2011

TR-2011002: Symbolic Lifting for Structured Linear Systems of Equations: Numerical Initialization, Nearly Optimal Boolean Cost, Variations, and Extensions

Victor Y. Pan

Symbolic Lifting for Structured Linear Systems of Equations: Numerical Initialization, Nearly Optimal Boolean Cost, Variations, and Extensions *

Victor Y. Pan

Department of Mathematics and Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Abstract

Hensel's symbolic lifting for a linear system of equations and numerical iterative refinement of its solution have striking similarity. Combining the power of lifting and refinement seems to be a natural resource for further advances, but turns out to be hard to exploit. In this paper, however, we employ iterative refinement to initialize lifting. In the case of Toeplitz, Hankel, and other popular structured inputs our hybrid algorithm supports Boolean (bit operation) time bound that is optimal up to logarithmic factor. The algorithm remains nearly optimal in its extensions to computing polynomial gcds and lcms and Padé approximations, as well as to the Berlekamp–Massey reconstruction of linear recurrences. We also cover Newton's lifting for matrix inversion, specialize it to the case of structured input, and combine it with Hensel's to enhance overall efficiency. Our initialization techniques for Hensel's lifting also work for Newton's. Furthermore we extend all our lifting algorithms to allow their initialization modulo powers of two, thus implementing them in the binary base.

Key Words: Linear systems of equations, Hensel's Lifting, Newton's lifting, Iterative refinement, Toeplitz matrices

*Supported by PSC CUNY Awards 61406-00-39 and 62230–0040. Proceedings version of this paper has been presented at the Third International Workshop on Symbolic-Numerical Computations (SNC'09), August 3–5, 2009, Kyoto, Japan.

1 Introduction

Hensel's lifting is a fundamental tool of symbolic computation [gg03], is highly effective for solving linear systems of equations [mc79], [d82], and has a natural and popular counterpart in numerical computations, called *iterative refinement*.

Both Hensel's lifting and iterative refinement are initiated with an approximate inverse Q of an input matrix M and recursively refine an initial approximate solution $\mathbf{x} = \mathbf{x}^{(0)}$ to a linear system $M\mathbf{x} = \mathbf{f}$, e.g., $\mathbf{x}^{(0)} = Q\mathbf{f}$. Every recursive step essentially amounts to multiplication of the matrices M and Q by two vectors and produces at least b new correct bits per an output value for a fixed positive b, although the two algorithms advance towards two different representations of such a value and from the two opposite sides.

For structured matrices multiplication steps are inexpensive, and we can advance towards the solution very rapidly as soon as we initialize lifting. This is precisely what we do by employing iterative refinement.

The similarity between lifting and refinement suggests that such hybrid algorithms are natural, but technically their derivation is not straightforward, and our work is the only known example where one takes advantage of the combined power of lifting and refinement.

Having solved the initialization problem, we arrive at a hybrid algorithm that supports a nearly optimal (up to logarithmic factor) Boolean (bit operation) time bound for the solution of Toeplitz, Hankel and other popular structured linear systems of equations, which are omnipresent in scientific and engineering computations and signal and image processing.

Furthermore our solution algorithms and our nearly optimal Boolean time bounds are extended to some fundamental polynomial computations such as the computation of the gcd, lcm, and Padé approximation for polynomials as well as the Berlekamp–Massey's reconstruction of linear recurrences (see Section 8).

Newton's lifting is a well known counterpart to Hensel's [y76], [g84]. For the computations with integer matrices both Newton's and Hensel's liftings were simultaneously proposed in [mc79]. Like Hensel's, Newton's lifting also has a celebrated numerical counterpart (see, e.g., [ps91]).

Newton's lifting computes the matrix inverse rather than the solution of a linear system of equations but otherwise is quite similar to Hensel's. In particular in the case of structured inputs we modify Newton's classical lifting algorithm to reduce its every lifting step essentially to multiplication of the input matrix and its approximate inverse by a small number of vectors, and thus in the case of structured input matrices we can also perform it in a nearly linear arithmetic time by exploiting matrix structure.

Here are the two main technical differences:

(a) instead of refining an approximate solution $\mathbf{x} = \mathbf{x}^{(0)}$ to a linear system $M\mathbf{x} = \mathbf{f}$, Newton's lifting recursively refines an approximate inverse $X = X^{(0)}$, that is an approximate solution to the matrix equation $MX = I$, and

(b) Newton's lifting incorporates the updated approximate inverses $X^{(i)}$ to accelerate the lifting process.

Due to these features, Newton's lifting closely approximates the inverse in

much fewer lifting steps than Hensel's lifting, although at the expense of performing computations with a higher precision. Overall this leads to a little inferior (although still nearly optimal) upper estimate for the Boolean complexity of the solution, but an appropriate combination of Newton's and Hensel's liftings yields a smaller cost bound under the most realistic model of the word operations, that is yields a practical acceleration (see Section 10).

The initialization with an approximate inverse is shared by Newton's and Hensel's liftings, and our recipes apply to both of them.

Besides customary lifting modulo a random prime, we propose *binary versions* for both Hensel's and Newton's liftings, that is we initialize lifting modulo a power of two, which can be implemented more effectively, with the decrease of the CPU time of the classical lifting by twice for $n = 64$. Our techniques can be of independent interest, as well as the related study in [pw08], which shows that our binary lifting is unlikely to degenerate on the average input.

Our lifting and binary lifting algorithms for Toeplitz and Toeplitz-like inputs were implememted in Lehman College of the City University of New York by Brian Murphy (with some assistance from Rhys Eric Rosholt). In experimental computations the algorithms performed in good accordance with our theoretical study.

Our earlier study of lifting for structured inputs was summarized in the Technical Report available at www.cs.gc.cuny.edu/tr/files/TR-2008003.pdf. The report also covers the distinct MBA approach (see [pmr10] on its journal version).

The speedup in lifting versus the MBA alternative divide-and-conquer algorithm in [m80] and [ba80] is by the factor $r \log n$ for an $n \times n$ input matrix having a displacement rank r. Symbolic implementation of this algorithm is rather straightforward, except that one must recursively compress the displacement generators of the auxiliary matrices [pmr10]. This has been done deterministically over any field in [p01, Section 4.6.2].

Technically the lifting approach is distinct and has some further advantages, besides the cited acceleration: it has a simpler code, involves just a pair of $n \times n$ matrices and a single random prime of the order $n \log n$, and supports binary lifting, whereas the MBA algorithm (although it also supports the solution in a nearly optimum Boolean time) is slightly slower, involves over $2n$ matrices of various sizes and n random primes of the order $n \log(n||M||)$, and does not work modulo powers of two because of degeneration.

We organize the paper as follows. We devote the next section to the definitions and preliminaries, recall Hensel's lifting for linear systems of equations in Section 3, reconstruct the rational solution from the output of lifting in Section 4 and Appendix A, and estimate the Boolean complexity of our computations in Section 5 and Appendix B. We initialize Hensel's lifting by means of iterative refinement in Section 6 and extend our algorithms to the case of singular input in Section 7 and to polynomial computations and the Berlekamp–Massey's problem in Section 8. We cover the generalized (in particular binary) Hensel's lifting in Section 9. In Section 10 we present both Newton's and generalized Newton's lifting. In Section 11 we compare lifting with the MBA divide-and-conquer algorithm. In Section 12 we present our concluding remarks. In Appendix C we

briefly comment on the history of the Method of Displacement Transformation.

2 Definitions and basic facts

$\mathbb{Z}$ denotes the ring of integers, $\mathbb{Z}_q$ the ring of integers modulo an integer $q > 1$, and $\mathbb{Q}$ the field of rational numbers.

$\mathrm{ord}_q(m)$, the order of q in m, is the maximal integer b such that q^b divides m.

$m \mod q$ for two integers $q > 1$ and m can denote the class of integers $\{m + hq\}$ defined over all integers $h \in \mathbb{Z}$ or the unique integer in this class lying in the range $[0, q)$.

We write log for $\log_2$, "op" for "arithmetic operation", and $\tilde{O}(f(n))$ for $O(f(n)(\log\log n)^d)$ (for a constant d). The latter definition is not standard but is convenient for expressing our complexity estimates.

Fact 2.1. *A multiplication (resp. addition or subtraction) modulo 2^d uses $\mu(d) = O((d\log d)\log\log d) = \tilde{O}(d\log d)$ (resp. $O(d)$) bit operations, whereas $m(n) = O(n(\log n)\log\log n) = \tilde{O}(n\log n)$ field operations (resp. n additions or subtractions) suffice to multiply (resp. add or subtract) two polynomials in x modulo x^n over any algebra or ring with unity.*

Proof. See [gg03]. □

The upper bound on $\mu(d)$ was further decreased in [f07].

