ТЕОРИЯ СЛОЖНОСТИ
ВЫЧИСЛЕНИЙ
IX
Сборник работ под редакцией
Э. А. Гирша
ПОСВЯЩАЕТСЯ 50-ЛЕТИЮ
ДМИТРИЯ ЙОРЬЕВИЧА ГРЮГОРЬЕВА
САНКТ-ПЕТЕРБУРГ
2004
Предисловие редактора

Настоящий том "Записок научных семинаров ПСМИ" продолжает серию "Теория сложности вычислений" и посвящается пятидесятилетию Дмитрия Юрьевича Грингвельда, одного из ведущих редакторов и постоянного автора этой серии.

Тематика статей сборника отражает заинтересованность математических интересов Юбидов. Статьи Л. Малена и В. И. Пана посвящены сложности алгоритмов для алгебраических задач (разложение многочленов на множители и проверка неприводимости), а также и т.о. В. Якубов и М. В. Руден в описании размерности полпредикативных множеств. Некоторые работы связаны с комбинаторной сложностью, работа С. А. Бадаева и Н. Н. Поломаренко об ассоциативных схемах, работа В. Варфоломеева и перестановочных, независимых относительно минимальной работы С. А. Верещагина, Ж. П. Эммер и В. С. Серебряев и Л. С. Миловидов о трехслойных. Статьи Р. Патрик и Л. Пулавский предлагают новый метод доказательства нижних оценок для арифметических схем.

Четыре статьи посвящены задачам математической логики. Это статьи Г. М. Ивахnenko и А. О. Ковачеков, освоение доказательств для интуиционистской логики, статьи Е. П. Орехова о новых разрешных фрагментах исчисления, статьи А. С. Куликов и С. С. Федия об автоматическом доказательстве оценок времени работы алгоритмов для задачи (максимальной) пропозициональной выполнимости, а также статьи И. Ишлинский и А. М. Скворцова о логических оценках в физических теориях.

Э. А. Гирш

http://www.pmi.ras.ru/zna

© Санкт-Петербургское отделение Российской Академии Наук, 2004

Contents

Arad A. Complexity bound of absolute factoring of parametric polynomials ........................................... 5
Bargachew V. On some properties of min-wise independent families and groups of permutations .................. 30
Basu S., Polack R., Roy M.-F. Computing the dimension of a semi-algebric set ............................................. 42
Erdokimov S. A., Ponomarenko I. N. On the vertex connectivity of a relation in association schemes .................. 55
Keinovich V., Finkelstein A. I. Towards applying computational complexity to foundations of physics .......... 63
Kulikov A. S., Fedin S. S. Automated proofs of upper bounds on the running time of splitting algorithms ........ 111
Mints G., Kojima K. Intuitionistic Frege systems are polynomially equivalent .............................................. 129
Orevkov V. P. A new decidable Horn fragment of the predicate calculus ......................................................... 147
Pan V. Y. On theoretical and practical acceleration of randomized computation of the determinant of an integer matrix ............................................... 163
Paterson R., Pudlák P. Circuit lower bound and linear codes ................................................................. 188
Efrat O., Reissayre J.-P., Simant P. On infinitesimal trace rational languages of maximum topological complexity ........................................ 205
Reviews .................................................................................................................................................................. 224
ON THEORETICAL AND PRACTICAL ACCELERATION OF RANDOMIZED COMPUTATION OF THE DETERMINANT OF AN INTEGER MATRIX

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 \( A = (a_{ij}) \) was studied recently in [15] and [30].

The paper [5] follows the approach in [37, Appendix], [9], [1], but contributes some novel ingenious techniques to yield a randomized Monte Carlo algorithm for \( \det A \) using \( O\left(n^{3/2}\log n\right) \) bit operations for the average input matrix \( A \) and \( O\left(n^{7/2}\log n\right) \) for the worst case input. Here and hereafter: \( L = \log \left(\max_{i,j} |a_{ij}| \right) \), \( O\left(f(n)\right) \) means \( f(k)\log f(k) \) for all \( k \geq 2 \), and Monte Carlo algorithms allow erroneous output, although with a smaller probability, bounded by a fixed nonnegative tolerance value \( \epsilon < 1 \). 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 MN). The practical accelerations of MN by Strassen in 1969, supporting the
arithmetic complexity bound in $O(n^{2.83})$, and more recently by Kaporin [23, 26] have so far led to no practical acceleration of computing the determinants because of the sub-stational overhead of the transition from fast MM to the computation of the determinant. Theorem decreases of all cited exponents for the complexity of computing the determinant, however, is possible based on asymptotically fast MM in [12, 11].

