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

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 record asymptotic bit operation cost of the solution, 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 entire computation in binary form, which promises to be valuable for practical computations. 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 applied to a larger class of Toeplitz/Hankel-like linear systems.

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 or 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 Kailath and Sayed [KS99], Pan [P00, Section 1.1] and Pan [P01]). Furthermore, the displacement transformation approach of Pan [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 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" algorithms by Levinson 1947/Durbin 1959 and by Trench 1964, and to $O(n \log^2 n)$ in the "superfast" BGY algorithm by Brent, Gustavson, and Yun [BGY80] and the divide-and-conquer MBA algorithm by Morf 1974/1980 and Bitmead/Anderson 1980 (cf. [P01]).

The more realistic measure 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 assign the cost of $\mu(d)$ bit operations (hereafter log stands for $\log_2$ unless specified otherwise), where

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

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

According to Tyrtyshnikov [T94], Toeplitz/Hankel matrices tend to be ill-conditioned, which motivates application of symbolic/algebraic methods for reducing the computational precision. The most popular is application of the CRA (Chinese remainder algorithm). The input is integral (or made integral by

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

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 (see [GG99] and Pan and Wang [PW02]), provided that the product $p_1 \cdots p_s$ exceeds $2\delta|\nu|$ for every rational output value $\nu/\delta, \delta \geq 1$. This property enables recovery of each value ν/δ from $(\nu/\delta) \bmod p_i, i = 1, \ldots, s$. 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 $((\det M)\mathbf{x}) \bmod p$ is cost-free.

To specify the bit cost bound for the MBA algorithm (and similarly for the BGY algorithm), let $m(n)$ denote the arithmetic cost of multiplying two polynomials of degree $n-1$,

$$2n - 1 \leq m(n) \leq (c_* n \log\ n) \log\log\ n, \tag{1.2}$$

for a constant c_* (Cantor and Kaltofen [CK91]). In the introduction, for simplicity let 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. Our new progress is twofold.

a) We decrease the randomized bit cost bound by roughly the logarithmic factor, thus reaching an information lower bound up to roughly logarithmic factor, and
b) we perform all computations in the binary form, modulo a fixed power of two; furthermore, this power of two is reasonably small on the average input; such an implementation is a substantial practical advantage versus computations modulo one or several random primes of the order of $\log n$.

Our progress should be viewed as surprising because we deal with a central problem of structured matrix computations open and intensively studied since 1980. Moreover (see [BGY80], [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].

Let us further specify our results. We rely 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 algorithm amounts to multiplication of the input matrix and its inverse modulo p by two vectors per step; these operations are quite simple for Toeplitz/Hankel-like matrices, and we arrive at the desired bit cost estimate for lifting. Lifting, however, must be initialized and be followed by the recovery of the rational output values from their truncated p-adic expansions. We study these problems here and in Pan [Pa].

At the stage of rational number reconstruction, $\det M$ is not available as by-product anymore, and until very recently the known algorithms required the order of $n^3 \log^2 n$ bit operations at this stage. In [PW02], however, the asymptotic cost bound of the extended Euclidean algorithm for integers has been improved dramatically, implying acceleration of our rational number reconstruction to $O(n\mu(n \log n) \log n)$. This theoretical progress enables us to match but still not to beat the BGY/MBA cost bound. To yield further progress, we extend the approach of Pan [P87], Pan [P88], Abbott, Bronstein and Mulders [ABM99], and Eberly, Giesbrecht and Villard [EGV00], which relates p-adic lifting to the computation of Smith's invariant factors of M. Now we recall the known trick of probabilistically computing the lcm of several integers $q_1, \ldots, q_k$ as the denominator of the random linear combination of the reciprocals $1/q_i$ (cf. Pan [P92], Bini and Pan [BP94], Cooperman, Feisel, von zur Gathen and Havas [CFG99]) and exploit this trick in a new context with a support from Smith's leading factor s_n. With this technique we decrease the bit cost of the recovery and the overall bit 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 integer vector $\mathbf{y} = s_n\mathbf{x}$ are output as p-adic numbers for a random prime p of the order of $\log n$. For practical purpose, however, the binary representation of all intermediate and output values or at least representation modulo a fixed (non random) prime or prime power p are desired.

