Записки научных
семинаров ПОМИ
Том 316, 2004 г.

V. Y. Pan

ON THEORETICAL AND PRACTICAL ACCELERATION OF RANDOMIZED COMPUTATION OF THE DETERMINANT OF AN INTEGER MATRIX

ABSTRACT. We reexamine the Wiedemann–Coppersmith–Kaltofen–Villard algorithm for randomized computation of the determinant of integer matrices and substantially simplify and accelerate its bottleneck stage of computing the minimum generating matrix polynomial, to make the algorithm practically promising while keeping it asymptotically fast.

1. INTRODUCTION

1.1. Some background.

Randomized Boolean (or bit operation) complexity of computing the determinant, $\det A$, of an $n \times n$ integer matrix $A = (a_{i,j})_{i,j}$ was studied recently in [15] and [30].

The paper [15] follows the approach in [37, Appendix], [38, 1] but contributes some novel ingeniuos techniques to yield a randomized *Monte Carlo* algorithm for $\det A$ using $O^*(n^3 L^2)$ bit operations for the average input matrix A and $O^*(n^{7/2} L^2)$ for the worst case input. Here and hereafter $L = \log(n \max_{i,j} |a_{i,j}|)$, $O^*(f(k))$ means $f(k)(\log f(k))^{O(1)}$ where $f(k) \longrightarrow \infty$ as $k \longrightarrow \infty$, and Monte Carlo algorithms allow erroneous output, although with a smaller probability, bounded by a fixed nonnegative tolerance value $\tau < 1$. Randomized algorithms are called *Las Vegas* algorithms if they fail with a smaller bounded probability but otherwise produce correct output, and it is assumed that the overall computational cost estimate of the algorithm covers the correctness certification.

For all complexity bounds that we cite and deduce, we assume the classical matrix multiplication (hereafter we use the abbreviation MM). The practical accelerations of MM by Strassen in 1969, supporting the

---

The results of this paper have been presented at the 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02), Juan-les-Pins, France, March 2002 and at the Fifth Annual Conference on Computer Algebra in Scientific Computing (CASC'2002), Yalta, Crimea, Ukraine 2002.

Supported by NSF Grant CCR 9732206 and PSC CUNY Award 65393-034.

arithmetic complexity bound in $O(n^{2.81})$, and more recently by Kaporin [23, 26] have so far lead to no practical acceleration of computing the determinants because of the substantial overhead of the transition from fast MM to the computation of the determinant. Theoretical decrease of all cited exponents for the complexity of computing the determinant, however, is possible based on asympotically fast MM in [12, 10].

1.2. The Wiedemann–Coppersmith–Kaltofen–Villard acceleration.

Further asymptotic acceleration has been achieved in [30] based on a distinct approach. The main result (Theorem 2) in [30] is a Las Vegas randomized estimate in $O^*(n^{10/3}L)$ bit operations for the complexity of computing the determinant of an $n \times n$ integer matrix. Kaltofen and Villard combine Wiedemann's algorithms in [56] and their block version proposed by Coppersmith in [9]. Like these two papers and unlike [15], the paper [30] allows additional acceleration where the input matrices can be multiplied by vectors fast. The paper exploits the known technique of baby steps/giant steps, shows how to compute the minimum generating matrix polynomial (hereafter we abbreviate *MGMP*), and supplies the bit complexity and the failure probability estimates. The computations in [30] essentially amount to the generation of the Wiedemann block sequence and computing its MGMP (see further details in Appendix A).

1.3. Our contribution.

Our present paper is the journal version of our proceeding papers [42] and [43] (see also [48]), more precisely, of the parts of [42] and [43] relevant to improving the algorithm in [30] by accelerating its computation of the MGMP. The paper [30] proposes to compute the MGMP

> "by a block version of the Berkekamp/Massey algorithm [9] or its variants, like by a matrix Padé approximation [6], by a matrix Euclidean algorithm [55], or by a block Toeplitz solver following the classical Levinson-Durbin approach [22]. The latter most easily elucidates the advantage of blocking: the number of sequence elements needed can be much shorter."

Our first observation is that the Levinson–Durbin block algorithm in [22] has the arithmetic cost unequivocally of the order of n^2. This leads to the overall bit complexity estimate in $O^*(n^{10/3}L)$ in the main theorem in [30], whereas the application of the MBA divide-and-conquer algorithm in [32], [33], [4] yields the arithmetic cost in $O^*(n)$ for computing the MGMP, and this decreases the exponent 10/3 in [30] to 16/5 [42, Theorem

5.1a]. An alternative derivation of the asymptotic arithmetic time bound in $O^*(n)$ for the MGMP and thus the overall exponent 16/5 can rely on the block version of the LKS half-gcd algorithm (due to Lehmer, Knuth, and Schönhage and extensively covered, e.g., in [3]). The LKS algorithm has been cited in [30] in conjunction with computing the determinant based on asymptotically fast MM but not for decreasing the exponent 10/3 to 16/5 in the main theorem in [30]. According to E. Kaltofen [25], this was because the main goal of [30] was practical computation of the determinants, whereas the inclusion of the block half-gcd algorithm would have ruined all hopes for achieving this goal.

Indeed, the LKS half-gcd algorithm has a complicated code and a large overhead. These deficiencies greatly limit the chances for its practical implementation in the polynomial case. The problems are aggravated in the block version of the polynomial half-gcd algorithm to make the MGMP computation by far the hardest stage for the implementation of the algorithm in [30]. As a result, the algorithm with the LKS block for computing the MGMP becomes impractical even for a moderately large input.

Including our approach in [42] and [43] instead of the LKS block, however, should salvage 16/5 as the record exponent supported by practical algorithms for the determinant computation. Recall that in [30], the MGMP is first computed with the Levinson–Durbin or LKS block algorithms modulo the order of nL distinct random primes and then (based on the Chinese remaindering) modulo their product. In contrast, we apply the MBA algorithm to compute the MGMP modulo a single random prime p and then lift the solution to yield it modulo p^h for h of the order of nL. Each lifting step is essentially equivalent to a small constant number of multiplications of block Hankel matrices by block vectors, versus quite a complex block LKS computational process or the order of n such multiplications in the Levinson–Durbin block algorithm. Since Hankel-by-vector product is just a polynomial product [41], our approach enables fast practical implementation of the MGMP stage, and thus of the entire algorithm because its another main stage of computing the Wiedemann block sequence causes no problems.

Furthermore, we have additional advantage since we use a single random basic prime versus the order of nL primes in [30]. This simplifies the random prime generation and gives us less chances to run into degeneracies. With the reduction modulo a single random prime we safely apply the effective diagonal preconditioner in [56], which would be a more risky adventure if we repeated it for the order of nL random primes. The analysis of the degeneration of computing the MGMP becomes much simpler and more transparent. We supply all details and complete this analysis in a distinct and shorter way versus the approach in [30] (see our proof of Theorem 3.4).

The resulting algorithm for computing the determinants of integer matrices may have advantage even versus numerical methods provided the input matrix is nearly singular, and this is the case in some important applications to geometric and algebraic geometric computations (cf.[8, 5, 50, 18]).

1.4. Organization of our paper.

We organize our paper as follows. In Sec. 2, we recall some definitions and elaborate upon the basic results. In Sec. 3, we cover and analyze the algorithm in [30] on computing the determinants amended with our accelerated computation of the MGMP. In Sec. 4, we discuss some extensions. We reproduce the algorithm of [30] in Appendix A and include some other related results in Appendices B and C.

2. Some definitions and Basic facts

Hereafter, log stands for $\log_2$ unless it is specified otherwise; $\ln = \log_e$ stands for the natural logarithms (with the base $e = 2.718281\ldots$).

2.1. Rings and matrices.

Hereafter $\mathbb{Z}$ and $\mathbb{Z}_q$ denote the rings of integers and of integers modulo q, respectively. $\mathbb{R}[\lambda]$ is the ring of polynomials in λ with their coefficients from a ring $\mathbb{R}$. $\mathbb{R}[\lambda]/\lambda^d$ is the same ring provided all its elements (that is, polynomials in λ) are reduced modulo λ^d. $\mathbb{R}^{k\times k}$ is the algebra of $k \times k$ matrices with their entries in a ring $\mathbb{R}$. $\mathbb{R}^k = \mathbb{R}^{k\times 1}$ is the space of k-dimensional vectors with their coordinates in a ring $\mathbb{R}$.

$D = \mathrm{diag}(d_1, \ldots, d_n) = \mathrm{diag}(d_i)_{i=1}^n$ is the diagonal matrix with diagonal entries $d_1, \ldots, d_n$. If d_i are blocks, then D is a block diagonal matrix. A matrix A is diagonalizable or similar to a diagonal matrix if $A = S^{-1}DS$ for a diagonal matrix D and a nonsingular matrix S. $I_k = \mathrm{diag}(1)_{i=1}^k$ is the $k \times k$ identity matrix. $\det A$, $\mathrm{adj}\, A$, $c_A(\lambda) = \det(\lambda I - A)$ and $\mu_A(\lambda)$ are the determinant, the adjoint and the characteristic and minimum polynomials of an $n \times n$ matrix A, respectively. M^T is the transpose of a matrix M. The vectors $\mathbf{v}$ such that $M\mathbf{v} = \mathbf{0}$ form the null space of a matrix M.