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

Furth, as, asymptotic acceleration has been achieved in [31] based on a distinct approach. The main result (Theorem 2) in [31] 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. Ketofen and Villard combine Wiedemann's algorithm in [56] with their block version proposed by Coppersmith in [9]. Like these two papers and unlike [15], the paper in [31] allows additional acceleration where the input matrices can be multiplied by vectors fast. The paper exploits the known techniques of baby steps/giant steps, show 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 [31] 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 version paper is the journal version of our proceeding papers [42] and [43] (see also [41]), more precisely, of the parts of [42] and [43] relevant to improving the algorithm in [31] by accelerating its computation of the MGMP. The paper [31] proposes to compute the MGMP

"by a block version of the Berlekamp/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] hast 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 [31], whereas the application of the MBA divide-and-conquer algorithm in [23, 24] yields the arithmetic cost in $O(n^2)$ for computing the MGMP and this decreases the exponent $30/3$ in [31] to $16/5$ [42, Theorem 5.a]. An alternative derivation of the asymptotic arithmetic time bound in $O^*(n)$ for the MGMP and thus the overall exponent $18/5$ can rely on the block version of the LKS half-gd algorithm (due to Lehmer, Kruth, and Schönhage and essentially covered, e.g., in [3]). The LKS algorithm has been cited in [31] 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 [31]. According to E. Kaltofen [25], this was because the main goal of [31] was practical computation of the determinants whereas the inclusion of the block half-gd algorithm would have ruined any hopes for achieving this goal.

Indeed, the LKS half-gd 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-gd algorithm to make the MGMP computation trivial, in the hardest stage of the implementation of the algorithm in [31]. 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 [31], the MGMP is first computed with the Levinson–Durbin or LKS block algorithms modulo the order of $n!$ 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 $n!$. 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 $n!$ primes in [31]. This simplifies the random prime generation and gives 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 \( n \) random primes. The analysis of the degeneration of computing the MGP 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 34).

The result 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.,[1], 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 without accelerated computation of the MGP. 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 \( \mathbb{Z}_s \) denote the rings of integers and of integers modulo \( q \), respectively. \( \mathbb{R}[\lambda] \) is the ring of polynomials \( \lambda \) with their coefficients from a ring \( \mathbb{R} \); \( \mathbb{R}[\lambda]/\lambda^k \) is the same ring provided all its elements (the polynomials \( \lambda \)) are reduced modulo \( \lambda^k \). \( \mathbb{R}^{k \times k} \) is the algebra of \( k \times k \) matrices with their entries in a ring \( \mathbb{R} \). \( \mathbb{R}^n = \mathbb{R}^{n \times 1} \) is the space of \( n \)-dimensional vectors with their coordinates in the ring \( \mathbb{R} \).

\( D = \text{diag}(d_1, \ldots, d_n) = \text{diag}(d_1^n, \ldots, d_n) \) 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 \( A = T^{-1}DT \) for a diagonal matrix \( D \) and a nonsingular matrix \( T \). \( I_k = \text{diag}(1, \ldots, 1) \) is the \( k \times k \) identity matrix; \( \det A = \text{det}(D) \) and \( \mu(A) = \det(D)^{-1} \). \( \mu(A) \) is the determinant, the adjoint and the characteristic and minimum polynomials of an \( n \times n \) matrix \( A \), respectively. \( M^T \) is the transpose of \( M \) for the null space of matrix \( M \). The vectors \( v \) such that \( Mv = 0 \) form the null space of a matrix \( M \).

ON THEORETICAL AND PRACTICAL ACCELERATION

\[ H = (h_{i,j})_{i,j=1}^{m,n} \] is an \( m \times n \) Hankel matrix if \( h_{i,j} = h_{i,j} \) whenever the entries \( h_{i,j} \) and \( h_{i,j+1} \) are defined. \[ T = (t_{i,j})_{i,j=1}^{m,n} \] is an \( m \times n \) Toeplitz matrix if \( t_{i,j+1} = t_{i,j} \) whenever the entries \( t_{i,j} \) and \( t_{i,j+1} \) 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. \( A = TB^T \)) where \( T, B \), and \( H \) are \( n \times (nL) \) matrices, \( H \) is a Hankel matrix, \( T \) and \( B \) 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 \( g \times g \) 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 \( g \).

2.2. Computational cost of multiplication.

\[ O^*(\kappa(n)) \] denotes the functions in \( (\log(n))^{O(1)} \). \( n \rightarrow \infty \). \( n \rightarrow \infty \). \( \nu(n) \) operations in a ring \( \mathbb{R} \) with unity are sufficient to multiply a pair of polynomials in the ring \( \mathbb{R} \). \( \nu(n) = O((n \log n)^{\log n} \log \log n) = O^*(n) \). \( \beta(\kappa) \) bit operations are sufficient to perform an arithmetic operation in \( \mathbb{Z}_s \) for a positive integer \( s \) such that \( \log s = \epsilon \). \( \beta(\kappa) = O((\log \log n) \log \log n) = O^*(\kappa) \). For the proofs and further details of the above estimates, see [3, 20]. \( \nu = \nu_{A,B} \) arithmetic operations in a ring \( \mathbb{R} \) are sufficient to multiply a fixed matrix \( A \) in \( \mathbb{R}^{n \times n} \) by a vector; \( \nu_A = \infty \). \( \nu \) for any matrix \( A \) in \( \mathbb{R}^{n \times n} \); \( \nu_A \) for sparse and/or structured matrices \( A \).