Our practical solution to this practical problem relies on our new binary version of Hensel's lifting (where the basic prime p can be two, even if $\det M$ is even). The power of this approach can be accentuated by combining it with perturbation of the input matrix by small rank random matrices. This enables binary computation of 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.) To initialize Hensel's lifting, we may apply a superfast (BGY or MBA) algorithm modulo a random prime p of the order of $\log n$. The bit cost is smaller than for lifting, but using a fixed prime or prime power p is desired (preferably $p = 2^g$ for smaller g), and we elaborate upon this in Pan [Pa], where g is nicely bounded on the average (but not for the worst) case input. In our Section 6, we propose two alternative methods, which are performed modulo $p = 2^g$ and modulo any fixed odd prime, respectively; their bit cost is higher than with the BGY/MBA algorithms modulo (random) p but still within the cost bound of lifting.

Our algorithms promise to be practical; moreover our work may even inspire reexamination of the general method of using random primes, to avoid singularity, and may suggest using fixed prime or prime power (e.g., $p = 2^g$) as an alternative. On the theoretical side, the techniques enable us to compute the determinant and all Smith's factors of an average general or structured matrix M at the same randomized asymptotic bit cost as 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 may 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 [Pa] 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. We also study their implementation modulo 2^g.

The solution of singular but 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. As a simple preliminary demonstration, we apply the algorithms of Sections 3 and 5 to selected 2×2 matrices in Section 7.

We conclude this section with some comments on possibility of further asymptotic 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 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 and Paterson 1974 (cf. Bini and Pan [BP94, Section 3.9]) could yield theoretical acceleration by the factor of $(\log\log n)\log\log\log n$. 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 write* $z_q = z \bmod q$ *if* q *divides* $z - z_q$ *and if* $-q/2 < z_q \leq q/2$*. (Clearly,* $z = z_q$ *if* $-|q|/2 < z \leq |q|/2$*.) We write* $y = \nu(y)/\delta(y)$ *for two coprimes* $\nu(y)$ *(numerator) and* $\delta(y)$ *(denominator).* $M = (m_{i,j})_{i,j=0}^{k-1,l-1}$ *is a* $k \times l$ *matrix.*

Definition 2.2. I *is the identity matrix of a proper size,* I_l *is the* $l \times l$ *matrix* I*.* $\det M$ *and* $\operatorname{adj} M = ((-1)^{i+j} d_{i,j})_{i,j=0}^{k-1,k-1}$ *denote the determinant and adjoint (adjugate) of a* $k \times k$ *matrix* $M = (m_{i,j})_{i,j=0}^{k-1,k-1}$*,* *respectively, 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. *$|M|$ is the column norm of M, $|M| = ||M||_1 = \max_j \sum_i |m_{i,j}|$ for $M = (m_{i,j})$. $|\mathbf{v}|$ is 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].

Theorem 2.4. *$|\det M| \le |M|^k$, $|\mathrm{adj}\, M| \le k|M|^{k-1}$ 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. *$d_k = d_k(M)$, the k-th determinantal divisor of M, is the greatest common divisor (gcd) of all $k \times k$ minors (subdeterminants) of a matrix $M \in \mathbf{Z}^{n\times n}$, $k = 1, \dots, n$. 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 $k = 1, \dots, n$.*

We have $s_1, \dots, s_n \in \mathbf{Z}$ and $|\det M| = s_1 \cdots s_n$, so (cf. Theorem 2.4) we have

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

2.2 The Bit-Complexity of Rational Number Reconstruction

Hereafter, $\rho(q)$ denotes the bit-operation cost of *modular rational roundoff*, that is, of recovering a rational number x/y from three integers k, q, and $r = (x/y) \bmod q$ 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, $x = |r|, y = 1, \rho(q) = 0$ if $k > 2|r|$.