$H = (h_{i,j})_{i,j=1}^{m,n}$ is an $m \times n$ Hankel matrix if $h_{i+1,j-1} = h_{i,j}$ wherever the entries $h_{i,j}$ and $h_{i+1,j-1}$ are defined. $T = (t_{i,j})_{i,j=1}^{m,n}$ is an $m \times n$ Toeplitz matrix if $t_{i+1,j+1} = t_{i,j}$ wherever the entries $t_{i,j}$ and $t_{i+1,j+1}$ are defined. If $h_{i,j}$ (or $t_{i,j}$) are matrices, then H (resp. T) is a block Hankel (resp. Toeplitz) matrix. A matrix A has a Hankel (resp. Toeplitz) displacement generator of length l if $A = TH^T$ (resp. $A = T_0T^T$) where T, T_0, and H are $n \times (nl)$ matrices, H is a Hankel matrix, T and T_0 are Toeplitz matrices.

Theorem 2.1. *No arithmetic operations are required to compute a Hankel (resp. Toeplitz) displacement generator of length $2t$ for a $t \times t$ block matrix with $g \times g$ Hankel (resp. Toeplitz) blocks.*

Proof. Examples 4.4.3 and 4.4.1 in [41] for $e = f = 0$ reduce the determination of a required generator to the representation of a $tg \times tg$ matrix which has only t nonzero columns and t nonzero rows as the sum of $2t$ outer products of the $2t$ pair of vectors of dimension tg. $\square$

2.2 Computational cost of multiplication.

$O^*(f(n))$ denotes the functions in $(\log f(n))^{O(1)} f(n)$, $n \longrightarrow \infty$, $f(n) \longrightarrow \infty$. $\nu(n)$ operations in a ring $\mathbb{R}$ with unity are sufficient to multiply a pair of polynomials in the ring $\mathbb{R}[\lambda]/\lambda^{n+1}$; $\nu(n) = O((n \log n) \log\log n) = O^*(n)$. $\beta(\zeta)$ bit operations are sufficient to perform an arithmetic operation in $\mathbb{Z}_s$ for a positive integer s such that $\lceil \log s \rceil = \zeta$; $\beta(\zeta) = O((\zeta \log \zeta) \log\log \zeta) = O^*(\zeta)$. For the proofs and further details of the above estimates, see [3, 20]. $v_A = v_{A,\mathbb{R}}$ arithmetic operations in a ring $\mathbb{R}$ are sufficient to multiply a fixed matrix A in $\mathbb{R}^{n\times n}$ by a vector; $v_A \le 2n^2 - n$ for any matrix A in $\mathbb{R}^{n\times n}$, $v_A = o(n^2)$ for sparse and/or structured matrices A.

Theorem 2.2. (a) *$v_A = O(\nu(n))$ for an $n \times n$ Hankel or Toeplitz matrix A. (b) $v_A = O(t\nu(tg))$ for a $g \times g$ block Hankel or block Toeplitz matrix A with $t \times t$ blocks.*

Proof. On part a) see, e.g.,[41, pp. 27–28]. To prove part (b), first interchange the $(i + jt)$th rows and columns of the matrix A according to the permutation $i + jt \longrightarrow j + ig$, $i = 0, \ldots, t-1$; $j = 0, \ldots, g-1$. This turns the matrix A into a $t \times t$ block matrix with Hankel or Toeplitz blocks. Now part (b) follows from part (a) and Theorem 2.1. $\square$

2.3. Invariant factors of a matrix.

The greatest common divisor (gcd) $d_k = d_k(A)$ of all $k \times k$ minors (subdeterminants) of a matrix $A \in \mathbb{Z}^{n\times n}$ is called the kth determinantal divisor of A for $k = 1, \ldots, n$. We write $s_0 = d_0 = 1$ and define the kth Smith invariant factor of A as $s_k = s_k(A) = d_k/d_{k-1}$, $k = 1, \ldots, n$.

It is easily deduced [36] that

$$s_1, \ldots, s_n \in \mathbb{Z}, \tag{2.1}$$

$$s_{i-1} | s_i \quad \text{for} \quad i = 1, \ldots, n, \tag{2.2}$$

$$|\det A| = s_1 s_2 \cdots s_n. \tag{2.3}$$

We also use the crude bound

$$|\det A| \le |A|^n, \ \ |A| = n \ \max_{i,j} |a_{i,j}|, \tag{2.4}$$

where $A = (a_{i,j})_{i,j}$, $|a_{i,j}| = \|\mathbf{a}_{i,j}\|_1$ if $a_{i,j}$ are polynomials with the coefficient vectors $\mathbf{a}_{i,j}$. Hereafter we write $L = \log |A|$.

The Smith factors $s_0(W) = 1$, $s_i(W) \in \mathbb{Z}(\lambda)$, $i = 1, \ldots, n$, can be defined for a matrix polynomial $W(\lambda) \in \mathbb{Z}^{n \times n}[\lambda]$. The properties (2.1)–(2.3) are extended. For $W(\lambda) = \lambda I - A$, we obtain the Frobenius invariant factors $f_i(A) = s_i(\lambda I - A) \in \mathbb{Z}[\lambda]$, $i = 0, \ldots, n$, which define the characteristic and minimum polynomials of the matrix A,

$$\mu_A(\lambda) = f_n(A), c_A(\lambda) = \det(\lambda I - A) = \prod_{i=1}^{n} f_i(A).$$

Hereafter $\mathbf{u}$ denotes the coefficient vector of a polynomial $u(\lambda)$, and $\delta(u)$ denotes its degree. Write $\delta_j = \delta(f_j)$.

2.4. Randomized algorithms.

As in the introduction, for a fixed nonnegative tolerance value $\tau < 1$, randomized algorithms fail to produce the correct output with a probability of at most τ and are partitioned into *Las Vegas* randomized algorithms, which either fail and stop or produce the correct output, and *Monte Carlo* randomized algorithms, which produce the output with no indication whether it is corrupted or not. One may decrease the tolerance bound τ to the level τ^k by allowing the k-fold increase of the number of bit operations and/or random bits involved. Theorems 2.3, 2.6, and 2.7 demonstrate this effect, which can also be achieved by rerunning the algorithms with new values of their random parameters. The overall computational cost estimates for all randomized algorithms in this paper include the cost of the generation of random parameters (see [20, Sec. 18.4] and [2] on bounding the cost of the generation of random primes in a fixed range).

Let us next recall some basic results and techniques.

Hereafter $|S|$ denotes the cardinality of a set S. We say that k values have been *randomly sampled* from this set S if they have been chosen from S at random, independently of each other, under the uniform probability distribution on S. $M^{(i)}$ is the $i \times i$ leading principal (that is, northwestern) submatrix of a matrix M. A matrix M of rank r has *generic rank profile* if the matrices $M^{(i)}$ are nonsingular for $i = 1, \dots, r$. A matrix M is strongly nonsingular if it is nonsingular and has generic rank profile.

Theorem 2.3 [14] (also cf. [58, 53]). *Suppose that a multivariate polynomial of degree d does not vanish identically. Let the values of its variables be randomly sampled from a fixed set S. Then the polynomial vanishes with a probability of at most $\frac{d}{|S|}$.*

In many cases the error/failure is equivalent to the vanishing of a polynomial (e.g., of the determinant of a matrix whose entries are the variables); then we may decrease the bound on the error/failure probability by factor k if we allow by factor of k more elements in the set S. If $S = \mathbb{Z}_{q-1}$ for an integer $q > 2$ or $S = \mathbb{Z}_q - \{0\}$ for $q > 1$, then $|S| = q$ and each element of the set S is represented with $\lceil \log q \rceil$ bits.

Theorem 2.4 [47, Lemma 10.3]. *Let $\pi(x)$ denote the number of primes not exceeding x. Let $y \geq 114$. Then $1 < \pi(y)(\ln y)/y < 1.25$, whereas $\pi(y) - \pi(y/20) > y/(\beta \ln y)$ where $1 - (1/\beta) = \ln 114/(16 \ln 5.7) = 0.17007650\dots$, $\beta = 1.2049303\dots$.*

Theorem 2.5 [47, Theorem 10.1]). *Fix $\epsilon > 0$. Suppose that $M \in \mathbb{Z}^{n \times n}$ is nonsingular, and a prime p is randomly sampled from the range $(y/20, y]$ under the uniform probability distribution in this range where $y = \frac{n\xi \ln |M|}{\nu\epsilon} \geqslant 114$ and $\xi = \frac{16 \ln 114}{16 \ln 5.7 - \ln 114} = 3.278885\dots$. Then we have*

$$P = \text{Probability}((\det M) \bmod p^{\nu} = 0) < \epsilon.$$

Theorem 2.6 [11, Theorem 4.3] (cf. [56]). *Let S be a set in a subfield $\mathbb{F}$ of the field of rational numbers having characteristic zero or greater than n. Assume that D and A are nonsingular matrices in $\mathbb{F}^{n \times n}$, D is a diagonal matrix whose diagonal entries are randomly sampled from the set S. Then with a probability of at least $1 - \frac{(2n-1)n}{|S|}$, the matrix DA has n distinct eigenvalues.*

Theorem 2.7 [29]. *Let S be a set in a ring $\mathbb{R}$. Assume that $A, V, U \in \mathbb{R}^{n\times n}$, rank $A = r$, and U^T and V are unit lower triangular Toeplitz matrices, defined by the $2n-2$ entries of their first columns. Let these entries be randomly sampled from the set S. Then the matrix UAV has generic rank profile with a probability of at least $1 - \frac{(r+1)r}{2|S|}$.*

2.5. Generalized Hensel's lifting for a linear system.

The following algorithm from [47] generalizes Hensel's lifting algorithm in [35, 13] by performing it in the ring $\mathbb{Z}_{qs}$ for two integers $q > 0$ and $s > 1$. The algorithm computes the first h terms in the s-adic expansion of the vector $qM^{-1}\mathbf{b} = q\sum_{i=0}^{\infty}\mathbf{u}^{(i)}s^i$ where $\mathbf{u}^{(i)} \in \mathbb{Z}_{qs}^n$, $i = 0, 1, \ldots$. In this paper we only need the case where $q = 1$ and, apart from Sec. 3.7, where s is a prime.

