] H. Brönnimann and 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, Nov. 1996.

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

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

[14] L. Chen, W. Eberly, E. Kaltofen, B.D. Saunders, W.J. Turner, and G. Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and its Applications,* special issue on *Infinite Systems of Linear Equations Finitely Specified. To appear.*

[15] K.L. Clarkson. Safe and effective determinant evaluation. In *Proc. 33rd IEEE Symp. Foundations Of Computer Science, Pittsburg, USA*, pages 387–395, 1992.

[16] H. Cohen. *A course in computational number theory.* Springer-Verlag, 1996.

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

[18] D. Coppersmith. Rectangular matrix multiplication revisited. *J. of Complexity*, 13:42–49, 1997.

[19] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *19 th. Annual ACM Symp. Theory Comp.*, pages 1–6, 1987.

[20] J. Demmel and P. Koev. Necessary and sufficient conditions for accurate and efficient rational function evaluation and factorizations of rational matrices. In V. Olshevsky, editor, *Structured matrices in mathematics, computer science and engineering II*, volume 281 of *Contemporary Mathematics Series.* American Mathematical Society, Providence, Rhode Island, 2001.

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

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

[23] P.D. Domich, R. Kannan, and L.E. Trotter Jr. Hermite normal form computation using modulo determinant arithmetic. *Mathematics of Operations Research*, 12(1):50–59, 1987.

[24] W. Eberly, M. Giesbrecht, and G. Villard. Computing the determinant and Smith form of an integer matrix. In *The 41st Annual IEEE Symposium on Foundations of Computer Science, Redondo Beach, CA*, pages 675–685. IEEE Computer Society Press, Nov. 2000.

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

[26] I.Z. Emiris. A complete implementation for computing general dimensional convex hulls. *Inter. J. Computational Geometry & Applications*, 8(2):223–253, 1998.

[27] S. Fortune and C.J. Van Wyk. Efficient exact arithmetic for computational geometry. In *Proc. 9th Ann. ACM Symp. Comput. Geom.*, pages 163–172, 1993.

[28] M.A. Frumkin. Polynomial time algorithms in the theory of linear diophantine equations. In *Fundamentals of Computation Theory*, LNCS 56, pages 386–392. Springer-Verlag, 1977.

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

[30] M. Giesbrecht. Fast computation of the Smith form of a sparse integer matrix. *Computational Complexity*. To appear.

[31] N.J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, Philadelphia, PA, 1996.

[32] X. Huang and V. Pan. Fast rectangular matrix multiplication and applications. *J. of Complexity*, 14:257–299, 1998.

[33] E. Kaltofen. On computing determinants without divisions. In *International Symposium on Symbolic and Algebraic Computation, Berkeley, California USA*, pages 342–349. ACM Press, July 1992.

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

[35] E. Kaltofen. An output-sensitive variant of the baby steps/giant steps determinant algorithm. Manuscript, Nov. 2001.

[36] E. Kaltofen and G. Villard. On the complexity of computing determinants. In K. Shirayanagi and K. Yokoyama, editors, *Proc. Fifth Asian Symposium on Computer Mathematics (ASCM 2001)*, volume 9 of *Lecture Notes Series on Computing*, pages 13–27, Singapore, 2001. World Scientific. Maple 6 complexity worksheet `http://www.math.ncsu.edu/~kaltofen/bibliography/01/KaVi01.mws`.

[37] J. Keyser, T. Culver, D. Manocha, and S. Krishnan. MAPC: A library for efficient and exact manipulation of algebraic points and curves. In *Proc. 15th Ann. ACM Symp. Comput. Geom.*, pages 360–369, 1999.

[38] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[39] R.T. Moenck and J.H. Carter. Approximate algorithms to derive exact solutions to systems of linear equations. In *Proc. EUROSAM*, LNCS 72, Springer Verlag, pages 63–73, 1979.

[40] T. Mulders and A. Storjohann. Certified dense linear system solving. Technical Report 356, Department of Computer Science, ETH Zurich, Switzerland, Dec. 2000.

[41] M. Newman. *Integral Matrices*. Academic Press, 1972.

[42] V. Pan. Complexity of parallel matrix computations. *Theoretical Computer Science*, 54:65–85, 1987.

[43] V.Y. Pan and Y. Yu. Certification of numerical computation of the sign of the determinant of a matrix. *Algorithmica*, 30:708–724, 2001.

[44] V.Y. Pan, Y. Yu, and C. Stewart. Algebraic and numerical techniques for the computation of matrix determinants. *Comput. Math. Appl.*, 34(1):43–70, 1997.

[45] Pan, V. Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations. *Inf. Proc. Letters*, 28:71–75, 1988.

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

[47] D. Priest. Algorithms for arbitrary precision floating point arithmetic. In *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–145. IEEE Computer Society Press, June 1991.

[48] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.

[49] A. Storjohann. Personal communication, Aug. 2001.

[50] A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. In *International Symposium on Symbolic and Algebraic Computation, Zurich, Switzerland*, pages 267–274. ACM Press, July 1996.

[51] A. Storjohann. *Algorithms for Matrix Canonical Forms.* PhD thesis, Institut für Wissenschaftliches Rechnen, ETH-Zentrum, Zurich, Switzerland, Nov. 2000.

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

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

[54] L.G. Valiant. Completeness classes in algebra. In *Proc. 11th Annual ACM Symp. Theory Comput.*, pages 249–261. ACM Press, 1979.

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

[56] G. Villard. Computing the Frobenius form of a sparse matrix. In *The Third International Workshop on Computer Algebra in Scientific Computing, Samarkand, Uzbekistan*, pages 395–407. Springer-Verlag, October 2000.

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