Likewise, $\bar\rho(\delta)$ denotes the bit operation cost of *numerical rational roundoff*, that is, of 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, $r_0, r_1, r_2, \dots$ [GG99], Zippel [Z93]. The classical Euclidean algorithm supports the bit cost bounds $\rho(q) \le cd^2$, $\bar\rho(\bar q) \le \bar c\bar\delta^2$, the algorithm in [PW02] yields

$$\rho(q) \le C\mu(d)\log d, \quad \bar\rho(\delta) \le C\mu(\bar d)\log \bar d, \tag{2.2}$$

where $\mu(d)$ is in (1.1), $d = \log q$, $\bar d = \log \delta$, $c < C$, $\bar c < \bar C$.

2.3 Toeplitz and Hankel Matrices

Definition 2.7. *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 $h + g \ne n-1$, reverses any vector $\boldsymbol{v} = (v_i)_{i=0}^{n-1}$, that is, $J\boldsymbol{v} = (v_{n-i-1})_{i=0}^{n-1}$, $J^2 = I$.*

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 the Toeplitz case.

The next well known theorem (cf., e.g., [P01, Chapter 2]) expresses the Toeplitz inverse via its products with two fixed vectors.

Theorem 2.8. *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 2.8 reduces $n \times n$ Toeplitz inversion to solving two fixed Toeplitz linear systems (each of n equations) and to multiplication of four triangular Toeplitz matrices by vectors.

Effective computations with Toeplitz matrices rely on fast multiplication of a Toeplitz matrix and its inverse by a vector. Here are the arithmetic cost bounds.

Theorem 2.9. *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 2.10. *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 2.8.*

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

In h steps of the next algorithm, p-adic expansion modulo p^h of the solution of a linear system $M\mathbf{x} = \mathbf{b}$ is computed by performing $v_M + v_{M^{-1}}$ lower precision arithmetic operations per step (see Definition 2.5, Theorems 3.2 and 3.4, and Corollary 3.3).

Algorithm 3.1. *Hensel's lifting for a linear system* [MC79], [D82] (see Example 7.1).
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 $Q = M^{-1} \bmod p$.
OUTPUT: $\mathbf{x}^{(h)} = M^{-1}\mathbf{b} \bmod p^h$.
INITIALIZE: $\mathbf{r}^{(0)} = \mathbf{b}$.
COMPUTATIONS: for $i = 0, 1, \ldots, h-1$, compute

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

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) $M \sum_{i=0}^{j-1} \mathbf{u}^{(i)} p^i = \mathbf{b} \bmod p^j, j = 1, 2, \ldots, h$;
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 $O((v_M \mu(\log(n\gamma)) + v_Q \mu(\log p))h)$ bit operations for $\mu(d)$ in (1.1) and v_S of Definition 2.5, to output the vector $\mathbf{x}^{(h)}$ in p-adic form.*

Let us next specify the integer parameters h and p and the bit cost of the recovery of the rational solution from its truncated p-adic expansion.

Theorem 3.4. *It is sufficient to choose $h = \lceil 2n \log_p(\gamma n) \rceil$ in Algorithm 3.1 and to perform $O(n\rho(p^h))$ 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*

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

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 $p^h \ge q = 2n|M|^{2n-1} > |\nu_j|\delta_j$ and $2 \le 2\delta_j < k = n|M|^{n-1}$ (see Theorem 2.4). So, every component x_j can be recovered from $x_j \bmod p^h$ 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 $n > 1$. □

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 $s_n M^{-1} \in \mathbf{Z}^{n \times n}$.*

In Section 4, based on a nontrivial algorithm for computing s_n, we prove the next theorem.

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 $i_M\mu(\log\ p)+O((v_{M^{-1}}\mu(\log p)+v_M\mu(\log\gamma))n\log_p\gamma+\rho(|M|^n))\log(1/\epsilon))$ bit operations (for $\mu(h)$ in (1.1), $\rho(q)$ in (2.2)) in order to compute a positive s_n^* dividing $s_n = s_n(M)$ and such that*

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

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 2.10). 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 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 (bounded in (2.1)) and assuming for simplicity that

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

we compute the vectors $\mathbf{x} \bmod p^h$ (by applying Algorithm 3.1 for the same prime p used in the computation of s_n) and then $\mathbf{z} = s_n\mathbf{x} = (s_n\mathbf{x}) \bmod p^h \in \mathbf{Z}^n$. The pair $s_n, \mathbf{z}$ defines $\mathbf{x} = \mathbf{z}/s_n \in \mathbf{Q}^n$. The nearly optimal overall bit cost of the solution of a linear system $M\mathbf{x} = \mathbf{b}$ is shown in the next theorem. It 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 $M\mathbf{x} = \mathbf{b}$. The solution may have an error with a 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 $\epsilon = 1/2$, we 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. *The 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 $1 \leq t^{(k)} \leq |M|^n$,
4. $s_n^* = \mathrm{lcm}\,(t^{(1)}, t^{(2)})$.

