2002

TR-2002001: Can We Optimize Toeplitz/Hankel Computations?

V. Y. Pan

Can We Optimize Toeplitz/Hankel Computations ?

V. Y. Pan *
Department of Mathematics and Computer Science
Lehman College of CUNY, Bronx, NY 10468, USA
vpan@lehman.cuny.edu

May 25, 2002

Abstract

The classical and intensively studied problem of solving a Toeplitz/Hankel linear system of equations is omnipresent in computations in sciences, engineering and communication. Its equivalent formulations include computing polynomial gcd and lcm, Padé approximation, and Berlekamp-Massey's problem of recovering the linear recurrence coefficients. To improve the current fastest divide-and-conquer algorithm by Morf 1974/1980 and Bitmead and Anderson 1980, we rely on Hensel's p-adic lifting. We accelerate its recovery stage by exploiting randomization and the correlation between lifting and the computation of Smith's invariant factors of the input matrix. Furthermore, for the average input, the 2-adic version of lifting is sufficient, allowing computations in binary form. Our resulting algorithms solve a nonsingular Toeplitz/Hankel linear system of n equations by using $O(m(n)n\mu(\log\ n))$ bit operations (versus the information lower bound of the order of $n^2\log\ n$), where $m(n)$ and $\mu(d)$ bound the arithmetic and Boolean cost of multiplying polynomials of degree n and integers modulo 2^d+1, respectively, and where the input coefficients are in $n^{O(1)}$. Our algorithms can be extended to solving nonsingular and consistent singular Toeplitz/Hankel-like linear systems of equations and

*Supported by NSF Grant CCR 9732206 and PSC CUNY Award 66383-0032

computing the resultant, gcd and lcm of two polynomials of bounded degrees as well as a fixed entry of a Padé approximation table and the linear recurrence coefficients provided that the input values are some bounded integers.

Key words: Toeplitz matrices, Hankel matrices, linear systems of equations, polynomial gcd, Padé approximation, Berlekamp-Massey problem, Hensel's p-adic lifting, rational number reconstruction, Smith invariant factors, randomized algorithms, bit complexity.

1 Introduction

Toeplitz and Hankel matrices and, more generally, matrices with the structure of Toeplitz/Hankel type are omnipresent in computations in sciences, engineering and communication. Solution of Toeplitz/Hankel-like linear systems of equations is required in the shift register synthesis and linear recurrence computation, inverse scattering, adaptive filtering, modelling of stationary and nonstationary processes, numerical computations for Markov chains, solution of PDE's and integral equations, polynomial rootfinding and many other fundamental problems in computer algebra such as computing resultants, Padé approximation, polynomial gcds and lcms (see more items and further bibliography in [KS99],[P00, Section 1.1] and [P01]). Furthermore, the displacement transformation approach of [P90] enables reduction of computations with matrices having structures of Cauchy, Vandermonde and other types to the Toeplitz/Hankel-like case.

Matrix structure can be intensively exploited in devising the solution algorithms to decrease the solution cost dramatically, from the order of n^3 flops in Gaussian elimination for a nonsingular Toeplitz/Hankel system of n equations $M\mathbf{x} = \mathbf{b}$ to $O(n^2)$ in the "fast" algorithm by Levinson 1947 and Durbin 1959, and further to $O(n \log^2 n)$ in the "superfast" divide-and-conquer MBA algorithm by Morf 1974/1980 and Bitmead/Anderson 1980 (cf. [P01]).

The more realistic measure, which we adopt, is the bit operation cost. To each arithmetic operation performed over the integers modulo q, that is, with d-bit precision, for $d = \lceil \log_2 q \rceil$, we asign the cost of $\mu(d)$ bit operations (hereafter log stands for $\log_2$ unless specified otherwise), where

$$\mu(d) \le C_{class} d^2, \mu(d) \le C_k d^{\log\ 3}, \mu(d) \le (C_{ss} d\ \log\ d) \log\ \log\ d, \qquad (1.1)$$

$\log\ 3 = 1.58496\ldots, 0 < C_{class} < C_k < C_{ss}$, and the above bounds are supported by the classical, Karutsuba's, and Schönhage-Strassen's algorithms, respectively [GG99].

The most popular way of controlling the precision is by applying the CRA (Chinese remainder algorithm). The input is integral (or made integral by scaling), and the computations are performed modulo distinct random primes $p_1, \ldots, p_s$ such that a nonsingular matrix M is very likely to remain nonsingular modulo $p_1, \ldots, p_s$. The output is recovered first modulo $p = p_1 \cdots p_s$ by using the CRA, and then in rational form based on the rational number reconstruction algorithms [GG99], [PW02], provided that the product $p_1 \cdots p_s$ exceeds $2\delta|\nu|$ for every rational output value $\nu/\delta, \delta \ge 1$. This property enables recovery of each value ν/δ from $(\nu/\delta) \mod p_i, i = 1, \ldots, s$. The latter stage of rational number reconstruction is generally considered quite hard but not for the MBA algorithm, which computes $\det(M)$ as by-product. The scaled output vector $\det(M)\mathbf{x}$ is an integer vector, and its reconstruction from $(\det(M)\mathbf{x}) \mod p$ is cost-free.

To specify the bit cost bound for the MBA, let $m(n)$ be the arithmetic cost of multiplying two polynomials of degree $n-1$,

$$2n - 1 \le m(n) \le (c_* n \log\ n) \log\log\ n, \qquad (1.2)$$

for a constant c_* [CK91]. In the introduction, we also assume for simplicity that all input values lie in the range $(-q, q)$ for q in $n^{O(1)}$. (Later on, we relax this assumption.) Then the MBA algorithm computes $\mathbf{x} = M^{-1}\mathbf{b}$ by using $O(n\mu(\log\ n)m(n)\log\ n)$ bit operations. Any further progress was supposed to be hard because we deal with a central problem of structured matrix computations open and intensively studied since 1980. Moreover (see [P01]), the solution of Toeplitz/Hankel linear system is equivalent to the computation of polynomial gcd/lcm and a fixed entry of Padé table and is closely related to computing the resultant of a univariate polynomial; these are even older problems, central and most intensively studied in computer algebra [GG99]. Berlekamp-Massey's problem of the recovery of the coefficients of a linear recurrence is another celebrated and intensively studied equivalent formulation of the same problem [BGY80]. Thus our new progress should be viewed as surprising.

Our first step is to rely on the distinct approach, based on p-adic Hensel's lifting. To its practical advantage versus the MBA algorithm, only a single random prime p in $n^{O(1)}$ is sufficient, and all lifting computations are with two matrices of the same size $n \times n$. Another advantage is that the bit cost of lifting decreases to $O(nm(n)\mu(\log\ n))$, thus approaching closer the lower bound of the order of $n^2 \log\ n$. This many bits are generally required already to represent the n output values, each with up to $n \log\ n$ bits.

The technical problems arise, however, at the stage of rational number reconstruction. $\det(M)$ is not available as by-product anymore, and the known algorithms require the order of $n^3 \log^2 n$ bit operations at this stage. Substantial help came very recently from [PW02] where the extended Euclidean algorithm for integers was improved dramatically. This immediately implied acceleration of our rational number reconstruction to $O(n\mu(n \log\ n) \log\ n)$, which matches but still does not beat the MBA cost bound.