Algorithm 2.8. Generalized lifting [47].

`INPUT:` a matrix $M \in \mathbb{Z}^{n\times n}$, a vector $\mathbf{b} \in \mathbb{Z}^n$, three positive integers h, q, and s, and a matrix $Q = (qM^{-1}) \bmod (qs)$ satisfying $MQ \mod (qs) = qI$.

`OUTPUT:` the vector $\mathbf{x}^{(h)} \in \mathbb{Z}^n$ such that $\mathbf{x}^{(h)} = (qM^{-1}\mathbf{b}) \bmod (qs^h)$, that is, such that $M\mathbf{x}^{(h)} = (q\mathbf{b}) \bmod (qs^h)$.

`INITIALIZATION:` $\mathbf{r}^{(0)} = \mathbf{b}$.

`COMPUTATIONS:` for $i = 0, 1, \ldots, h-1$, compute the vectors

$$\mathbf{u}^{(i)} = Q\mathbf{r}^{(i)} \bmod (qs), \quad \mathbf{r}^{(i+1)} = (q\mathbf{r}^{(i)} - M\mathbf{u}^{(i)})/(qs).$$

Output the vector $\mathbf{x}^{(h)} = \sum_{i=0}^{h-1}\mathbf{u}^{(i)}s^i$.

The following theorem shows correctness of the algorithm (see part b) and bounds the precision of its computations. For $q = 1$ and a prime s, Algorithm 2.8 and the theorem have appeared in [13].

Theorem 2.9 [47]. *For $\mathbf{r}^{(i)}$ and $\mathbf{x}^{(h)}$ in Algorithm 2.8, we have*

(a) $\mathbf{r}^{(i)} \in \mathbb{Z}^n$ *for all i;*

(b) $M\mathbf{x}^{(h)} = q\mathbf{b} \bmod (qs^h)$*;*

(c) *all components $r_j^{(i)}$ of all vectors $\mathbf{r}^{(i)} = (r_j^{(i)})_j$ satisfy the bounds $|r_j^{(i)}| \leqslant |b_j|/s^i + \alpha n\frac{qs-1}{q}\sum_{k=1}^{i}s^{-k} < \beta/s^i + \alpha n(qs-1)/(qs-q) < \gamma$ where $M = (m_{i,j})_{i,j=1}^n$, $\mathbf{b} = (b_j)_{j=1}^n$*

$$\beta = \beta(\mathbf{b}) = \max_j |b_j|, \quad \alpha = \alpha(M) = \max_{i,j}|m_{i,j}|, \quad \gamma = 2\alpha n + \beta.$$

Proof. (a) $(q\mathbf{r}^{(i)} - M\mathbf{u}^{(i)}) \bmod (qs) = (qI - MQ)\mathbf{r}^{(i)} \bmod (qs)$, and the claim follows because $MQ = qI \bmod (qs)$.

(b) $M\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} M\mathbf{u}^{(i)}s^i = \sum_{i=0}^{h-1}(q\mathbf{r}^{(i)} - qs\mathbf{r}^{(i+1)})s^i = q\mathbf{b} - qs^h\mathbf{r}^{(h)} = q\mathbf{b} \bmod (qs^h)$.

(c) By definition, all components $u_j^{(i)}$ of all vectors $\mathbf{u}^{(i)}$ satisfy $|u_j^{(i)}| \leqslant qs-1$, and so $qs|r_j^{(i+1)}| \leqslant q|r_j^{(i)}| + \alpha n \max_k |u_k^{(i)}| \leqslant q|r_j^{(i)}| + (qs-1)\alpha n$. The claim now follows by induction on i. □

2.6. Multiplication of a block Hankel matrix and its inverse by a vector.

Theorem 2.10. *Suppose we are given a strongly nonsingular $g \times g$ block Hankel (resp. Toeplitz) matrix $B = (b_{i,j})_{i,j=0}^{tg-1}$ with $t \times t$ blocks in $\mathbb{Z}_p^{t\times t}$, and a block vector V with $t \times t$ blocks in $\mathbb{Z}_p^{t\times t}$. Then the matrix $B^{-1}V$ can be computed in $\mathbb{Z}_p$ by using $O(t^2\nu(gt)\beta(\log p)) = O^*(t^2 n \log p)$ bit operations for $n = gt$ and $\nu(n)$ and $\beta(\xi)$ in Sec. 2.2.*

Proof. Interchange the rows and columns of the matrix B according to the permutation $i + jt \longrightarrow j + ig$, $i = 0, \ldots, t-1$; $j = 0, \ldots, g-1$. This preserves strong nonsingularity of B and turns B into a $t \times t$ block matrix $\widetilde{B} = PBP$ with $g \times g$ Hankel blocks. Here P is the matrix of the above permutation, and $\widetilde{B}$ can be immediately represented with its Hankel (resp. Toeplitz) displacement generator of length $2t$ (in virtue of Theorem 2.1). By applying the MBA algorithm in $\mathbb{Z}_p$ [32, 33, 4, 22], [41, Chapter 5], we obtain a Hankel (resp. Toeplitz) displacement generator of length $2t$ for the matrix $\widetilde{B}^{-1}$ by using $O(t^2\nu(n)\log n)$ arithmetic operations in $\mathbb{Z}_p$. Now Theorem 2.10 immediately follows from Theorem 2.2(b) since $B^{-1}V = P^{-1}\widetilde{B}^{-1}P^{-1}V$. □

3. Las Vegas computation of the determinant of an integer matrix

3.1. The basic theorem.

Theorem 3.1. *Given a matrix $A = (a_{i,j})_{i,j=1}^n$ in $Z^{n\times n}$, $n > 1$, $t = \min\{n^{2/5}, \nu_A^{1,3}\}$, and four positive numbers τ, q, s, and w such that $w \geqslant 114$,*

$$s > 5n(n+1)/(2\tau), \tag{3.1}$$

$$w > 16.5(n^2/\tau)\ln(s^2n^2|A|), \tag{3.2}$$

$$w/20 \geqslant q > (10n-5)n/\tau, \tag{3.3}$$

$$w/(\ln w) > 6.025(n/\tau)(L+(2n-1)\log(1+|A|)), \tag{3.4}$$

one can devise a Las Vegas randomized algorithm which

(a) *outputs either FAILURE with a probability of at most τ or otherwise the correct value of* $\det A$,

(b) *generates a random prime in the range $(w/20, w]$ and randomly samples n elements in $\mathbb{Z}_q - \{0\}$, $2tn$ elements in $\mathbb{Z}_{q-1}$, and $2n$ elements in $\mathbb{Z}_{s-1}$, that is, generates $R = (2t+1)n\log q + 2n\log s + \log w$ random bits overall, and*

(c) *performs B bit operations such that $B \leqslant \min\{B_0, B_1\}$, $B_0 = O^*(n^{16/5}L)$, $B_1 = O^*(n^2 v_A^{2/3} L)$.*

Here $L = \log|A|$, $|A| = n\max_{i,j}|a_{i,j}|$ (cf. (2.4)), and v_A is the arithmetic cost of multiplying the matrix A by a vector (cf. Sec. 2.2).

Proof. Theorem 3.1 is supported by the main algorithm in [30] complemented by the acceleration of its stage of computing the MGMP, the minimum generating matrix polynomial. For the sake of completeness, we reproduce this algorithm in Appendix A; readers not familiar with the subject may find it instructive to read this appendix next. We supply more details and describe and analyze this algorithm and its acceleration in the next subsections.

3.2. Deciding the singularity.

Let us first elaborate upon the sketch in [31, Remarks 1 and 3] to test if the matrix A is nonsingular. We fix a sufficiently large random prime $p \leqslant w$ of length $\lceil \log p \rceil = O(\log(nL))$ and apply Gaussian elimination with pivoting to compute $(\det A) \bmod p$ (this involves $O^*(n^3L)$ bit operations). If $(\det A) \bmod p \neq 0$, then $\det A \neq 0$.