Clearly, s_n^* divides s_n. To prove that $s_n^* = s_n$ with a probability of at least 1/2, we combine the proof of Theorem 2 in [EGV00] with the next lemma, which for every k validates using the denominator $t^{(k)}$ of a linear combination of $x_0^{(k)}, \dots, x_{n-1}^{(k)}$ instead of the lcm of all denominators $\delta(x_0^{(k)}), \dots, \delta(x_{n-1}^{(k)})$. Hereafter, write $l = \operatorname{ord}_p(z)$ 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 $\delta^{(k)} = \operatorname{lcm}(\delta(x_0^{(k)}), \dots, \delta(x_{n-1}^{(k)}))$, so $t^{(k)}$ divides $\delta^{(k)}$; $\delta^{(k)}$ divides s_n. Then for any prime $\bar{p}$,*

a) Probability$(\operatorname{ord}_{\bar{p}}(s_n) \neq \operatorname{ord}_{\bar{p}}(\delta^{(k)})) \leq \max\{1/m, 1/\bar{p}\}$;

b) Probability$(\operatorname{ord}_{\bar{p}}(t^{(k)}) \neq \operatorname{ord}_{\bar{p}}(\delta^{(k)})) \leq \max\{1/m, 1/\bar{p}\}$.

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 $h_{i,j} = \operatorname{ord}_{\bar{p}}(d_{i,j}), h = \operatorname{ord}_{\bar{p}}(d) = \min_{i,j} d_{i,j}$. We have $h = \operatorname{ord}_{\bar{p}}(d_{u,v})$ 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 $\operatorname{ord}_{\bar{p}}(\bar{d}_{0,0}) = 0$ and $b_0^{(k)}$ is randomly chosen in $\mathbf{Z}_m$, part a) of the lemma follows.

To prove part b) first write $x_i = \nu_i/\delta_i, y = \sum_{i=0}^{n-1} c_i \nu_i/\delta_i, \sigma_i = \delta/\delta_i = (\operatorname{lcm}(\delta_i)_i)/\delta_i$, where ν_i and δ_i are coprime, for all i. (We drop the superscripts k of $x_i^{(k)}, y_i^{(k)}, c_i^{(k)}, \delta^{(k)}, t^{(k)}, \delta_i^{(k)}$, and $\nu_i^{(k)}$, to simplify the notation.) Clearly, $\min_i\{\operatorname{ord}_{\bar{p}}(\sigma_i)\} = 0$ for any prime $\bar{p}$. W.l.o.g., let $\operatorname{ord}_{\bar{p}}(\sigma_0) = 0$. If $\bar{p}$ divides ν_0, then $\operatorname{ord}_{\bar{p}}(\delta_0) = 0$, so $0 = \operatorname{ord}_{\bar{p}}(\delta) \geq \operatorname{ord}_{\bar{p}}(t) \geq 0$, that is, $\operatorname{ord}_{\bar{p}}(\delta) = \operatorname{ord}_{\bar{p}}(t) = 0$. It remains to cover the case where

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

Observe that $y = \sum_{i=0}^{n-1} c_i \nu_i \sigma_i / \delta_i$, so $\operatorname{ord}_{\bar{p}}(t) = \operatorname{ord}_{\bar{p}}(\delta)$ if $\operatorname{ord}_{\bar{p}}(\sum_{i=0}^{n-1} c_i \nu_i \sigma_i) = 0$. Under (4.1), the latter equation holds with a probability of at least $\max\{1/m, 1/\bar{p}\}$ for c_0 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 $M^{-1} \bmod p$ at stage 1 (once for all k); $O((v_{M^{-1}} \mu(\log p) + v_M \mu(\log \gamma))h)$, where $h = O(n \log_p \gamma)$, in Hensel's p-adic lifting applied for each fixed k to compute $M^{-1}\mathbf{b}^{(k)} \bmod p^h$ (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 Computations in Binary Form

Performing Algorithms 3.1 and 4.1 modulo a fixed (rather than random) prime or prime power p leads to substantial benefits in practical implementation. The most desired choice is $p = 2^g$, which means binary representation. We wish to have g of the word size, and we achieve this on the average, but not for the worst case input because of the singularity problems.