2.1 Rational number reconstruction

Definition 2.1. *$\nu(y)$ is the numerator and $\delta(y) \geq 1$ is the denominator in the ratio $y = \nu(y)/\delta(y)$ of two coprime integers $\nu(y)$ and $\delta(y)$.*

Modular rational reconstruction is the recovery of a rational number x/y from three integers k, l, and $r = (x/y) \mod l$ provided x and y are coprime unless $r = 0$, l and y are coprime, $|x| < k \leq l$, and $0 < y \leq l/k$ (we can write $x = r$, $y = 1$ if $k > |r|$).

$\rho(\log l)$ is the bit operation complexity of this recovery.

Fact 2.2. *The pair (x, y) is unique if $2|x| < k$.*

Proof. See [gg03]. □

Theorem 2.1. *For $\mu(d)$ in Fact 2.1 and constants C and c, $C > c > 0$, we have*

$$\rho(d) \leq cd^2, \quad \rho(d) \leq C\mu(d)\log d. \tag{2.1}$$

Proof. See [wp03]. □

The reconstruction is immediate if $\delta(y) = 1$.

Fact 2.3. *For three integers q, m, and $z = m \mod q$ such that $-0.5\ q < m \leq 0.5\ q$, we have $m = z$ if $2|z| \leq q$, $m = z - q$ otherwise.*

2.2 General matrices

$M = (m_{i,j})_{i,j=1}^{k,l} \in \mathcal{R}^{k\times l}$ and $\mathbf{v} = (v_i)_{i=1}^{k} \in \mathcal{R}^{k\times 1}$ for matrices and vectors with entries $m_{i,j}$ and v_i in a ring $\mathcal{R}$ of integers $\mathbb{Z}$ or $\mathbb{Z}_q$ or rationals $\mathbb{Q}$. I_k denotes the $k \times k$ identity matrix, with the columns $\mathbf{e}_i$, $i = 1, \dots, k$. We write just I where k is defined by the context.

Definition 2.2. *(K, L) is a 1×2 block matrix with the blocks K and L. M^T is the transpose of a matrix M. $M^{(h)}$ is its $h\times h$ leading principal block.* $\det M$ *and* $\operatorname{adj} M$ *are the determinant and the adjoint of a square matrix M, respectively. (*$\operatorname{adj} M = M^{-1} \det M$ *if M is a nonsingular matrix.) A matrix M of a rank ρ has* generic rank profile *if $\det M^{(k)} \neq 0$ for $k = 1, \dots, \rho$. M is* strongly nonsingular *if it is nonsingular and has generic rank profile.*

Definition 2.3. *For a matrix $M = (m_{i,j})_{i,j}$ and a vector $\mathbf{v} = (v_i)_i$, write $\alpha(M) = |M| = \max_{i,j} |m_{i,j}|$ and $|\mathbf{v}| = \beta(\mathbf{v}) = \max_i |v_i|$, so that $|\mathbf{v}|$ is the maximum norm of a vector $\mathbf{v}$.*

Definition 2.4. *m_S is the minimum number of ops sufficient to multiply a matrix S by a vector.*

Fact 2.4. *Let $M = (m_{i,j})_{i,j=1}^{n}$. Then $|\det M| \le \prod_j (\Sigma_i m_{i,j}^2)^{1/2} \le (\alpha(M)\sqrt{n})^n$ and $|\operatorname{adj} M| \le (\alpha(M)\sqrt{n-1})^{n-1}$.*

Hereafter $\mathbf{b} \neq \mathbf{0}$, $n > 2$, $|M| > 2$ (and so $\log n > 1$, $\log |M| > 1$).

2.3 Structured matrices

We apply Hensel's lifting where the input matrices have the popular and highly important structures of Toeplitz, Hankel, Vandermonde and Cauchy types. We call this class *THVC matrices*. The reader can find their extensive study in the book [p01] and the bibliography therein. These matrices generalize the four fundamental classes of Toeplitz, Hankel, Vandermonde and Cauchy matrices in Table 2.1. Such an $n \times n$ matrix is defined by cn parameters for $c \le 2$ (rather than by its n^2 entries) and can be multiplied by a vector fast (see Fact 2.5).

To extend the structures of the matrices M of the four basic classes, one can associate with them four classes of linear displacement operators L such that the *L-displacement* $L(M)$ has rank one for Vandermonde and Cauchy matrices and at most two for Toeplitz and Hankel matrices. (We specify some of these classes below and refer to [p01] for further information.) The four larger classes of matrices M for which the same operators L define displacements $L(M)$ of small ranks r make up the matrix class THVC. The notion "small" depends on the context, e.g., one may require that $r \le c$ or $r \le c \log n$ for a constant c and $n \times n$ matrices M.

The popular choices are the operators L of the Stein type such that $L(M) = \Delta_{A,B}(M) = M - AMB$ and of the Sylvester type such that $L(M) = \nabla_{A,B}(M) = AM - MB$, for fixed pairs of operator matrices A and B. The rank $r = \operatorname{rank}(L(M))$ is called the *displacement rank* of a matrix M. The following simple fact enables easy transition between the Stein and Sylvester representations.

Table 2.1: Four classes of structured matrices

Toeplitz matrices $(t_{i-j})_{i,j=0}^{n-1}$ $\begin{pmatrix} t_0 & t_{-1} & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}$	Hankel matrices $(h_{i+j})_{i,j=0}^{n-1}$ $\begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & h_n \\ \vdots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}$
Vandermonde matrices $\left(t_i^j\right)_{i,j=0}^{n-1}$ $\begin{pmatrix} 1 & t_0 & \cdots & t_0^{n-1} \\ 1 & t_1 & \cdots & t_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & t_{n-1} & \cdots & t_{n-1}^{n-1} \end{pmatrix}$	Cauchy matrices $\left(\frac{1}{s_i-t_j}\right)_{i,j=0}^{n-1}$ $\begin{pmatrix} \frac{1}{s_0-t_0} & \cdots & \frac{1}{s_0-t_{n-1}} \\ \frac{1}{s_1-t_0} & \cdots & \frac{1}{s_1-t_{n-1}} \\ \vdots & & \vdots \\ \frac{1}{s_{n-1}-t_0} & \cdots & \frac{1}{s_{n-1}-t_{n-1}} \end{pmatrix}$

Theorem 2.2. *(See [p01, Theorem 1.3.1].)* $\nabla_{A,B} = A\Delta_{A^{-1},B}$ *if the operator matrix* A *is nonsingular, and* $\nabla_{A,B} = -\Delta_{A,B^{-1}}B$ *if the operator matrix* B *is nonsingular.*

For $n \times n$ THCV matrices M one can choose appropriate pairs of $n \times n$ operator matrices A and B of shift $Z_f = \begin{pmatrix} 0 & & & & f \\ 1 & \ddots & & & 0 \\ & \ddots & \ddots & & \vdots \\ & & \ddots & 0 & 0 \\ & & & 1 & 0 \end{pmatrix}$ defined by scalars f and diagonal scaling $D_{\mathbf{s}} = \mathrm{diag}(s_i)_{i=1}^n$ defined by vectors $\mathbf{s} = (s_i)_{i=1}^n$. Hereafter $Z_f(\mathbf{v})$ denotes the Toeplitz matrix $\sum_{i=1}^n v_i Z_f^{i-1}$ defined by its first column $\mathbf{v} = (v_i)_{i=1}^n$, whereas $Z_f^T(\mathbf{v}) = (Z_f(\mathbf{v}))^T$ denotes the transpose of this matrix. $Z_0(\mathbf{v})$ (resp. $Z_0^T(\mathbf{v})$) is a lower (resp. an upper) triangular Toeplitz matrix. $Z_f(\mathbf{v})$ is an f-circulant matrix for $f \neq 0$.

Assuming the Stein displacements $L(M) = \Delta_{A,B}(M)$ and writing $r = \mathrm{rank}(L(M))$ we have $r < 3$ for Toeplitz matrices $M = T = (t_{i-j})_{i=0,j=0}^{m-1,n-1}$ provided $A = Z_e$, $B = Z_f^T$, and $ef \neq 1$; $r < 3$ for Hankel matrices $M = H = (h_{i+j})_{i=0,j=0}^{m-1,n-1}$ provided $A = Z_e$, $B = Z_f$, and $ef \neq 1$; $r = 1$ for Vandermonde matrices $M = V(\mathbf{t}) = (t_i^j)_{i=0,j=0}^{m-1,n-1}$ provided $A = D_{\mathbf{t}}$, $B = Z_f^T$, and $f \neq t_i^n$ for all i, and $r = 1$ for Cauchy matrices $M = C(\mathbf{s},\mathbf{t}) = (\frac{1}{s_i-t_j})_{i=0,j=0}^{m-1,n-1}$ provided $s_i t_j \neq 0$ for all pairs (i,j), $A = D_{\mathbf{s}}$ and $B = D_{\mathbf{t}_-} = \mathrm{diag}(t_j^{-1})_{j=0}^{n-1}$.