Theorem 2.2. \((a) \nu_A = O(\nu(\kappa)) \) for an \( n \times n \) Hankel or Toeplitz matrix \( A \). \((b) \nu_B = O(\nu(\kappa)) \) 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 \( (i+j) \)th rows and columns of the matrix \( A \) according to the permutation \( i+j \rightarrow i+j+g, i = 0, \ldots, t-1 \); \( j = 0, \ldots, g-1 \). This turns the matrix \( A \) into a \( t \times t \) block matrix with Hankel and Toeplitz blocks. Now part \( b \) follows from part \( a \) and Theorem 2.1.

2.3. Invariant factors of a matrix.

The greatest common divisor (gcd) \( d_k = d_k(A) \) of all \( k \times k \) minors (subdeterminants) \( d_k(A) \in \mathbb{Z}^{n \times n} \) is called the \( k \)th determinantal
divisor of $f$ for $k = 1, \ldots, n$. We write $s_0 = d_0 = 1$ and define the $k$th 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

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

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

$$| \det A| = s_1 s_2 \cdots s_n.$$  \hfill (2.1)

We also use the crude bound

$$| \det A| \leq |A|^{n}, |A| = n \max_{i,j} |a_{i,j}|,$$  \hfill (2.4)

where $A = (a_{i,j})_{n \times n}$, $|a_{i,j}| = \|a_{i,j}\|_1$ if $a_{i,j}$ are polynomials with the coefficient vectors $a_{i,j}$. 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 $W(\lambda) \in \mathbb{Z}[\lambda]$, the properties (2.1)–(2.3) are extended. For $W(A) = A - I$, we obtain the Frobenius invariant factors $f_i(A) = s_i(A - I) \in \mathbb{Z}[\lambda]$, $i = 1, \ldots, n$, which define the characteristic and minimum polynomials of the matrix $A$,

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

Hereafter $u$ denotes the coefficient vector of a polynomial $u(\lambda)$, and $\delta_j$ 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 $\tau$ and are partitioned into Las Vegas randomized algorithms, which either fail or stop and produce the correct output, and Monte Carlo randomized algorithms, which produce the output with no indication whether it is correct or not. One may decrease the tolerance bound $\tau$ to the level $\tau^*$ 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 running 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 [30, Sec. 18.1] 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 $i$th leading principal (that is, the (northwest)-submatrix of a matrix $M$. A matrix $M$ of rank $r$ has generic rank profile if the matrices $M^{(i)}$ are nonsingular for $i = 1, \ldots, r$. A matrix $M$ is strongly nonsingular if it is nonsingular and has generic rank profile.

Theorem 2.3 [11] (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 $\frac{1}{d^n}$.

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 $\ell$ if we allow by factor of $k$ more elements in the set $S$.

Theorem 2.4 [7, Lemma 10.3]. Let $\pi(n)$ denote the number of primes not exceeding $n$. Let $y \geq 114$. Then $1 < \pi(y) < y < 1.25$, whereas $\pi(y) > y/\log y$ where $y = (1/\beta) \in [n^{1/4}]$. Then we have

$$F = \text{Probability}(\det(M) \mod \rho^k = 0) < \epsilon.$$

Theorem 2.6 [1, Theorem 4.3] (cf. [56]). Let $\ell$ be a set in a subfield $F$ of the field of rational numbers having characteristic zero or greater than 1. Assume that $D$ and $A$ are nonsingular matrices in $F^{n \times n}$. $D$ is a diagonal matrix whose diagonal entries are randomly sampled from the set $S$. Then with a probability of at least $1 - \frac{1}{(2n) \cdot 16^n}$, the matrix $DA$ has $n$ distinct eigenvalues.
Theorem 2.7 [29]. Let \( S \) be a set in a ring \( R \). Assume that \( \lambda, \nu, U \in R^{m \times n}, \) \( \text{rank } A = r, \) and \( U^T \) and \( V \) are unit lower triangular Toeplitz matrices, defined by the \( 2n - 1 \) entries of their first column. Let these entries be randomly sampled from the set \( S \). Then the matrix \( UV \) has generic rank profile with a probability of at least \( 1 - \frac{25}{23} \).

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

The following algorithm from [47] generalizes Hensel's lifting algorithm in [35, 1] by performing it in the ring \( Z_q \), for two integers \( q > 0 \) and \( s > 1 \). The algorithm computes the first \( h \) terms in the \( s \)-adic expansion of the vector \( q^{-1} = q \sum_{i=0}^{s-1} u^{(i)}s^i \) where \( u^{(i)} \in \mathbb{Z}_q^2, i = 0, 1, \ldots, s \). 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 \( b \in \mathbb{Z}^n \), three positive integers \( h, q, \) and \( s \), and a matrix \( Q = (qM)^{-1} \) mod \( (q^s) = q^l \).