Algorithms 3.1 and 4.1 can be applied with $p = 2$ if the Smith leading factor $s_n = s_n(M)$ is odd. The next algorithm extends the application to any s_n.

Algorithm 5.1. *Linear solver in binary form via small rank perturbation* (see Examples 7.2 and 7.3).

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 the 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 $n^{O(1)}, k = 1, 2, \dots, l$, and apply Algorithm 4.1 to compute Smith's leading invariant factors $s_{n,k} = s_n(M_k)$ for the matrices $M_k = M - U_k V_k, k = 0, 1, \dots, l$. Stop for the smallest l for which $s_{n,l} = s_n(M_l)$ is odd. (This l can be computed with a binary search.)

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 $\mathbf{u} = M_l^{-1}\mathbf{b}, \mathbf{v} = M_l^{-1} U_l (I_l + V_l W_l)^{-1} V_l \mathbf{u}$, 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 $M^{-1} = (M_l + UV)^{-1}$ (Golub and Van Loan [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 $k = 1, \dots, l$. The nonsingularity property, together with the (Sherman–Morrison–)Woodbury formula, implies correctness of the algorithm.

We now assume a random integer input matrix M, for which equations $s_{n-k} = 1$ are likely to hold for all but $O(\log n)$ smallest values of k [EGV00], and deduce that $l = O(\log n)$ from the above expression of $s_{n-k}(M)$ as the gcd.

To estimate the arithmetic cost of the computation, observe that the matrices M_k have displacement rank 3, so the definition of a generator of the inverse and Corollary 2.10 are extended (see the definition of the displacement rank and proofs in [P01]) to yield that $v_{M_k} \le 4m(n) + n$ and $v_{M_k^{-1}} \le 6m(n) + 2n$ for v_S in Definition 2.5.

Now, reexamination of Algorithm 4.1 (with U_k replaced by U_kJ or JU_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 (up to the factor of* $\log \log n$*) 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 $\det M$.

h steps of the generalized version of Hensel's lifting below lift the input solution vector modulo p^g to output the solution modulo p^{g+kh} where g, h, k and $p-1$ are four positive integers such that $s_n(M)/p^g$ is an integer coprime with p. The latter condition holds with a high probability for random integer matrix M and reasonably small nonnegative g. Generalized lifting combined with Algorithm 5.1 can be applied if $s_{n-l}(M)/p^g$ is an integer coprime with p. This condition is very likely to hold for random M and relatively small g and l. The generalized lifting is still performed with a quite low precision of the order of $(k + g)\log p$ bits.

Algorithm 5.3. *Lifting without coprimality* (cf. Examples 7.2 and 7.3).

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 $MQ = p^gI \bmod p^{g+k}$.

OUTPUT: $\mathbf{x}^{(h)} \in \mathbf{Z}^n$ such that $\mathbf{x}^{(h)} = p^gM^{-1}\mathbf{b} \bmod p^{g+kh}$.

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

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

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

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

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) $M\mathbf{x}^{(h)} = p^g\mathbf{b} \bmod p^{g+kh}$*;*

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) $p^g\mathbf{r}^{(i)} - M\mathbf{u}^{(i)} = (p^gI - MQ)\mathbf{r}^{(i)} \bmod p^{g+k}$, and the claim follows because $MQ = p^gI \bmod p^{g+k}$.

b) $M\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} M\mathbf{u}^{(i)}p^{ki} = \sum_{i=0}^{h-1}(p^g\mathbf{r}^{(i)} - p^{g+k}\mathbf{r}^{(i+1)})p^{ki} = p^g\mathbf{b} - p^{g+kh}\mathbf{r}^{(h)} = p^g\mathbf{b} \bmod p^{g+kh}$.