As we said earlier, the same linear displacement operators L define four classes of matrices M with the structures of Toeplitz, Hankel, Vandermonde,

and Cauchy types, respectively, such that $r = \text{rank}(L(M))$ is small. To take advantage of this property, recall that an $m \times n$ matrix W of a rank r can be expressed as the product GH^T of a pair of an $m \times r$ matrix G and an $r \times n$ matrix H^T. One can apply this observation to the displacements $W = L(M)$ and then yield bilinear or trilinear expressions for $n \times n$ matrices M of the above classes via the columns of the respective $n \times r$ matrices G and H that generate the displacements GH^T of a rank r. In particular we have the following result.

Theorem 2.3. *Assume two scalars e and f such that $ef \neq 1$, an $n \times n$ matrix M, and a pair of $n \times r$ matrices $G = (\mathbf{g}_j)_{j=1}^r$ and $H = (\mathbf{h}_j)_{j=1}^r$. Then $M = \sum_{j=1}^r Z_e(\mathbf{g}_j) Z_f(\mathbf{h}_j)^T$ if and only if $\Delta_{Z_e, Z_f^T}(M) = GH^T$.*

Proof. See [kkm79], [go94], [p01, Example 4.4.1]. □

A variety of such expressions in [go94] and [p01, Example 1.4.1 and Section 4.4] define $n \times r$ structured matrices M in terms of $2rn$ entries of their displacements GH^T (versus the n^2 entries of M). For $n \gg r$ this means a great saving of memory space and enables fast multiplication of such matrices by vectors.

Fact 2.5. *Assume an $n \times n$ matrix M with the structure of Toeplitz or Hankel (resp. Vandermonde or Cauchy) type and a displacement rank r. Then $m_M = O(rm(n))$ (resp. $m_M = O(rm(n) \log n)$) for m_M in Definition 2.4.*

Proof. See [p01, Section 4.2]. □

Fact 2.6. *For $n \times n$ matrices M and N with the structures of Toeplitz or Hankel types and displacement ranks r_M and r_N, resp., the matrices MN and $M + aN$ for a scalar a have structure of Toeplitz or Hankel type and have displacement ranks in $r_M + r_N + O(1)$. Their displacement generators of lengths in $O(r_M + r_N + 1)$ can be obtained in $m_M r_N + m_N r_M$ ops for MN, in $n \min\{r_M, r_N\}$ ops for $M + aN$, and op-free for $M + N$.*

Proof. See [p01, Theorems 1.5.1 and 1.5.4]. □

Here are some basic expressions for the displacement of the inverse.

Theorem 2.4. *Let M be a nonsingular matrix. Then*

$$\nabla_{B,A}(M^{-1}) = -M^{-1} \nabla_{A,B}(M) M^{-1}.$$

Furthermore,

$$\Delta_{B,A}(M^{-1}) = BM^{-1} \Delta_{A,B}(M) B^{-1} M^{-1},$$

if B is a nonsingular matrix, whereas

$$\Delta_{B,A}(M^{-1}) = M^{-1} A^{-1} \Delta_{A,B}(M) M^{-1} A,$$

if A is a nonsingular matrix.

Proof. See [p01, Theorem 1.5.3]. □

Corollary 2.1. *Let M be a nonsingular matrix with L-displacement $L(M) = GH^T$. Then $L_-(M^{-1}) = G_-H_-^T$ where*

(a) $G_- = -M^{-1}G$ and $H_-^T = H^TM^{-1}$ for $L = \nabla_{A,B}$ and $L_- = \nabla_{B,A}$,

(b) $G_- = BM^{-1}G$ and $H_-^T = H^TB^{-1}M^{-1}$ for $L = \Delta_{A,B}$, $L_- = \Delta_{B,A}$, and a nonsingular matrix B, and

(c) $G_- = M^{-1}A^{-1}G$ and $H_-^T = H^TM^{-1}A$ for $L = \Delta_{A,B}$, $L_- = \Delta_{B,A}$, and a nonsingular matrix A.

The corollary implies that the inversion of a matrix preserves its displacement rank.

Based on these and other results in [p01, Sections 1.4, 1.5, and 4.2–4.4], one can perform computations with the THVC matrices in terms of their displacement generators, thus dramatically saving the memory space and the computational time and unifying the algorithms for all these matrix structures.

Remark 2.1. *Further support for such a unification comes from the techniques of* displacement transformation *proposed in [p89/90] (see also [p01, Sections 1.7, 4.7–4.9]). These techniques reduce a linear system $M\mathbf{x} = \mathbf{f}$ for a matrix M with the structure of Vandermonde or Cauchy type and a displacement rank r to equivalent systems $MV_0\mathbf{y} = \mathbf{f}$ or $VMV_0\mathbf{y} = V\mathbf{f}$, respectively, where $\mathbf{x} = V_0\mathbf{y}$ and V (or V^T) and V_0 (or V_0^T) are appropriate Vandermonde matrices. Then displacement generators of length at most $r+2$ for the matrices MV_0 and VMV_0 are obtained at a lower cost and represent the structure of Toeplitz or Hankel type (see [p01, Section 4.7]). Therefore we can compute the solution vector $\mathbf{x}$ in $O(r^2m(n)\log^2 n)$ ops, due to the MBA algorithm in [m80], [ba80]. One can similarly extend any algorithm for computing the inverse or determinant of the matrices with the structure of any of Toeplitz, Hankel, Vandermonde, or Cauchy types to all THCV input matrices. The method has become celebrated when it served as the basis for devising effective practical algorithms for solving Toeplitz, Hankel, Toeplitz-like and Hankel-like linear systems of equations (see [gko95]).*

2.4 Multiplication of Toeplitz matrices and their inverses by vectors

Theorem 2.5. *Multiplication of an $n \times n$ Toeplitz matrix T by a vector is a subproblem of multiplication of two polynomials of degrees $2n-2$ and $n-1$ whose coefficients are given by the entries of the input matrix and vector, respectively, that is $m_T \le m(3n-3)$ for $m(n)$ in Fact 2.1 and m_T in Definition 2.4. If the Toeplitz matrix T is triangular and $m = n$, then both of these polynomials have degree $n-1$, that is in this case $m_T \le m(2n-2)$.*

Proof. See, e.g., [p01, pages 27–28]. □

The following theorem in [h79] (and also in [hr84]) extends the Gohberg–Semencul celebrated formula of 1972.

Theorem 2.6. *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 $t_{i-j} = t_{i,j}$ for $i, j = 0, \dots, n-1$;*

$p_n = -1$, $\mathbf{t} = (t_{i-n})_{i=0}^{n-1}$, $\mathbf{p} = (p_i)_{i=0}^{n-1} = T^{-1}\mathbf{t}$, $\mathbf{q} = (p_{n-i})_{i=0}^{n-1}$, $\mathbf{v} = T^{-1}\mathbf{e}_1$, ${\mathbf{e_1}}^T = (1, 0, \dots, 0)^T$, $\mathbf{u} = ZJ\mathbf{v}$. *Then* $T^{-1} = Z(\mathbf{p})Z^T(\mathbf{u}) - Z(\mathbf{v})Z^T(\mathbf{q})$.

Hereafter the $n \times 2$ matrix $(\mathbf{v}, \mathbf{p})$ for the above vectors $\mathbf{v}$ and $\mathbf{p} = \mathbf{p}(T, t_{-n})$ is called a *generator* for T^{-1}. The following theorem is a corollary of Theorems 2.5 and 2.6.

Theorem 2.7. *$i_T = m_{T^{-1}} \le 4m(2n-2) + n$ for i_S and m_S in Definition 2.4 and a nonsingular Toeplitz matrix T provided the matrix T^{-1} is given with its generator, that is, the vectors $\mathbf{v}$ and $\mathbf{p}$ in Theorem 2.6.*

2.5 Randomization

Randomized algorithms produce correct output with a probability of at least $1 - \epsilon$ for a fixed tolerance ϵ. The randomized complexity estimates are of the *Las Vegas* type if they cover the cost of the correctness verification. Otherwise they are of the *Monte Carlo* type.

Theorem 2.8. *For a positive ϵ, a nonsingular matrix $M \in \mathbb{Z}^{n\times n}$, and the scalar $\xi = \frac{16 \ln 114}{16 \ln 5.7 - \ln 114} = 3.278885\dots$, let $y = \frac{n\xi \ln |M|}{\epsilon} \ge 114$ and let a prime p be randomly sampled from the range $(y/20, y]$ under the uniform probability distribution in it. Then* Probability$((\det M) \bmod p = 0) < \epsilon$.