Otherwise it is still possible that $\det A \neq 0$, although only within a claimed probability bound for a sufficiently large value of $\log p$ in $O(\log(nL))$, as follows from Theorem 2.5. To turn this Monte Carlo result into a Las Vegas result, we certify that $\det M$ vanishes (if it does), by computing a nontrivial rational solution to the linear system $M\mathbf{x} = \mathbf{0}$. Here $M = UAV$ where V^T and U are two random unit upper triangular Toeplitz matrices defined by the $2n-2$ entries of their first columns. We randomly sample these entries in $\mathbb{Z}_{s-1}$, and deduce from (3.1) and Theorem 2.7 that in both rings $\mathbb{Z}$ and $\mathbb{Z}_{s-1}$ the matrix M has generic rank profile with a probability of at least $1 - \frac{n(n+1)}{2s}$, that is, within the failure probability bound $\tau/5$.

We only need a Monte Carlo solution of the system $M\mathbf{x} = \mathbf{0}$ because the multiplication of the matrix AV by the vector $\mathbf{x}$ immediately enables us to determine if $\mathbf{x}$ is a solution to the system or not. We first compute $r = \text{rank}M$ by applying the binary search for the largest nonsingular leading principal submatrix of M. This takes $O(r^3 \log r)$ arithmetic operations where $r \leqslant n$. Then again if $r = n$, we know that $\det A = \det M \neq 0$. Otherwise, we fix the values of the last $n - r$ components of the vector $\mathbf{x}$ and compute its remaining r components by solving a nonsingular linear system of r equations with the coefficient matrix $M^{(r)}$. This gives us the desired vector $\mathbf{x}$ in $O^*(n^3)$ arithmetic operations.

To keep the bit cost in $O^*(n^3 L)$, however, we first perform all these computations modulo p. Then we keep the selected values of the last $n - r$ components of the vector $\mathbf{x}$, and apply Algorithm 2.8 for $q = 1$, $s = p$ to lift the solution of the linear system of r equations to compute it modulo p^k for $k = \lfloor 2n \log |M| \rfloor + 2$. This takes $O(kn^2 \nu(\log p)) = O(n^3 L)$ bit operations. Due to (2.4), we may recover the rational solution at deterministic bit cost in $O(n\beta(k \log p)) = O^*(nk \log p) = O^*(n^2 \log p)$ for $\beta(\xi)$ in Sec. 2.2, by applying any of the algorithms in [57, 49, 34]. Thus we satisfy the first r equations in the system $M\mathbf{x} = \mathbf{0}$. The $n - r$ last equations are also satisfied unless rank M (which is equal to r in $\mathbb{Z}_p$) increases in $\mathbb{Z}$. Since the matrix has generic rank profile in $\mathbb{Z}$, the rank increase would imply that at least one of the $n - r$ values of $\det M^{(i)}$ for $i = r + 1, \ldots, n$ must vanish in the transition from $\mathbb{Z}$ to $\mathbb{Z}_p$. This, however, may only occur within a probability bound $\tau/5$ due to (3.2), Theorem 2.5, and the upper bound $n^2 s^2 |A|$ on $|M|$. Thus within the Las Vegas bit cost bound $2\tau/5$, we have determined whether $\det A$ vanishes or does not vanish. Now, it remains to compute it assuming that the matrix A is nonsingular.

3.3. The accelerated algorithm.

Algorithm 3.2. Computing the determinant of a nonsingular integer matrix.

`INPUT:` A matrix $A \in \mathbb{Z}^{n \times n}$ and a positive $\tau < 1$.

`OUTPUT:` `FAILURE` with a probability of at most τ or $\det A$.

`INITIALIZATION:`

1. Fix three integers s, q, and w satisfying (3.1)–(3.4).

2. Fix a value t, $1 \leqslant t \leqslant n$, to be specified in Sec. 3.5, and compute $g = \lceil n/t \rceil$ and $l = (2t+1)n$. (Here $t \times t$ is the size of the block entries of X, Y, and A.)
3. Randomly sample $2tn$ elements in $\mathbb{Z}_{q-1}$ and use them as the entries of two matrices $X \in \mathbb{Z}_{q-1}^{t\times n}$ and $Y \in \mathbb{Z}_{q-1}^{n\times t}$.
4. Randomly sample a prime p in the range $(w/20, w]$.
5. Randomly sample n elements in $\mathbb{Z}_q - \{0\}$ and use them as the diagonal entries of the $n \times n$ diagonal matrix D. Compute $d = \det D$. (Note that $p > w/20 \geqslant q$, and so p does not divide d.)

`COMPUTATIONS:`

1. Compute the minimum integer $\eta \geq 0$ such that t divides $\eta + n$. Redenote $n \longleftarrow \eta + n$, $A \longleftarrow \operatorname{diag}(DA, I_\eta)$. (Without changing $\det A$, this turns A into a $g \times g$ block matrix with $t \times t$ blocks for $g = \frac{n}{t}$. Furthermore, it is likely that $\mu_A(\lambda) = n$ due to Theorem 2.6.)
2. Compute the matrices $B^{[i]} = XA^iY$ for $i = 0, 1, \ldots, 2g$, $g = \frac{n}{t}$. (See Stage 1 of Algorithm A.1 in Appendix A.) Form the matrix $B = (B^{[i+j]})_{i,j=0}^{g-1,g}$, which is a $g \times (g+1)$ block Hankel matrix with $t \times t$ blocks $B^{[i+i]}$.
3. Compute $\overline{d} = (\det B) \mod p$ by applying Gaussian elimination. Use pivoting to avoid divisions by zero in $\mathbb{Z}_p$ if possible. If $\overline{d} = 0$, stop and output FAILURE. Otherwise, rank $B = n$ in $\mathbb{Z}_p$. Apply the algorithm supporting Theorem 2.10 to compute in $\mathbb{Z}_p^{t\times t}[\lambda]/\lambda^g$ the block coefficient vector $F = (F^{(i)})_{i=0}^{g}$ of the MGMP, the unique monic minimum generating matrix polynomial $F(\lambda) = F_X^{A,Y}(\lambda) = \Sigma_{i=0}^{g} F^{(i)}\lambda^i$ for the matrix sequence $B^{[0]}, \ldots, B^{[2g-1]}$ or output FAILURE. (Unlike [30] (cf. Algorithm A.1, Stage 3) we require at this stage that $F(\lambda)$ have degree n and be monic.)
4. Apply h steps of the Hensel's lifting Algorithm 2.8 for $q = 1$, $s = p$, $h = \lfloor \log_p(2|A|^n \rfloor + 1$ to compute the coefficient vector of the MGMP $F(\lambda)$ in $\mathbb{Z}_{p^h}$.
5. Compute modulo p^h the value $f_0 = \det F(0) = \det F^{(0)}$ and $f = f_0/d$. If $f \leqslant |A|^n$, output $(-1)^n f$; otherwise output $(-1)^n(f - p^h)$.

3.4. Correctness of the Algorithm.

To prove correctness of Algorithm 3.2, we need the following theorem.

Theorem 3.3 [30, Theorem 1].

(a) *In a fixed field $\mathbb{F}$ the Smith ith leading invariant factor $s_{t+1-i}(F)$ of the $t \times t$ matrix polynomial $F(\lambda) = F_X^{A,Y}(\lambda)$ divides the Frobenius ith leading factor $f_{n+1-i}(\lambda)$ of an $n \times n$ matrix A for any pair of X and Y and for every i, $i = 1, \ldots, t$.*

(b) *For some pair of matrices $X = W$ and $Y = V$, we have $f_{n+1-i}(\lambda) = s_{t+1-i}(F)$, $i = 1, \ldots, t$;*

$$\deg \det F_W^{A,V}(\lambda) = \operatorname{rank} B = \max_{X,Y} \deg \det F_X^{A,Y}(\lambda).$$

Due to the latter equations and to (2.3), we have $\mu_A(\lambda) = f_n(\lambda) = s_n(\lambda I - A) = s_t(F)$. Under the assumption that $\mu_A(\lambda) = c_A(\lambda) = \det(\lambda I - A)$, we have $f_{n-i}(\lambda) = s_{t-i}(F) = 1$ for $i > 0$, $\det(\lambda I - A) = s_t(F) = \det F_X^{A,Y}(\lambda) = \det F(\lambda)$. Therefore, the output value $\det A$ is correct if $\deg \det F(\lambda) = \operatorname{rank} B = n$ in $\mathbb{Z}$.

Suppose that at Stage 3 of Algorithm 3.2, the input matrix B is strongly nonsingular in $\mathbb{Z}_p$. Then Algorithm 3.2 correctly computes the monic MGMP $F(\lambda) \mod p$. By construction, $\det F(\lambda)$ has degree n in $\mathbb{Z}_p$ and therefore in $\mathbb{Z}$. Then Theorem 3.3 implies that the output value of $\det A$ is correct.

To complete the correctness proof it remains to estimate that the overall failure probability is within the bound τ of Theorem 3.1 assuming a nonsingular input matrix A and the specified random choice of a prime p and matrices X, Y, and D. We may have failure only at Stage 3, namely, if $\operatorname{rank} B < n$ and/or the MBA algorithm fails. Neither of these may occur if the matrix $B^{(g)}$ is strongly nonsingular. The following theorem enables us to estimate the likelihood of such a strong nonsingularity.