In this paper we propose two new randomization techniques. Both of them exploit and extend the approach of [P87],[P88],[ABM99],[EGV00], which relates p-adic lifting to the computation of Smith's invariant factors of M. These techniques decrease the bit cost of the recovery of the output from its modular value to or below the level of the lifting cost. Our first technique includes the known trick of probabilistically computing the lcm of several integers $q_1, \dots, q_k$ as the denominator of the random linear combination of the reciprocals $1/q_i$ [P92],[BP94],[CFG99], but exploits it in a new context with support from Smith's leading factor s_n. With this technique we decrease the recovery cost to $O(\mu(n \log\ n) \log\ n)$; the output is represented as a pair $\mathbf{y}, s_n$, where the components of the vector $\mathbf{y} = s_n\mathbf{x}$ are output as p-adic numbers. For practical purpose, however, the binary representation of all output values is desired.

Our second technique (actulally a combination of three techniques) handles the problem of conversion to the binary base in a novel way, including a new binary version of Hensel's lifting (where the basic prime can be 2 even if $\det(M)$ is even) combined with perturbation of the input matrix by small rank random matrices and the variable diagonal or modular continuation techniques. So, we arrive at a practical solution to this practical problem by computing the output within the desired cost bound for an average Toeplitz/Hankel-like matrix M. We cannot trace this solution and its techniques back to any previous works, except for the variable diagonal techniques we proposed in 1985 (see the details and the bibliography in [P00]).

Our algorithms promise to be practical, and the techniques are also of

independent theoretical interest: they enable us to compute the determinant and all Smith's factors of a general or structured average matrix M at the same randomized asymptotic bit cost as we achieve for solving linear systems.

For simplicity, we specify our algorithms and complexity estimates for Toeplitz matrices, but the extension to the Toeplitz/Hankel-like case is straightforward. Furthermore, our algorithms can be extended to solving a consistent but singular general or Toeplitz/Hankel-like linear system $M\mathbf{x} = \mathbf{b}$ and computing a vector from (or a basis for) the null space of a singular general or Toeplitz/Hankel-like matrix M. The latter extensions are straightforward as soon as a nonsingular submatrix of M of the maximum size is computed, and we compute such a submatrix probabilistically by applying the MBA algorithm modulo a single random prime p in $n^{O(1)}$ to a randomly preconditioned input matrix M [P01]. The arithmetic cost of the MBA algorithm is $O(n \log^2 n)$, so the bit cost is small as long as the algorithm is performed modulo p, that is, with the precision of $O(\log\ n)$ bits. In our next paper we specify the MBA processes and detail the estimates for the error/failure probability due to the randomization as well as the resulting record randomized bit complexity bounds for singular Toeplitz/Hankel-like computations. The solution of consistent Toeplitz linear systems actually covers the solution of the equivalent problems of computing the gcd and lcm of polynomials as well as a fixed entry of Padé approximation table and recovering the linear recurrence coefficients from a sequence of the recurrence terms (Berlekamp–Massey's problem), whereas the computation of the determinant of a Toeplitz-like matrix covers the computation of the univariate resultant [P01].

We organize our paper as follows. After definitions and preliminary results in the next section, we recall and then modify Hensel's lifting algorithm for a linear system of equations in Sections 3–5. In Section 6 we apply the variable diagonal and modular continuation techniques to initialize lifting.

We conclude this section with some comments on possibility of further acceleration. The factor of $m(n)$ in our estimates comes from our basic operation of Toeplitz/Hankel matrix-by-vector multiplication or equivalently polynomial multiplication. It is highly unlikely that any efficient algebraic computation scheme for our tasks could dispense with this operation. (Try to imagine such a scheme, e.g., for polynomial gcd.) This informal argument suggests that improvement of our bounds by the factor $m(n)/n$ is unlikely. On the other hand, our basic operation can be viewed as multiplication of polynomials with bounded integer coefficients, so the binary segmentation

technique of Fischer-Paterson 1974 (cf. [BP94, Section 3.9]) could yield theoretical acceleration by the factor of $(\log\log\ n)\log\log\ n$. The resulting bit cost bound of $O(n\mu(n\log\ n))$, however, does not seem to be practically attractive unless n is huge because the overhead constant C_{ss} is large, whereas with C_{class} and C_k in (1.1) the overall bit cost bounds become n^α for $\alpha > 2.5$.

2 Definitions and basic facts

2.1 Integers, rationals, matrices

Definition 2.1. $\mathbf{Z}$ *is the ring of integers,* $\mathbf{Z}_q$ *is the ring of integers modulo* q, $\mathbf{Q}$ *and* $\mathbf{R}$ *are the fields of rational and real numbers, respectively. For* $z, q \in \mathbf{Z}, q > 1$, *we define* $z \mod q$ *as a unique number* z_q *such that* q *divides* $z - z_q$ *and* $-q/2 \le z_q < q/2$. *(Clearly,* $z = z_q$ *if* $-|z|/2 \le z_q < |z|/2$.*)* $\nu(y)$ *denotes the numerator, and* $\delta(y)$ *denotes the denominator in the ratio* $y = \nu(y)/\delta(y)$ *of two coprimes* $\nu(y)$ *and* $\delta(y)$. $M = (m_{i,j})_{i,j=0}^{k-1,l-1}$ *is a* $k \times l$ *matrix with rational or integer entries* $m_{i,j}$; $M \in \mathbf{Q}^{k\times l}$ *or* $M \in \mathbf{Z}^{k\times l}$, *respectively.*

Definition 2.2. I *and* 0 *are the identity and null matrices of proper sizes,* I_l *is the* $l \times l$ *matrix* I. $\det(M)$ *and* $\mathrm{adj}(M) = ((-1)^{i+j}d_{i,j})_{i,j=0}^{k-1,k-1}$ *denote the determinant and adjoint (adjugate) of* $k \times k$ *matrix* $M = (m_{i,j})_{i,j=0}^{k-1,k-1}$, *where* $d_{i,j}$ *is the determinant of the submatrix* $M_{i,j}$, *obtained by deleting the* i*-th row and* j*-th column of* M. M^T *is the transpose of* M.

Definition 2.3. $|M|$ *denotes the column norm of* M, $|M| = ||M||_1 = \max_j \sum_i |m_{i,j}|$ *for* $M = (m_{i,j})$. $|\mathbf{v}|$ *denotes the* ℓ_1*-norm* $\sum_i |v_i|$ *of a vector* $\mathbf{v} = (v_i)_i$.

The next well known estimate is an overestimate on the average, according to [ABM99].

Fact 2.4. $|\det(M)| \le |M|^k$ *and* $|\mathrm{adj}(M)| \le k|M|^{k-1}$ *for a* $k \times k$ *matrix* M.