Proof. See [pw08]. □

3 Hensel's lifting

The first adaptations of Hensel's classical lifting [gg03] to the symbolic solution of an integer linear system of equations $M\mathbf{x} = \mathbf{f}$ were proposed in [mc79], [d82]. The lifting algorithm computes the first h terms in the vector expansion

$$M^{-1}\mathbf{f} = \sum_{i=0}^{\infty} \mathbf{u}^{(i)} s^i, \quad \mathbf{u}^{(i)} \in \mathbb{Z}_s^n, \quad i = 0,\ 1,\ \dots.$$

Algorithm 3.1. Hensel's lifting [d82].
INPUT: *a matrix $M \in \mathbb{Z}^{n\times n}$, a vector $\mathbf{f} \in \mathbb{Z}^n$, two integers $h > 0$ and $s > 1$, and a matrix $Q \in \mathbb{Z}_s^{n\times n}$ such that*

$$MQ = I \mod s. \tag{3.1}$$

OUTPUT: *the vector $\mathbf{x}^{(h)} \in \mathbb{Z}^n$ such that $M\mathbf{x}^{(h)} = \mathbf{f} \bmod (s^h)$.*
INITIALIZATION: $\mathbf{r}^{(0)} = \mathbf{f}$.
COMPUTATIONS: *for $i = 0, 1, \dots, h-1$, compute the vectors $\mathbf{u}^{(i)} = Q\mathbf{r}^{(i)} \bmod s$, $\mathbf{r}^{(i+1)} = (\mathbf{r}^{(i)} - M\mathbf{u}^{(i)})/s$. Output the vector $\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} \mathbf{u}^{(i)} s^i$.*

Part (b) of the following theorem shows correctness of the algorithm. Part (c) bounds the precision of the computations by this algorithm.

Theorem 3.1. *For $M = (m_{i,j})_{i,j=1}^{n}$, $\mathbf{f} = (f_j)_{j=1}^{n}$, $\mathbf{r}^{(i)}$ and $\mathbf{x}^{(h)}$ in Algorithm 3.1, we have*
(a) $\mathbf{r}^{(i)} \in \mathbb{Z}^n$ for all i,
(b) $M\mathbf{x}^{(h)} = \mathbf{f} \bmod (s^h)$,
(c) all components $r_j^{(i)}$ of all vectors $\mathbf{r}^{(i)} = (r_j^{(i)})_j$ satisfy the bounds

$$|r_j^{(i)}| \le \frac{1}{s^i}|f_j| + (s-1)\alpha n \sum_{k=1}^{i} s^{-k} < \frac{\beta}{s^i} + \alpha n < \gamma$$

where

$$\beta = \max_j |f_j|, \quad \alpha = \max_{i,j} |m_{i,j}|, \quad \gamma = 2\alpha n + \beta. \tag{3.2}$$

Proof. See [d82]. □

Lemma 3.1. *Algorithm 3.1 operates with integers in the range $[-2^{d_1}, 2^{d_1})$ where*

$$d_1 \le \lceil \log(2s\gamma) \rceil \quad \text{for } \gamma \text{ in } \quad (3.2). \tag{3.3}$$

Proof. The lemma follows from parts (a) and (c) of Theorem 3.1 because the vectors $\mathbf{u}^{(i)}$ are computed in $\mathbb{Z}_s$. □

Theorem 3.2. *Each lifting step performs $m_M + O(n)$ ops with the precision d_1 in (3.3) and m_Q ops (for the vectors $\mathbf{u}^{(i)}$) with the precision $d_0 = \lceil \log s \rceil$, which means $(m_M + O(n))\mu(d_1) + m_Q\mu(d_0)$ bit operations per step for m_M in Definition 2.4 and $\mu(d)$ in Fact 2.1.*

If λ is the length of a computer word and $d_1 < \lambda$, then all ops in the algorithm are word operations, that is performed within the computer precision. We can save lifting steps and word operations by applying *saturated initialization* such that s maximizes d_1 subject to the bound $d_1 < \lambda$.

4 Reconstruction of rational solutions

Our next task is the reconstruction of the rational solution $\mathbf{x}$ to the equation $M\mathbf{x}$=$\mathbf{f}$ from the vector $\mathbf{x} \bmod p^h = M^{-1}\mathbf{f} \bmod s^h$ for a larger h. The techniques go back to [p87, Appendix] and [p88] and more recently were used in [abm99], [cfg99], [egv00], and [ms04]. Next we outline these techniques. On further details see Appendix A.

Theorem 4.1. *Let $\mathbf{x} = M^{-1}\mathbf{f}$ be a unique solution to the linear system $M\mathbf{x} = \mathbf{f}$. Assume $\rho(d)$ in Fact 2.1 and α, β and γ in (3.2). Write*

$$l = \lceil \log(2(\alpha\sqrt{n})^{2n-1}\beta) \rceil = O(n \log \gamma), \tag{4.1}$$

$$h = \lfloor \log_s(2(\alpha\sqrt{n})^{2n-1}\beta) \rfloor + 1. \tag{4.2}$$

Let the vector $\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} \mathbf{u}^{(i)}p^i = \mathbf{x} \bmod (s^h)$ be computed in $h-1$ steps of Algorithm 3.1. Then one can recover the vector $\mathbf{x}$ from the vector $\mathbf{x}^{(h)}$ by performing $B = n\rho(l)$ bit operations.

Proof. Suppose two coprimes $\nu_j = \nu(x_j)$ and $\delta_j = \delta(x_j)$ define the rational components $x_j = \nu_j/\delta_j$ of the vector $\mathbf{x} = (x_j)_j = M^{-1}\mathbf{f}$. Fix the smallest integer k exceeding $2(\alpha\sqrt{n-1})^{n-1}\beta$. Note that $s^h > 2(\alpha\sqrt{n})^{2n-1}\beta$ for h in (4.2). Recall that $M^{-1}\det M = \operatorname{adj} M$ and deduce from Fact 2.4 that $s^h > 2|\nu_j|\delta_j$ and $2|\nu_j| < k \le s^h$. Then according to Section 2.1, we can uniquely recover every component x_j from $qx_j \bmod (qs^h)$ in $\rho(l)$ bit operations. □

Remark 4.1. *We can accelerate reconstruction by the factor* $\log l$ *by applying Las Vegas randomization. Namely for* $\mathbf{x} = (x_i)_i$*, random integers* $c_i^{(j)}$*, and fixed (reasonably small) integer* K*, we can compute* δ_{lcm}*, the lcm of the integer denominators* δ_j *of the rationals* $\nu_j/\delta_j = \sum_i c_i^{(j)} x_i$ *for* $j = 1, 2, \dots, K$*. One can estimate that this lcm is likely to coincide with the lcm of all denominators* $\delta(x_1), \dots, \delta(x_n)$ *of the* n *rational coordinates* x_i *of the vector* $\mathbf{x}$*. If indeed so, then the vector* $\delta_{\mathrm{lcm}}\mathbf{x}$ *is filled with integers and can be readily reconstructed from its value modulo* p^h*, which we can verify by checking whether* $M\mathbf{x} = \mathbf{f}$*. We specify these techniques and the resulting complexity estimates in Appendix A.*

Remark 4.2. *If the vector* $\mathbf{x} \bmod p^h$ *and the integer* $(\det M) \bmod p^h \neq 0$ *are available, we can compute the integer vector* $(\det M)\mathbf{x}$ *without applying Theorem 2.1. Then reconstruction needs no randomization and goes faster by logarithmic factor, and similarly where the output is known to be integral, as this occurs, e.g., for Berlekamp–Massey's problem (see Definition 8.2) and at the final stage of Wiedemann's algorithm [w86], which computes the minimal polynomial, determinant, and Smith's factors of an integer matrix.*

5 Computational complexity estimates

By combining equation (4.2), Theorems 2.1, 3.2, and 4.1, and Remark 4.1 we obtain the following estimates.