Theorem 3.4 [30]. *Suppose that a matrix A in $\mathbb{Z}^{n\times n}$ has n distinct nonzero eigenvalues (in an algebraic extension of $\mathbb{Z}$). Let X and Y be two matrices in $\mathbb{Z}^{n\times t}$ whose entries have been randomly sampled in $\mathbb{Z}_{q-1}$. Then the $n \times n$ leading principal submatrix $B^{(n)} = (B^{[i+j]})_{i,j=0}^{g-1,g-1}$ of the matrix B, where $B^{[k]} = X^T A^k Y$, $k = 0, \ldots, 2g-2$, is strongly nonsingular in $\mathbb{Z}$ with a probability of at least $1 - \frac{(n+1)n}{q}$.*

Proof. Let us first assume generic vectors X and Y. Let $\overline{D} = S^{-1}AS$ be a diagonal matrix; $X^T = \widetilde{X}^T S^{-1}$, $Y = S\widetilde{Y}$. ($\overline{D}$ has n distinct diagonal entries, which are the eigenvalues shared by A and $\overline{D}$.) We have $X^T A^i Y = \widetilde{X}^T \overline{D}^i \widetilde{Y}$ for $i = 0, \ldots, g-1$. Therefore it is sufficient to prove the theorem for $A = \overline{D}$.

The factorization $B^{(tk)} = (X^T A^i)_{i=0}^{k-1}(A^i Y)_{i=0}^{k-1}$ implies that rank $B^{(tk)} = n$ if and only if $\mathrm{rank}(X^T A^i)_{i=0}^{k-1} = \mathrm{rank}(A^i Y)_{i=0}^{k} = n$. Now, for $A = \overline{D} = \mathrm{diag}(d_i)_{i=0}^{n-1}$ (with distinct diagonal entries $d_0, \dots, d_{n-1}$), the choice of $X = (x_{i,j})_{i,j=0}^{n-1,g-1}$, $Y = (y_{i,j})_{i,j=0}^{n-1,g}$, $x_{i,j} = d_i^{gj}$, $y_{i,j} = d_i^{(g+1)j}$, turns the matrices $(X^T A^i)_{i=0}^{k-1}$ and $(A^i Y)_{i=0}^{k-1}$ into Vandermonde matrices (up to their column permutation). This implies nonsingularity of the matrices $B^{(h)}$ for $h = 1, \dots, n$.

Thus the matrix $B^{(n)}$ is strongly nonsingular for generic X and Y, that is, $\det B^{(i)} \neq 0$, $i = 1, \dots, n$. Recall that $\det B^{(i)}$ is a polynomial in the entries of X and Y of degree of at most $2i$. If the entries of X and Y are randomly sampled in $\mathbb{Z}_{q-1}$, then with a probabilty of at least $1 - \frac{(n+1)n}{q}$, neither of the polynomials $\det B^{(i)}$ vanishes, in virtue of Theorem 2.3. $\square$

In virtue of bound (3.3) and Theorem 3.4, the unlucky choice of the matrices X and Y may cause the failure with a probability of at most $\tau/5$ under our choice of q. Otherwise, the cause of the failure is restricted to the case where the matrix $(DA) \mod p$ has multiple or zero eigenvalues. Suppose this is due to the transition from A to $(DA) \mod p$. We have $\det(DA) \neq 0$ in $\mathbb{Z}$ (by assumption); in virtue of (3.3) and Theorem 2.6 it is unlikely enough (that is, within the probability bound $\tau/5$) that the matrix DA has multiple eigenvalues. Finally, in virtue of Theorem 2.4, a nonzero integer vanishes in the transition from $\mathbb{Z}$ to $\mathbb{Z}_p$ with a probability of $\pi(w) - \pi(w/20) > w/(\beta \ln w)$, $\beta = 1.2049303\dots$ for our choice of a random prime p in the range $(w/20, w]$. In our case, this should be applied to the two nonzero integers: $\det A$ and the discriminant d_A of $\det(\lambda I - A)$. (2.4) implies that $|d_A| \leqslant (1 + |A|)^{(2n-1)n}$, $|d_A \det A| \leqslant |A|^n (1+|A|)^{(2n-1)n}$. Therefore, the integer $d_A \det A$ can be divided by less than $nL + (2n-1)n\log(1+|A|)$ distinct primes. Combining this estimate with (3.4) and Theorem 2.4 implies that $(d_A \det A) \mod p$ vanishes with a probability of at most $\tau/5$ for our choice of p. Summarizing, we obtain the overall bound τ on the failure probability for Algorithm 3.2, thus completing our proof of its correctness.

Remark 3.1. We apply the MBA divide-and-conquer algorithm in [32, 33, 4] at Stage 3 of Algorithm 3.2 to compute the block coefficient vector $F = (F^{(i)})_{i=0}^{g}$ of an MGMP $F(\lambda) = \sum_{i=0}^{g} F^{(i)}\lambda^i$. Alternatively (see also Sec. 4) we can compute the block vector F by applying to the block matrix $B^{(g)}$ the block versions of the generalized Berlekamp–Massey algorithm [9], the Levinson–Durbin algorithm or the LKS half-gcd algorithm (as proposed in [30]). Any of these algorithms may fail only if the matrix

B does not have the generic rank profile or is rank deficient. Otherwise, by avoiding redundant extra multipliers, one can keep the matrix $F^{(g)}$ nonsingular for generic X and Y [9, 30], and therefore, with a probability of at least $1 - n/q$, for random X and Y in $\mathbb{Z}_{q-1}^{n \times t}$. Based on unimodular diagonalization of the matrix $F^{(g)}$, we deduce from its nonsingularity that $\det F^{(g)}$ has degree n again, as in the MBA case. The application of the LKS block half-gcd algorithm supports the same asymptotic bit cost bounds for computing the MGMP as we have in our Algorithm 3.2 with the application of the MBA algorithm to the matrix $PB^{(n)}P$. With the Levinson–Durbin and generalized Berlekamp–Massey block algorithms, the computation of the MGMP slows down by the factor of $n^2/\nu(n)$ for $\nu(n)$ in Sec. 2.2 but the overall asymptotic bit operation count still matches the bound B_0 in Theorem 3.1.

3.5. Bounding the bit operation cost.

Finally, let us estimate the overall bit operation cost including also the cases where at Stage 3 we replace the MBA algorithm by the alternative algorithms in Remark 3.1. To yield the bound B_0 in Theorem 3.1, we use the estimates in [30] at Stages 1 and 2 (cf. Stage 1 in Algorithm A.1). Clearly, the bit complexity at these stages is dominated by $O(n^3 \log n)$ arithmetic operations with integers of the length $O(rL)$ bits required for computing the power A^r and $O(n^3/r)$ arithmetic operations with integers of the length $O(gL)$ bits required for computing the Krylov sequence $A^{jr}Y$, $j = 1, \ldots, g/r$, where r is any positive integer, $r \leqslant g$. This gives us the overall bound of $O^*((r + g/r)n^3L)$ bit operations at Stages 1 and 2.

At Stage 3, we compute in $\mathbb{Z}_p$ the block vector F which satisfies the block linear system $BF = 0$. At this stage we apply the MBA algorithm to the matrix $PB^{(n)}P$; this gives us the MGMP in $O^*(gt^3) = O^*(nt^2)$ operations in $\mathbb{Z}_p$, that is, $O^*(nt^2L)$ bit operations since $\log p = \log(nL)$. This bound is dominated by the cost bound at the lifting Stage 4. We need $h = O(nL)$ lifting steps, each requiring $O^*(nt^2)$ arithmetic operations with $O(\log p)$-bit integers (see Theorem 2.10), that is, $O^*(nt^2L)$ bit operations. This means $O^*(n^2t^2L)$ bit operations at Stage 3 for all lifting steps. Clearly, this bound, together with that in $O((r + \frac{n}{rt})n^3L)$ at Stage 2 (which holds for any choice of positive integers r and t such that $r \leqslant g = n/t \leqslant n$) dominates the bit cost at Stage 5 and, therefore, the overall bit cost estimate. The choice of $r = \lceil n^{1/5} \rceil$, $g = r^3$ implies the overall bit cost bound B_0. To yield the bound B_1, we choose $r = 1$, $t = \lceil v_A^{2/3} \rceil$ and trivialize Stage 2 of Algorithm 4.2 as follows: write $X_0 = X$ and recursively compute the matrices $B^{[i]} = X_i^T Y^T$, $X_{i+1}^T = X_i^T A$ for $i = 0, 1, \ldots, 2g$, $g = n/t$. This supports the bit cost bound $O((1 + v_A/t)n^2 L)$ at Stage 2. Write $t = \lceil v_A^{2/3} \rceil$ and obtain the overall bit cost bound B_1. This completes the proof of Theorem 3.1. □

3.6. Saving word operations.

Assume the realistic model where the length λ of a computer word is fixed and the computational cost is measured by the number of word operations involved. Then a single arithmetic operation with a precision ξ is a word operation if $\xi \leqslant \lambda$ and requires $O^*(\xi/\lambda)$ word operations otherwise.