Definition 2.5. $v_S \le 2n^2 - n$ *and* i_S *arithmetic operations are sufficient to multiply a given* $n \times n$ *matrix* S *by a vector and to invert it, respectively.*

Definition 2.6. *The* k*-th determinantal divisor of* M, *for* $k = 1, \dots, n$ *is the greatest common divisor (gcd)* $d_k = d_k(M)$ *of all* $k \times k$ *minors (subdeterminants) of a matrix* $M \in \mathbf{Z}^{n\times n}$. *We write* $s_0 = d_0 = 1$ *and define the* k*-th* Smith invariant factor *of* M *as* $s_k = s_k(M) = d_k/d_{k-1}$ *for* $k = 1, \dots, n$.

It is easily deduced that $s_1, \dots, s_n \in \mathbf{Z}$ and $|\det(M)| = s_1 \cdots s_n$, so (cf. Fact 2.4)

$$s_n \le |\det(M)| \le |M|^n. \tag{2.1}$$

2.2 The bit-complexity of rational number reconstruction

Hereafter, $\rho(q)$ denotes the bit-operation cost of recovering a rational number x/y from three integers k, q, and $r = (x/y) \mod q$ provided q and y are coprime, x and y are coprime, k is an integer, $1 \le k \le q$, $|x| < k$, and $0 < y \le q/k$. (See [GG99] on conditions of existence of the number x/y.) If in addition $2|x| < k$, then the pair (x, y) is unique [GG99]. Clearly, $x = 1, y = r, \rho(q) = 0$ if $q > 2r$.

Likewise, $\bar{\rho}(\delta)$ denotes the bit operation cost of *rational roundoff*, that is, the recovery of a unique rational number x/y from its approximation ν/δ and a positive integer k, provided that $1 \le y \le k, |x| < y, x$ and y are coprime, and $|x/y - \nu/\delta| < 1/(2k^2)$ for fixed ν, δ and k.

Both of the recovery problems can be solved by applying the extended Euclidean algorithm to the input pair r_0, r_1 being q, r or ν, δ, respectively, and by stopping for the smallest positive i such that $r_i < k$ in the computed remainder sequence, $r_0, r_1, r_2, \dots$. In both cases the desired output x/y is recovered immediately from the triple r_i, s_i, t_i (that is, from r_i and t_i, where q and $r = (x/y) \mod q$ are given, or from s_i and t_i, for the rational roundoff) [GG99],[Z93].

Due to the recent acceleration of the extended Euclidean algorithm in [PW02], we obtain the following estimates.

Fact 2.7. *[PW02]. $\rho(q) = O(\mu(d) \log\ d)$ and $\bar{\rho}(\delta) = O(\mu(d) \log\ d)$, for $\mu(d)$ in (1.1) and $d = \log\ q$ or $d = \log\ \delta$, respectively, if there exists a pair (x, y) satisfying the requirements of the recovery problem.*

2.3 Toeplitz and Hankel matrices

Definition 2.8. *A matrix $T = (t_{i,j})$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$ for every pair of its entries $t_{i,j}$ and $t_{i+1,j+1}$. $Z(\boldsymbol{v})$ is the lower triangular Toeplitz matrix defined by its first column $\mathbf{v}$. $H = (h_{i,j})$ is a Hankel matrix if $h_{i,j} = h_{i-1,j+1}$ for every pair of its entries $h_{i,j}$ and $h_{i-1,j+1}$. The unit Hankel*

(reflection) matrix $J = (j_{g,h})$, $j_{g,n-1-g} = 1$, *for* $g = 0, \dots, n-1$, $j_{g,h} = 0$ *for* $h+g \neq n-1$, *reverses any vector* $\boldsymbol{v} = (v_i)_{i=0}^{n-1}$, *that is,* $J\boldsymbol{v} = (v_{n-i-1})_{i=0}^{n-1}$, $J^2 = I$.

Note that for any Toeplitz matrix T, there exist nonunique pairs $(Z(\mathbf{w}), Z(\mathbf{x}))$ such that $T = Z(\mathbf{w}) + Z^T(\mathbf{x})$. Furthermore, TJ and JT are Hankel matrices if T is a Toeplitz matrix, and HJ and JH are Toeplitz matrices if H is a Hankel matrix. Therefore, the problems of solving Toeplitz and Hankel linear systems of equations are immediately reduced to each other. We only specify Toeplitz case.

The next well known results (see, e.g., [P01], Chapter 2) cover multiplication of a Toeplitz matrix and its inverse by a vector. In particular, Theorem 2.9 extends the Gohberg-Semencul formula of 1972, Theorem 2.10 is easy to verify.

Theorem 2.9. *Let* $T = (t_{i-j})_{i,j=0}^{n-1}$ *be an* $n \times n$ *nonsingular Toeplitz matrix, let* t_{-n} *be any scalar (e.g.,* $t_{-n} = 0$*), and write* $p_n = -1, \mathbf{t} = (t_{i-n})_{i=0}^{n-1}, \mathbf{p} = (p_i)_{i=0}^{n-1} = T^{-1}\mathbf{t}, \mathbf{q} = (p_{n-i})_{i=0}^{n-1}, \mathbf{v} = T^{-1}(1, 0, \dots, 0)^T, \mathbf{u} = ZJ\mathbf{v}$. *Then* $T^{-1} = Z(\mathbf{p})Z^T(\mathbf{u}) - Z(\mathbf{v})Z^T(\mathbf{q})$.

Hereafter the pair of the above vectors $\mathbf{p} = \mathbf{p}(t_{-n})$ (for a fixed t_{-n}) and $\mathbf{v}$ is called a *generator* for T^{-1}.

Theorem 2.10. *Given an* $m \times n$ *Toeplitz matrix* T, *its multiplication by a vector is a subproblem of multiplication of two polynomials of degrees* $m+n-2$ *and* $n-1$, *whose coefficients are given by the entries of the input matrix and vector, respectively. If* T *is triangular and* $m = n$, *then both of these polynomials have degrees of at most* $n-1$.

Corollary 2.11. *An* $n \times n$ *Toeplitz matrix* T *can be multiplied by a vector in* $2m(n)$ *arithmetic operations for* $m(n)$ *in (1.2); the bound decreases to* $m(n)$ *if* T *is a triangular matrix.* $4m(n) + n$ *arithmetic operations suffice to multiply* T^{-1} *by a vector provided that* T *is nonsingular and is given with its generator, that is, with the vectors* $\mathbf{p}$ *and* $\mathbf{v}$ *in Theorem 2.9.*

3 Hensel's Lifting for General and Toeplitz/Hankel-like Linear Systems

Algorithm 3.1. *Hensel's lifting for a linear system [MC79],[D82].*

INPUT: $M \in \mathbf{Z}^{n\times n}$, an integer p coprime with $\det(M), \mathbf{b} \in \mathbf{Z}^n$, an integer $h > 1$, and $Q = M^{-1} \mod p$.

OUTPUT: $\mathbf{x}^{(h)} = M^{-1}\mathbf{b} \mod p^h$.