Theorem 5.1. *Lifting and the rational reconstruction together use at most* $((m_M + O(n))\mu(d_1) + m_Q\mu(\log s))h + O(n\rho(l))$ *bit operations for* m_M *in Definition 2.4,* $\mu(d)$ *in Fact 2.1,* $\rho(d)$ *in Section 2.1,* d_1 *in (3.3),* l *in (4.1), and* h *in (4.2). This covers the solution of a nonsingular linear system of* n *equations* $M\mathbf{x} = \mathbf{f}$ *given an integer* $s > 1$*, a vector* $\mathbf{f}$*, and two matrices* M *and* Q *satisfying the equation* $MQ = I \bmod s$*. By allowing Las Vegas randomization at the reconstruction stage, one can decrease the term* $O(n\rho(l))$ *by the factor* $\log l$*.*

Table 5.1 summarizes the estimates for the overall randomized Las Vegas complexity of the exact solution of a nonsingular linear system $M\mathbf{x} = \mathbf{f}$ where

$$\log_n p = O(1), \ \log_n(1/\epsilon) = O(1), \ \log_p \gamma = O(1), \tag{5.1}$$

for $s = p$ being a prime and ϵ denoting a fixed tolerance to the error probability in the randomized rational reconstruction.

In Appendix B we do not assume equations (5.1) and specify more detailed estimates.

Table 5.1: Randomized Boolean complexity under equations (5.1).

Lifting	$O((m_M + m_Q)n\mu(\log n)) = \tilde{O}((m_M + m_Q)n \log n)$
Reconstruction	$O(n\mu(n \log n) + \rho(n \log n) \log n) = \tilde{O}(n^2 \log^2 n)$

Theorem 5.2. *Recall the definitions of $\mu(d)$ and $m(n)$ in Fact 2.1. Assume a nonsingular $n \times n$ integer linear system of n equations whose coefficient matrix has structure of Toeplitz, Hankel, Vandermonde, or Cauchy type and is given with a displacement of a rank r. Assume equations (5.1) and Las Vegas randomization with $O(n \log n)$ random bits. Then the lifting cost estimate in Table 5.1 turns into $O(rm(n)n\mu(\log n)) = \tilde{O}(rn^2 \log^2 n)$.*

Proof. For an input with the structure of Toeplitz or Hankel type the theorem follows from Theorem 5.1 and Fact 2.5. To extend this result to an input matrix M with the structure of the Vandermonde or Cauchy type, apply the method of displacement transformation (see Remark 2.1) and Fact 2.5. □

Remark 5.1. *(a) Recall that $\gamma = 2\alpha n + \beta$, and so equations (5.1) imply that $\log_p(\alpha + \beta) = O(1)$.*

(b) The same equations imply that $\log_n \gamma = O(1)$.

(c) They also imply that the output size (that is the logarithm of the maximum absolute value of the numerators and denominators of the rational output values) is in $O(n \log n)$.

(d) No randomization is needed at the lifting stage, and one can also reconstruct the rational solution deterministically at the cost of performing $n\rho(n \log n)$ ops in $O(n^2 \log^3 n)$ bit operations (see Theorem 4.1); this slows down our randomized recovery just by the factor $\log n$.

(e) Randomized cost of the initialization of lifting (not covered in Table 5.1 and Theorem 5.2) is estimated in the next section.

The estimates of Theorem 5.2 for lifting, of Section 4 for the solution reconstruction, and of Table 5.1 for both stages sum to $\tilde{O}(n^2 \log^2 n)$ bit operations if $r = O(1)$.

If λ, the length of a computer word, exceeds $\log\lceil 2\gamma p \rceil$, so that lifting and initialization are performed within the computer precision (see Lemma 3.1), then the word operation cost of performing these stages is by the factor of λ smaller than the bit–complexity estimates.

6 Initialization by means of iterative refinement

6.1 Introductory comments

Given a matrix M, a prime p, and its power $m = p^b$, we seek an integer matrix Q such that $MQ = I \mod m$. More precisely, we assume that the matrix M is given with a generator $G = (\mathbf{g}_i)_{i=1}^r$, $H = (\mathbf{h}_i)_{i=1}^r$ of a length r for its Sylvester displacement $GH^T = \sum_{i=1}^r \mathbf{g}_i \mathbf{h}_i^T$ and we seek a displacement generator $YZ^T = \sum_{i=1}^r \mathbf{y}_i \mathbf{z}_i^T$ of length r for the matrix $Q = M^{-1} \mod m$.

Due to Theorem 2.4, the problem is reduced to the solution of $2r$ linear systems of equations $-M\mathbf{y}_i = \mathbf{g}_i$ and $\mathbf{z}_i^T M = \mathbf{h}_i^T$, $i = 1, \dots, r$. It remains to supply an initialization algorithm for lifting, that is for solving linear systems $M^T\mathbf{y} = \mathbf{f} \mod m$ and $M\mathbf{z} = \mathbf{f} \mod m$ for any vector $\mathbf{f}$, and then we can solve the $2r$ linear systems above based on Hensel's lifting.

We cover just the case of the latter system (for the former system the solution is similar). We employ numerical iterative refinement in [gl96, Section 3.5.3] and [pgmq, Section 10] to yield the desired solution vector $\mathbf{y}$ in $O((n \log(|M| + p^b) + \log |\mathbf{f}|)m_M)$ ops.

6.2 The basic algorithm

Our basic initialization algorithm employs the algorithm in [wp03] for *numerical rational reconstruction*, which recovers a unique rational number $\frac{x}{y}$ from three integers ν, δ, and k provided $1 \le y \le k$, $|x| < k$, $|x|$ and y are coprime unless $x = 0$; $|\frac{x}{y} - \frac{\nu}{\delta}| < \frac{1}{2k^2}$, and $|\nu| < \delta$. The bit-complexity bound of Theorem 2.1 for $d = \delta$ can be applied (see [wp03]). The algorithm employs the following rounding policy (see Lemma 6.1 below).

Definition 6.1. *Represent all components of a vector* $\mathbf{v}$ *as fixed-point numbers with the common fixed point placed in front of the first nonzero digit of the absolutely largest component, so that the fractions of the other components can begin with zeros. Then round all fractions to the closest* t*-digit numbers that all share this fixed point, so that the zeros that follow the fixed point are counted among the* t *digits. The resulting vector* $\tilde{\mathbf{v}}$ *is said to approximate the vector* $\mathbf{v}$ *with the* t*-digit fixed-point precision and with rounding to the closest values.*

Lemma 6.1. *If a vector* $\tilde{\mathbf{v}}$ *approximates the vector* $\mathbf{v}$ *with the* t*-digit fixed-point precision and with rounding to the closest values, then* $|\tilde{\mathbf{v}} - \mathbf{v}| \le 0.5|\mathbf{v}|/\phi^t$ *assuming* ϕ*-ary digits.*

Algorithm 6.1. *Initialization of lifting based on iterative refinement.*
INPUT: *A nonsingular matrix* $M \in \mathbb{Z}^{n \times n}$*, a vector* $\mathbf{f} \in \mathbb{Z}^n$*, a prime* p*, and positive integers* b*,* m*, and* t *such that* $m = p^b \ge 2^{t+2}|M|$.
OUTPUT: *either FAILURE if* $\delta((M)^{-1}\mathbf{f})_j) \mod p = 0$ *for some* j*,* $j = 1, \dots, n$ *(see Definition 2.1), or the vector* $\mathbf{z} = (M^{-1}\mathbf{f}) \mod m$ *otherwise.*
INITIALIZATION: *Write* $\mathbf{r}_0 = \mathbf{f}$*,* $M_0 = M + mI$*, and* $Q = I/m$.
COMPUTATIONS:

1. Write $\mathbf{w}_i = Q\mathbf{r}_i = \mathbf{r}_i/m$ *and recursively, for* $i = 0, 1, \dots, \bar{h} - 1$ *and (cf. (4.2))*

$$\bar{h} = \lceil ((2n-1)\log(|M| + m) + \log(2|\mathbf{f}|^2))/t \rceil, \tag{6.1}$$

compute (a) the vectors $\tilde{\mathbf{w}}_i = \mathbf{w}_i + \mathbf{e}_i$ *that approximate the vectors* $\mathbf{w}_i$ *with the* t*-bit fixed-point precision and with rounding to the closest values (see Definition 6.1) and (b) the error-free residual vectors* $\mathbf{r}_{i+1} = \mathbf{r}_i - M_0\tilde{\mathbf{w}}_i = \mathbf{r}_i - M\tilde{\mathbf{w}}_i - m\tilde{\mathbf{w}}_i$.
2. Recover the vector $\mathbf{z} = M_0^{-1}\mathbf{f}$ *from the vector* $\mathbf{z}_{\bar{h}} = \sum_{i=0}^{\bar{h}-1} \tilde{\mathbf{w}}_i$*, by using the numerical rational reconstruction algorithm in [wp03].*
3. If p *divides at least one of the integers* $\delta((M_0^{-1}\mathbf{f})_j)$*,* $j = 1, \dots, n$*, then output FAILURE. Otherwise compute and output the vector* $\mathbf{z} = (M_0^{-1}\mathbf{f}) \bmod m = (M^{-1}\mathbf{f}) \bmod m$.