**Output:** the vector \( x^{(h)} \in \mathbb{Z}^n \) such that \( x^{(i)} = (qM^{-1}b) \) mod \( (q^s) \), that is, such that \( Mx^{(h)} = b \) mod \( (q^h) \).

**Initialization:** \( x^{(0)} = b \).

**Computation:** for \( i = 0, 1, \ldots, h - 1 \), compute the vectors
\[
x^{(i+1)} = Qx^{(i)} \text{ mod } (q^s), \quad x^{(i+1)} - Mx^{(i)} = (q^s)^{i+1} / (q^s).
\]

Output the vector \( x^{(h)} = x^{(i+1)} / (q^h) \).

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 \( x^{(h)} \) and \( x^{(h)} \) in Algorithm 2.8, we have

(a) \( x^{(i)} \in \mathbb{Z}^n \) for all \( i \);

(b) \( Mx^{(h)} = q^h \) mod \( (q^h) \);

c) all components \( x^{(i)} \) of all vectors \( x^{(i)} = (x^{(i)})^T \) satisfy the bounds
\[
|x^{(i)}_j| \leq |x|_s + \alpha(M), \quad \sum_{i=1}^{s-k} < |x|_s + \alpha(M)/q < \gamma
\]
where \( M = (m_{ij}), b = (b_j)^n \), \( \alpha = \beta(b) = \max |b_j|, \quad a = \alpha(M) = \max |m_{ij}|, \quad \gamma = \gamma(n) \).

3. On Theoretical and Practical Acceleration

3.1. The basic theorem.

Theorem 3.1. Given a matrix \( A = (a_{ij})_{n \times n} \) in \( \mathbb{Z}^{n \times n}, n > 1, \) \( t = \min(t, \tau(n, A)), \) and four positive numbers \( \tau, \sigma, \) and \( w \) such that \( w \geq 114, \)

\[
s > 5n(n+1)/(2\tau),
\]

(c) By definition, all components \( u^{(i)} \) of all vectors \( u^{(i)} \) satisfy \( |u^{(i)}| \leq q^{-1} \), and so \( q^{-1} \leq |u^{(i)}| \leq q^{-1} + \alpha(n, a) \leq q^{-1} + (q^{-1} - 1)n \). 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 \( B = (b_{ij})_{i,j=0}^{t-1} \) with \( t \times t \) blocks in \( \mathbb{Z}_p \), and \( t \) block vector \( V \) with \( t \times t \) blocks in \( \mathbb{Z}_p \). Then the matrix \( B^TV \) can be computed in \( \mathbb{Z}_p \) by using \( O(t^2 n \log p) \) bit operations for \( t = q \) and \( r = \beta \) in Sec. 2.2.

Proof. Interchange the rows and columns of the matrix \( B \) according to the permutation \( i + j - j + i, i = 0, \ldots, t - 1; j = 0, \ldots, t - 1, \) which preserves strong nonsingularity of \( B \) and turns it into a \( t \times t \) block matrix \( B = PBP \) with \( g \times g \) Hankel blocks. Here \( P \) is the matrix of the above permutation, and \( B \) can be immediately represented with its Hankel (resp. Toeplitz) displacement generator of length \( 2t \) (in virtue of Theorem 2.11). By applying the MBA algorithm in \( \mathbb{Z}_p \) [32, 31, 22, 4], (Chapter 5), we obtain a Hankel (resp. Toeplitz) displacement generator of length \( 2t \) for the matrix \( B^T \) by using \( O(t^2 n \log n) \) arithmetic operations in \( \mathbb{Z}_p \). Now Theorem 2.10 immediately follows from Theorem 2.2.(i) since \( B^T = B^{t-1}B^{-1} \).
\( w > 16.5 (n^2/r) \ln (s^n^2 |A|), \)  
\( w/20 \geq \gamma > (10n-3)n/r, \)  
\( w/\ln w > 6.025(n/r) (L + (2n-1) \log(1 - |A|)), \)

one can devise a Las Vegas randomized algorithm which

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

(b) generates a random prime in the range \( (w/20, w] \) and randomly samples \( h \) elements in \( \mathbb{Z}_q \) \( \setminus \{0\}, \) \( 2n \) elements in \( \mathbb{Z}_q', \) and \( 2n \) elements in \( \mathbb{Z}_q'' \), that is, generates \( R = [2h+1] n \log q + 2n \log s + \log w \) random bits overall, and

(c) performs \( B \) bit operations such that \( B \leq \min\{B_0, B_1\}, B_0 = O^* (n^{10/5} L), B_1 = O^* (n^3 L^3). \)

Here \( L = \Omega (|A|), |A| = n \max_{i,j} |a_{i,j}| \) (cf. (3.4)), and \( v_4 \) is the arithmetic cost of multiplying a matrix \( A \) by a vector (cf. Sec. 2.2).