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

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

$$\mathbf{u}^{(i)} = Q\mathbf{r}^{(i)} \mod p, \mathbf{r}^{(i+1)} = (\mathbf{r}^{(i)} - M\mathbf{u}^{(i)})/p.$$

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

Theorem 3.2. *[D82].*

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

b) $M\sum_{i=0}^{j-1} \mathbf{u}^{(i)}p^i = \mathbf{b} \mod p^j, j = 1, 2, \ldots, h$*;*

c) $\mathbf{r}^{(i)} = (r_j^{(i)})_{j=0}^{n-1}, |r_j^{(i)}| \le n\gamma p/(2p-2)$ *for all* i *and* j *if* $M = (m_{ij})_{i,j}, \mathbf{b} = (b_j)_j$ *for all* i *and* j*, and* $\gamma = \max_{i,j}\max\{p, |m_{i,j}|, |b_j|\}$.

Corollary 3.3. *Algorithm 3.1 uses* $O((v_M + v_Q)h\mu(\log(n\gamma)))$ *bit operations for* $\mu(d)$ *in (1.1) and* v_S *of Definition 2.5, to output* $\mathbf{x}^{(h)}$ *in p-adic form.*

Proof. Combine Theorem 3.2 and Corollary 2.11.

□

Let us next specify the integer parameters h and p.

Theorem 3.4. *It is sufficient to choose* $h = \lceil 2n\log_p(\gamma n)\rceil$ *in Algorithm 3.1 and to perform* $O(n\mu(h\log p)\log(h\log p))$ *bit operations to recover a unique solution* $\mathbf{x} = M^{-1}\mathbf{b}$ *to the linear system* $M\mathbf{x} = \mathbf{b}$ *from the vector*

$$\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} \mathbf{u}^{(i)}p^i = \mathbf{x} \mod p^h.$$

Proof. According to Section 2.2, we may uniquely recover the pair of coprimes $\nu_j = \nu(x_j)$ and $\delta_j = \delta(x_j)$ for a rational component $x_j = \nu_j/\delta_j$ of the vector $\mathbf{x} = M^{-1}\mathbf{b}$ if $p^h > |\nu_j|\delta_j$ and $2 \le 2\delta_j < p^h$. To bound ν_j and δ_j, recall Fact 2.4 and obtain that the rational roundoff applies in our case with $k = n|M|^{n-1}, q = 2n|M|^{2n-1}$, so every component x_j can be recovered from $x_j \mod p^h$ if $p^h > 2n|M|^{2n-1} \ge 2n(\gamma n)^{2n-1}$, that is, if $h > \log_p(2n) + (2n-1)\log_p(\gamma n) > 2n\log_p(\gamma n)$ for $n > 1$. Now, Fact 2.7 supports the claimed bit cost bound for the recovery of $\mathbf{x}$ from $\mathbf{x}^{(h)}$, which dominates the bit operation cost of the transition from p-adic to binary representation of $\mathbf{x}^{(h)}$.

□

The following simple theorem is the basis for faster randomized recovery.

Theorem 3.5. *For a nonsingular matrix $M \in \mathbf{Z}^{n\times n}$ and its leading Smith's invariant factor $s_n = s_n(M)$, we have $s_n M^{-1} \in \mathbf{Z}^{n\times n}$.*

Theorem 3.6. *For a nonsingular matrix $M \in \mathbf{Z}^{n\times n}$ such that $v_M \geq n$, γ in Theorem 3.2, and a positive $\epsilon < 1$, it is sufficient to generate a random prime p in the range $(a, na]$ (for $a = (Cn/\epsilon)\log|M|$ and a constant C) and K random vectors $\mathbf{b}^{(k)}, \mathbf{c}^{(k)} \in \mathbf{Z}_m^n$ (for $k = 1, 2, \ldots, K, K = O(\log(1/\epsilon))$, and $m = \max\{\left\lceil \sqrt{n\log|M|}\right\rceil, 4000\}$), that is, a total of $O((n\log(n\log|M|))\log(1/\epsilon))$ random bits, and in addition to perform $i_M\mu(\log\ p) + O((v_M + v_{M^{-1}})\mu(\log\ \gamma)n\log_p\ \gamma + \rho(|M|^n))\log(1/\epsilon))$ bit operations (for $\mu(h)$ in (1.1), $\rho(q)$ in Fact 2.7) in order to compute a positive s_n^* dividing $s_n = s_n(M)$ and such that*

$$\text{Probability}(s_n^* = s_n) \geq 1 - \epsilon.$$

Outline of Proof. The divisor of $s_n(M)$ (equal to $s_n(M)$ with a high probability) is computed as the least common denominator of the rational numbers $\eta_k = \mathbf{c}^{(k)T}\mathbf{y}^{(k)}$ where $M\mathbf{y}^{(k)} = \mathbf{b}^{(k)}$ and $\mathbf{b}^{(k)}$ and $\mathbf{c}^{(k)}$ are random vectors, $k = 1, 2, \ldots, O(\log(1/\epsilon))$; the linear systems $M\mathbf{y}^{(k)} = \mathbf{b}^{(k)}$ are solved based on Hensel's lifting. The bit cost of the recovery decreases by the factor of $n/\log(1/\epsilon)$ versus Theorem 3.4 because we only recover $O(\log(1/\epsilon))$ scalars η_k. Full details of the proof are presented in the next section.

□

Let us next specify the cost estimates in terms of $n, |M|$, and ϵ , for general and Toeplitz matrices. For a non-singular $n \times n$ matrix M, we have $i_M = O(n^3), v_M \leq 2n^2 - n, v_{M^{-1}} \leq 2n^2 - n$. If M is a Toeplitz matrix, we have $i_M = O(m(n)\ \log\ n), v_M = O(m(n)), v_{M^{-1}} = O(m(n))$, provided M^{-1} is given with its generator (see Corollary 2.11). Substituting these bounds, we observe that the lifting cost dominates the cost of inversion modulo p and the recovery cost for both general and Toeplitz matrices M. So we specialize Theorem 3.6 as follows.

Corollary 3.7. *Let $M \in \mathbf{Z}^{n\times n}, \det(M) \neq 0, |M| \geq n$, and $0 < \epsilon < 1$. Then a divisor s_n^* of the leading Smith factor s_n such that $s_n^* = s_n$ with a probability of at least $1 - \epsilon$ can be computed by generating $O((n \log(n \log |M|)) \log(1/\epsilon))$ random bits and in addition performing $\alpha = O((n^3 \mu(\log\ \gamma) \log_p\ \gamma) \log(1/\epsilon))$ bit operations for $p = O((n^2/\epsilon) \log |M|), \mu(h)$ in (1.1), and γ in Theorem 3.2. For a Toeplitz matrix M the bit operation cost bound is bounded by $\beta = O((n\ \log_p \gamma) m(n) \mu(\log\ \gamma) \log\ (1/\epsilon))$ for $m(n)$ in (1.2). If $\log_p \gamma = O(1)$ and $1/\epsilon = O(1)$, then $\alpha = O(n^3 \mu(\log\ n))$, $\beta = O(m(n) n \mu(\log\ n))$.*

Now, having s_n and assuming for simplicity that

$$\log\ |\mathbf{b}| \leq n\ \log\ |M|, \tag{3.1}$$

we compute the vectors $\mathbf{x}\mod p^h$ (by applying Algorithm 3.1 for the same prime p used in the computation of s_n) and then $\mathbf{z} = s_n\mathbf{x} = (s_n\mathbf{x}) \mod p^h \in \mathbf{Z}^n$. The pair $s_n, \mathbf{z}$ defines $\mathbf{x} = \mathbf{z}/s_n \in \mathbf{Q}^n$. The overall bit cost of the solution of a linear system $M\mathbf{x} = \mathbf{b}$ is dominated by the estimates in Theorem 3.6 and Corollary 3.7.