FAILURE is output if and only if $v = \max_j \operatorname{ord}_p(\delta((M_0^{-1}\mathbf{f})_j)) > 0$. Stage 1 produces the output values beyond the double precision by extending the customary numerical algorithm for iterative refinement.

Theorem 6.1. $\max_j \operatorname{ord}_p(\delta((M_0^{-1}\mathbf{f})_j)) \le \operatorname{ord}_p(\det M_0) = \operatorname{ord}_p(\det M)$.

Proof. The theorem follows because the integers $\delta((M_0\mathbf{f})_j)$ divide $\det M$ for all j. □

For a fixed nonsingular integer matrix M and random prime p in a reasonably large range, the integers p and $\det M$ are likely to be coprime (see Fact 2.4 and Theorem 2.8), and if they are coprime the algorithm does not fail.

If it fails, one can apply some heuristic recipes from [pw08] or reapply the algorithm either for a distinct prime p or for its larger power p^b. Also see [pq10], [pqa], and the references therein on the alternative methods of randomized preprocessing.

6.3 Correctness of the algorithm

With no loss of generality assume that $M \in \mathbb{Z}_m^{n\times n}$ and $\mathbf{f} \in \mathbb{Z}_m^n$.

Lemma 6.2. *We have* $\mathbf{z} - \mathbf{z}_{\bar{h}} = M_0^{-1}\mathbf{r}_{\bar{h}}$.

Proof. Combine the equations $\mathbf{z}_{\bar{h}} = \sum_{i=0}^{\bar{h}-1} \tilde{\mathbf{w}}_i$ and $M_0\tilde{\mathbf{w}}_i = \mathbf{r}_i - \mathbf{r}_{i+1}$ for all i to obtain $M_0\mathbf{z}_{\bar{h}} = \mathbf{r}_0 - \mathbf{r}_{\bar{h}}$, so that $\mathbf{r}_0 - M_0\mathbf{z}_{\bar{h}} = \mathbf{r}_{\bar{h}}$. Premultiply this equation by M_0^{-1} and substitute $M_0^{-1}\mathbf{r}_0 = M_0^{-1}\mathbf{f} = \mathbf{z}$. □

Lemma 6.3. *We have* $|\mathbf{r}_{i+1}| \le |\mathbf{r}_i|/2^t \le |\mathbf{f}|/2^{(i+1)t}$ *for all* i*,* $i = 0, 1, \dots$.

Proof. Recall that $\mathbf{r}_{i+1} = \mathbf{r}_i - M_0\tilde{\mathbf{w}}_i = \mathbf{r}_i - M_0\mathbf{w}_i - M_0\mathbf{e}_i$, whereas $\mathbf{w}_i = \mathbf{r}_i/m$, and so $\mathbf{r}_i - M_0\mathbf{w}_i = (I - M_0/m)\mathbf{r}_i = -(M/m)\mathbf{r}_i$. It follows that $\mathbf{r}_{i+1} = -(M/m)\mathbf{r}_i - M_0\mathbf{e}_i$, and so $|\mathbf{r}_{i+1}| \le |(M/m)\mathbf{r}_i| + |M_0\mathbf{e}_i|$. Let us estimate both terms on the right hand side.

We have $|\mathbf{e}_i| \le |\mathbf{w}_i|/2^{t+1}$ in virtue of Lemma 6.1 applied for $\phi = 2$. Now deduce that $|\mathbf{r}_{i+1}| = |(M/m)\mathbf{r}_i| \le |\mathbf{r}_i|/2^{t+2}$ and $|M_0\mathbf{e}_i| \le |M + mI|\ |\mathbf{w}_i|/2^{t+1} = |M + mI|\ |\mathbf{r}_i|/(m2^{t+1}) \le 3|\mathbf{r}_i|/2^{t+2}$.

Finally combine the two latter bounds with the bound $|\mathbf{r}_{i+1}| \le |(M/m)\mathbf{r}_i| + |M_0\mathbf{e}_i|$ above. □

Corollary 6.1. *We have* $|\mathbf{z} - \mathbf{z}_{\bar{h}}| \le |M_0^{-1}|\ |\mathbf{f}|/2^{t\bar{h}}$.

Numerical rational reconstruction ensures correct recovery of the vector $\mathbf{z}$ from $\mathbf{z}_{\bar{h}}$ if $|\mathbf{z} - \mathbf{z}_{\bar{h}}| < 1/(2|M_0|^{2n-1}|\mathbf{f}|)$. Due to Corollary 6.1, this bound is reached under (6.1).

6.4 The computational precision and the Boolean cost

The complexity of Algorithm 6.1 has already been estimated in Section 5, but for M_0 replacing M and with the distinct precision of the computations. It remains to estimate the adjusted precision.

By the definition of the vectors $\tilde{\mathbf{w}}_i$, the binary representations of the components of the vector $\mathbf{r}_{i+1} = \mathbf{r}_i - M_0\tilde{\mathbf{w}}_i$ extend rightward by at most $t + \lceil \log m \rceil$ bits from the leading bit of the value $|\mathbf{r}_i|$. At the same time this leading bit itself moves rightward by at least t bits when we move from $\mathbf{r}_i$ to $\mathbf{r}_{i+1}$ because $|\mathbf{r}_{i+1}| \le |\mathbf{r}_i|/2^t$ (see Lemma 6.3). Thus it is sufficient to use a precision of at most $\lceil \log m \rceil$ bits for all components of the vectors $\mathbf{r}_i$ for all i. We increase this precision by at most $\lceil \log |M| \rceil$ and $\lceil \log m \rceil$ bits when we compute the vectors $M\tilde{\mathbf{w}}_i$ and $m\tilde{\mathbf{w}}_i$, respectively. It follows that the asymptotic complexity estimates in Section 5 can be extended to Algorithm 6.1 for $\log m = b \log p = O(\log(n\alpha))$.

Our resulting randomized bit operation complexity estimates for solving structured linear systems are record low. Furthermore, for an $n \times n$ matrix M with the displacement of a constant rank and with entries having the absolute values in $n^{O(1)}$, the estimates are nearly optimal because $n^2 \log n$ bits are required to represent the n rational coordinates of the vector $\mathbf{x}$, and so computing these values takes at least as many bit operations. Without randomization we need to increase the cost bound by the factor n to avoid degeneration of the matrix $M \mod s$.

7 Computations with singular matrices

Assume that the matrix M has the generic rank profile property. (This property holds with a probability near one if we apply randomized preprocessing in Theorem 7.2 below.) Apply Algorithm 6.1 recursively to the $j \times j$ leading principal submatrices of the matrix M for $j = j(i) = 2^i$, $i = 0, 1, \dots, k$ until for some integer $i = k + 1$ the algorithm fails. Then apply binary search in the range $[2^k, 2^{k+1})$ for the maximum integer $\tilde{\rho}$ for which the algorithm does not fail. This integer is likely to equal the rank ρ and cannot be less than the rank. Such a search of the rank increases the overall cost by a factor in $O(\log \rho)$ (see Corollary 7.2 and Section 11).

Theorem 7.1. *Let a singular integer THVC matrix M have generic rank profile and be given with its displacement generator of length r. Then at a randomized*

Las Vegas cost within the factor $O(r \log \rho)$ from the estimates in Section 5, one can compute the rank ρ of this matrix and a shortest displacement generator for a matrix whose columns form a basis for the null space of the matrix M.

Proof. Due to the method of displacement transformation (see Section 2.3 and recall that our Vandermonde multipliers are nonsingular), we can assume the structure of Toeplitz or Hankel type for the matrix M. Now we can verify whether the candidate integer $\tilde{\rho}$ is indeed equal to the rank ρ as follows. Represent the matrix M as the 2×2 block matrix $\begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix}$ with the nonsingular block $M_{00} = M^{(\tilde{\rho})}$ and compute a displacement generator for its Schur complement $S = M_{11} - M_{10}M_{00}^{-1}M_{01}$. We immediately observe that $\tilde{\rho} = \rho$ if and only if $S = 0$ (and then the displacement of the matrix S vanishes as well). If $S = 0$ indeed, then the columns of the matrix $\begin{pmatrix} M_{00}^{-1}M_{01} \\ -I_{n-\rho} \end{pmatrix}$ form a basis for the null space of the matrix M.