Proof. Theorem 3.1 is supported by the main algorithm in [3] complemented by the acceleration of its stage of computing the MGIP, 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 \( p \) of length \( \log p = O (\log (n L)) \) and apply Gaussian elimination with pivoting to compute \( \det A \) modulo \( p \) (this involves \( O^* (r^2 L) \) bit operations). If \( \det A \mod p \neq 0 \), 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 \( O (\log (n L)) \), 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 \( M x = 0 \). Here \( M = AV \) 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 \( \mathbb{Z}_q \), and deduce from (3.1) and Theorem 2.7 that in both rings \( Z \) and \( \mathbb{Z}_q \) the matrix \( M \) has generic

### 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 \( r < 1 \).

**OUTPUT:** FAILURE with a probability of at most \( \epsilon \) or \( \det A \).

**INITIALIZATION:**

1. Fix three integers \( s, q, \) and \( w \) satisfying (3.1)–(3.4).
2. Fix a value \( t \), \( 1 \leq t \leq n \), 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 \( X \in \mathbb{Z}_{q}^{n \times n} \) and \( Y \in \mathbb{Z}_{q}^{n \times n} \).

4. Randomly sample a prime \( p \) in the range \( (w/20, w) \).

5. Randomly sample \( n \) elements in \( \mathbb{Z}_{q} \setminus \{0\} \) and use them as the diagonal entries of the \( n \times n \) diagonal matrix \( D \). Compute \( d = \det D \). (Note that \( p > w/20 \geq q \), and so \( p \) does not divide \( d \).

**Computation:**

1. Compute the minimum integer \( \eta \geq 0 \) such that \( t \) divides \( \eta + n \). Redefine \( n \rightarrow \eta + n \) and \( A \rightarrow \text{diag}(DA, I_{n}) \). (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 \( B[0] = XAY \) for \( i = 0, \ldots, 2g, g = \frac{n}{t} \). (See Stage 1 of Algorithm A.1 in Appendix A.) Form the matrix \( B[i] = \left[ B[i-1]^k \right]^T \), where \( k = (g+1) \) block Hankel matrix with \( t \times t \) blocks \( B[i] \).

3. Compute \( \hat{d} \equiv (\det B) \mod p \) by applying Gaussian elimination. Use pivoting to avoid divisions by zero in \( \mathbb{Z}_{p} \). If possible, \( \hat{d} \neq 0 \), stop, and output success. Otherwise, output failure. If \( \hat{d} = 0 \), stop and output failure. Otherwise, rank \( B = \mathbb{Z}_{p} \). Apply the algorithm supporting Theorem 2.10 to compute \( \mathbb{Z}^{d}[X]/\lambda^{d} \) the block coefficient vector \( F = \left( F^{(i)} \right)_{i=0}^{d} \) of the MGMP, the unique monic minimum generator matrix polynomial \( F(X) = F_{X}^{A}(\lambda) = \sum_{i=0}^{d} F^{(i)} \lambda^{i} \) for the matrix sequence \( B[0], \ldots, B[2g-1] \). Output failure. (Unlike [30] (c) Algorithm A.1, Stage 3) we require at this stage that \( F(A) \) have degree \( d \) and be monic.

4. Apply \( h \) steps of the Hensel's lifting Algorithm 2.8 for \( q = 1, s = p \), \( h = \log_{2}(2A^{q}) + 1 \) to compute the coefficient vector of the MGMP \( F_{B}(\lambda) \) in \( \mathbb{Z}_{p}^{d} \).

5. Compute modulo \( p^{h} \) the value \( f_{0} = \det F(0) = \det F(0) \) and \( f_{0} = f_{0} \). If \( f_{0} \neq 0 \), output \( (-1)^{d}[f_{0}] \), otherwise output \( (-1)^{d}[f_{0}] \).

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

(b) For some pair of matrices \( X = N \) and \( Y = V \), we have \( f_{i+1-1}(\lambda) = s_{i+1-1}(F) \).

\[ \det F_{W}^{A}(\lambda) = \det F_{X}^{A}(\lambda). \]

Due to the latter equations and to (2.3), we have \( \mu_{A}(\lambda) = f_{n}(\lambda) = s_{n}(I_{A} - A) = s_{n}(F) \). Under the assumption that \( \mu_{A}(\lambda) = c_{A}(\lambda) = \det(AI - A) \), we have \( f_{n-1}(\lambda) = s_{n-1}(F) = 1 \) for \( i > 0 \), \( \det(AI - A) = s_{1}(F) \). Therefore, the output value \( \det \lambda \) is correct if \( \deg \det F(\lambda) = \text{rank} B = n \in \mathbb{Z} \).