Theorem 3.8. *Given a nonsingular matrix $M \in \mathbf{Z}^{n\times n}$, a vector $\mathbf{b} \in \mathbf{Z}^n$ satisfying (3.1), and a positive ϵ, the bit cost bounds of Theorem 3.6 and Corollary 3.7 apply to the solution of the linear system $M\mathbf{x} = \mathbf{b}$ with an error probability of at most ϵ. The bit operation cost bound covers the cost of verifying correctness of the computed solution $\mathbf{x}$.*

4 Computation of the leading Smith factor

To support Theorem 3.6 for $\epsilon = 1/2$, let us modify the algorithm **Largest Invariant Factor** in [EGV00, Section 2] by changing its parameters m and $t_n^{(k)}$. (We write m instead of M in [EGV00] and then M instead of A in [EGV00].) As in [EGV00], the extension to any fixed $\epsilon, 0 < \epsilon < 1$, is by increasing the parameter K by the factor of $\log(1/\epsilon)$.

Algorithm 4.1. *Leading Smith Factor.*
INPUT: A nonsingular matrix $M \in \mathbf{Z}^{n\times n}$.
OUTPUT: A positive integer s_n^* dividing s_n.

INITIALIZATION: $m, p, \mathbf{b}^{(k)}$ and $\mathbf{c}^{(k)}$ are as in the Theorem 3.6 for $K = 2$, and $h = 1 + \lceil 2\log_p(2|M|^{2n-1}m) \rceil$ such that $p^h > 2|M|^{2n-1}m$.
COMPUTATION: For $k = 1, 2$, first compute in $\mathbf{Z}_q$ the vectors $\mathbf{x}^{(k)}$ and scalars $y^{(k)}$, then compute the integers $t^{(k)}$ and s_n^* as follows:
1. $\mathbf{x}^{(k)} = (x_i^{(k)})_{i=0}^{n-1} = M^{-1}\mathbf{b}^{(k)} \in \mathbf{Z}_q^n$,
2. $y^{(k)} = \mathbf{c}^{(k)T}\mathbf{x}^{(k)} = \sum_{i=0}^{n-1} c_i^{(k)} x_i^{(k)} \in \mathbf{Z}_q$,
3. $t^{(k)} = \delta(y^{(k)})$, so $1 \le t^{(k)} \le |M|^n$,
4. $s_n^* = \mathrm{lcm}(t^{(1)}, t^{(2)})$.

Clearly, s_n^* divides s_n. Let us prove that $s_n^* = s_n$ with a probability of at least 1/2. It is sufficient to repeat the proof of Theorem 2 in [EGV00] complemented by the next lemma, which for every k validates using the denominator $t^{(k)}$ of a linear combination of $x_0^{(k)}, \ldots, x_{n-1}^{(k)}$ instead of the lcm of all denominators $\delta(x_0^{(k)}), \ldots, \delta(x_{n-1}^{(k)})$. Hereafter, write $l = \mathrm{ord}_p(z)$ if $p, z \in \mathbf{Z}, p > 1$, p^l divides z, but p^{l+1} does not.

Lemma 4.2. *Fix $k = 1$ or $k = 2$ and write $\delta^{(k)} = \mathrm{lcm}(\delta(x_0^{(k)}), \ldots, \delta(x_{n-1}^{(k)}))$, so $t^{(k)}$ divides $\delta^{(k)}$; $\delta^{(k)}$ divides s_n. Then for any prime $\bar{p}$,*
a) $\mathrm{Probability}(\mathrm{ord}_{\bar{p}}(s_n) \ne \mathrm{ord}_{\bar{p}}(\delta^{(k)})) \le \max\{1/m, 1/\bar{p}\}$;
b) $\mathrm{Probability}(\mathrm{ord}_{\bar{p}}(t^{(k)}) \ne \mathrm{ord}_{\bar{p}}(\delta^{(k)})) \le \max\{1/m, 1/\bar{p}\}$.

Proof of Lemma 4.2. Part a) follows from Theorem 2 in [ABM99], but here is a simpler proof. We have
$x_i^{(k)} = \sum_j (-1)^{i+j} d_{i,j} b_j^{(k)} / \det(M)$, $s_n = |\det(M)/d|$, $d = \gcd(d_{i,j})_{i,j}$ for $d_{i,j}$ in Definition 2.2 and $\mathbf{b}^{(k)} = (b_j^{(k)})_{j=1}^n$. Write
$h_{i,j} = \mathrm{ord}_{\bar{p}}(d_{i,j})$, $h = \mathrm{ord}_{\bar{p}}(d) = \min_{i,j} d_{i,j}$. We have $h = \mathrm{ord}_{\bar{p}}(d_{u,v})$ for some u, v; w.l.o.g., let $u = v = 0$. Furthermore, write $\bar{d}_{i,j} = d_{i,j}/d$ for all i and j. Then it follows that $s_n x_0^{(k)} = \bar{d}_{0,0} b_0^{(k)} + r$, where $r = \sum_{j=1}^{n-1} (-1)^j \bar{d}_{0,j} b_j^{(k)} \in \mathbf{Z}$. Since $\mathrm{ord}_{\bar{p}}(\bar{d}_{0,0}) = 0$ and $b_0^{(k)}$ is randomly chosen in $\mathbf{Z}_m$, part a) of the lemma follows.
To prove part b) first write
$x_i^{(k)} = \nu_i/\delta_i$, $y^{(k)} = \sum_{i=0}^{n-1} c_i^{(k)} \nu_i/\delta_i$, $\sigma_i = \delta^{(k)}/\delta_i = (\mathrm{lcm}(\delta_i)_i)/\delta_i$, where ν_i and δ_i are coprime, for all i. (We drop the superscripts k of $\delta_i^{(k)}$ and $\nu_i^{(k)}$ to simplify the notation.) Clearly, $\min_i\{\mathrm{ord}_{\bar{p}}(\sigma_i)\} = 0$ for any prime $\bar{p}$. W.l.o.g., let $\mathrm{ord}_{\bar{p}}(\sigma_0) = 0$. If $\bar{p}$ divides ν_0, then $\mathrm{ord}_{\bar{p}}(\delta_0) = 0$, so $0 = \mathrm{ord}_{\bar{p}}(\delta^{(k)}) \ge \mathrm{ord}_{\bar{p}}(t^{(k)}) \ge 0$, that is, $\mathrm{ord}_{\bar{p}}(\delta^{(k)}) = \mathrm{ord}_{\bar{p}}(t^{(k)}) = 0$. It remains to cover the case where

$$\mathrm{ord}_{\bar{p}}(\nu_0) = \mathrm{ord}_{\bar{p}}(\sigma_0) = 0. \tag{4.1}$$

Now observe that $y^{(k)} = \sum_{i=0}^{n-1} c_i^{(k)} \nu_i \sigma_i / {\delta_i^{(k)}}_i$, so $\mathrm{ord}_{\bar{p}}(t^{(k)}) = \mathrm{ord}_{\bar{p}}(\delta^{(k)})$ if $\mathrm{ord}_{\bar{p}}(\sum_{i=0}^{n-1} c_i^{(k)} \nu_i \sigma_i) = 0$. Under (4.1), the latter equation holds with a probability of at least $\max\{1/m, 1/\bar{p}\}$ for $c_0^{(k)}$ randomly chosen in $\mathbf{Z}_m$.