Theorem 3.5. *Algorithm* 3.2 *supporting Theorem* 3.1 *can be modified to support parts* (a) *and* (b) *of Theorem* 3.1 *by performing* W *word operations where* $W \leqslant \min\{W_0, W_1\}$, $W_0 = O^*(n^{16/5}L/\lambda)$ *if* $\lambda = O(n^{1/5}L)$, $W_1 = O^*(n^2 \nu A^{2/3} L)$ *if* $\lambda = O(nL)$, *and* λ *is the length of a computer word.*

Proof. Clearly, the word complexity at Stages 1, 2, and 5 of Algorithm 3.2 decreases by the factor λ versus the bit complexity under the assumptions $\lambda = O(n^{1/5}\lambda)$ or $\lambda = O(nL)$, respectively. The MBA algorithm uses $O(t^2 n)$ arithmetic operations which are also operations at Stage 3. Stage 4 requires only $O^*(t^2 n^2/\lambda)$ word operations if we perform Stage 3 modulo p^v for $v = \lfloor \lambda/\log p \rfloor$ and then apply the generalized lifting Algorithm 2.8 for $g = 1$, $s = p^v$ at Stage 4. The degeneration modulo p^v cannot occur unless it occurs modulo p, and so our failure analysis remains valid. The word cost at Stage 3 does not change, and it is dominated at Stage 4 if $\lambda = O(nL)$. □

4. Discussion

In spite of the competition with numerical methods, our amendment of the algorithm in [30] promises to be practically useful, particularly where the input matrices A are sparse and/or structured. In [17] and [18] this approach, including Theorem 3.1 and Algorithm 3.2, was extended to computing the determinant of univariate and multivariate matrix polynomials with applications to computing the univariate and multivariate resultants and solving multivariate polynomial systems of equations. The resultant matrices are special (they are both sparse and structured); the efficient algorithm must exploit this fact. For the applications to the resultants, the papers [17], [18] employ a trivialized version of the algorithm in [30] (dropping the baby steps/giant steps techniques, since the application of these techniques would destroy the structure and sparsity). The papers also follow the approach in [5, Section 4] to decrease the cost of the evalulation of a scalar determinant where its absolute value is small. This yields the output-sensitive acceleration where $\det A$ vanishes, as is the case for the solution of a polynomial system.

The algorithms in [17] and [18] rely essentially on the algorithm in [30], on extending it to the multivariate input in [42], and on exploiting its output-sensitive version. By incorporating our present algorithm instead of the algorithm in [30], we can enhance the overall efficiency.

The extension from computing the determinant to computing all Smith's factors is quite straightforward due to [19]. Storjohann in [51] and [52] achieves a breakthrough by computing the determinants of $n \times n$ integer and polynomial matrices at the asymptotically optimal Las Vegas bit cost in $O^*(MM(n)L)$, where $MM(n)$ is the arithmetic complexity of $n \times n$ matrix multiplication, $MM(n) = O(n^{2.376})$. The technique in [54] and the algorithm in [51] combined with the one in [30] enable the Monte Carlo extension of the bit cost bound $O^*(n^{16/5}L)$ to computing the minimum and characteristic polynomials and all Frobenius factors of an $n \times n$ integer matrix A.

One may compute the eigenvalues of the input matrix A as the roots of its minimum and characteristic polynomials $\mu_A(\lambda)$ and $c_A(\lambda)$. The algebraic and geometric multiplicities of the eigenvalues are given by their multiplicities as the roots of $c_A(\lambda)$ and $\mu_A(\lambda)$, respectively. The record (and optimal up to polylog factors) bit complexity bounds at the root-finding stage have been achieved in [40]. The application of the root-finder from this paper yields the following result (cf. [42, Corollary 5.8]).

Theorem 4.1. *All eigenvalues of a matrix $A \in \mathbb{Z}^{n\times n}$ can be approximated within 2^{-b} for $b \geqslant 2n\log(n|A|)$ by using $O(R)$ random bits and $O(B + (\mu(n)\log n)(\log^2 n + \log b)bn)$ bit operations for $\beta(\zeta) = O(\zeta\log\zeta\log\log\zeta)$ and B and R from Theorem 3.1. The same bit cost bound for a sufficiently large value of b covers the computation of the geometric and algebraic multiplicities of the eigenvalues.*

The extension from the polynomial case in [51] to the integer case in [52] involves a practically promising approach, which has technical similarity to [39] and [16], but the practical value of the algorithms for computing the determinant and the minimum and characteristic polynomials in the papers [51] and [52] is not clear so far. Their power is restricted to the case of the dense and unstructured input. Like the baby step/giant step stage in the algorithm in [30], the algorithms in [51] and [52] rely on fast computation of the powers of the input matrix. Thus they do not preserve and do not exploit its sparsity and structure, and so they do not compete with the bit complexity bounds and the algorithms in [17] and [18] for sparse and structured input.

Trying to yield more efficient variant of the algorithms in this paper, we may apply the block version of the MBA algorithm directly to the matrix $B^{(n)}$ considered as $g \times g$ block Hankel matrix, instead of applying the scalar version of this algorithm to the matrix $PB^{(n)}P$. We still need to elaborate upon the time-complexity and degeneration analysis of this variation.

Another tentative practical improvement is via performing the lifting computations modulo 2^w where $\log w$ approximates the length λ of a computer word below. The paper [47] elaborates upon this approach and estimates that it is unlikely that the resulting algorithm would degenerate for random integer input matrices and larger exponent w, but we must also elaborate upon the computation of the initial inverse for lifting. We may rely on the MBA algorithm, its block version, or the algorithms in Remark 3.1, but we must estimate the probability of degeneration (possibly just with the same tools as in [47]) and control the time complexity at this stage (cf. [46]).

Finally, our further study of the determinant algorithm and attempts of its improvement and extension has lead us to some interesting technical problems, which we state in the Appendix, parts B and C.

Appendix

A. The Wiedemann–Coppersmith–Kaltofen–Villard algorithm

Assume $\det A \neq 0$. The algorithm in [30] for computing $\det A$ extends the Wiedemann–Coppersmith's approach. Wiedemann in [56] first preconditions the matrix A, by using random diagonal scaling, to ensure probabilistically that $\det \mu_A(\lambda) = n$, so that $\mu_A(\lambda) = c_A(\lambda)$, and therefore $\det A = (-1)^n \mu_A(0)$. This means a Las Vegas reduction of the determinant problem to computing $\mu_A(\lambda)$, the minimum polynomial of A. In [56] this polynomial is computed as the generator for the linear recurrence sequence of scalars $b^{(i)} = \mathbf{x}^T A \mathbf{y}$ where $\mathbf{x}$ and $\mathbf{y}$ are random integer vectors. The overall bit complexity of this computation is dominated at the stage of computing the Krylov sequence $A^i\mathbf{y}$, $i = 1, \ldots, 2n-1$, since the generator $\mu_A(\lambda)$ is rapidly computed based on the Berlekamp–Massey celebrated algorithm for the recovery of the linear recurrence coefficients. Instead of using the latter algorithm, one may compute the coefficient vector of $\mu_A(\lambda)$ as a vector from the right null space of the Hankel matrix $H^{(r)} = (b^{(i+j)})_{i,j=0}^{r-1,r}$ where the matrix $H = (b^{(i+j)})_{i,j=0}^{n-1,n}$ has rank r and the matrix $H^{(r)}$ is nonsingular [28, 7]. Wiedemann proposed this approach for both computing $\det A$ and solving linear systems $A\mathbf{x} = \mathbf{f}$. Coppersmith in [9] proposed an acceleration for the linear systems by using block vectors (that is, matrices) X and Y. Kaltofen and Villard in [30] extended this acceleration to computing $\det A$ and supplied the probabilistic and bit cost analysis. They also employed the known techniques of baby steps/giant steps to yield additional acceleration and elaborated upon the transition from $F(\lambda)$, the MGMP for the sequence of block scalars $X^T A^i Y$, $i = 0, 1, \ldots$, to $\det A$. Kaltofen and Villard compare the Smith invariant factors for both matrices $F(\lambda)$ and $I - \lambda A$ and show that, up to scaling, they are likely to be identical for random X and Y, and therefore, $\Delta(\lambda) = \det F(\lambda) = \mathrm{lc}(\Delta)\det(\lambda I - A)$, $\det F(0) = (-1)\det A$ where $\mathrm{lc}(\Delta)$ is the leading coefficient of $\Delta(\lambda) = \det F(\lambda)$.

Here is the resulting algorithm where the matrix A is nonsingular.

Algorithm A.1. The determinant of a nonsingular matrix [30].

`Input:` preconditioned matrix $A \in \mathbb{Z}^{n\times n}$, positive integers m, r and s, $m \leqslant n$, $r = \lceil (2n/m + 3)/s \rceil$. (Preconditioning is by Wiedemann's technique or its extensions.)

`Output:` "failure" with a low probability or $\det A$.

`Initialization:` select positive integers m, r, s, $\gamma = n^{O(1)}$, $m \leqslant n$, $r = \lceil (2n/m + 3)/s \rceil$ and random matrices $X, Y \in \mathbb{Z}_\gamma^{n\times m}$.