Suppose that at Stage 3 of Algorithm 3.2, the output matrix \( B \) is strongly nonsingular in \( \mathbb{Z} _{p}^{d} \). Then Algorithm 3.2 correctly computes the monic MGMP \( F_{B}(\lambda) \). By construction, \( \det F_{B}(\lambda) \) and \( \det F_{B}(\lambda) \) have degree \( n \) in \( \mathbb{Z} _{p}^{d} \) 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 \( r \) of Theorem 31 assuming a nonsingular input matrix \( A \) and the specified random choice of a prime \( p \) and matrices \( X, Y \) and \( E \). We may have failures only at Stage 3, namely, if rank \( B < n \) and/or the MBA algorithm fails. Neither of these may occur if the matrix \( B(A) \) 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 nonmatrices in \( \mathbb{Z}^{n \times n} \) whose entries have been randomly sampled in \( \mathbb{Z}_{q}^{n} \). Then the \( n \times n \) leading principal submatrix \( B(k) = (B^{(i+1)}(j))_{i,j=0}^{n-1} \) of the matrix \( B(k) \), where \( B^{k} = X^{T}A^{T}Y, k = 0, \ldots, 2g-1 \), is strongly nonsingular in \( \mathbb{Z} \) with a probability of at least \( 1 - \left( \frac{4+n}{4} \right)^{n} \).

**Proof.** Let \( \mathbf{u}_{i}^{T} \) be generic vectors \( X \) and \( Y \). Let \( \mathbf{D} = S^{-1} AS \) be a diagonal matrix; \( X^{T} = S^{T}X, Y \) and \( Y \). (The \( \mathbf{D} \) has distinct diagonal entries, which are the entries shared by \( A \) and \( \mathbf{D} \).) We have \( X^{T}A^{T}Y = X^{T}D^{T}Y \) for \( i = 0, \ldots, g-1 \). Therefore it is sufficient to prove the theorem for \( A = \mathbf{D} \).
The factorization \( B^{(k)} = (A^T A)^{-1} (A^T Y) A^{k-1} \) implies that \( \det B^{(k)} = n \) if and only if \( \det (A^T A)^{-1} (A^T Y) A^{k-1} = n \). Now, let \( B^{(k)} = \frac{A^T Y}{A^{k-1}} \) (with distinct diagonal entries \( d_0, \ldots, d_{n-1} \)), the choice of \( X = (x_{ij})_{i,j=0}^{n-1} \) with \( y_{0j} = x_{0j} = \frac{d_j}{d_0} \), \( y_{ij} = d^{i-j}_0 \), turns the matrices \( (X^T A)^{k-1} \) and \( (A^T Y) A^{k-1} \) into Vandermonde matrices (up to their column permutation). It implies nonsingularity of the matrices \( B^{(k)} \) for \( k = 1, \ldots, n \).

Thus, the matrix \( B^{(n)} \) is strongly nonsingular for generic \( X \) and \( Y \), that is, \( \det B^{(n)} \neq 0 \), \( i = 1, \ldots, n \). Recall that \( B^{(k)} \) 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 probability of at least \( 1 - \frac{(n+1)n}{2} \), neither of the polynomials \( B^{(k)} \) vanishes in virtue of Theorem 2.3.

In virtue of bound (3.3) and Theorem 1.4, 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 \( \epsilon \). Otherwise, the cause of the failure is restricted to the case where the matrix \( (DA) \) mod \( p \) has multiple or zero eigenvalues. Suppose \( \epsilon \) 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 \( (DA) \) has multiple eigenvalues. Finally, in virtue of Theorem 2.4, a nonzero integer vanishes in the transition from \( Z \) to \( Z_p \) with a probability of \( \pi(w) - \pi(w/20) > w/\beta \ln w \), \( \beta = 1.304930 \ldots \) for our choice of 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 \( \det(XI - A) \). (2.4) implies that \( |d_A| \leq |1 + |A||^{2(1-1)n} \); \( |d_A| \leq |A|^{n(1+|A|)^{2(1-1)n}} \). Therefore, the integer \( d_A \) of \( A \) mod \( p \) can be divided by less than \( (2n) - (2n-1)n \) distinct primes. Combining this estimate with (3.4) and Theorem 2.4 implies that \( (\det A) \) vanishes in virtue of (3.4) and Theorem 2.4 for our choice of \( p \). Summarying, we obtain the overall bound \( \tau \) on the failure probability for Algorithm 3.2, thus completing our proof of its correctness.

**Remark 3.1.** We apply the MRA divide-and-conquer algorithm in [32, 34, 4] at Stage 3 of Algorithm 3.2 to compute the block coefficient vector \( F = (F^{(k)})_{k=0}^{n-1} \) of an MGP \( F(A) = \sum_{k=0}^{n-1} F^{(k)} A^k \). Alternatively, see also Sec. 4) we can compute the block vector \( F \) by applying to the block matrix \( H^{(k)} \) 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