□

To prove Theorem 3.6 it remains to estimate from above the number of bit operations used in Algorithm 4.1. We have the following upper bounds: $i_M \mu(\log\ p)$ for computing $M^{-1} \mod p$ at stage 1 (once for all k); $O((v_M + v_{M^{-1}})\mu(\log\ \gamma)h)$, where $h = O(n\ \log_p\ \gamma)$, in Hensel's p-adic lifting applied for each fixed k to compute $M^{-1}\mathbf{b}^{(k)} \mod p^h$ (also at stage 1), and $O(\mu(n \log |M|) \log(n \log |M|))$ for each k at stage 3. The cost of lifting dominates the cost at stages 2 and 4 (recall our assumption that $v_M \geq n$). Summarizing, we complete the proof of Theorem 3.6.

5 Conversion into binary form

Based on Algorithms 3.1 and 4.1, we compute the vector $\mathbf{x} = M^{-1}\mathbf{b}$ in the form of a pair $\mathbf{y}, s_n$ such that s_n and all n components of $\mathbf{y}$ are integers represented as p-adic numbers, and $\mathbf{x} = \mathbf{y}/s_n$. Practically, it is desired to convert these integers into the binary form. For each integer the convergence requires the order of $\mu(n \log\ p) \log\ p$ bit operations [GG99]. So, the conversion cost for s_n and $O(1)$ entries of $\mathbf{y}$ is dominated by the cost of performing the algorithms in Sections 3 and 4, but for all n components of $\mathbf{y}$, the conversion would increase the overall bit cost bound by logarithmic factor. If s_n is an odd integer, this small but practically undesirable cost increase can be avoided as we simply change p into 2 when we solve the linear system $M\mathbf{y} = s_n\mathbf{b}$. Indeed, in this case $\det(M)$ is odd, so the matrix $M \mod p$ remains nonsingular, p-adic means binary, and no conversion is needed. The next algorithm extends our latter recipe to any s_n.

Algorithm 5.1. *Linear solver in binary form.*
INPUT: A nonsingular matrix $M \in \mathbf{Z}^{n \times n}$ and a vector $\mathbf{b} \in \mathbf{Z}^n$.
OUTPUT: Scalar $s_n = s_n(M)$ and vector $\mathbf{y} = s_n M^{-1}\mathbf{b}$, both in binary form.
COMPUTATIONS:
1. Recursively generate 2ℓ random Toeplitz matrices $U_k, V_k^T \in \mathbf{Z}_q^{n \times k}$ for a fixed q in $n^{O(1)}, k = 1, 2, \ldots, l$, and apply Algorithm 4.1 to compute Smith's leading invariant factors $s_{n,k} = s_n(M_k)$ for the matrices

$M_k = M - U_k V_k, k = 0, 1, \ldots, l$. Stop for the smallest l for which $s_{n,l} = s_n(M_l)$ is odd.
2) Apply the algorithms of the preceding sections, for $p = 2$ and for a fixed sufficiently small positive h, to compute the $n \times l$ matrix $M_l^{-1} U_l = W_l$ and the vectors $\mathbf{u} = M_l^{-1}\mathbf{b}, \mathbf{v} = M_l^{-1} U_l (I + V_l W_l)^{-1} V_l \mathbf{u}$, and finally $\mathbf{x} = \mathbf{u} - \mathbf{v} = M^{-1}\mathbf{b} = (M_l^{-1} - M_l^{-1} U_l (I_l + V_l M_l^{-1} U_l)^{-1} V_l M_l^{-1})\mathbf{b}$ in the binary (2-adic) form.

The latter matrix equation relies on the (Sherman–Morrison–)Woodbury formula for $M^{-1} = (M_l + UV)^{-1}$ [GL96]. We also need Lemma 3.2 and Theorem 3.13 in [EGV00] by which with a high probability all matrices M_k are nonsingular and $s_{n-k}(M) = \gcd(s_n(M), s_n(M_k))$ for $k = 1, \ldots, l$. The first property, together with the (Sherman–Morrison–)Woodbury formula, implies correctness of the algorithm.
We now combine the second property with the well known fact that for a random integer matrix M, we almost always have $s_{n-k} = 1$ for all but $O(1)$ smallest values of k [ABM99]. By combining this fact with the above expression for $s_{n-k}(M)$, we deduce that $l = O(1)$ for the average random integer matrix M.
To estimate the arithmetic cost, observe that the matrices M_k have displacement rank 3, so the definition of a generator of the inverse and Corollary 2.11 are extended (see the definition of the displacement rank and proofs in [P01]) to yield that $v_{M_k} \le 4m(n) + n$ and $i_{M_k} \le 6m(n) + 2n$ for v_S in Definition 2.5.
Now, reexamination of Algorithm 4.1 (with U_k replaced by $U_k J$ or $J U_k$ in the Hankel-like case) leads us to the following estimates.

Theorem 5.2. *For random average (general or Toeplitz) integer matrix M, the asymptotic cost bounds of Corollary 3.7 apply to the bit cost of performing Algorithm 5.1, except that $O(n)$ additional random entries of the matrices $U_k, V_k, k = 1, \ldots, l$, for $l = O(1)$, must be generated in $\mathbf{Z}_q, q \in n^{O(1)}$. The same bounds cover the bit cost of computing all Smith invariant factors of the average M and, consequently,* $\det(M)$.

Let us extend Hensel's lifting by relaxing the assumption that the basic prime p is coprime with $\det(M)$.

Algorithm 5.3. *Lifting without coprimality.*
INPUT: $M \in \mathbf{Z}^{n \times n}$, a prime p, the integer $g = \mathrm{ord}_p(s_n(M))$, two positive integers h and k, and a matrix $Q \in \mathbf{Z}^{n \times n}$ such that $MQ = p^g I \mod p^{g+k}$.

OUTPUT: $\mathbf{x}^{(h)} \in \mathbf{Z}^n$ such that $\mathbf{x}^{(h)} = p^g M^{-1}\mathbf{b} \mod p^{g+kh}$.
INITIALIZE: $\mathbf{r}^{(0)} = p^g\mathbf{b}$.
COMPUTATIONS: for $i = 0, 1, \ldots, h-1$,
compute $\mathbf{u}^{(i)} = Q\mathbf{r}^{(i)} \mod p^{g+k}, \mathbf{r}^{(i+1)} = (p^g\mathbf{r}^{(i)} - M\mathbf{u}^{(i)})/p^{g+k}$.
Output $\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} \mathbf{u}^{(i)} p^{g+ki}$.

For $g = 0$ and $k = 1$, Algorithm 5.3 turns into Algorithm 3.1.