`Computations:`

1. Compute $B^{[i]} = X^T A^i Y$, $i = 0, 1, \ldots, \lfloor 2n/m + 3 \rfloor$.
2. Compute the minimal matrix generating polynomial $F_X^{A,Y}(\lambda)$ for the sequence $\{B^{[i]}\}_{i\geqslant 0}$.
3. Compute the leading and constant coefficients of $\Delta(\lambda) = \det(F_X^{A,Y}(\lambda))$. If $\deg(\Delta) < n$ and $\Delta(0) \neq 0$ then return "failure"; otherwise return $\det A = \Delta(0)/(\text{leading coefficient of } \Delta(\lambda))$. (In Algorithm 3.2, the latter division is avoided since the polynomial $\Delta(\lambda)$ is monic.)

Stage 1 is performed by using the baby steps/giant steps technique as follows (V^T is the transpose of V):

1.1 **For** $j = 1, 2, \ldots, r-1$, **Do** $V^{[j]} \leftarrow A^j Y$;

1.2 $Z \leftarrow A^r$;

1.3 **For** $k = 1, 2, \ldots, s$, **Do** $(U^{[k]})^T \leftarrow X^T Z^k$;

1.4 **For** $j = 0, 1, \ldots, r-1$, **Do**
For $k = 0, 1, \ldots, s$, **Do** $B^{[kr+j]} \leftarrow (U^{[k]})^T V^{[k]}$.

The computation at Stages 2 and 3 are performed modulo sufficiently many random primes and the ouput value of det A is recovered by applying the Chinese remainder algorithm.

B. "Bad" Primes: Two Theorems

The following results can additionally back up and help to refine our estimates for the failure probability.

Theorem B.1 [21, Lemma 2.3]**.** *Let $A \in \mathbb{Z}^{n \times n}$. Let $f_i(A)$ and $f_{i,p}(A)$ denote the ith invariant factors of A in $\mathbb{Z}$ and $\mathbb{Z}_p$, respectively, for a prime p and $i = 1, \ldots, n$. Let P denote the product of all primes p for which $f_i(A) \bmod p \neq f_{i,p}(A)$ (we call such primes "bad"). Then $P \leq (n^2|A|)^{n^2}$.*

Theorem B.2. *Random sampling in the range $[x, 20x)$ for $x = kn^2|A| \geqslant 5.7$ produces a single "bad" prime with a probability of at most*

$$P = \frac{\ln(20kn^2|A|)}{14k|A|}. \tag{B.1}$$

(This bound decreases to zero as $k|A|$ grows to the infinity, and any prime chosen in the range has at most $\log(20x) \leq \lceil \log(n^2|A|) + \log(20k) \rceil$ bits.)

Proof. By Theorem 2.4, there are at least

$$\frac{20x}{\beta \ln(20x)} - \frac{1.25x}{\beta \ln(x)} > \frac{17x}{\beta \ln(20x)} > \frac{14x}{\ln(20x)}$$

primes in the above range. Due to Theorem B.1, at most n^2 of them can be "bad." □

C. Some Numerical Bounds on the Smith and Frobenius Factors

Can we decrease the bit operation complexity of matrix computations by extending the domain of our study from integers to real and complex numbers? Such an extension helped us a little in the proof of Theorem 3.4. We have the following results, which seem to be relevant to our study.

Given that $A = (a_{i,j})$, $|a_{i,j}| = \|\mathbf{a}_{i,j}\|_1$ if $a_{i,j}$ are polynomials with the coefficient vectors $\mathbf{a}_{i,j}$, we have

$$r_i = s_i/s_{i-1}, \quad r_i^{(q)} = r_i \bmod q, \quad i = 1, \ldots, n \tag{C.1}$$

and

$$|\det A| \leqslant |A|^n, \quad \|A\| = \|A\|_1 = \max_j \sum_i |a_{i,j}|. \tag{C.2}$$

(2.3), (C.1), and (C.2) together imply that

$$s_i = \prod_{j=1}^{i} r_j, \quad |\det A| = \prod_{g=1}^{n} r_g^{n-g+1} \leqslant \|A\|^n, \tag{C.3}$$

$$r_g \leqslant r_g^+ = \|A\|^{n/(n-g+1)}, \quad g = 1, \ldots, n. \tag{C.4}$$

Having any lower bounds on r_g and/or s_i, we yield an improvement.

Lemma C.1. *Let* $\widetilde{r}_i \leqslant r_i$, $i = 1, \ldots, l$; $\widetilde{s}_i \leqslant s_i$ *for* $i = l+1, \ldots, n$. *Then*

$$r_g \leqslant \widetilde{r}_g c_g, \quad c_g = \left(\|A\|^n \Big/ \left(\prod_{i=1}^{l} \widetilde{r}_i^{\,l-i+1} \prod_{j=l+1}^{n} \widetilde{s}_j\right)\right)^{1/(l-g+1)}, \quad g = 1, \ldots, l.$$

Write $r_i(\lambda) = f_{n-i+1}(\lambda)/f_{n-i}(\lambda)$, $\delta_j = \delta(f_j)$.

Lemma C.2. $|\mathbf{f}_j| \leqslant (1 + \|A\|)^{\delta_j}$, $|\mathbf{r}_j| \leqslant (1 + \|A\|)^{\delta(r_j)}$ *for all* j.

Proof. Each $r_j(\lambda)$ and $f_j(\lambda)$ is a polynomial $\prod_g (\lambda - \lambda_{i_g})$ where λ_{i_g} are the eigenvalues of A, so $|\lambda_{i_g}| \leqslant \|A\|$ for all j. $\square$

Lemma C.3. $\delta_{n-k+1} \leqslant n/k$, $k = 1, \ldots, n-1$.

Proof. We have $\sum_{i=0}^{n} \delta_{n-i} = n$, $\delta_{n-k+1} \leqslant \delta_{n-k+i}$ for $i \geqslant 1$, so $\delta_{n-k+1} \leqslant n - \sum_{i=0}^{k} \delta_{n-i} \leqslant n - (k+1)\delta_{n-k} + 1$, $\delta_{n-k+1} \leqslant n/k$. $\square$

Our attempts to use these bounds failed due to some flaws in Sec. 4 and 5 in [42]. How much can the latter lemmas help in computing the matrix determinants and minimum and characteristic polynomials?

Acknowledgements. I am grateful to the referees for their helpful comments.

References

1. J. Abbott, M. Bronstein, T. Mulders, *Fast Deterministic Computations of the Determinants of Dense Matrices.* — In: *Proceeding of International Symposium on Symbolic and Algebraic Computation* (ISSAC'99), ACM Press, New York (1999), pp. 197–204.
2. M. Agrawal, N. Kayal, N. Saxena, *Primes in* P. Preprint, 2002. Available from http:/www.cse.iitk.ac.in/news/primality.pdf
3. D. J. Bernstein. *Fast Multiplication and Its Applications.* — Preprint, 2003. Available from http://cr.yp.to/papers.html
4. R. R. Bitmead, B. D. O. Anderson, *Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations.* — Linear Algebra and Its Applications **34** (1980), 103–116.
5. H. Brönnimann, I. Z. Emiris, V. Y. Pan, S. Pion, *Sign Determination in Residue Number Systems.* — Theoretical Computer Science **210**, **1** (1999), 173–197.
6. B. Beckermann and G. Labahn, *A Uniform Approach for Fast Computation of Matrix-Type Padé Approximants.* — SIAM J. Matrix Anal. Appl. **15**, **3** (1994), 804–823.
7. D. Bini, V. Y. Pan, *Polynomial and Matrix Computations, Volume. 1: Fundamental Algorithms.* — Birkhäuser, Boston (1994).
8. K. L. Clarkson, *Safe and Efficilent Determinant Evaluation.* — In: *Proceedings* 33*rd Annual Symp. Foundations of Comp. Sci.* (FOCS92), Los Alamitos, California, IEEE Computer Society Press, (1992), pp. 387–395.
9. D. Coppersmith, *Solving Homogeneous Linear Equations over $GF(2)$ via block Wiedemann Algorithm.* — Math. of Computation **62**, **205** (1994), 333–350.
10. D. Coppersmith, *Rectangular Matrix Multiplication Revisited.* — J. Complexity **13** (1997), 42–49.
11. 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.
12. D. Coppersmith, S. Winograd, *Matrix Multplication via Arithmetic Progressions.* — J. Symolic Comput. **9**, **3** (1990), 251–281.
13. J. D. Dixon, *Exact Solution of Linear Equations Using p-adic Expansions.* — Numerische Math. **40** (1982), 137–141.
14. R. A. DeMillo, R. J. Lipton, *A Probabilistic Remark on Algebraic Program Testing.* — Information Process. Letters **7**, **4** (1978), 193–195.
15. W. Eberly, M. Giesbrecht, G. Villard, *On Computing the Determinant and Smith Form of an Integer Matrix.* — In: *Proc.* 41*st Annual Symposium on Foundations of Computer Science* (FOCS'2000), IEEE Computer Society Press, Los Alamitos, California (2000), pp. 675–685.
16. I. Z. Emiris, V. Y. Pan, Y. Yu, *Modular-Arithmetic for Linear Algebra Computations in the Real Field.* — J. of Symblolic Computation, **26** (1998), 71–87.
17. I. Z. Emiris, V. Y. Pan, *Improved Computation of Determinants and Resultants.* — In: *Proc. of the* 6*th Intern. Workshop on Computer ALgebra in Scientific Computing* (CACS'03) (E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov Editors) *Tech. Univ. München*, Germany (2003), pp. 81–99. (Also Technical Report 2002-019, Ph. D. Program in Computer Science, The Graduate Center, CUNY (2002).

