2002

TR-2002017: Nearly Optimal Toeplitz/Hankel Computations

Victor Y. Pan

Nearly Optimal Toeplitz/Hankel Computations *

Victor 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 signal processing. We study this subject as a computer algebra problem assuming a nonsingular integer input and rely on Hensel's p-adic lifting to improve the current fastest divide-and-conquer algorithm by Morf 1974/1980 and Bitmead and Anderson 1980. With randomization, lifting enables solution of 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 $n^2 \log n$), 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 the input coefficients are integers in $n^{O(1)}$. Furthermore, we extend our algorithms and cost bound to q-adic lifting for a fixed integer q, in particular $q = 2^g$ allowing computations in binary form. This enhances practical value of our study. In Pan 2002 (Pa) the algorithms and the bit cost estimates are extended to solving nonsingular and consistent singular Toeplitz/Hankel-like linear systems and computing the rank and a vector from or a basis for the null space of a Toeplitz/Hankel-like matrix as well as the resultant, gcd and lcm of two polynomials of bounded degrees, a fixed entry of a Padé approximation table, and the Berlekamp-Massey linear recurrence coefficients provided that the input values are some bounded integers. To yield this extension, Hensel's lifting is combined with the Morf-Bitmead-Anderson's divide-and-conquer algorithm, where again we allow computations both modulo a random prime and a fixed prime power, in particular a power of 2.

2000 Math. Subject Classification: 68W30, 68W20, 65F05, 68Q25

*Supported by NSF Grant CCR 9732206 and PSC CUNY Awards 66383-0032 and 64406-0033

Key Words: Toeplitz matrices, Hankel matrices, solving linear systems, Hensel's lifting, computations in rings.

Acknowledgement: I thank Mark Giesbrecht and Arne Storjohann for (p)reprints of their papers and helpful comments.

0. Introduction

0.1. Toeplitz/Hankel computations

Toeplitz and Hankel and, more generally, Toeplitz/Hankel-like matrices (with the structure of Toeplitz/Hankel types) are omnipresent in computations in sciences, engineering, and signal processing. 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 Kailath and Sayed (eds.) 1999 (KS99), Pan 2000 (P00, Section 1.1) and Pan 2001 (P01)). The displacement transformation approach of Pan 1990 (P90) enables reduction of computations with matrices having structures of Cauchy, Vandermonde and other types to the Toeplitz/Hankel-like case. Moreover (see Brent et al. (BGY80), (P01)), the solution of a Toeplitz/Hankel linear system is equivalent to the computation of polynomial gcd/lcm as well as 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 (see von zur Gather and Gerhard 1999 (GG99)). Berlekamp-Massey's problem of recovering the coefficients of a linear recurrence is another celebrated and intensively studied equivalent formulation of the same problem.

Exploiting matrix structure enables a dramatic decrease of the solution cost, 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 two "fast algorithms", by Levinson 1947 and Durbin 1959 and by Trench 1964, and further to $O(n \log^2 n)$ in the two "superfast algorithms" by Brent et al. 1980 (BGY80) and by Morf 1974/1980 (M74), (M80) and Bitmead/Anderson 1980 (BA80). Hereafter we refer to the latter algorithms as the BGY and MBA algorithms, respectively. More precisely, these algorithms use $O(m(n) \log n)$ flops where $m(n)$ denotes the arithmetic cost of multiplying two polynomials of degree $n-1$, $m(n) \geq 2n-1$ (an information lower bound),

$$m(n) \leq c_{class} n^2, \quad m(n) \leq c_k n^{\log 3}, \quad m(n) \leq (c_{ck} n \log n) \log\log n, \tag{0.1}$$

c_{class}, c_k and c_{ck} are three constants, $0 < c_{class} < c_k < c_{ck}$, and the above bounds are supported by the classical, Karatsuba's, and Cantor and Kaltofen's

algorithms (see Bernstein, to appear (Ba)). Here and hereafter log stands for $\log_2$ unless specified otherwise.

The BGY algorithm applies to Toeplitz/Hankel linear systems, whereas the MBA algorithm covers also the solution of the more general class of Toeplitz/Hankel-like linear systems (KS99), (P01), and involves their solution as a part of the algorithm even where it applies to Toeplitz/Hankel systems.

0.2. The bit complexity model, CRA and rational number reconstruction

Our goal is to optimize the Toeplitz/Hankel and Toeplitz/Hankel-like computations under the more realistic bit complexity model. Under this model, to each arithmetic operation performed over integers modulo q, that is, with d-bit precision, for $d = \lceil \log_2 q \rceil$, one assigns the cost of $\mu(d)$ bit operations, where $\mu(d) \geq 2d - 2$ (an information lower bound),

$$\mu(d) \leq C_{class} d^2, \quad \mu(d) \leq C_k d^{\log 3}, \quad \mu(d) \leq (C_{ss} d \log d) \log\log d, \tag{0.2}$$

$\log 3 = 1.58496\ldots$, C_{class}, C_k and C_{ss} are three constants, $0 < C_{class} < C_k < C_{ss}$, and the above bounds are supported by the classical algorithm and those of Karutsuba 1963 and Schönhage and Strassen 1971 (see (Ba)).

The most popular algebraic way of controlling the precision length 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 $p_1, \ldots, p_s$. From the integers $(\nu/\delta) \bmod p_i$, $i = 1, \ldots, s$, the output values ν/δ, $\delta \geq 1$, are recovered uniquely, at 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), Pan and Wang 2002 (PW02) and (WPa)). To ensure uniqueness, sufficiently many primes are chosen so that the product $p = p_1 \cdots p_s$ exceeds $2\delta|\nu|$ for every rational output value η/δ. The stage of rational number reconstruction is generally considered quite hard, but it is simple for the MBA algorithm, which computes $\det M$ as by-product. $(\det M)\mathbf{x}$ is an integer vector, and its reconstruction from $(\det M)\mathbf{x} \bmod p$ is cost-free.

To simplify our bit cost estimates, we assume in the introduction that all input values are integers in $n^{O(1)}$, and in the paper after stating our detailed estimates in the general form, we usually restate them in a simplified form (under similar simplifying assumptions).

0.3. Hensel's lifting versus the MBA algorithm

Applying the MBA algorithm (with the CRA) we have to avoid divisions by 0, which occur if some leading principal (northwestern) submatrices of the input matrix M are singular. The known recipes Kaltofen and Saunders 1991 (KS91), (P01) work for general matrices M and random primes supporting the CRA, and we have to extend these recipes in two ways: for computations with

structured matrices represented by their displacement generators and for computations modulo a fixed integer q. The choice of a fixed integer q (rather than a random prime) is practically preferred, particularly with $q = 2^g$, which means binary computations. We address these issues in the companion paper Pan 2002 (Pa), but next we shift to a distinct approach where the problems become simpler, that is, we rely on Hensel's lifting in Moenck and Carter 1979 (MC79) and Dixon 1982 (D82). This implies several advantages: only a single random prime in $n^{O(1)}$ and a pair of $n \times n$ matrices (instead of many primes and auxiliary matrices of various sizes) are sufficient throughout the computations, the singularity problem is restricted to a single input matrix, and the bit operation cost bound decreases by the factor of $\log n$ to yield the bound of

$$B = O(nm(n)\mu(\log n)) \tag{0.3}$$

(see (6.11) in Section 6) to be compared with $n^2 \log n$ bits, generally required already to represent the n output values, each with up to $n \log n$ bits.

To support this upper bound and the transition to the ring $\mathbb{Z}_q$ of integers modulo a fixed integer q, however, various issues must be addressed such as the initialization of lifting, singularity of the input matrix in $\mathbb{Z}_q$, and the recovery of the rational output from its computed p-adic expansion. For the latter problem, we use the known recipes, due to Pan 1988 (P88), Cooperman et al. 1999 (CFG99), and Mulders and Storjohann, to appear (MSa), but addressing the two former issues leads us to comprehensive study, which combines various old and new techniques. The latter ones, of independent interests, include extension of Hensel's lifting to $\mathbb{Z}_q$ where $\det M$ is not coprime with q, the variable diagonal and the modular continuation techniques for the initialization of lifting, and a small rank random additive Toeplitz perturbation for avoiding singularities in $\mathbb{Z}_q$. We also supply some probabilistic estimates for degeneracy, including a shorter proof of a main result in Kaltofen and Lobo 1996 (KL96).

Our cost estimates for the solution of an integer Toeplitz linear system $M\mathbf{x} = \mathbf{b}$ with computations modulo p^{g+k} for a fixed prime p (e.g., $p = 2$) depend on positive integers k and $g+k$, which for practical computations we should choose as small as possible provided that M/p^g has the inverse modulo p^{g+k}. Statistically (see (Wa)), the value $g + k$ tends to be not large for a random M, and then the nearly optimal randomized bit cost bound in (0.3) can be extended. In the unlikely case where a large $g+k$ is required to avoid degeneracy ($g+k$ is always in $O(\log_p |\det M|)$), the bit cost bound grows roughly proportionally to $\mu(g+k)/k$, but at most to the level of

$$B = O(m(n)\mu(n \log n)) \tag{0.4}$$

(see (6.10) in Section 6), and we yield this bound performing all computations modulo a power of a fixed prime p, e.g., $p = 2$, except of course, for the final recovery stage. Note that the information lower bound of $n^2 \log n$ bits (required to represent the output) lies within the factor of $m(n)\mu(n)/n^2$ from B. This

factor is quite small if $m(n)$ and $\mu(n)$ are nearly linear (see (0.1), (0.2)). For large n the estimate (0.4) is only interesting if we perform multiplication of integers and polynomials in nearly linear time (see (0.1), (0.2)). This can be a problem in practice, and then keeping k small and imposing a reasonably small upper bound g_+ on g is the best strategy. It is expected that g rarely exceeds g_+, but if this occurs, the computations can be repeated with a distinct (fixed or random) prime p.

Taking into account the significance of the problems, long and intensively attacked by a huge army of researchers, it is surprising to achieve progress by rather elementary and simple means. So we seek credit first of all for finding a surprisingly simple path to progress in two important areas (of Toeplitz/Hankel solving and lifting modulo a nonprime base q). We also note that this work substantially advances our paper (PW02) in the ISSAC'02 proceedings; the overlap is essentially confined to the first four sections of the present paper, whereas its technically and conceptually more innovative part of the extension to the rings $\mathbb{Z}_{p^g}$ and the initialization of lifting was not in (PW02) yet.