### 3.5. Bounding the Bit Operation Cost

Finally, let us estimate the overallbit operation cost including also the case where at Stage 3 we replace the MRA algorithm by the alternative algorithm in Remark 3.1. To yield the bound \( \eta_0 \) in Theorem 3.1, we use the estimates in [30] at Stages 1 and 2 (cf. Stage 1 in Algorithm A1). Clearly, the bit complexity at these stages is dominated by \( O(n^3 \ln n) \); arithmetic operations with integers of the length \( O(r \ln p) \) are required for computing the power \( A^r \) and \( O(n^3 \ln r) \) arithmetic operations with integers of the length \( O(g) \) bits required for computing the Jordan sequence \( A^{g/p}, j = 1, \ldots, g/p \). This gives us the overall bound of \( O^{*}(r + g/r) \) 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 MRA algorithm to the matrix \( PB^{(n)} P \); this gives us the MGP \( F \) in \( O^{*}(g^2) = O(n^{*} n^2) \) operations in \( \mathbb{Z}_p \), that is, \( O(\eta_0 n^2) \) bit operations. This bound is dominated by the cost bound at the lifting Stage 4. We need \( h = O(n^l) \) lifting steps, each requiring \( O^*(n^{l^2}) \) arithmetic operations with \( O(\log p) \) bit integers (see Theorem 2.10), that is, \( O^{*}(n^{l^2}) \) bit operations. This means \( O^*(n^{l^2} n) \) bit operations at Stage 3 for all lifting steps. Clearly, this bound, together with the bound in \( O^*(r + g/r + n l) \) at Stage 2 (which holds for any choice of positive integers \( r \) and \( l \) such that \( r \leq g = n l \)), dominates the bit cost at Stage 3 and, therefore, the overall bit cost estimate. The choice \( \eta_1 = \eta_0 \) implies the overall bit cost bound \( B_0 \). To yield the bound \( B_1 \), we choose \( r = 1, t = \lceil \sqrt{\eta_0} \rceil \), and trivialize Stage 2 of Algorithm 42 as follows: write \( X_0 = X \) and re-
cursively compute the matrices $Y^0 = X^T Y^T$, $X_{i+1}^T A$ for $i = 0, 1, \ldots, 2g$, $g = n/t$. This supports the bit cost bound $O((1 + v_A) t^2 L)$ at Stage 2. Write $t = [n^{2/3}]$ and obtain the overall bit cost bound $L_1$. This completes the proof of Theorem 3.1. 

3.6. Saving word operations 

Assume the realistic model where the length $\lambda$ 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 $\xi$ is a word operation if $\xi \leq \lambda$ 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 $W \leq mn\{W_0, W_1\}$, $W_0 = O(n^{3/5} L/\lambda)$, if $\lambda = O(n^{1/10} L)$, $W_1 = O(n^{1/2} t^{1/2} L)$ if $\lambda = O(n^{1/4} L)$, and $\lambda$ 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 $\lambda$ versus the bit complexity under the assumptions $\lambda = O(n^{1/10} L)$ or $\lambda = O(n^{1/4} L)$, respectively. The MFA algorithm uses $O(t^2 n)$ arithmetic operations which are also operations at Stage 2. Stage 4 requires only $O(t^2 n^{1/2})$ word operations if we perform Stage 3 modulo $p^s$ for $v = [n/\log p]$ and then apply the generalized lifting Algorithm 2.8 for $g = 1, s = p^s$ at Stage 4. The degeneration modulo $p^s$ 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 $\lambda = O(n^{1/4} L)$.

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 technique, since the application of these techniques would destroy the structure and sparsity). The papers also follow the approach in [1, Section 4] to decrease the cost of the evaluation 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 algorithm in [17] and [18] rely essentially on the algorithms in [30], on extending $t$ to the multivariate input in [4], 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]. Storpfinn in [31] 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 $O(n M M(n) L)$, where $M M(n)$ is the arithmetic complexity of $n \times n$ matrix multiplication, $M M(n) = O(n^{2.376})$. The techniques in [54] and the algorithm in [51] combined with the one in [30] enable the Monte Carlo extension of the bit cost bound $O(n^{2/3} L)$ 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 characteristic polynomials $q_A(t)$ and $p_A(t)$. The algebraic and geometric multiplicities of the eigenvalues are given by their multiplicities as the roots of $q_A(t)$ and $p_A(t)$, respectively. The record (and optimal up to polylog factors) bit complexity bounds at the root-finding stage have been achieved in [30]. 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 $b \geq b(n, |A|)$ by using $O(b)$ random bits and $O(b + \mu(n, \log b + \log |A| + b) n + b)$ bit operations for $b = O(\log \log n)$ and $b \geq 3$ and $b \geq 3$ 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 similarities to [30] and [16], but the practical value of the algorithm 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 structured input. Like the baby
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 \( H^{(r)} = (b^{(r)})_{i,j=0}^{n-1} \) where the matrix \( H = (b^{(r)})_{i,j=0}^{n-1} \) has rank \( r \) and the matrix \( H^{(r)} \) is nonsingular [28, 7]. Wedemann proposed this approach for both computing \( \det A \) and solving linear systems \( Ax = b \). Coppersmith in [9] proposed an acceleration for the linear systems \( y \) using block vectors (that is, matrices \( X \) and \( Y \)). Kaltofen and Villard in [30] extended this acceleration to computing \( \det A \) and improved 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 MCM defining the sequence of block scalars \( XTAY \), to \( \det A \). Kaltofen and Villard compare the Smith invariant factors for the matrices \( F(\lambda) \) and \( I - \lambda A \) and show that, up to scaling, they are likely to be identical for random \( A \) and \( Y \), and therefore, \( \Delta(\lambda) = \det F(\lambda) = \det(\lambda I - A) \), \( \det F(0) = (-1)^{r} \det A \) when \( 0 \in \Delta(\lambda) \) is the leading coefficient of \( \Delta(\lambda) \). Let \( F(\lambda) \).