The arithmetic cost of computing displacement generators for the matrices $M_{00}^{-1}M_{01}$ and S is in $O(rm_M)$ (see Fact 2.6). It remains to bound the precision of computing based on Fact 2.4. □

Remark 7.1. *If we agree to use the Monte Carlo estimates for the rank computation we can decrease the bound of Theorem 7.1 by roughly the factor $\frac{\log h}{\log \gamma}$ for h in equation (4.2) and γ in (3.2) (cf., e.g., [p96], [pw08, Appendix A]).*

Theorem 7.2. *Suppose that a finite set S of a sufficiently large cardinality $|S|$ lies in a field $\mathbb{F}$. Further assume that a matrix M belong to $\mathbb{F}^{n \times n}$. Define randomized preprocessing $M \leftarrow XMY$ where $X = X_g$, $Y = Y_h$ for $g, h \in \{1, 2\}$, $X_1 = (\frac{x_j}{s_i - t_j})_{i=1}^n$, $Y_1 = (\frac{y_j}{u_i - v_j})_{i=1}^n$, s_i, t_j, u_i, and v_j are $4n$ distinct values, $X_2 = Z_0^T(\mathbf{x})$, $Y_2 = Z_0(\mathbf{y})$, $x_1 = y_1 = 1$, and the other $2n - 2$ coordinates of the vectors $\mathbf{x} = (x_i)_{i=1}^n$ and $\mathbf{y} = (y_i)_{i=1}^n$ are randomly sampled from the set S. Then both matrices X_2 and Y_2 are nonsingular and with a probability of at least $(1 - \rho/|S|)^2$ both matrices X_1 and Y_1 are nonsingular. If the matrices X and Y are nonsingular, then with a probability of at least $1 - (\rho + 1)\rho/|S|$ the matrix XMY has generic rank profile (and therefore is strongly nonsingular if the matrix M is nonsingular).*

Proof. See [ks91] where $X = X_2$, $Y = Y_2$ and [p01, Corollary 5.6.3] where $X = X_1$, $Y = Y_1$. □

Each of the multipliers X_1, Y_1, X_2, and Y_2 has displacement rank one assuming the displacement operators $\nabla_{D_{\mathbf{s}}, D_{\mathbf{t}}}$, $\nabla_{D_{\mathbf{u}}, D_{\mathbf{v}}}$, $\Delta_{Z_0^T, Z_0}$, and Δ_{Z_0, Z_0^T}, respectively (cf. Section 2.3), and so Fact 2.6 and the above theorem imply the following result.

Corollary 7.1. *To extend Theorem 7.1 to the case of input matrices not having generic rank profile it is sufficient to perform $O(m_M)$ additional ops (which preserves the Boolean and word cost bounds) and to generate $2n - 2$ random parameters in the field $\mathbb{F}$.*

Proof. Extend the dispacement structure of the matrix M to the matrix XMY by choosing appropriate matrices $X = X_i$ and $Y = Y_i$ for $i = 1, 2$ (see [p01, Sections 5.6 and 5.7]). □

Corollary 7.2. *Let an $n \times n$ integer THVC matrix M satisfy the assumptions of Theorem 7.2 and bounds (5.1), and suppose we seek the integer $\rho = \operatorname{rank} M$ and displacement generators of length $O(r)$ for a nonsingular $\rho \times \rho$ submatrix W of the matrix XMY, for the inverse W^{-1}, and for a matrix whose columns make up a basis for the null space of the matrix M. To solve all these problems it is sufficient to apply Las Vegas randomization with $O(n \log n)$ random bits that define the two matrices X and Y in Theorem 7.2 and in addition to perform $\tilde{O}(r^2 n^2 (\log^2 n)(\log \rho))$ bit operations.*

8 Padé approximation and related computations

Definition 8.1. Padé approximation.
For two nonnegative integers m and n and a polynomial $t(x) = \sum_{i=0}^{m+n} t_i x^i$, an (m, n) Padé approximation is a pair of coprime polynomials $r(x) = \sum_{i=0}^{m} r_i x^i$ and $v(x) = \sum_{i=0}^{n} v_i x^i$ satisfying equation $r(x) = t(x)v(x) \mod x^{m+n+1}$. (The ratio $\frac{r(x)}{v(x)}$ is unique.)

Theorem 8.1. *A randomized Las Vegas algorithm for an (m, n) Padé approximation of a polynomial $t(x) = \sum_{i=0}^{m+n} t_i x^i$ generates $O(N \log N)$ random bits and in addition performs $O(N^2 \log^3 N)$ bit operations provided $N = n + m$ and equations (5.1) hold for $\gamma = \max_{i=0}^{N} |t_i|$.*

Proof. First recall from [bgy80] or [p01, Section 2.11] that the task of computing an (m, n) Padé approximation can be reduced to the solution of the Toeplitz linear system $T\mathbf{v} = -v_0\mathbf{t}$ where $T = (t_{m+i-j})_{i,j=0}^{n-1}$, $\mathbf{v} = (v_{j+1})_{j=0}^{n-1}$, $\mathbf{t} = (t_{j+1})_{j=0}^{n-1}$, $v_0 = 1$ if $\rho = \operatorname{rank}(T) = n$, and $v_0 = 0$ if $\rho = \operatorname{rank}(T) < n$, in which case $\det(T^{(\rho)}) \neq 0$, that is the $\rho \times \rho$ leading principal block of the matrix T is nonsingular. It remains to apply the algorithms supporting Corollary 7.2 for $r < 3$ to solve this Toeplitz linear systems. □

Definition 8.2. Berlekamp–Massey's problem.
Given a positive integer s and $2s$ numbers $a_0, a_1, \ldots, a_{2s-1}$, compute the minimum integer $n \leq s$ and n numbers $c_0, c_1, \ldots, c_{n-1}$ such that $a_i = c_{n-1}a_{i-1} + \cdots + c_0 a_{i-n}$ for $i = n, n+1, \ldots, 2s-1$.

Fact 8.1. *Berlekamp–Massey's problem has a unique solution, given by the degree n and the coefficients $c_0, c_1, \ldots, c_{n-1}$ of the minimum span polynomial $c(x) = x^n - \sum_{i=0}^{n-1} c_i x^i$ such that for some polynomial $r(x)$ the pair of polynomials $(r(x), c(x))$ is an $(s-1, s-1)$ Padé approximation to the polynomial $a(x) = \sum_{i=0}^{2s-1} a_i x^i$.*

Proof. See [bgy80]. □

Corollary 8.1. *A randomized Las Vegas algorithm for Berlekamp–Massey's problem, for a positive integer s and $2s$ numbers a_0, a_1, ..., a_{2s-1}, generates $O(s \log s)$ random bits and in addition performs $O(s^2 \log^3 s)$ bit operations if equations (5.1) hold for $\gamma = \max_{i=0}^{2s-1} |a_i|$.*

Definition 8.3. $\gcd(\mathrm{u}, \mathrm{w})$*, the greatest common divisor of two polynomials $u(x) = \sum_{i=0}^{m} u_i x^i$ and $w(x) = \sum_{i=0}^{n} w_i x^i$ is their common divisor of the largest degree, whereas their least common multiple* $\mathrm{lcm}(\mathrm{u}, \mathrm{w}) = u(x)w(x)/\gcd(\mathrm{u}, \mathrm{w})$ *is their common multiple of the smallest degree. (Monic gcd and monic lcm are unique.)*

Fact 8.2. *Let the pair $(r(x), v(x))$ be an (m, n) Padé approximation to the polynomial $t(x) = \sum_{i=0}^{m+n} t_i x^i$ such that $t(x)w(x) \mod x^{m+n+1} = u(x)$ and let $d(x)$ be a polynomial such that $u(x) = d(x)r(x)$. Then $w(x) = d(x)t(x)$ and $d(x)$ is a $\gcd(u, w)$.*

Theorem 8.2. $g(x) = \gcd(\mathrm{u}, \mathrm{w})$*, greatest common divisor and* $\mathrm{lcm}(\mathrm{u}, \mathrm{w})$*, least common multiple of two polynomials $u(x) = \sum_{i=0}^{m} u_i x^i$ and $w(x) = \sum_{i=0}^{n} w_i x^i$ can be computed by a randomized Las Vegas algorithm that generates $O(N \log N)$ random bits and in addition performs $\tilde{O}(N^2 \log^3 N)$ bit operations provided equations (5.1) hold for $\gamma = \max\{\max_{i=0}^{m} |u_i|, \max_{i=0}^{n} |w_i|\}$ and $N = n + m$.*

Proof. We first compute the degree $k = \deg g(x)$ of the polynomal $g(x) = \sum_{i=0}^{k} g_i x^i = \gcd(\mathrm{u}, \mathrm{w})$ by applying the reduction gcd $\rightarrow$ Padé (see Fact 8.2) and then applying our algorithm that supports Theorem 8.1. To avoid the growth of the coefficients and the cost bounds in the transition gcd $\rightarrow$ Padé, we compute $\deg g(x)$ in $\mathbb{Z}_p$ for a reasonably large prime p, so that the bit-cost stays within the claimed bounds and the resulting degree value is likely to withstand the transition to $\mathbb{Z}$.