0.4. Some extensions

Hensel's lifting can be rather easily extended to a Toeplitz/Hankel-like nonsingular input. Hybrid algorithms combining lifting with the MBA algorithm handle singular input as well. In this case we process many auxiliary matrices but other advantages of lifting versus the MBA algorithm are preserved; in particular the MBA stage is only used modulo a single moderately large integer (to initialize lifting), so its bit cost is dominated at the lifting stage. Many technical details of this approach are elaborated upon in (Pa), in particular, solving the singularity problems in $\mathbb{Z}_q$ (via recursive scaling) and compression of displacement generators in $\mathbb{Z}_q$. The hybrid algorithm is immediately extended to computing the matrix rank and a vector from or a basis for the null space of a Toeplitz/Hankel-like matrix and whence to computing the gcd, lcm, and resultant of univariate polynomials, their Padé approximations, and the solution of Berlekamp-Massey's problem.

The algorithms and computational cost estimates can be extended easily to other classes of input matrix M. The computaional cost is proportional to the arithmetic cost of multiplication of M and M^{-1} by a vector, which is also small for several other classes of structured matrices, e.g., of Vandermonde and Cauchy–Pick types. The latter matrices of Cauchy–Pick type have rational entries, and to avoid possible degeneration in the reduction modulo a fixed prime power, it is probably most promising to examine the cited approach of the displacement transformation to the Toeplitz/Hankel-like case (P90), (P01) followed by rounding and scaling.

To apply the algorithms in this paper and (Pa) to general matrix, we only need to remove the block of compression of the matrices using their displacement generators. This simplifies the study substantially but increases the bit cost by

the factor of $n^2/\log^2 n$ (or, theoretically, a little less in the asymptotics based on fast matrix multiplication).

Hensel's lifting can be replaced by Newton's; furthermore, our accelerated generalized version of Hensel's lifting yields the same effect (see our Remark in Section 5), that is, we proceed with by the factor of $n/\log n$ fewer lifting steps but a higher precision of computing, in particular the precision increases by the same factor at the final lifting step. Overall, this may be an advantage for parallel acceleration (P00) but no advantage for sequential computation, except may be in the variant of the high order Newton's lifting proposed by Storjohann in 2002 (S02), the latter distinct approach so far was unrelated to ours (see our Remark at the end of Section 6).

Theoretical support of our statistical data in (Wa) is an interesting technical challenge, but even more important directions for further study are the experimental tests and implementation of our algorithms in the rings $\mathbb{Z}_q$ for $q = p^g$ and a fixed p, particularly for $p = 2$ (including their extensions to gcd, lcm, etc.).

Let us also briefly comment 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 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 and Paterson 1974 (FP74) (cf. (BP94, Section 3.9)) could yield theoretical acceleration by the factor of $(\log\log n)\log\log n$. The resulting cost bound of $O(n\mu(n\log n))$, however, does not seem to be attractive unless n is huge because the overhead constant C_{ss} is large, whereas with C_{class} and C_k in (0.1) the overall bit cost bounds become N^α for $\alpha > 2.5$.

0.5. Organization of our paper

We organize our paper as follows. We state definitions and preliminary results in the next two sections. We recall and then modify Hensel's lifting algorithm for a linear system of equations in Sections 3–5. In Section 6 we estimate the probability that the modified lifting does not degenerate for a random integer input. In Section 7, we apply the variable diagonal and modular continuation techniques to initialize lifting. In Section 8, we apply random additive Toeplitz perturbation (cf. Villard 2000 (V00) and Eberly et al. 2000 (EGV00)) to avoid degeneration. In Appendix A, we briefly comment on an extension to computing the determinant and the leading Smith factor of a Toeplitz matrix. In (Wa) we present the results of our experimental estimates of how frequently a random integer Toeplitz matrix is a) singular and b) strongly nonsingular in $\mathbb{Z}_{2^g}$, that is, modulo 2^g. These estimates motivate our extension to $\mathbb{Z}_{2^g}$ of a) Hensel's lifting

in this paper and b) the hybrid of lifting with the MBA algorithm in (Pa). (Wa) is due to the second author, all other parts of the paper to the first author.

Acknowledgements. Our thanks go to Mark Giesbrecht and Arne Storjohann for (p)reprints of their papers and helpful comments and to Richard Isaac for suggesting format for statistical tests.

1. Definitions and Basic Facts: Rational Number Reconstruction

Definition: $\mathbb{Z}$ is the ring of integers, $\mathbb{Z}_q$ is the ring of integers modulo q, $\mathbb{Q}$ is the field of rational numbers. For $z, q \in \mathbb{Z}$, $q > 1$, we define $z \bmod q$ as a unique integer 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 $\delta(y)$.

Hereafter, we call by the *modular rational roundoff* the recovery of 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, $1 \le k \le q$, $|x| < k$ and $0 < y \le q/k$. $\rho(q)$ denotes the bit-operation cost of this recovery. Clearly we may write $x = r, y = 1$ if $k > |r|$. The pair x, y is unique under the additional assumption that $2|x| < k$ (GG99). In this paper we recover the rational coordinates x/y of the solution to a linear system of equations, where two coprimes $|x|$ and y are bounded from above based on Fact 2.1 in the next section. We choose q and k such that $2|x| < k$, so the solution is unique.

Likewise, we call by the *numerical rational roundoff*, 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 integers ν, δ and k, where $1 \le |\nu| < \delta$. $\bar{\rho}(\delta)$ denotes the bit-operation cost of the recovery.

Both of the recovery problems can be solved by applying the *extended Euclidean algorithm* (hereafter referred to as the *EEA*) to the input pair r_0, r_1 being q, r or δ, ν, respectively, $|r_1| < |r_0|$, and stopping for the smallest positive i such that $r_i < k$ in the remainder sequence $r_0, r_1, r_2, \ldots$. The remainder sequence can be complemented by two dual sequences of *convergents*, denoted $s_0, s_1, s_2, \ldots$ and $t_0, t_1, t_2, \ldots$ in (GG99) (cf. also Schrijver 1986 (S86) and Zippel 1993 (Z93)); for both problems of modular and numerical rational roundoff, the desired rational solution can be easily obtained from the two triples $r_{i-1}, s_{i-1}, t_{i-1}$ and r_i, s_i, t_i, each made up of a remainder and two convergents (here again we use the notation in (GG99)).

Hereafter, $\sigma(r_0)$ denotes the bit operation cost of computing the above triples for given values of r_0, r_1 and k. We have (GG99), (S86), (Z93):

$$\rho(q) \le \sigma(q) + O(q), \quad \bar{\rho}(\delta) \le \sigma(\delta), \tag{1.1}$$

$$\sigma(q) \le cd^2, \tag{1.2}$$

where $d = \log q$ and c is a constant. The asymptotic bound (1.2) can be improved as follows.

THEOREM 1.1: *(PW02), (WPa).*

$$\sigma(q) \le C\mu(d)\log d, \quad d = \log q \tag{1.3}$$

for $\mu(d)$ in (0.2) and a positive constant C.

Presently the practical choice is the classical algorithm supporting (1.2). The recent algorithms in (PW02) and (WPa) may eventually become competitive, although only for larger values of d because C exceeds c.

2. Definitions and Basic Facts: Toeplitz and Hankel Matrices

Definition: $M = (m_{i,j})_{i,j=0}^{k-1,l-1}$ is a $k \times l$ matrix with rational or integer entries $m_{i,j}$; $M \in \mathbb{Q}^{k\times l}$ or $M \in \mathbb{Z}^{k\times l}$, respectively. I and 0 are the indentity and null matrices of the proper sizes, I_l is the $l \times l$ identity matrix. M^T is the transpose of M.

Definition: $\det M$ and $\operatorname{adj} M$ denote the determinant and adjoint (adjugate) of a $k \times k$ 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.

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

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

Definition: 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 $M \in \mathbb{Z}^{n\times n}$. 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 $k = 1, \dots, n$.

Definition: 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 $t_{i+1,j+1}$. $Z(\mathbf{v})$ 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 $\mathbf{v} = (v_{i,j})$, that is, $J\mathbf{v} = (v_{n-i-1})_{i=0}^{n-1}$, $J^2 = I$.

The next well known estimate is an overestimate on the average, according to Abbott et al. 1999 (ABM99).

FACT 2.1: $|\det M| \le |M|^n$ *and* $|\operatorname{adj} M| \le n|M|^{n-1}$ *for an* $n \times n$ *matrix* M.

It is easily deduced that $s_1, \dots, s_n \in \mathbb{Z}$ and $|\det M| = s_1 \cdots s_n$, so

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

Clearly, for any Toeplitz matrix T there exist non-unique pairs $(Z(\mathbf{w}), Z(\mathbf{x}))$ such that $T = Z(\mathbf{w}) + Z^T(\mathbf{x})$, so T is defined by its first row and first column. 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 are immediately reduced to each other. *We only specify the 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.

THEOREM 2.1: *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 degree* $n - 1$.

COROLLARY 2.1: *An* $n \times n$ *Toeplitz matrix* T *can be multiplied by a vector in* $2m(n)$*arithmetic operations for* $m(n)$ *in* (0.1)*; the bound decreases to* $m(n)$ *if* T *is a triangular matrix.*

The next theorem of Heinig 1979 (H79) extends the Gohberg–Semencul formula of 1972.

THEOREM 2.2: *Let* $T = (t_{i,j})_{i,j=0}^{n-1}$ *be a 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}. The next theorem is easy to verify.

THEOREM 2.3: $4m(n) + n$ *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 2.2.*

COROLLARY 2.2: *Representing* T^{-1} *via the generator* $\mathbf{p}, \mathbf{v}$ *requires* $2n$ *units of memory (versus the order of* n^2 *entries) and enables fast multiplication of* T^{-1} *by a vector.*

3. Hensel's lifting for linear systems

Given a prime p, a matrix M, a vector $\mathbf{b}$, and (the generator for) the first term in the p-adic expansion of M^{-1}, the next algorithm computes the first h terms in the p-adic expansion of $M^{-1}\mathbf{b}$.