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 $O((v_M\mu(\log(n\gamma)) + v_Q\mu(\log(p^{g+k}))h)$ 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 larger than the word length, then we may shift to the matrices $M_k = M - U_kV_k$, $k = 1, 2, \ldots$, defined in Algorithm 5.1, and combine Algorithms 5.1 and 5.3 (see Example 7.3 b)). This solves the problem probabilistically except for a smaller class of matrices M, for which $s_{n-k}(M)/2^g$ is even for large k and g.

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 $n^{O(1)} \log|M|$, keeping computational precision in $O(\log(n \log|M|))$. Then the bit cost at this stage is dominated at the lifting stage. If we fix $p = 2$ or $p = 2^g$ (instead of choosing a random p), then some complications arise where $\det M$ is even, but for the average input matrix M, we handle the problem by extending the techniques of Algorithms 5.1 and 5.3 (see [Pa]).

6 Initialization of Hensel's Lifting with Variable Diagonal and Modular Continuation

Let us next show two alternative algorithms for computing generator for $M^{-1} \bmod p$ 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 $\mathbf{x}^{(l)} = (p^g M^{-1}\mathbf{b}) \bmod p^l$.

INITIALIZE: Compute the matrices $M_0 = M + p^l I$ and $Q = p^{-l}I$, so $QM_0 - I = p^{-l}M$. Write $\mathbf{z}_0 = 0, \mathbf{r}_0 = \mathbf{b}$.

COMPUTATIONS:

1. Recursively compute the vectors $\mathbf{z}_{i+1} - \mathbf{z}_i = Q\mathbf{r}_i = p^{-l}\mathbf{r}_i, \mathbf{r}_{i+1} = \mathbf{b} - M_0\mathbf{z}_{i+1} = \mathbf{r}_i - M_0 Q\mathbf{r}_i = -p^{-l}M\mathbf{r}_i, i = 0, 1, \ldots, h-1$.
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 $\mathbf{x}^{(l)} = (p^g M_0^{-1}\mathbf{b}) \bmod p^l$. 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 $p \geq 2$,

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

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

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

Therefore, every iteration step in Stage 1,

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

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 error analysis of the iterative improvement algorithm (see Skeel [S80], Higham [H96]); in our simplified case 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

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

This inequality holds for $h \geq 4n - 2 + (4n - 2 + 2\log n)/\log p^l$. 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 $|M| + |b|$ and n as follows,

$$\eta = O(nm(n)\mu(\log(|M| + |b| + 2))). \tag{6.4}$$

Let us also express η in terms of n, g and p. If initially $l \leq g$, then finally, $g \leq l < 2g$, so

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

Numerical rational roundoff at Stage 2 requires $O(\mu(\bar{d})\log\bar{d})$ bit operations per an entry of $\mathbf{z}$ for $\bar{d} = n\log|M_0| = n\log|M+p^lI| = O(nl\log p)$. To decrease the overall bit cost of the recovery 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 the vector $s_n\mathbf{z}$ 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, similar to Algorithm 6.1, first computes $M_0^{-1} \bmod q$ for a fixed prime q distinct from p and then $M_0^{-1} \bmod q^h$, $M_0^{-1}\mathbf{b}$ and $M_0^{-1}\mathbf{b} \bmod p$. Its analysis is simpler, but for $p=2$ this algorithm requires computations modulo a fixed odd prime q (say $q=3$ or $q=5$), whereas Algorithm 6.1 performs all computations with binary numbers.

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) and an integer $p \geq 2$.

OUTPUT: $g = \max_j(\mathrm{ord}_p(\delta(M^{-1}\mathbf{b})_j))$ and $(p^g M^{-1}\mathbf{b}) \bmod p$.

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 $s = p^{-1} \bmod q, t = q^{-1} \bmod p$, and the matrix $M_0 = pI + qM$, so $Q = M_0^{-1} \bmod q = sI$.
2. Apply Algorithm 3.1 for M and p replaced by M_0 and q, respectively, to compute $M_0^{-1}\mathbf{b} \bmod q^h$. 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 $p^g(M_0^{-1}\mathbf{b}) \bmod p = (tp^gM^{-1}\mathbf{b}) \bmod p$.