Theorem 5.4. *(Cf. Theorem 3.2.)*
a) $\mathbf{r}^{(i+1)} \in \mathbf{Z}^n$*;*
b) $M\mathbf{x}^{(h)} = p^g\mathbf{b} \mod p^{g+kh}$*;*
c) all components $r_j^{(i)}$ *of all vectors* $\mathbf{r}^{(i)}$ *satisfy* $|r_j^{(i)}| \le n\gamma p^k/(2p^k - 2)$ *for* γ *in Theorem 3.2.*

Proof. a) $p^g\mathbf{r}^{(i)} - M\mathbf{u}^{(i)} = (p^g I - MQ)\mathbf{r}^{(i)} \mod p^{g+k}$, and the claim follows because $MQ = p^g I \mod p^{g+k}$.
b) $M\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} M\mathbf{u}^{(i)} p^{g+ki} = \sum_{i=0}^{h-1} (p^g\mathbf{r}^{(i)} - p^{g+k}\mathbf{r}^{(i+1)})p^{ki} =$
$p^g\mathbf{b} - p^{g+kh}\mathbf{r}^{(h)} = p^g\mathbf{b} \mod p^{g+kh}$.
c) By definition, all components $u_j^{(i)}$ of all vectors $\mathbf{u}^{(i)}$ satisfy $2|u_j^{(i)}| \le p^{k+g}$, and so $p^{k+g}|r_j^{(i+1)}| \le p^g|r_j^{(i)}| + n\gamma|u_j^{(i)}| \le p^g|r_j^{(i)}| + n\gamma p^{k+g}/2$. The claim now follows by induction on i.

□

Corollary 5.5. *Algorithm 5.3 outputs* $\mathbf{x}^{(h)}$ *in p-adic form by performing* $O((v_Q\mu(\log(n\gamma)) + v_M\mu(\log(p^{g+k}))h)$ *bit operations for* $\mu(d)$ *in (1.1) and* v_S *in Definition 2.5.*
For $p = 2$ the entire algorithm can be performed in binary form. If $g = g(M)$ is small or moderate, the bit cost is reasonable. If $g = g(M)$ is large, we may shift to the matrices $M_k = M - U_k V_k, k = 1, 2, \ldots$, defined earlier in this section.
The input matrix Q defines the integer g. To compute Q, we may apply the algorithms in the next section. An alternative way is to apply some fast or superfast algorithm modulo a random prime p in $n^{O(1)}\log|M|$, still keeping computational precision in $O(\log(n\log|M|))$. Then the bit cost at this stage may even decrease, but some complications arise if we choose $p = 2$ or $p = 2^g$, where $\det(M)$ is even.

6 Initialization of Hensel's lifting with variable diagonal and modular continuation

Let us next show two alternative algorithms for computing generator for $M^{-1} \mod p$ given a Toeplitz matrix M. This task requires the solution of two linear systems with matrix M, and we show how to solve such a system.

Algorithm 6.1. *Initialization of Toeplitz–Hensel's lifting with variable diagonal (cf. [P00]).*
INPUT: $M \in \mathbf{Z}^{n\times n}, \mathbf{b} \in \mathbf{Z}^n$ satisfying (3.1), an integer $p \geq 2$, and two sufficiently large integers h and l (specified later on).
OUTPUT: $g = \max_j(\mathrm{ord}_p(\delta(M^{-1}\mathbf{b})_j))$ and $\mathbf{x}^{(l)} = (p^g M^{-1}\mathbf{b}) \mod p^l$.
INITIALIZE: Compute the matrices $M_0 = M + p^l I$ and $Q_0 = p^{-l} I$, so $Q_0 M_0 - I = p^{-l} M$. Write $\mathbf{z}_0 = 0, \mathbf{r}_0 = \mathbf{b}$.
COMPUTATIONS:
1. Recursively compute the vectors $\mathbf{z}_{i+1} - \mathbf{z}_i = Q_0\mathbf{r}_i = p^{-l}\mathbf{r}_i, \mathbf{r}_{i+1} = \mathbf{b} - M_0\mathbf{z}_{i+1} = \mathbf{r}_i - M_0 Q_0 \mathbf{r}_i = -p^{-l} M\mathbf{r}_i, i = 0, 1, \ldots, h-1$.
2. Recover $\mathbf{z} = M_0^{-1}\mathbf{b}$ from $\mathbf{z}_h$ by using the rational roundoff algorithm in Section 2.2.
3. Compute $g_0 = \max_j(\mathrm{ord}_p(\delta(M_0^{-1}\mathbf{b})_j))$. If $g_0 < l$, output $g = g_0$ and $\mathbf{x}^{(l)} = (p^g M_0^{-1}\mathbf{b}) \mod p^l$. Otherwise double l and reapply the algorithm.

Stage 1 is the customary residual correction algorithm for iterative improvement of approximation to $\mathbf{z}$ [GL96]. We have

$$\mathbf{z} - \mathbf{z}_h = M_0^{-1}(\mathbf{b} - M_0\mathbf{z}_h) = M_0^{-1}\mathbf{r}_h,$$
$$\mathbf{r}_h = -p^{-l} M\mathbf{r}_{h-1} = (-p^{-l}M)^h \mathbf{r}_0 = (-p^{-l}M)^h \mathbf{b}.$$

So $|\mathbf{z} - \mathbf{z}_h| \leq (p^{-l}|M|)^h |\mathbf{b}|/|M_0|$. Let $p \geq 2$,

$$l = 1 + \max\{1, \left\lfloor 2\log_p |M| \right\rfloor, \left\lfloor 2\log_p |\mathbf{b}| \right\rfloor\} \geq g. \tag{6.1}$$

Then $p^{l/2-1} \leq |M| < p^{l/2}, |\mathbf{b}| < p^{l/2}$, $|M_0| \geq p^l - p^{l/2} \geq (p^{l/2} - 1)p^{l/2}$, and

$$|\mathbf{z} - \mathbf{z}_h| < p^{-hl/2}. \tag{6.2}$$

Therefore, every iteration step in Stage 1,

$$\mathbf{z}_{i+1} - \mathbf{z}_i = p^l \mathbf{r}_i, \mathbf{r}_{i+1} = -p^{-l} M\mathbf{r},$$

contributes $l/2$ additional correct p-digits in the p-adic representation of $\mathbf{z}$. Rounding the components of $\mathbf{r}_{i+1}$ to (say) l leading p-digits may destroy at most a single correct p-digit per component, whereas the computational precision would stay bounded to $O(l \log\ p)$ bits. (This argument is a simplification of the routine backward error analysis of the iterative improvement algorithm which proves its numerical stability in a more general form [S80],[H96] versus our simplified case where Q is in a very special form of $p^{-l}I$.)
Now to ensure correct recovery of $\mathbf{z}$ with using rational roundoff, it is sufficient to approximate $\mathbf{z}$ by $\mathbf{z}_h$ within the error norm less than $1/(n|M_0|^{2n-1})$. Due to (6.1),(6.2), we achieve this as soon as