ALGORITHM 3.1: *Hensel's lifting for a linear system (MC79), (D82).*

INPUT: $M \in \mathbb{Z}^{n\times n}$, *an integer* p *coprime with* $\det M$, $\mathbf{b} \in \mathbb{Z}^n$, *an integer* $h > 1$, *and* $Q = M^{-1} \bmod p$ *(see Section 6 and (Pa) on computing* Q*).*

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

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

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

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

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

THEOREM 3.1: *(D82).*

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

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

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*

$$\gamma = \max_{i,j} \max\{p, |m_{i,j}|, |b_j|\}. \tag{3.1}$$

Algorithm 3.1 uses $O((v_M + v_Q)h\mu(\log(n\gamma)))$ bit operations, for $\mu(d)$ in (0.2) and v_S defined in Section 2, and outputs $\mathbf{x}^{(h)}$ in p-adic form.

THEOREM 3.2: *It is sufficient to choose*

$$h = \lceil 2n \log_p(\gamma n) \rceil \tag{3.2}$$

in Algorithm 3.1 and to perform

$$B_0 = O(n\rho(q)) \tag{3.3}$$

bit operations, for $\rho(q)$ *in (1.1)–(1.3) and* $q = p^h < (\gamma_n)^{2n} + 1$, *to recover a unique solution* $\mathbf{x} = M^{-1}\mathbf{b}$ *to the linear system* $M\mathbf{x} = \mathbf{b}$ *from the vector*

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

Proof: According to Section 1, 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 $\mathbf{x} = (x_j)_j = M^{-1}\mathbf{b}$ if $q = p^h > |\nu_j|\delta_j$ and $2|\nu_j| < k \le p^h$. To bound ν_j and δ_j, recall Fact 2.1, so we may choose $k = 2n|M|^{n-1}|\mathbf{b}|$ and any $q > 2n|M|^{2n-1}|\mathbf{b}|$. Therefore, every component x_j can be recovered from $x_j \bmod p^h$ if $p^h > 2n(\gamma n)^{2n} \ge 2n|M|^{2n-1}|\mathbf{b}|$, that is, if $h > \log_p(2n) + 2n\log_p(\gamma n)$. Now Theorem 1.1 supports the claimed bit cost bound for the recovery of $\mathbf{x}$ from $\mathbf{x}^{(h)}$. □

THEOREM 3.3: *Fix h of (3.2). Then Algorithm 3.1 performs*

$$\begin{aligned} B_1 &= O((v_Q\mu(\log p) + v_M\mu(\log(|M|np)) + n\log(n\gamma))h) \\ &= O((v_M + v_Q)n\mu(\log(n\gamma))\log_p(\gamma n)) \end{aligned}$$

bit operations to output $\mathbf{x}^{(h)} = \mathbf{x} \bmod p^h$ in the p-adic form for $\mu(d)$ in (0.2), γ in (3.1), and v_S defined in Section 2. If

$$v_M = O(m(n)), \quad v_Q = O(m(n)), \quad \log\gamma = O(\log n), \tag{3.4}$$

then

$$B_1 = O(m(n)n\mu(\log n)\log_p n)$$

for $m(n)$ in (0.1) where $1/\log_p n = O(1)$ due to (3.1) and (3.4).

Hereafter let $\mathbf{b} \ne \mathbf{0}$, $n \ge 2$, $|M| \ge 2$, so $\log n \ge 1$, $\log|M| \ge 1$.

4. Faster recovery of the rational solution

Hereafter we write $l = \mathrm{ord}_p(z)$ if $p, z \in \mathbb{Z}$, p^l divides z but p^{l+1} does not.

In this section we apply some known randomization techniques to accelerate the recovery of $\mathbf{x}$ from $\mathbf{x}^{(h)}$ by the factor of $\rho(q)/\mu(d)$ for $\mu(d)$ in (0.2) and $\rho(q)$ in (1.1)–(1.3). If both $\rho(q)$ and $\mu(d)$ are of the order of d^2, then this is just a constant factor. With $\mu(d)$ of a smaller order in $O(d^{\log 3})$ or $O((d\log d)\log\log d)$ and $\rho(q)$ in (1.1), (1.3), this factor grows within $O(\log d)$. The acceleration relies on two observations:

(a) The vector $\mathbf{y} = \delta\mathbf{x}$ is in $\mathbb{Z}^n$ where

$$\delta = \mathrm{lcm}_i\,\delta(x_i), \quad 0 \le i \le n-1, \tag{4.1}$$

is the least common denominator of all coordinates x_i of the solution $\mathbf{x} = (x_i)$ to the system $M\mathbf{x} = \mathbf{b}$, whereas the recovery of $\mathbf{y}$ from $\mathbf{y} \bmod p^h$ is immediate if $p^h > 2\delta|\mathbf{x}| = 2|\mathbf{y}|$. Since $\delta \le s_1(M) \le |\det M| \le |M|^n$ (see (2.1)), it is sufficient to double h of (3.2), which little affects the overall bit cost bounds. Multiplication of $\mathbf{x}$ by δ, however, requires the order of $n\mu(d)$ bit operations, thus limiting the gain versus $B_0 = O(n\rho(q))$ in (3.3).

(b) Computation of δ can be accelerated with randomization, because δ is likely to equal the least common denominator of a smaller number $K < n$ of random linear combinations $\mathbf{c}_k^T\mathbf{x}$ of the coordinates $x_1, \dots, x_n$ of $\mathbf{x}$.

The approach can be traced back to (P88, Section 6). Its recent studies include (CFG99) and (MSa). The next algorithm specifies Hensel's lifting with such a randomized recovery.

ALGORITHM 4.1: *Hensel's lifting with randomized recovery of the rational output.*

INPUT: *a positive ε, otherwise as in Algorithm 3.1.*

OUTPUT: *FAILURE with a probability of at most ε or a positive integer δ and an integer vector $\mathbf{y}$ such that*

$$M\mathbf{y} = \delta\mathbf{b}. \tag{4.2}$$

INITIALIZATION: *compute*

$$K = 2\lceil \log(1/\varepsilon)\rceil, \tag{4.3}$$
$$m = \max\{\sqrt{n\log|M|}, 4000\}, \tag{4.4}$$
$$h = 1 + \lfloor \log_p(n|M|^{2n-1}m|\mathbf{b}|)\rfloor. \tag{4.5}$$

Then sample K random vectors

$$\mathbf{c}_k = (c_{ik})_{i=0}^{n-1} \in \mathbb{Z}_m^n, \quad k = 1, \dots, K. \tag{4.6}$$

COMPUTATIONS:

1. *Apply Algorithm 3.1 to compute $\mathbf{x}^{(h)} = (x_i^{(h)})_{i=0}^{n-1}$.*
2. *Compute the K integers*

$$v_k = \mathbf{c}_k^T\mathbf{x}^{(h)} = \sum_{i=0}^{n-1} c_{ik}x_i^{(h)}, \quad k = 1, \dots, K.$$

3. *Recover a unique set of integers δ_k such that*

$$\begin{gathered}(\eta_k/\delta_k) \bmod p^h = v_k; \quad \gcd(\eta_k, \delta_k) = 1,\\ 1 \le \delta_k \le |M|^n, \quad |\eta_k| < |M|^{n-1}m|\mathbf{b}|, \quad k = 1, \dots, K.\end{gathered} \tag{4.7}$$

4. *Compute the least common denominator*

$$\delta_{l.c.d.} = \operatorname{lcm}_k \delta_k, \quad 1 \le k \le K. \tag{4.8}$$

5. *Compute $\mathbf{y} = \delta_{l.c.d.}\mathbf{x}^{(h)} \bmod p^h$. Write $\delta = \delta_{l.c.d.}$. If (4.2) holds, output $\mathbf{y}$ and δ; otherwise output FAILURE.*

Equations (4.4)–(4.6) and Fact 2.1 ensure uniqueness of coprimes δ_k and η_k satisfying (4.7). Now, correctness of Algorithm 4.1 follows from the next result.

THEOREM 4.1: *$\delta_{l.c.d.}$ in (4.8) divides δ in (4.1). Furthermore,*

$$\text{Probability}(\delta_{l.c.d.} \neq \delta) \leq \varepsilon.$$

Theorem 4.1 is deduced similarly to Theorem 2.1 in (EGV00) based on (4.3)–(4.7) and the next lemma.

LEMMA 4.1: *For a prime p, an integer k, $1 \leq k \leq K$, δ in (4.1), and δ_k in (4.7), we have*

$$\text{Probability}(\text{ord}_p(\delta_k) < \text{ord}_p(\delta)) \leq \max\{\tfrac{1}{p}, \tfrac{1}{m}\}.$$

Proof: Let $l = \text{ord}_p(\delta) = \max_i \text{ord}_p(\delta(x_i))$ for $0 \leq i \leq n-1$. W.l.o.g., let $l = \text{ord}_p(\delta(x_0))$ and let c denote the first coordinate of the vector $\mathbf{c} = \mathbf{c}_k$. Then we have

$$\mathbf{c}^T\mathbf{x} = \frac{cu}{ap^l} - \frac{v}{p^h b} = \frac{cub - avp^{l-h}}{abp^l}$$

where $l \geq h$ and a, b, u and v are integers coprime with p. Clearly, $\text{ord}_p(\delta_k)$ for δ_k in (4.7) never exceeds l; it equals l if and only if $cub - avp^{l-h}$ is coprime with p. Since ub is coprime with p and c is random, the probability bound follows. □

We state some detailed bit cost bounds in the next theorem and in a more observable form in Corollary 4.1.

THEOREM 4.2: *Define $\varepsilon > 0$, $\mu(d)$ in (0.2), $d = \log(|M|^n|\mathbf{b}|)$, $h = O(d/\log p)$ in (4.5), and γ, v_M and v_Q in (3.4). Then Algorithm 4.1 fails with a probability of at most ε and otherwise correctly computes a rational vector $\mathbf{x} = M^{-1}\mathbf{b}$. The algorithm generates $R = O((n\log(1/\varepsilon))\log(n\log|M|))$ random bits and in addition performs B_i bit operations at stage (i), $i = 1, \ldots, 5$, where B_1 represents the bit cost of lifting and is bounded in Theorem 3.3, $B_0 = B_2 + B_3 + B_4 + B_5$ represents the rational reconstruction bit cost, $B_2 = O((nd\mu(\log d)/\log d)\log(1/\varepsilon))$, $B_3 + B_4 = O((\mu(d)\log d)\log(1/\varepsilon))$, and $B_5 = O(n\mu(d))$.*

COROLLARY 4.1: *Let ε, M, Q and p be as in Algorithm 3.1, let (3.1) and (3.4) hold, so $1/\log_p n = O(1)$, and let*

$$\log(1/\varepsilon) = O(\log n), \tag{4.9}$$

$$\mu(n) = O(m(n)) \tag{4.10}$$

for $m(n)$ in (0.1) and $\mu(d)$ in (0.2). Then Algorithm 4.1 outputs FAILURE with a probability of at most ε and otherwise correctly computes the solution $\mathbf{x} = M^{-1}\mathbf{b}$ to a linear system $M\mathbf{x} = \mathbf{b}$; the algorithm requires generating $R = O((n\log n)\log(1/\varepsilon)) = O(n\log^2 n)$ random bits and in addition performing

$$B = O(m(n)n\mu(\log n)\log_p n) \tag{4.11}$$

bit operations.