Here is the resulting algorithm where the matrix \( A \) is nonsingular.

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

**Input:** preconditions \( x \in \mathbb{Z}^{n} \times m \), positive integers \( m \), \( n \), \( s \), \( m \leq n \), \( r = \lceil (2n/m + 3)/s \rceil \). (Preconditioning is by Wiedemann's technique or its extensions)

**Output:** "failure" with a low probability or \( \det A \).

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

**Computation:**
1. Compute \( H^{(i)} = X^{T}AY \), \( i = 0, 1, \ldots, \lceil (2n/m + 3)/s \rceil \).
2. Compute the minimal matrix generating polynomial \( F^{A} \) for the sequence \( \{H^{(i)}\}_{i \geq 0} \).
3. Compute the leading and constant coefficients of \( \Delta(\lambda) = \det F^{A}(\lambda) \).

If \( \deg(\Delta) < n \) and \( \Delta(0) \neq 0 \) then return "failure"; otherwise return \( \det A = \Delta(1)/\text{(leading coefficient of } \Delta(\lambda)) \). (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 \( i = 1, 2, \ldots, r - 1 \), do \( V[i] \leftarrow \beta V[i] \);
1.2 \( Z \leftarrow A^r \);
1.3 For \( k = 0, 1, \ldots, s \), do \( (U[k])^T \leftarrow X^T Z[k] \);
1.4 For \( i = 0, 1, \ldots, r - 1 \), do
   \[ B[i] = B[i] - (U[i])^T V[i] \]

The computation at Stages 2 and 3 are performed modulo sufficiently many random primes and the output 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 or 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 \( f_i(A) \) and \( f_{i,p}(A) \) (we call such primes “bad”). Then \( P \leq \lceil n \| A \| \rceil \).

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

\[
P = \frac{\lfloor 20k^2 \| A \| \rfloor}{14k \| A \|}.
\]

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

**Proof.** By Theorem 2.4, there are at least
\[
20x \geq 1.25x > \frac{17x}{\beta \log(2x)} > \frac{17x}{\beta \log(50x)} > \frac{17x}{\log(50x)}
\]
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.

**ON THEORETICAL AND PRACTICAL ACCELERATION**

Given that \( A = (a_{ij}), \| a_{ij} \| = \| A \| \) if \( a_{ij} \) are polynomials with the coefficient vectors \( a_{ij} \), we have

\[
r_i = s_i/s_{i-1}, \quad r_i^{(q)} = r \mod q, \quad i = 1, \ldots, s \tag{C.1}
\]

and

\[
|\det A| \leq |A|^n, \quad \| A \| = \| A \| = \max \sum_i |a_{ij}|. \tag{C.2}
\]

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

\[
s_i = \prod_{j=1}^{i} r_j, \quad |\det A| = \prod_{j=1}^{n} r_j^{x_i-1} \leq |A|^n, \tag{C.3}
\]

\[
r_g \leq r_g^{n/(n-g+1)}, \quad g = 1, \ldots, n. \tag{C.4}
\]

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

**Lemma C.1.** Let \( r_i \leq r_i, i = 1, \ldots, l; a_i \leq a_i \) for \( i \leq l-1, \ldots, n \). Then

\[
r_i \leq \beta c_i, \quad c_i = \left( \frac{1}{|A|^n} \right)^{\left( \frac{1}{\beta} \right)^{l-i}} \tag{C.5}
\]

Write \( r_i(\lambda) = f_{n-i+1}(\lambda)/f_{n-i}(\lambda), \delta_i = \delta(f_i) \).

**Lemma C.2.** \( |f_j| \leq (1 + |A|)|^{\delta_j}, |r_i| \leq (1 + |A|)^{\delta_i} \) for all \( j \).

**Proof.** Each \( f_i(\lambda) \) is a polynomial \( \prod_{i} (\lambda - \lambda_i) \) where \( \lambda_i \) are the eigenvalues of \( A \), so \( |\lambda_i| \leq |A| \) for all \( i \) □

**Lemma C.3.** \( \delta_{n-k+1} \leq n/k, k = 1, \ldots, n-1 \)

**Proof.** We have \( \sum_{i=0}^{k} \delta_{n-i} = n, \delta_{n-k+1} \leq \delta_{n-i}, i \geq 1, \) so \( \delta_{n-k+1} \leq n - \sum_{i=0}^{k} \delta_{n-i} \leq (k+1)\delta_{n-k+1} + 1, \delta_{n-k+1} \leq n/k \) □

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

**Acknowledgments.** I am grateful to the referees for their helpful comments.
REFERENCES


On Theoretical and Practical Acceleration

Department of Mathematics and Computer Science
Lehman College of CUNY, USA
http://comet.lehman.cuny.edu/vfan/
E-mail: victor.fan@lehman.cuny.edu