$$p^{hl/2} > n(p^l + p^{l/2})^{2n-1} \geq n|M|^{2n-1},$$

that is, as soon as

$$(hl/2)\log\ p \geq (2n-1)(l/2)\log\ p + \log\ n.$$

This inequality holds for $h \geq 2n - 1 + (2\log n)/(\log p^l)$. Therefore, the arithmetic and Boolean bit cost of performing stage 1 are bounded by

$$\xi = O(hm(n)) = O(nm(n)), \quad \eta = \xi\mu(l \log\ p), \tag{6.3}$$

respectively. Under (6.1), we express the bit cost in terms of $|M|$ and n as follows,

$$\eta = O(nm(n)\mu(\log|M|)). \tag{6.4}$$

If initially $l \leq g$, then finally, $g \leq l < 2g$, so the bit cost bound turns into

$$\eta = \xi\mu(g \log\ p) = O(nm(n)\mu(g \log\ p)). \tag{6.5}$$

Rational roundoff at Stage 2 requires $O(\mu(d)\log\ d)$ bit operations for $d = n\log|M_0| = n\log|M + p^l I| = O(nl \log\ p)$ per an entry of $\mathbf{z}$. To decrease the overall bit cost of the recovery to the level below the lifting cost, we employ our techniques of Sections 3 and 4 again. That is, we first apply Algorithm 6.1 with randomization to computing $s_n = s_n(M)$ probabilistically (by using random vectors $\mathbf{b}^{(k)}$ and $\mathbf{c}^{(k)}$ as in Sections 3 and 4), and then all components of $s_n(\mathbf{z})$ become integers and are recovered cost-free. Thus the asymptotic bit cost bounds (6.3)–(6.5) cover the entire cost of performing Algorithm 6.1.

The next algorithm is similar to Algorithm 6.1. It solves the same computational problem by first computing $M_0^{-1} \mod q$ for a fixed prime q distinct from p and then computes $M_0^{-1} \mod q^h$, $M_0^{-1}\mathbf{b}$ and $M_0^{-1}\mathbf{b} \mod p$. Its analysis is simpler, but this algorithm requires computations modulo another prime q, whereas Algorithm 6.1 for $p = 2$ performs all computations with binary numbers and also outputs binary integers.

Algorithm 6.2. *Initialization of Toeplitz–Hensel's lifting with modular continuation.*
INPUT: $M \in \mathbf{Z}^{n\times n}, \mathbf{b} \in \mathbf{Z}^n$, satisfying (3.1) an integer $p \geq 2$.
OUTPUT: $g = \max_j(\mathrm{ord}_p(\delta(M^{-1}\mathbf{b})_j))$ and $(p^g M^{-1}\mathbf{b}) \mod p$.
INITIALIZE: Choose an integer $q > 1$ coprime with p. (Sample choices are given by $p, q \in \{2, 3, 5\}$ or by p and q being powers of $2, 3$, or 5.)
COMPUTATIONS:
1. Compute $s = p^{-1} \mod q, t = q^{-1} \mod p$, and the matrix $M_0 = pI + qM$, so $Q = M_0^{-1} \mod q = sI$.
2. Apply Algorithm 3.1 for M replaced by M_0 and p by q to compute $M_0^{-1}\mathbf{b} \mod q^h$. Choose h sufficiently large and recover $M_0^{-1}\mathbf{b}$.
3. Compute and output $g = \max_j(\mathrm{ord}_p\, \delta((M_0^{-1}\mathbf{b})_j))$ and $p^g(M_0^{-1}\mathbf{b}) \mod p = (tp^g M^{-1}\mathbf{b}) \mod p$.

Including Algorithm 6.2 as a block in the parallel algorithm in [P00] (with Newton's lifting replacing Hensel's lifting) enables dramatic simplification because most part of [P00] is devoted to fast parallel computation of the initial matrix $M^{-1} \mod p$. Furthermore, some complications in [P00] are due to using the MBA algorithm, which involves many auxiliary Toeplitz-like matrices for a Toeplitz input matrix M, whereas Algorithm 6.2 avoids these complications by operating only with M and its inverse.

References

[ABM99] J. Abbott, M. Bronstein, T. Mulders. Fast Deterministic Computations of the Determinants of Dense Matrices, *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC '99)*, 197-204, ACM Press, New York, 1999.

[BGY80] R. P. Brent, F. G. Gustavson, D. Y. Y. Yun, Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximations, *J. Algorithms*, **1**, 259-295, 1980.

[BP94] D. Bini and V. Y. Pan, *Polynomial and Matrix Computations, Volume. 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[CFG99] G. Cooperman, S. Feisel, J. von zur Gathen, G. Havas, GCD of Many Integers, *Computing and Combinatorics, Lecture Notes in Computer Science*, **1627**, 310–317, Springer, Berlin, 1999.

[CK91] D.G. Cantor, E. Kaltofen, On Fast Multiplication of Polynomials over Arbitrary Rings, *Acta Informatica*, **28(7)**, 697-701,1991.

[D82] J. D. Dixon, Exact Solution of Linear Equations Using p-adic Expansions, *Numerische Math.*, **40**, 137–141, 1982.

[EGV00] W. Eberly, M. Giesbrecht, G. Villard, On Computing the Determinant and Smith Form of an Integer Matrix, *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS'2000)*, 675–685, IEEE Computer Society Press, Los Alamitos, California, 2000.

[GL96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1996.

[GG99] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, UK, 1999.

[H96] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM Publications, Philadelphia, 1996.

[KS99] T. Kailath, A. H. Sayed (editors), *Fast Reliable Algorithms for Matrices with Structure*, SIAM Publications, Philadelphia, 1999.

[MC79] R. T. Moenck, J. H. Carter, Approximate Algorithms to Derive Exact Solutions to Systems of Linear Equations, *Proceedings of EUROSAM, Lecture Notes in Computer Science*, **72**, 63–73, Springer, Berlin, 1979.

[P87] V. Y. Pan, Complexity of Parallel Matrix Computations, *Theoretical Computer Science*, **54**, 65-85, 1987.

[P88] V. Y. Pan, Computing the Determinant and the Characteristic Polynomials of a Matrix via Solving Linear Systems of Equations, *Information Processing Letters*, **28**, 71–75, 1988.

[P90] V. Y. Pan, On Computations with Dense Structured Matrices, *Mathematics of Computation*, **55(191)**, 179–190, 1990.

[P92] V. Y. Pan, Parametrization of Newton's Iteration for Computations with Structured Matrices and Applications, *Computers and Mathematics (with Applications)*, **24(3)**, 61–75, 1992.

[P00] V. Y. Pan, Parallel Complexity of Computations with General and Toeplitz-like Matrices Filled with Integers and Extensions, *SIAM J. Comput.*, **30(4)**, 1080-1125, 2000.

[P01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/NewYork, 2001.

[PW02] V. Y. Pan, X. Wang, Acceleration of Euclidean Algorithm and Extensions, preprint, 2001.

[S80] R. D. Skeel, Iterative Refinement Implies Numerical Stability for Gaussian Elimination, *Math. of Computation*, **35**, 817–832, 1980.

[Z93] R. Zippel, *Effective Polynomial Computation*, Kluwer, Boston, 1993.