Proof: Let $d = \log(|M|^n|\mathbf{b}|)$ as in Theorem 4.2. Then (3.3) and (4.9) together imply that

$$(\log d)\log(1/\varepsilon) = O(\log^2 n), \quad d\mu(\log d)\log(1/\varepsilon) = O(\mu(d),$$

and consequently,

$$B_0 = B_2 + B_3 + B_4 + B_5 = O(n\mu(d)). \tag{4.12}$$

It remains to show that $n\mu(d)$ is in $O(m(n)n\mu(\log n)\log_p n)$, which bounds B_1 in Theorem 3.3. This bound on $n\mu(d)$ follows from (4.10) and the equations $d = O(n\log n)$ and $\log p = O(\log n)$ both implied by (3.4). □

5. Generalized lifting

Algorithm 3.1 works under the assumption that $\det M \neq 0$ in $\mathbb{Z}_p$, required for the existence of the input matrix $Q = M^{-1}$ in $\mathbb{Z}_p$. Let us extend the algorithm to allow the following weaker initial assumption:

$$MQ = p^g I \bmod p^{g+k}, \tag{5.1}$$

where g and $k > 0$ are two fixed integers of our choice, the number of lifting steps decreases by the factor of k, whereas the precision of computing increases by $\lceil (k-1)\log p \rceil$ bits as k increases from 1,

$$g \geq g_- = g_p(M) = \mathrm{ord}_p(s_n(M)) = \max_{i,j} \mathrm{ord}_p(\delta((M^{-1})_{i,j})), \tag{5.2}$$

$$g_- \leq \log_p s_n(M) \leq n\log_p |M|$$

(see (5.3) and the last Remark in this section). For $g = 0$, $k = 1$ we come back to Algorithm 3.1; our algorithms in the next section as well as in (Pa) compute Q, g and $g + k$ satisfying (5.1).

Hereafter we write $Q = p^g M^{-1} \bmod p^{g+k}$ if (5.1) and (5.2) hold, so Q/p^g is the first term in the p^k-adic expansion of M^{-1}. Given (a generator for) this term, the next algorithm computes the first h terms in the p^k-adic expansion of $M^{-1}\mathbf{b}$.

ALGORITHM 5.1: *Generalized lifting (see Examples 9.2 and 9.3).*

INPUT: *$M \in \mathbb{Z}^n$, $\mathbf{b} \in \mathbb{Z}^n$, a prime p, the integer $g = g_p(M)$ in (5.2), two positive integers h and k, and matrix $Q = p^g M^{-1} \bmod p^{g+k}$ satisfying (5.1).*

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

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

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

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

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

For $p = 2$, Algorithm 5.1 operates with binary values. For $g = 0$ and $k = 1$, it turns into Algorithm 3.1. Note that

$$p^{g_-} \le s_n(M) \le |\det M| \le |M|^n \tag{5.3}$$

due to (5.2) and (2.1), so we assume that

$$g = O(g_-) = O(n \log_p |M|). \tag{5.4}$$

THEOREM 5.1: *(Cf. Theorem 3.1.)*

a) $\mathbf{r}^{(i+1)} \in \mathbb{Z}^n$;

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 (3.1).*

Proof:

a) $(p^g\mathbf{r}^{(i)} - M\mathbf{u}^{(i)}) \bmod p^{g+k} = (p^gI - MQ)\mathbf{r}^{(i)} \bmod p^{g+k}$, 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 $2|u_j^{(i)}| \le p^{g+k}$, and so $p^{g+k}|r_j^{(i+1)}| \le p^g|r_j^{(i)}| + n\gamma \max_k |u_k^{(i)}| \le p^g|r_j^{(i)}| + n\gamma p^{g+k}/2$. The claim now follows by induction on i.

□

The choice of any

$$h \ge 1 + \lfloor (1/k) \log_p(2|M|^{2n-1}|\mathbf{b}|) \rfloor \tag{5.5}$$

ensures that $p^{g+hk} > 2p^g|M|^{2n-1}|\mathbf{b}|$ and thus enables the recovery of a unique vector $\mathbf{y} = p^g\mathbf{x}$ from $\mathbf{x}^{(h)} \bmod p^{kh}$.

COROLLARY 5.1: *(Cf. Theorem 3.3.) Algorithm 5.1 for* h *in (5.4) uses* $B_1 = O((v_M + v_Q)\mu(\log(n\gamma p^{g+k}))(\log q)/\log p^k)$ *bit operations for* $q = |M|^{2n-1}|\mathbf{b}|$, $\mu(d)$ *in (0.1), and* v_S *defined in Section 2, so*

$$B_1 = O(m(n)\mu(\log(n\gamma p^{g+k}))(n \log n)/\log(p^k)) \tag{5.6}$$

for $m(n)$ in (0.2) if (3.4) holds;

$$B_1 = O((m(n)n\log n)\mu(k\log p)/(k\log p)) \tag{5.7}$$

if k is chosen such that

$$\log(\gamma np^g) = O(\log p^k), \tag{5.8}$$

$B_1 = O(m(n)\mu(\log(n\gamma p^{g+k}))n)$ *if* $k\log p = O(\log n)$. *Furthermore, the asymptotic bit cost of the recovery of* $\mathbf{x}$ *from* $\mathbf{x}^{(h)}$ *is still covered by the bounds in Theorems 3.2 and 4.2 and Corollary 4.1, due to (5.3).*

Remark: Algorithm 5.1 can be interpreted as Hensel's lifting computations applied to the matrix M/p^g and its inverse in $\mathbb{Z}^{n\times n}_{p^{g+k}}$, that is, $\mathbf{u}^{(i)} = Q^{(i)}\mathbf{r} \bmod p^{g+k}$ (this is unchanged), $\mathbf{r}^{(i+1)} = (\mathbf{r}^{(i)} - (M/p^g)\mathbf{u}^{(i)})/p^k$.

Remark: (5.2) defines the minimum integer $g = g_-$ such that M/p^g can be inverted modulo p^{g+1}. For a random M, this g is likely to be quite small (see (Wa)). If p^{g+k} exceeds the value p^{g_+} representing the size of a computer word, however, the computations may become inefficient, and shifting to backup algorithms for a distinct (fixed or random) base prime p is in order. In a sample policy, we first seek a smaller g satisfying (5.2) by applying a binary search process; then if $g < g_+$, we may choose $k = g_+ - g$, otherwise we shift to a backup lifting algorithm with a distinct p.

Remark: Accelerated generalized Hensel's lifting. One may apply a single step of Algorithm 5.1 to compute the matrix $Q_1 = M^{-1} \bmod p^{g+2k}$, then restart the algorithm with Q_1 replacing $Q = Q_0$ and repeat this trick recursively. In h steps such as accelerated generalized lifting algorithm computes the matrix $Q_h = M^{-1} \bmod p^{g+2^h k}$. This is about the same progress as in 2^h steps of Algorithm 5.1, but the precision of computing is now roughly doubled in each step, reaching the level of $((g + 2^h k)\log p)$-bit precision in h steps.

6. Initialization of Hensel's lifting

To complete the solution of a Toeplitz system $M\mathbf{x} = \mathbf{b}$ by means of Hensel's lifting, it remains to compute a generator for matrix Q satisfying (5.1). This task requires the solution of two linear systems with the matrix M, and each of our next two algorithms as well as the algorithm in (Pa) solve the problem.

ALGORITHM 6.1: *Initialization of Toeplitz–Hensel's lifting by using modular continuation.*

INPUT: $M \in \mathbb{Z}^{n\times n}$, $\mathbf{b} \in \mathbb{Z}^n$, *and an integer* $p \geq 2$.

OUTPUT: *two integers:* $k > 0$ *and* $g \geq \max_j(\mathrm{ord}_p(\delta((M^{-1}\mathbf{b})_j)))$, *and vector* $(p^g M^{-1}\mathbf{b}) \bmod p^{g+k}$.

INITIALIZATION: *Choose an integer $q > 1$ coprime with p. (Sample choices are given by $q \in \{2, 3, 5\}$ or the powers of $2, 3$, or 5.) Write $l = 1$.*

COMPUTATIONS:

1. *Compute $s_l = p^{-l} \bmod q^l$, $t_l = q^{-l} \bmod p^l$, and the matrix $M_0 = M_{0,l} = p^l I + q^l M$, so $Q = M_0^{-1} \bmod q^l = s_l I$, $M_0^{-1} \bmod p^l = (t_l M^{-1} \bmod p^l)$.*
2. *Choose h sufficiently large and apply Algorithm 5.1 for M replaced by M_0 and p by q^l to compute $M_0^{-1}\mathbf{b} \bmod q^{hl}$; recover $M_0^{-1}\mathbf{b}$.*
3. *Compute $g = \max_j(\mathrm{ord}_p(\delta((M_0^{-1}\mathbf{b})_j)))$. If $g < l$, then output g, compute and output $k = l - g$ and $p^g(M_0^{-1}\mathbf{b}/t_l) \bmod p^l = (p^g M^{-1}\mathbf{b}) \bmod p^l$. Otherwise double l and reapply the algorithm.*

THEOREM 6.1: *Assume that (4.9) holds and that*

$$\log p = O(\log q), \quad \log(|M| + |\mathbf{b}|) = O(\log n), \quad v_M = O(m(n)). \tag{6.1}$$

Then Algorithm 6.1 involves B_1 bit operations in all lifting stages for

$$B_1 = O((m(n)n\mu((g+k)\log p)\log(g+k)/\log p^k), \tag{6.2}$$

$\mu(d)$ in (0.1) and $m(n)$ in (0.2), B_1 satisfies (5.6) if k is chosen at least of the order of $g + 1$ (cf. (5.7)). Furthermore, the algorithm requires reconstruction of $\kappa = \lceil \log(g+k) \rceil = O(\log n)$ vectors $M_0^{-1}\mathbf{b}$ from their q^l-adic expansions (for κ values of l); this stage involves either performing

$$B_0 = O(n\kappa\rho(s)) \tag{6.3}$$

bit operations, for $\rho(q)$ in (1.1)–(1.3), $s = 2^d$, and $d = O(n \log n)$, or generating $R = O(n\log^2 n)$ random bits and in addition performing

$$B_0 = O(n\kappa\mu(d)) \tag{6.4}$$

bit operations.