18. I. Z. Emiris, V. Y. Pan, *Improved Algortihms for Computing Determinants and Resultants.* — J. of Complexity (2004) (in press).
19. M. Giesbrecht, *Fast Computation of the Smith Form of a Sparse Integer Matrix.* — Computational Complexity **10** (2001), 41–69.
20. J. von zur Gathen, J. Gerhard, *Modern Computer Algebra.* — Cambridge University Press (2nd edition), Cambridge, UK (2003).
21. M. Giesbrecht, A. Storjohann, *Computing Rational Forms of Integer Matrices.* — J. of Symbolic Computation **34**, **3** (2002), 157–172.
22. E. Kaltofen, *Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems.* — Math. of Computation **64**, **210** (1995), 777–806.
23. I. Kaporin, *A Practical Algorithm for Faster Matrix Multiplication.* — Numerical Linear Algebra with Applications **6**, **8** (1999) 687–700.
24. E. Kaltofen, *An Output-sensitive Variant of the Baby Steps/Giant Steps Determinant Algorithm.* — In: *Proc. ACM Intern. Symp. on Symbolic & Algebraic Computation* (ISSAC'02), ACM Press, New York (2002), pp. 138–144.
25. E. Kaltofen, *Private communication*, May (2002).
26. I. Kaporin, *The Aggregation and Cancellation Techniques as a Practical Tool for Faster Matrix Multiplication.* — Theoretical Computer Science **315**, **2–3** (2004), 469–510.
27. E. Kaltofen, M. S. Krishnamoorthy, B. D. Saunders, *Parallel Algorithms for Matrix Normal Forms.* — Linear Algebra and Applications **136** (1990), 189–208.
28. E. Kaltofen, V. Y. Pan, *Processor Efficient Parallel Solution of Linear Systems over an Abstract Field.* — In: *Proceedings of 3rd Annual ACM Symposium on Parallel Algorithms and Architectures* (SPAA'91) , ACM Press, New York (1991), pp. 180–191.
29. E. Kaltofen, B. D. Saunders, *On Wiedemann's Method for Solving Sparse Linear Systems.* — In: *Proceedings of AAECC–5, Lecture Notes in Computer Science*, **536**, Springer, Berlin (1991), pp. 29–38.
30. E. Kaltofen, G. Villard, *On the Complexity of Computing Determinants.* — In: *Proc. Fifth Asian Symposium on Computer Mathematics (ASCM 2001)*, (Shirayanagi, Kiyoshi and Yokoyama, Kazuhiro, editors), *Lecture Notes Series on Computing*, **9**, World Scientific, Singapore (2001), pp. 13–27.
31. E. Kaltofen, G. Villard, *Computing the Sign or the Value of the Determinant of an Integer Matrix, a Complexity Survey.* — J. Computational Applied Math. **162**, **1** (2004), 133–146.
32. M. Morf, *Fast Algorithms for Multivariable Systems.* — Ph. D. Thesis, Department of Electrical Engineering, Stanford University, Stanford, California (1974).
33. M. Morf, *Doubling Algorithms for Toeplitz and Related Equations.* — In: *Proceedings of IEEE International Conference on ASSP*, IEEE Press, Piscataway, New Jersey (1980), pp. 954–959.
34. M. Monahan, *Maximal Quotient Rational Reconstruction: an Almost Optimal Algorithm for Rational Reconstruction.* — In: *Proc. International Symp. on Algebraic and Symbolic Computation* (ISSAC'04), ACM Press, New York (2004), pp. 243–249.
35. R. T. Moenck, J. H. Carter.*Approximate Algorithms to Derive Exact Solutions to Systems of Linear Equations.* — In: *Proceedings of EUROSAM, Lecture Notes*

in Computer Science, **72**, Springer, Berlin (1979), pp. 63–73.
36. M. Newman, *Integral Matrices.* — Academic Press, New York (1972).
37. V. Y. Pan, *Complexity of Parallel Matrix Computations.* — Theoretical Computer Science **54** (1987), 65–85.
38. V. Y. Pan, *Computing the Determinant and the Characteristic Polynomials of a Matrix via Solving Linear Systems of Equations.* — Information Processing Letters **28** (1988), 71–75.
39. V. Y. Pan, *Can We Utilize the Cancelation of the Most Significant Digits?.* — Tech. Report TR-92-061. The International Computer Science Institute, Berkeley, California (1992).
40. V. Y. Pan, *Univariate Polynomials: Nearly Optimal Algorithms for Numerical Factorization and Rootfinding.* — J. of Symbolic Computation **33**, **5** (2002), 701–733. (Proc. version in *Proc. International Symposium on Symbolic and Algebraic Computation* (ISSAC'01), ACM Press, New York (2001), pp. 253–266.)
41. V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms.* Birkhäuser/Springer, Boston–New York (2001).
42. V. Y. Pan, *Randomized Acceleration of Fundamental Matrix Computations.* — In: *Proc. Symp. on Theoretical Aspects of Computer Science (STACS)*, Lect. Notes in Comput. Sci. **2285**, Springer, Heldelberg, Germany (2002), pp. 215–226.
43. V. Y. Pan, *Can We Optimize Toeplitz/Hankel Computations?* — In: Proc. of the 5th Intern. Workshop on Computer Algebra in Scientific Computing (CASC'03). E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov (eds.), Tech. Univ. München, Germany (2002), pp. 253–264.
44. V. Y. Pan, *Nearly Optimal Toeplitz/Hankel Computations.* — Technical Reports 2002-001 and 2202-017. Ph. D. Program in Computer Science, The Graduate Center, CUNY, New York (2002).
45. V. Y. Pan, *Superfast Algorithms for Singular Integer Toeplitz/ Hankel-like Matrices.* — Technical Reports 2002-002 and 2003-004. Ph. D. Program in Computer Science, The Graduate Center, CUNY, New York (2002–2003).
46. V. Y. Pan, *Superfast Divide-and-Conquer Algorithms for Integer Toeplitz-like Matrices.* — Technical Report 2004-015. Ph. D. Program in Computer Science, The Graduate Center, CUNY, New York (2004).
47. V. Y. Pan, B. Murphy, R. E. Rosholt, X. Wang, *Toeplitz and Hankel Meet Hensel and Newton Nearly Optimal Algorithms and Their Practical Acceleration with Saturated Initialization.* — Technical Report 2004-013. Ph. D. Program in Computer Science, The Graduate Center, CUNY, New York (2004).
48. V. Y. Pan, X. Wang, *Acceleration of Euclidean Algorithm and Extensions.* — In: *Proceedings of the International Symposium on Symbolic and Algebraic Computation* (ISSAC'02), (Teo Mora editor), ACM Press, New York (2002), pp. 207–213.
49. V. Y. Pan, X. Wang, *On Rational Number Reconstruction and Approximation.* — SIAM J. on Computing **33**, **2** (2004), 502–503.
50. V. Y. Pan, Y. Yu, *Certification of Numerical Computation of the Sign of the Determinant of a Matrix.* — Algorithmica **30** (2001), 708–729.
51. A. Storjohann, *High Order Lifting and Integrality Certification.* — J. of Symbolic Computation **36**, **3–4** (2003), 613–648. (Proc. version in *Proc. of Intern. Symposium on Symbolic and Algebraic Computation*, ACM Press, New York (2002),

pp. 246–254.)

52. A. Storjohann, *The Shifted Number System for Fast Linear Algebra on Integer Matrices.* — Technical Report TR CS-2004-18, School of Computer Science, University of Waterloo, Canada, April (2004).
53. J. Schwartz, *Fast Probabilistic Algorithms for Verification of Polynomial Identities.* — J. ACM **27**, **4** (1980), 701–717.
54. A. Storjohann, *Computing the Frobenius Form of a Sparse Integer Matrix.* — Preprint.
55. E. Thomé, *Fast Computation of Linear Generators for Matrix Sequences and Application to the Block Wiedemann Algorithm.* — In: *Proc.* 2001 *Internat. Symp. Symbolic Algebraic Comput.* (ISSAC 2001), ACM Press, New York (2001), pp. 323–331.
56. D. Wiedemann, *Solving Sparse Linear Equations over Finite Fields.* — IEEE Trans. Inf. Theory IT-**32** (1986), 54–62.
57. X. Wang, V. Y. Pan, *Acceleration of Euclidean Algorithm and Rational Number Reconstruction.* — SIAM J. on Computing **32**, **2** (2003), 548–556.
58. R. Zippel, *Probaliistic Algorithms for Sparse Polynomials.* — Lect. Notes Comp. Science (Proc. EUROSAM'79) Springer, Berlin (1979), 216–226.

Department of Mathematics
and Computer Science
Lehman College of CUNY, USA
http://comet.lehman.cuny.edu/vpan/

E-mail: victor.pan@lehman.cuny.edu

Поступило 27 ноября 2004 г.