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 $M^{-1} \bmod p$. 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.

7 Examples

Example 7.1. $M = \begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, so $\mathbf{x} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$. By applying Algorithm 3.1 for $p=2$, $\mathbf{r}^{(0)} = \mathbf{b}$, we successively compute $Q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \mathbf{u}^{(0)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \mathbf{r}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{u}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{r}^{(2)} = \begin{pmatrix} -1 \\ -2 \end{pmatrix}, \mathbf{u}^{(2)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ldots$. So, $\mathbf{x}^{(3)} = 2\mathbf{x} \bmod 8 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} + 2\begin{pmatrix} 1 \\ 1 \end{pmatrix} + 4\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Example 7.2. $M = \begin{pmatrix} 4 & 1 \\ 6 & 2 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, so $\mathbf{x} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

a) By applying Algorithm 5.3 for $p=2$, $g=k=1$, $\mathbf{r}^{(0)} = \mathbf{b}$, we successively compute $Q = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}, \mathbf{u}^{(0)} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \mathbf{r}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{u}^{(1)} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \mathbf{r}^{(2)} = \begin{pmatrix} -1 \\ -2 \end{pmatrix}, \mathbf{u}^{(2)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \ldots$. So, $\mathbf{x}^{(3)} = 2\mathbf{x} \bmod 8 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} + 2\begin{pmatrix} 1 \\ 2 \end{pmatrix} + 4\begin{pmatrix} 2 \\ 2 \end{pmatrix}$, $(M\mathbf{x}^{(h)} - 2\mathbf{b}) \bmod 2^{h+1} = 0$ for $h = 1,2,3$.

b) Alternatively, by observing that $s_2(M) = 2$, $s_1(M) = 1$ and applying Algorithm 5.1 to $M_1 = M - U_1V_1$, $U_1 = V_1^T = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and $\mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, we reduce computation of $\mathbf{x}$ to applying Algorithm 3.1 three times (according to Sherman-Morrison-Woodbury formula), with the right-hand-side vectors $\mathbf{b} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, $\mathbf{b}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and $\mathbf{b}^{(2)} = (1/3)\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} M_1^{-1}\begin{pmatrix} 3 \\ 4 \end{pmatrix}$, respectively. We have $M_1 = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix}, M_1^{-1}\begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, M_1^{-1}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/3 \\ -2/3 \end{pmatrix}$, (these vectors can be computed by applying Algorithm 3.1, we omit the details), so $\mathbf{b}^{(2)} = \mathbf{0}, M_1^{-1}\mathbf{b}^{(2)} = \mathbf{0}, M^{-1}\mathbf{b} = M_1^{-1}\mathbf{b} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

Example 7.3. $M = \begin{pmatrix} 32 & 2 \\ 48 & 4 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 24 \\ 32 \end{pmatrix}$. So, $\mathbf{x} = \begin{pmatrix} 1 \\ -4 \end{pmatrix}, s_2(M) = 32, s_1(M) = 2$. We may

a) apply Algorithm 5.3 to M and $\mathbf{b}$ for $p = 2, g = 5, k = 1$ or

b) apply Algorithm 5.1 to $M_1 = M - U_1V_1, U_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, V_1^T = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, M_1 = \begin{pmatrix} 30 & 0 \\ 46 & 2 \end{pmatrix}$. For solving the equations $M_1^{-1}\mathbf{b}^{(i)}, i = 1, 2, 3$ (cf. Example 7.2 b)), apply Algorithm 5.3 for $p = 2, g = k = 1$.

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.

[Pa] V. Y. Pan, Randomized and Derandomized Singular Toeplitz/Hankel-like Computations, preprint, 2002.

[PW02] V. Y. Pan, X. Wang, Acceleration of Euclidean Algorithm and Extensions, *Proc. Intern. Symposium on Symb. and Algebraic Computation (ISSAC'2002)*, 207-213, ACM Press, New York, 2002.

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

[T94] E. E. Tyrtyshnikov, How Bad Are Hankel Matrices? *Numerische Mathematik*, **67, 2**, 261–269, 1994.

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