Proof: To recover a unique vector $\mathbf{x} = \mathbf{x}_l = M_0^{-1}\mathbf{b}$ from $\mathbf{x}_l \bmod p^h$ at stage 2 (for a fixed l), it is sufficient to choose $h = h_l = 2 + \lfloor (2n - 1)\log_{q^l}(q^l|M| + p^l) + \log_{q^l}(2|\mathbf{b}|))(1/k) \rfloor$, so $h = O((n \log n)/\log q^l)$ under (6.1). Then the bit cost of lifting is bounded by $B_{1,l} = O(m(n)\mu(\log q^l)h) = O((m(n)n\mu(l \log q)\log n)/\log q^l)$ for a fixed l and by

$$B_1 = \sum_l B_{1,l} = \sum_{i=0}^{\kappa} B_{1,2^i}$$

for all l. If $\mu(d) \geq bd^a$ for $a > 1, b > 0$, we have $B_1 = O(B_{1,g+k})$, whereas generally $\mu(d) \geq d$, so $B_1 = O(B_{1,g+k}\log(g+k))$, which proves (6.2). (6.3) and (6.4) are proved similarly. □

With Algorithm 6.1 we cannot keep the computation of $\mathbf{x} = M^{-1}\mathbf{b}$ in binary form (which is practically most desirable) because the primes p and q are distinct and thus cannot both equal 2. The next algorithm is free of this deficiency and still supports Theorem 6.1 although requires a refined analysis to bound the number $h = h(l)$ of lifting steps for each l.

ALGORITHM 6.2: *Initialization of Toeplitz–Hensel's lifting by using variable diagonal (cf. (P00)).*

INPUT: $M \in \mathbb{Z}^{n\times n}$, $\mathbf{b} \in \mathbb{Z}^n$, *and an integer* $p \geq 2$.

OUTPUT: *two integers:* $k > 0$ *and* $g \geq \max_j(\mathrm{ord}_p(\delta((M^{-1}\mathbf{b})_j)))$, *and vector* $(p^g M^{-1}\mathbf{b}) \bmod p^{g+k}$.

INITIALIZE: *Compute an integer*

$$l_0 \geq 1 + \max\{1, \lfloor 2\log_p |M| \rfloor, \lfloor 2\log_p |\mathbf{b}| \rfloor\}, \tag{6.5}$$

so $|M| + |\mathbf{b}| = O(\log p^{l_0})$, *and write* $\mathbf{z}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}$, $l = l_0$.

COMPUTATIONS:

1. *Compute the matrices* $M_0 = M + p^l I$ *and* $Q = p^{-l} I$, *so* $M_0 Q - I = QM_0 - I = p^{-l} M$; *compute the integer* $h = \lceil 2\log_p(2n|M_0|^{2n-1}|\mathbf{b}|)/l \rceil$.
2. *Recursively compute the vectors* $\mathbf{z}_{i+1} - \mathbf{z}_i = Q\mathbf{r}_i = p^{-l}\mathbf{r}_i$, $\mathbf{r}_{i+1} = \mathbf{b} - M_0\mathbf{z}_{i+1} = \mathbf{r}_i - M_0 Q\mathbf{r}_i = -p^{-l} M\mathbf{r}_i$, $i = 0, 1, \ldots, h-1$.
3. *Recover* $\mathbf{z} = M_0^{-1}\mathbf{b}$ *from* $\mathbf{z}_h$ *by using the numerical rational roundoff algorithm supporting Theorem 1.1 (see (6.6) below).*
4. *Proceed as in stage 3 of Algorithm 6.1, that is, compute* $g = \max_j(\mathrm{ord}_p(\delta(M_0^{-1}\mathbf{b})_j))$. *If* $g < l$, *output* g, *compute and output* $k = l - g$ *and* $(p^g M_0^{-1}\mathbf{b}) \bmod p^{g+k}$. *Otherwise double* l *and reapply the algorithm.*

Let us analyze this algorithm. Stage 1 is the customary residual correction algorithm for iterative improvement of approximation to $\mathbf{z}$ (see Skeel 1980 (S80), (GL96, Section 3.5.3), Higham 1996 (H96)), which we employ in the place of Hensel's lifting. (Compare numerical computations with binary computations modulo p^{-u} where u is the machine precision or the word length.) 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 $|\mathbf{z} - \mathbf{z}_h| \leq (p^{-l}|M|)^h |\mathbf{b}|/|M_0|$.

For $l \geq l_0$ in (6.5), we have $|M| < p^{l/2}$, $|\mathbf{b}| < p^{l/2}$, $p^l + p^{l/2} > |M_0| > p^l - p^{l/2} \geq (p^{l/2} - 1)p^{l/2}$, and

$$|\mathbf{z} - \mathbf{z}_h| < p^{-hl/2}. \tag{6.6}$$

Therefore, average iteration step in stage 1,

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

contributes at least $l - \log_p |M| \geq l/2$ additional correct terms to the p-adic representation of $\mathbf{z}$. Rounding the components of $\mathbf{r}_{i+1}$ to (say) l leading terms destroys at most a single correct p-term per component, whereas the computational precision stays bounded to $O(l \log\ p)$ bits. (This argument is a simplification of the routine error analysis of the iterative improvement algorithm (S80), (H96); in our case Q is in the very special simple form of $p^{-l}I$.)

Now, to ensure correct recovery of $\mathbf{z}$ with using the numerical rational roundoff algorithm supporting Theorem 1.1, it is sufficient to approximate $\mathbf{z}$ by $\mathbf{z}_h$ within the error norm less than $1/(2|M_0|^{2n-1})|\mathbf{b}|$. Due to (6.5), (6.6), we achieve this as soon as

$$p^{hl/2} > 2(p^l + p^{l/2})^{2n-1}p^{l/2} \geq 2|M_0|^{2n-1}|\mathbf{b}|,$$

that is, as soon as

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

This inequality holds for $h \geq 4n - 1 + 4n/\log p^l$.

By using this bound on h and the bound $O(l \log p)$ on the precision of computing for a fixed l, we deduce the same bit cost estimates of Theorem 6.1 for p replacing q.

The value of g computed by Algorithms 6.1 and 6.2 may exceed g_- in (5.2) but not the value $2\,\mathrm{ord}_p(\det M)$. By our experiments in (Wa), even $\mathrm{ord}_p(\det M)$ is rarely large, and in the rare cases where it is large, we may shift to a slower backup algorithm. We may detect this case by imposing an upper bound on l and stopping if this bound is exceeded. For completeness, we cover our algorithms in this unlikely case as well, by allowing l grow unbounded.

In the alternative initialization algorithm in (Pa), the recovery is confined to the very end of the solution algorithm and the bit cost of the initialization decreases dramatically to the bound

$$B = O((m(n)\log n)\mu(\log(g+k)\log p)), \tag{6.7}$$

for a fixed p (see Theorem 12.1 in (Pa)). The overall bit cost bound, however, does not decrease that dramatically because of the impact of the lifting and rational number reconstruction stages. The next asymptotic estimates are based on combining (3.3), (4.12), (5.4), (5.6), (5.7) and (6.7).

THEOREM 6.2: *Suppose M is a nonsingular $n \times n$ integer Toeplitz matrix, $\mathbf{b} \in \mathbb{Z}^n$, $\epsilon > 0$. Let (3.1), (3.4), (4.9), (4.10), and (5.7) hold. Then it is sufficient to perform $B_0 + B_1 + B$ bit operations for B_1 in (5.6), B in (6.7), and B_0 in (3.3) or, alternatively, to generate $O((n \log n)\log(1/\epsilon))$ random bits and in addition to perform $B_0 + B_1 + B$ bit operations for B_1 in (5.6), B in (6.7), and B_0 in (4.12) to compute the vector $\mathbf{x} = M^{-1}\mathbf{b}$ and to verify that $M\mathbf{x} = \mathbf{b}$ holds true, allowing a probability of failure of at most ϵ.*

Theorem 6.2 can be supported by a hybrid algorithm using our lifting, the fast rational number reconstruction in (PW02), (WPa), and initilaization in (Pa) but alternatively just by the algorithms in (Pa) performed modulo a sufficiently high prime power and by the subsequent fast recovery of the rational solution by the algorithms in (PW02), (WPa).

Let us express B_0, B_1 and B in terms of n. Recall that $d = \log s = O(n \log(|M| + |\mathbf{b}|))$, so $d = O(n \log n)$ if $\log \gamma = O(\log n)$. Thus

$$B_0 = O(n\mu(n \log n)) \text{ in (4.12)}; \tag{6.8}$$

$$B_0 = O((n \log n)\mu(n \log n)) \text{ in (3.3)}. \tag{6.9}$$

Furthermore, $B_1 = O((m(n)n \log n)\mu(k \log p)/(k \log p))$ in (5.6) and $B = O((m(n) \log n)\mu(k \log p))$ in (6.7) assuming (5.7) where $k \log p = O(n \log n)$. Summarizing, we have

$$B_0 + B_1 + B = O((m(n)\mu(n \log n)), \tag{6.10}$$

whereas

$$B_0 + B_1 + B = O((m(n)n \log n)\mu(\log n)) + B_0 \tag{6.11}$$

where $g = O(1)$ and B_0 is in (6.8) or (6.9) (cf. (0.3)).

Remark: Generalized Newton's iteration. We may perform generalized Newton's iteration for matrix inversion,

$$X_0 = Q, \quad X_{i+1} = 2p^g X_i - X_i M X_i \bmod p^{(g+k)2^{i+1}}, \quad i = 0, 1, \ldots, h,$$

with the precision of $\lceil ((g + k)2^{i+1}) \log p \rceil$ bits in the i-th step; in h steps it outputs the matrix X_h satisfying $MX_h = p^{g2^h} I \bmod p^{(g+k)2^h}$. Roughly the same progress as in $k2^h$ steps of generalized Hensel's lifting algorithms is now achieved in h steps. This is similar to the effect of the accelerated generalized Hensel's lifting defined in our Remark in the previous section, and this effect can be exploited for parallel acceleration of lifting (P00) where the number of lifting steps is a bottleneck. The transition from Hensel's to Newton's lifting for matrix inversion does not generally decrease the overall bit operation cost, because the precision of computing grows dramatically. The parallel algorithm in (P00) uses lifting and is quite involved, to a large extent because it uses Algorithm 6.2 for the initialization of lifting; since Algorithm 6.1 is much simpler conceptually, its substitution for Algorithm 6.2 would substantially simplify the algorithm analysis in (P00).

7. Probability of degeneration in $\mathbb{Z}_{p^{g+1}}$

By Theorem 6.1, the bit cost of solving the system $M\mathbf{x} = \mathbf{b}$ is roughly proportional to $\mu(g+1)$ where $\mu(d)$ is in (0.1) and g is expressed via p and M in (5.2).

Next, we let a positive integer $g+1$ and a nonsingular matrix $M \in \mathbb{Z}^{n\times n}$ be fixed and a prime p be randomly sampled in a fixed range, then we estimate the probability that M becomes singular in $\mathbb{Z}^{n\times n}_{p^{g+1}}$. We begin with some auxiliary estimates where $\ln = \log_e$ stands for the natural logarithms (with the base $e = 2.718281\ldots$) and $\pi(y)$ denotes the number of primes not exceeding y.

LEMMA 7.1: *(See also (7.3) in this section.) If* $y > 114$, *then* $1 < (\ln y)\pi(y)/y < 1.25$.

Proof: See Rosser and Schoenfeld 1962 (RS62). □

LEMMA 7.2: *Let* $y \geq 114$, *then* $\pi(y) - \pi(y/20) > y/(\beta \ln y)$ *for*

$$\beta = 1/(1-\alpha) = 1.2049303\ldots, \quad \alpha = \ln 114/(16\ln 5.7) = 0.17007650\ldots. \tag{7.1}$$

Proof: By Lemma 7.1, we have $\pi(y) - \pi(y/20) > y/\ln y - 1.25y/(20\ln(y/20))$. Observe that $\ln(y/20)/\ln y$ is monotone increasing as y grows. So $1.25/(20\ln(y/20)) \leq \alpha/\ln y$, for α in (7.1) and $y \geq 114$. Combine the above estimates. □

LEMMA 7.3: *(Cf. Corollary 7.8.2 in (P01).) Let* y, g, h, *and* k *be positive integers such that*

$$y \geq 114, \quad 0 < h^{1/k} \leq y/20. \tag{7.2}$$

Let p *be a random prime selected in the range* $(y/20, y]$ *under the uniform probability distribution. Then* $\text{Probability}(h \bmod p^{g+1} = 0) < (\beta k \ln y)/((g+1)y)$ *for* β *in (7.1).*

Proof: Suppose that in the above range there are exactly l distinct primes whose $(1+g)$-th powers divide h. Then the product of these powers also divides h, and therefore we have $h \geq (y/20)^{(g+1)l}$ because each of the l primes lying in the range $[y/20, y]$ is at least as large as $y/20$. On the other hand, $h \leq (y/20)^k$ by assumption. Therefore, $(g+1)l \leq k$, that is, $l \leq k/(g+1)$. Compare the latter upper bound on l with the lower bound given by Lemma 7.2. □

THEOREM 7.1: *(Cf. Corollary 7.8.3 in (P01).) Suppose that* $g+1$ *is a positive integer,* $M \in \mathbb{Z}^{n\times n}$ *is nonsingular, and a prime* p *is randomly sampled from the range* $(y/20, y]$ *under the uniform probability distribution in this range where* $y = n^b \ln|M| \geq 114$, $b \geq 1$. *Then we have*

$$P = \text{Probability}((\det M) \bmod p^{g+1} = 0) < cn^{1-b}/(g+1) = (cn\ln|M|)/(g+1)$$

for $c = 16\alpha/(1-\alpha)$ *and* α *in (7.1), so* $c = 3.278885445\ldots$.

Proof: Write $y = n^b \ln |M|$, $h = |\det M|$, $k = (n \ln |M|)/\ln((n^b \ln |M|)/20)$. Recall that $h \le |M|^n$ and deduce that $k \ln(y/20) \ge \ln h$, which implies (7.2). Apply Lemma 7.3 and deduce that

$$P < \frac{\beta n \ln |M|}{(g+1)\ln((n^b \ln |M|)/20)} \frac{\ln(n^b \ln |M|)}{n^b \ln |M|} = \frac{\beta n^{1-b}}{(g+1)(1 - \ln 20/\ln y)}.$$

Note that

$$\frac{1}{1 - \frac{\ln 20}{\ln y}} = \frac{\ln y}{\ln(y/20)} \le \frac{\ln 114}{\ln 5.7}$$

for $y \ge 114$, so

$$P \le \frac{n^{1-b}\beta}{(g+1)} \frac{\ln 114}{\ln 5.7} = \frac{16\alpha\beta n^{1-b}}{g+1}$$

for α and $\beta = 1/(1-\alpha)$ in (7.1). □

To extend the above results to smaller y, one may exploit the known extensions of Lemma 7.1, e.g.,

$$1 + \frac{1}{2\ln y} < (\ln y)\pi(y)/y < 1 + \frac{3}{2\ln y} \tag{7.3}$$

for $y \ge 59$ (see (GG99, Theorem 18.7)).

By combining Theorem 7.1 with our algorithms in Sections 3–6, we deduce the following estimates.

THEOREM 7.2: *Assume that (4.10) holds for a given ε, a nonsingular Toeplitz matrix $M \in \mathbb{Z}^{n\times n}$, and a vector $\mathbf{b} \in \mathbb{Z}^n$ such that*

$$\log \max\{|M|, |\mathbf{b}|, 1/\varepsilon\} = O(\log n).$$

Then there are algorithms which output either FAILURE (with a probability of at most ε) or the correct solution $\mathbf{x}$ to the linear system $M\mathbf{x} = \mathbf{b}$ and which generate R random bits and perform $B = B_0 + B_1$ bit operations where $B_1 = O(nm(n)\mu(\log n))$ and one may yield either

a) $R = O(n \log^2 n)$, $B_0 = O(B_1)$, or

b) $R = O(\log n)$, $B_0 = O(n\rho(n \log n))$

for $m(n)$ in (0.1), $\mu(d)$ in (0.2), and $\rho(d)$ in (1.1)–(1.3).

Proof: (Cf. also Remark below.) Combine Theorems 4.2, 6.1 and 7.1, Corollaries 4.1 and 5.1, and our first Remark in Section 5 with either Theorem 3.3 to support part(a) or Theorem 3.2 to support part(b). □

Remark: The latter asymptotic estimates can be proved based on Theorem 7.1 for $g = 0$; by increasing g, we may decrease the bound y (which defines the range for a random prime p) at the expense of increasing B by a constant factor.

8. Avoiding degeneration for a fixed p

In practice, computations in $\mathbb{Z}_{p^{g+1}}$ can be substantially simplified by preconditioning if p^{g+1} is fixed or varies in a relatively small range. Furthermore, for $p = 2$, the computations are binary, and their simplification is most significant even with no preconditioning.

These observations raise the derandomization problem of implementing our algorithm for fixed p^{g+1}. Clearly, we cannot supply the input matrix $Q = p^g M^{-1} \bmod p^{g+1}$ if $\det M = 0$ in $\mathbb{Z}_{p^{g+1}}$, so we should apply our faster algorithms in the case where $(\det M) \bmod p^{g+1} \neq 0$ and otherwise back them up by slower algorithms which are insensitive to vanishing $\det M$ in $\mathbb{Z}_{p^{g+1}}$. Now our problem is twofold:

a) estimating which fraction of all Toeplitz matrices in $\mathbb{Z}_{p^{g+1}}^{n\times n}$ are nonsingular in $\mathbb{Z}_{p^{g+1}}^{n\times n}$ and

b) modifying our lifting algorithms to extend them to all or a large part of the remaining Toeplitz matrices in $\mathbb{Z}_{p^{g+1}}^{n\times n}$.

Here are some relevant comments.

a) For $g = 0$ and any p and n, the singular matrices make up the fraction of exactly $1/p$ in the space of all matrices in $\mathbb{Z}_p^{n\times n}$.

THEOREM 8.1: *(Kaltofen and Lobo 1996 (KL96).) For any pair of a prime p and a natural n, the singular matrices make up a fraction of $1/p$ in the space of all Hankel (or Toeplitz) matrices in $\mathbb{Z}_p^{n\times n}$.*

Proof: We give a shorter proof versus (KL96) by obtaining it as a corollary of Proposition 9.1 on page 158 in (BP94). This proposition defines a bijection map of all pairs (h, H) of $h = h_{2n-1} \in \mathbb{Z}$ and Hankel matrices H in $\mathbb{Z}_p^{n\times n}$ to all ratios $\frac{u(x)}{v(x)}$ of polynomials $u(x)$ and $v(x)$ over $\mathbb{Z}_p$ where $\deg v(x) = n$, $\deg u(x) < n$; the map is also a bijection where it is restricted to mapping the pairs of integer h and nonsingular H to coprime polynomials $u(x)$ and $v(x)$. Clearly, coprime pairs make up a fraction of $1/p$ in the set of all polynomials $u(x)$ and $v(x)$ of this class. (Indeed, fix all coefficients of $u(x)$ and $v(x)$ except for the x-free term of $v(x)$, which would vary in $\mathbb{Z}_p$.) The bijection extends the estimate $1/p$ to Hankel matrices, and the map $HJ = T$ extends it further to Toeplitz matrices. □

Both proofs in (KL96) and above do not easily extend to the rings $\mathbb{Z}_{p^g}$ for $g > 1$ but our experimental computations for $p = 2$ show that the singular matrices also make up a small fraction in the space of all $n \times n$ Hankel (or Toeplitz) matrices in $\mathbb{Z}_{p^g}$ for larger g (see (Wa)).

b) A trivial extension of our algorithms to cover a larger subclass of Toeplitz inputs is by increasing g, and each of Algorithms 6.1 and 6.3 computes g such that a fixed M nonsingular in $\mathbb{Z}$ remains nonsingular in $\mathbb{Z}_{p^{g+k}}$ for a fixed positive k. If this g is large, however, then the computations in $\mathbb{Z}_{p^{g+k}}$ may become too costly, so we stop increasing g and report failure as soon as the algorithm detects

that g exceeds a fixed bound g_+. This detection is a by-product of Algorithms 6.1 and 6.3 which we simplify in this case by terminating them earlier and thus avoiding costly computations with a high precision. The problem arises, however, whether after the early termination we may resume computations by modifying our algorithms and still solve the system $M\mathbf{x} = \mathbf{b}$ efficiently.

In this section, we show a recipe towards this goal. We reduce the solution of the system $M\mathbf{x} = \mathbf{b}$ in $\mathbb{Z}^n_{p^{g+1}}$ (and consequently computation of a generator for the inverse Q of M in $\mathbb{Z}^{n\times n}_{p^{g+1}}$) to solving a reasonably small number of linear systems with the coefficient matrices of the form

$$M_i = M - U_i V_i \tag{8.1}$$

where U_i, V_i^T are random Toeplitz matrices in $\mathbb{Z}_q^{n\times i}$, q is a sufficiently large integer in the range

$$q = O(n^5), \tag{8.2}$$

and i is relatively small, $i = O(1)$. The recipe relies on the following result from (EGV00).

THEOREM 8.2: *For two positive integers i and n, $i < n$, a prime p, and a nonsingular matrix M in $\mathbb{Z}^{n\times n}$, let $U_i = U_{i,j}$ and $V_i^T = V_{i,j}^T$ denote pairs of two random matrices in $\mathbb{Z}_q^{n\times i}$ for $j = 1, 2, \dots, 450$ and a sufficiently large q in (8.2) and let the matrices $M_i = M_{i,j}$ be defined by (8.1). Then with a probability of at least $1/2$, we have $s_{n-i}(M) = \gcd(s_n(M), \gcd_{j=1}^{450}(s_n(M_{i,j}))$. To increase the probability bound above $1 - \varepsilon$ for a fixed positive ε, it is sufficient to use $j_+ = O(\log(1/\varepsilon))$ matrices $M_{i,j}$.*

Besides Theorem 8.2, we rely on a well known statistical observation (ABM99), (EGV00) whose formally proved support is so far limited to an average case result for general but not for Toeplitz matrices). More precisely, according to this result (EGV00), the value $s_{n-i}(M)$ for a random integer matrix M, is unlikely to exceed 1 except for a few smaller i. Thus we fix sufficiently large constants i_+ and j_+ and suppose (based on Theorem 8.2 and the cited statistical evidence) that

$$s = \gcd\Big(s_n(M), \gcd_{j=1}^{j_+}(s_n(M_{i,j}))\Big) = 1 \tag{8.3}$$

for some $i \le i_+$. (In fact we just need a weaker assumption that s is coprime with a fixed base prime p.) More precisely, we call a fixed (Toeplitz) matrix $M \in \mathbb{Z}^{n\times n}$ by (i_+, j_+)-*extraordinary* unless (8.3) holds.

Finally, we also need the (Sherman–Morrison)–Woodbury formula (GL96, page 50):

$$M^{-1} = (M_i + U_i V_i)^{-1} = M_i^{-1} - M_i^{-1} U_i (I_i + V_i M_i^{-1} U_i)^{-1} V_i M_i^{-1}, \tag{8.4}$$

which holds provided that the matrices M_i and at least one of M and

$$W_i = I_i + V_i M_i^{-1} U_i \tag{8.5}$$

are nonsingular for $n \times i$ matrices U_i and V_i^T. The formula follows from the equations

$$M_i U_i V_i M^{-1} = U_i W_i^{-1} V_i = M^{-1} U_i V_i M_i, \tag{8.6}$$

which can be easily proved by using infinitesimal transformation of U_i and V_i into $n \times n$ nonsingular matrices.

ALGORITHM 8.1: *(See our Examples 9.2 and 9.3.)*

INPUT: *a prime p, four positive integers $1+g_+, i_+, j_+$, and k_+, a nonsingular Toeplitz matrix $M \in \mathbb{Z}^{n\times n}$ such that $s_1(M)$ is coprime with p (this condition can be ensured by scaling M), a vector $\mathbf{b} \in \mathbb{Z}^n$, and a Subroutine* **S** *(see (Pa) and our Remark at the end of this section), which for a given matrix $W \in \mathbb{Z}^{n\times n}$ computes the minimum integer $g(W)$ and a matrix $Q = Q(W) = p^{g(W)}W^{-1} \bmod p^{g(W)+k}$ such that $WQ = p^{g(W)}I \bmod p^{g(W)+k}$ for a fixed positive integer k.*

OUTPUT: *EXTRAORDINARY (that is, M is found to be an (i_+, j_+)-extraordinary matrix) or the minimum integer g and a vector $\mathbf{y} = p^g M^{-1}\mathbf{b} \bmod p^{g+k}$, that is, such that $M\mathbf{y} = p^g\mathbf{b} \bmod p^{g+k}$ and $g \le g_+$.*

INITIALIZE: *write $i = 0$ and compute a sufficiently large q satisfying (8.2).*

COMPUTATIONS:

1. *Write $j = 1$. Sample two random Toeplitz matrices $U_{i,j}$ and $V_{i,j}^T$ from $\mathbb{Z}_q^{n\times i}$, then compute the matrix $M_{i,j} = M - U_{i,j}V_{i,j}$ and invoke the Subroutine* **S** *for $W = M_{i,j}$. If $g(W) \le g_+$, write $g_i = g(W)$, $Q_i = Q(W)$. (In this case $Q_i = p^{g_i}M_{i,j}^{-1} \bmod p^{g_i+k}$, and it remains to apply (8.3)–(8.5) to compute the vector $(p^g M^{-1}\mathbf{b}) \bmod p^{g+k}$, for $M = M_i + U_iV_i$, with $M_i = M_{i,j}$, $U_i = U_{i,j}$, $V_i = V_{i,j}$. This computation is specified in stages 2–6 below.) If $g(W) > g_+$ but $j < j_+$, then increase j by 1 and repeat stage 1. If $j = j_+$ and $i < i_+$, then write $j = 1$, increase i by 1, and repeat stage 1. (One may also consider a policy of more rapidly increasing, e.g., doubling i.) Otherwise $j = j_+$, $i = i_+$, then stop and output EXTRAORDINARY.*
2. *Compute the $i \times i$ matrix $W_i = p^{g_i}I + V_iQ_iU_i$.*
3. *Apply Gaussian elimination with pivoting to compute the minimum integer $\bar{g}$ and the matrix T_i such that $W_iT_i = p^{\bar{g}-g_i}I \bmod p^{\bar{g}-g_i+k}$, so $T_i = p^{\bar{g}-g_i}W_i^{-1} \bmod p^{\bar{g}-g_i+k}$. ($i$ is expected to be quite small, and if so the computation is not expensive.)*
4. *Compute the vectors $\mathbf{u}_i = p^{g_i}M_i^{-1}\mathbf{b} \bmod p^{g_i+k}$ and $\mathbf{v}_i = p^{\bar{g}}U_iT_iV_i\mathbf{u}_i \bmod p^{\bar{g}+k}$. (By (8.6), we have $\mathbf{v}_i = p^{\bar{g}}M^{-1}U_iV_iM_i\mathbf{u}_i \bmod p^{\bar{g}+k} = p^{\bar{g}}M^{-1}U_iV_i\mathbf{b} \bmod p^{\bar{g}+k}$.)*
5. *Compute the vectors $\mathbf{w}_i = -p^{\bar{g}+g_i}M^{-1}\mathbf{v}_i \bmod p^{\bar{g}+g_i+k}$, $\mathbf{y}_i = \mathbf{w}_i + p^{\bar{g}}\mathbf{u}_i \bmod p^{\bar{g}+g_i+k}$.*

6. *Compute and output the minimum integer* g *and the vector* $\mathbf{y}$ *such that* $M\mathbf{y} = p^g\mathbf{b}$ *and* $p^{a_i}\mathbf{y} = \mathbf{y}_i$ *for some integer* a_i.

Correctness of the algorithm is verified by using Theorem 8.2 and equations (8.3)–(8.6). The asymptotic computational bit cost is dominated by the cost of invoking Subroutine **S** at most $(1 + i_+)j_+$ times (assuming that i_+ is small enough to keep the bit cost at stage 3 lower) and thus (up to the factor of $(1 + i_+)j_+$) is bounded as in Theorem 6.1 applied for $g = g_+$. Observe that $v_{M_i} \le 4m(n) + n$ and the matrix M_i has displacement rank 3, so the definition of a generator of the inverse and Theorem 2.11 and Corollary 2.2 are extended (see the definition of the displacement rank and proofs in (Pa) or (P01)) to yield that $i_{M_i} \le 6m(n) + 2n$ for v_S and i_S defined in Section 2. The number of random bits can be decreased by reusing the same bits for all i, so it may stay as in Theorem 6.1 for $g = g_+$.

For the same upper bound $1 + g_+$ on the order of p and for larger i_+ and j_+, the algorithm handles a larger class of input Toeplitz matrices than Algorithms 6.1 and 6.2 do, that is, bounding $g_p(s_n(M))$ by g_+ is now replaced by bounding $g_p(s_{n-i_+}(M))$ by g_+, which already for moderate i_+ is a substantially weaker requirement according to the cited statistical evidence.

Remark: Subroutine **S** is the only missing ingredient in Algorithm 8.1, for which each of Algorithms 6.1 and 6.2 can be used. The assumption about minimization of $g(W)$ is not supported by these algorithms in general but we only need it in a special case where matrix $M_{i,j}$ is nonsingular modulo p. Indeed the problem is trivial unless p divides $s_n(M)$, but p cannot divide both $s_n(M)$ and $s_n(M_{i,j})$ for all j due to (8.3). Thus we may just write $g^+ = 0$ and apply Algorithm 8.1 with Algorithm 6.1 or 6.2 as Subroutine **S**.

9. Demonstration of Algorithms 3.1, 5.1, and 8.1 by Examples

EXAMPLE 9.1: $M = \left(\begin{smallmatrix} 2 & 1 \\ 3 & 2 \end{smallmatrix}\right)$, $\mathbf{b} = \binom{3}{4}$, *so* $\mathbf{x} = \binom{2}{-1}$. *By applying Algorithm 3.1 for* $p = 2$, $\mathbf{r}^{(0)} = \mathbf{b}$, *we successively compute* $Q = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $\mathbf{u}^{(0)} = \binom{0}{1}$, $\mathbf{r}^{(1)} = \binom{1}{1}$, $\mathbf{u}^{(1)} = \binom{1}{1}$, $\mathbf{r}^{(2)} = \binom{-1}{-2}$, $\mathbf{u}^{(2)} = \binom{0}{1}, \ldots$. *So,* $\mathbf{x}^{(3)} = 2\mathbf{x} \bmod 8 = \binom{0}{1} + 2\binom{1}{1} + 4\binom{0}{1}$.

EXAMPLE 9.2: $M = \left(\begin{smallmatrix} 4 & 1 \\ 6 & 2 \end{smallmatrix}\right)$, $\mathbf{b} = \binom{3}{4}$, *so* $\mathbf{x} = \binom{1}{-1}$.

a) By applying Algorithm 5.1 for $p = 2$, $g = k = 1$, $\mathbf{r}^{(0)} = \mathbf{b}$, *we successively compute* $Q = \left(\begin{smallmatrix} 0 & 1 \\ 2 & 0 \end{smallmatrix}\right)$, $\mathbf{u}^{(0)} = \binom{0}{2}$, $\mathbf{r}^{(1)} = \binom{1}{1}$, $\mathbf{u}^{(1)} = \binom{1}{2}$, $\mathbf{r}^{(2)} = \binom{-1}{-2}$, $\mathbf{u}^{(2)} = \binom{2}{2}, \ldots$. *So,* $\mathbf{x}^{(3)} = 2\mathbf{x} \bmod 8 = \binom{0}{2} + 2\binom{1}{2} + 4\binom{2}{2}$, $(M\mathbf{x}^{(h)} - 2\mathbf{b}) \bmod 2^{h+1} = 0$ *for* $h = 1, 2, 3$.

b) Alternatively, by observing that $s_2(M) = 2$, $s_1(M) = 1$ *and applying Algorithm 8.1 to* $M_1 = M - U_1V_1$, $U_1 = V_1^T = \binom{1}{1}$, *and* $\mathbf{b} = \binom{3}{4}$, *we reduce computation of* $\mathbf{x}$ *to applying Algorithm 3.1 three times (according*

to the (Sherman–Morrison)–Woodbury formula), with the right-hand-side vectors $\mathbf{b} = \binom{3}{4}$, $\mathbf{b}^{(1)} = \binom{1}{1}$, *and* $\mathbf{b}^{(2)} = (1/3)\begin{pmatrix}1&1\\1&1\end{pmatrix} M_1^{-1}\binom{3}{4}$, *respectively. We have* $M_1 = \begin{pmatrix}3&0\\5&1\end{pmatrix}$, $M_1^{-1}\binom{3}{4} = \binom{1}{-1}$, $M_1^{-1}\binom{1}{1} = \binom{1/3}{-2/3}$, *(these vectors can be computed by applying Algorithm 3.1, we omit the details), so* $\mathbf{b}^{(2)} = \mathbf{0}, M_1^{-1}\mathbf{b}^{(2)} = \mathbf{0}$, $M^{-1}\mathbf{b} = M_1^{-1}\mathbf{b} = \binom{1}{-1}$.

EXAMPLE 9.3: $M = \begin{pmatrix}32&2\\48&4\end{pmatrix}$, $\mathbf{b} = \binom{24}{32}$. *So,* $\mathbf{x} = \binom{1}{-4}$, $s_2(M) = 32$, $s_1(M) = 2$. *We may*

a) apply Algorithm 5.1 to M *and* $\mathbf{b}$ *for* $p = 2, g = 5, k = 1$ *or*

b) apply Algorithm 8.1 to $M_1 = M - U_1V_1$, $U_1 = \binom{1}{1}$, $V_1^T = \binom{2}{2}$, $M_1 = \begin{pmatrix}30&0\\46&2\end{pmatrix}$. *For solving the equations* $M_1^{-1}\mathbf{b}^{(i)}, i = 1, 2, 3$ *(cf. Example 9.2 b)), apply Algorithm 5.1 for* $p = 2, g = k = 1$.

A. Extension to computing Smith's factors and the determinant

In (EGV00), computing Smith's factors and the determinant of a general integer matrix M is reduced to solving a small number of linear systems $M\mathbf{x}_k = \mathbf{b}_k$ for random vectors $\mathbf{b}_k$. The reduction is immediately extended to a Toeplitz matrix M. Here are the resulting bit cost estimates.

THEOREM A.1: *Allow output errors with a probability of at most* $\nu > 0$, *also allow an additional factor of* $\log(1/\nu)$ *in all asymptotic estimates in Theorems 3.3, 3.4, and 4.4 and Corollary 4.5 for the numbers of random bits and bit operations. Then the resulting (increased) estimates apply to the computation of Smith's leading factor* $s_n(M)$, *not including the correctness verification cost. Up to increasing the bit operation cost bounds by the factor of* k *and sampling* $O(kn \log n)$ *additional random bits, the same bounds cover the computation of the next* k *distinct leading Smith's factors of* M; *with the* l*-fold increase, the bit operation cost bounds of Theorem 4.2 and Corollary 4.1 cover the computation of all Smith's factor of* M *and* $\det M$ *(without correction verification), where* l *is the overall number of distinct Smith's factors,* $l \le \sqrt{\log\det|M|} \le \sqrt{n\log|M|}$ *for every matrix* $M \in \mathbb{Z}^{n\times n}$.

The theorem is supported by the algorithm in (EGV00) complemented by the smaller complexity bounds for Toeplitz matrix computation. We only state the next basic lemma and give its shorter proof.

LEMMA A.1: *Let* $\mathbf{b}$ *be a random vector in* $\mathbb{Z}^n$. *Then* δ *in (4.1) divides* $s_n(M)$, *and furthermore, for any prime* p,

$$\text{Probability}(\text{ord}_p(\delta) < \text{ord}_p(s_n(M))) \le \max\{1/m, 1/p\}.$$

Proof: The theorem follows from Theorem 2 in (ABM99), but here is a simpler proof. We have $x_i = \sum_j (-1)^{i+j} d_{i,j} b_j / \det M$, $s_n = |(\det M)/d|$, $d = \gcd(d_{i,j})_{i,j}$ for $d_{i,j}$ in Definition 2.1 and $\mathbf{b} = (b_j)_{j=1}^n$. Write $h_{i,j} = \mathrm{ord}_p(d_{i,j})$, $h = \mathrm{ord}_p(d) = \min_{i,j} d_{i,j}$. We have $h = \mathrm{ord}_p(d_{u,v})$ 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 $r = \sum_{j=1}^{n-1} (-1)^j \bar{d}_{0,j} b_j^{(k)} \in \mathbb{Z}$. It remains to recall that $\mathrm{ord}_p(\bar{d}_{0,0}) = 0$ and b_0 is randomly sampled from $\mathbb{Z}_m$, independently of $\bar{d}_{0,0}$. □

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.

[Ba] D. J. Bernstein, Multidigit Multiplication for Mathematicians, *Advances in Applied Mathematics*, to appear.

[BA80] R. R. Bitmead, B. D. O. Anderson, Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations, *Linear Algebra and Its Applications*, **34**, 103–116, 1980.

[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.

[FP74] M. J. Fischer, M. S. Paterson, String Matching and Other Problems, *SIAM-AMS Proceedings*, **7**, 113–125, 1974.

[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.

[H79] G. Heinig, Beitrage zur spektraltheorie von Operatorbuschen und zur algebraischen Theorei von Toeplitzmatrizen, Dissertation **B**, *TH Karl-Marx-Stadt*, 1979.

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

[KL96] E. Kaltofen and A. Lobo, On Rank Properties of Toeplitz Matrices over Finite Fields, *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'96)*, 241-249, ACM Press, New York, 1996.

[KS91] E. Kaltofen, B. D. Saunders, On Wiedemann's Method for Solving Sparse Linear Systems, *Proceedings of AAECC–5, Lecture Notes in Computer Science*, **536**, 29–38, Springer, Berlin, 1991.

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

[M74] M. Morf, *Fast Algorithms for Multivariable Systems*, Ph.D. Thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, 1974.

[M80] M. Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proceedings of IEEE International Conference on ASSP*, 954–959, IEEE Press, Piscataway, New Jersey, 1980.

[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.

[MSa] T. Mulders, A. Storjohann, Certified Dense Linear System Solving, Preprint, 2001.

[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.

[P02] V. Y. Pan, Can We Optimize Toeplitz/Hankel Computations? *Proc. of the Fifth International Workshop on Computer Algebra in Scientific Computing (CASC'02)*, Yalta, Crimea, Sept. 2002 (E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov, Editors), 253–264, Technische Universitat München, Germany, 2002.

[Pa] V. Y. Pan, Superfats Algorithms for Singular Integer Toeplitz/Hankel-like Matrices, Preliminary Report, 2002.

[PW02] V. Y. Pan, X. Wang, Acceleration of Euclidean Algorithm and Extensions, *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC'02)*, (Teo Mora editor), 207–213, ACM Press, New York, 2002.

[RS62] J. B. Rosser, L. Schoenfeld, Approximate Formulas of Some Functions of Prime Numbers, *Illinois J. of Math.*, **6**, 64–94, 1962.

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

[S86] A. Schrijver, *Theory of Linear and Integer Programming*, Wiley, New York, 1986.

[S02] A. Storjohann, High-Order Lifting, *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'02)*, 246–254, ACM Press, New York, 2002.

[V00] G. Villard, Computing the Froberius Normal Form of a Sparse Matrix, *Proc. 3rd Intern. Workshop on Computer Algebra in Scientific Computing (CASC'00)*, Samarkand, Uzbekistan, October 2000 (E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov, Editors), Springer, 2000.

[WPa] X. Wang, V. Y. Pan, Acceleration of Euclidean Algorithm and Rational Number Reconstruction, accepted by *SIAM J. on Computing*.

[Wa] X. Wang, How frequently is a matrix nonsingular? Technical Report, Ph.D. Program in Computer Science, The Graduate Center of City University of New York, New York, 2002.

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