To yield the Las Vegas estimates, we must verify that the degree indeed remains the same in this transiton. For a fixed candidate value $k = \deg g(x)$ let U_k and W_k denote the Toeplitz matrices of the sizes $(m + n - k) \times (n - k)$ and $(m + n - k) \times (m - k)$, respectively, which are simultaneously upper and lower triangular, that is, are filled with zeros above their upper diagonals and below their lower diagonals. Thus they are defined by their first columns $(u_0, \ldots, u_m, 0, \ldots, 0)^T$ and $(w_0, \ldots, w_n, 0, \ldots, 0)^T$, respectively. The $(m + n - k) \times (m+n-2k)$ matrix (U_k, W_k) (called subresultant matrix) has Toeplitz-like structure, has a displacement rank at most two, and has rank k for $k = \deg g(x)$. Thus our algorithms supporting Theorem 5.2 enable us to verify the equation $k = \deg g(x)$ within the claimed cost bounds.

Now, having certified the degree k, we wish to compute the gcd. We recall from [bgy80] and [p01, Section 2.10] that $g(x) = s(x)u(x) + t(x)w(x)$ where $s(x) = \sum_{i=0}^{n-k} s_i x^i$, $t(x) = \sum_{i=0}^{m-k} t_i x^i$, the coefficient vectors $\mathbf{s} = (s_i)_{i=0}^{n-k}$ and $\mathbf{t} = (t_i)_{i=0}^{m-k}$ satisfy the subresultant equation $\tilde{U}_k \mathbf{s} + \tilde{W}_k \mathbf{t} = \mathbf{e}_1$, and $\tilde{U}_k$ and $\tilde{W}_k$ are Toeplitz matrices of the sizes $(m+n-2k) \times (n-k)$ and $(m+n-2k) \times (m-k)$, respectively, which we obtain by deleting the last k rows of the matrices U_k and W_k, respectively.

It follows (see [p01, Definitions 4.1.1 and 4.1.3]) that the coefficient matrix $(\tilde{U}_k, \tilde{W}_k)$ of the above subresultant system is again a Toeplitz-like matrix with a displacement rank at most three. Now the claimed complexity bounds for computing the gcd $g(x) = \gcd(\mathrm{u}, \mathrm{w})$ follow from Corollary 7.2.

They are immediately extended to the task of computing the polynomial $\mathrm{lcm}(\mathrm{u}, \mathrm{w}) = u(x)w(x)/\gcd(\mathrm{u}, \mathrm{w})$. □

9 Generalized Hensel's lifting

9.1 The algorithm

If $(\det M) \mod s = 0$, then we cannot initialize Algorithm 3.1. How frequently does this equation hold for a random structured integer matrix M and a fixed nonrandom integer $s < |\det M|$? According to the analysis and tests in [pw08], such degeneracy is unlikely if $s = p^b$ is a fixed reasonably large prime power, even where the prime p is small and divides $\det M$. Next we generalize our lifting and initialization algorithms to the case where $(\det M) \mod s \neq 0$ for $s = p^b$. In particular, for $p = 2$ this covers binary lifting, having implementation advantages. We rely on the following concept.

Definition 9.1. *For two integers $q > 0$ and $s > 1$, a matrix M in $\mathbb{Z}_{qs}^{n\times n}$ is* factor-q nonsingular *modulo qs if there exists a matrix Q in $\mathbb{Z}_{qs}^{n\times n}$ such that*

$$MQ \bmod (qs) = qI. \tag{9.1}$$

For $q = 1$ equation (9.1) turns into $MQ = I \mod s$ and thus brings us back to Hensel's lifting. Next we rely on Definition 9.1 for $q > 1$ to generalize our constructions and analysis in Sections 3–6. In fact we mimic them quite closely.

Let us be given a vector $\mathbf{f}$ and black box subroutines for multiplying by vectors a factor-q nonsingular matrix M in $\mathbb{Z}_{qs}^{n\times n}$ (see Definition 9.1) and matrix Q satisfying (9.1). Then the following algorithm computes the first h terms in the vector expansion $M^{-1}\mathbf{f} = \sum_{i=0}^{\infty} \mathbf{u}^{(i)} s^i$, $\mathbf{u}^{(i)} \in \mathbb{Z}_{qs}^n$, $i = 0, 1, \ldots$.

Algorithm 9.1. Generalized Hensel's lifting.
INPUT: *a matrix $M \in \mathbb{Z}^{n\times n}$, a vector $\mathbf{f} \in \mathbb{Z}^n$, three positive integers h, q, and s, and a matrix $Q \in \mathbb{Z}_{qs}^{n\times n}$ satisfying (9.1).*
OUTPUT: *the vector $\mathbf{x}^{(h)} \in \mathbb{Z}^n$ such that $M\mathbf{x}^{(h)} = (q\mathbf{f}) \bmod (qs^h)$.*
INITIALIZATION: $\mathbf{r}^{(0)} = \mathbf{f}$.
COMPUTATIONS: *for $i = 0, 1, \ldots, h-1$, compute the vectors*

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

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

The following extension of Theorem 3.1 shows correctness of the algorithm (see part b) and bounds the precision of its computations.

Theorem 9.1. *For M, $\mathbf{f}$, $\mathbf{r}^{(i)}$, and $\mathbf{x}^{(h)}$ in Algorithm 9.1, we have*
(a) $\mathbf{r}^{(i)} \in \mathbb{Z}^n$ for all i,
(b) $M\mathbf{x}^{(h)} = q\mathbf{f} \bmod (qs^h)$,
(c) all components $r_j^{(i)}$ of all vectors $\mathbf{r}^{(i)} = (r_j^{(i)})_j$ satisfy the bounds

$$|r_j^{(i)}| \le \frac{1}{s^i}|f_j| + \alpha n \frac{qs-1}{q}\sum_{k=1}^{i} s^{-k} < \frac{1}{s^i}\beta + \alpha n \frac{qs-1}{qs-q} < \gamma$$

where β, α, and γ are defined in Theorem 3.1.

Proof.
(a) $(q\mathbf{r}^{(i)} - M\mathbf{u}^{(i)}) \bmod (qs) = (qI - MQ)\mathbf{r}^{(i)} \bmod (qs)$, and the claim follows because $MQ = qI \bmod (qs)$.
(b) $M\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} M\mathbf{u}^{(i)}s^i = \sum_{i=0}^{h-1}(q\mathbf{r}^{(i)} - qs\mathbf{r}^{(i+1)})s^i = q\mathbf{f} - qs^h\mathbf{r}^{(h)} = q\mathbf{f} \bmod (qs^h)$.
(c) By definition, all components $u_j^{(i)}$ of all vectors $\mathbf{u}^{(i)}$ are integers in the range $[0, qs-1]$, and so

$$qs|r_j^{(i+1)}| \quad \le \quad q|r_j^{(i)}| + \alpha n \max_k |u_k^{(i)}| \le q|r_j^{(i)}| + (qs-1)\alpha n.$$

Now the claim follows by induction on i. □

Next we change an upper bound on d_1 into $\lceil \log(2qs\gamma) \rceil$, change s into qs, and then readily extend Lemma 3.1 and Theorem 3.2. Here is the respective generalization of Theorem 4.1.

Theorem 9.2. *Let $\mathbf{x} = qM^{-1}\mathbf{f}$ be a unique solution to the linear system $M\mathbf{x} = q\mathbf{f}$. Assume $\rho(d)$ in (2.1), α, β and γ in (3.2), and h in (4.2). Write*

$$l = \lceil \log(2(\alpha\sqrt{n})^{2n-1}n\beta q) \rceil = O(n \log \gamma + \log q). \tag{9.2}$$

(These expressions extend equations (4.1) to the case of generalized lifting.) Let the vector $\mathbf{x}^{(h)} = \sum_{i=0}^{h-1} \mathbf{u}^{(i)}p^i = \mathbf{x} \bmod (qs^h)$ be computed in $h-1$ steps of Algorithm 9.1. Then one can recover the vector $\mathbf{x}$ from the vector $\mathbf{x}^{(h)}$ by performing $B = n\rho(l)$ bit operations.

The proof mimics the proof of Theorem 4.1 except that now we define k to be the smallest integer exceeding $2(\alpha\sqrt{n-1})^{n-1}n\beta q$. Likewise we restate Theorems 5.1 and 5.2 for l in equations (9.2) (rather than in (4.1)) and Q satisfying equation (9.1) (rather than $MQ = I \bmod s$).

Remark 9.1. *Algorithm 9.1 and Theorem 9.2 can be immediately extended to the variation where the vectors $\mathbf{f}$, $\mathbf{x}^{(h)}$, $\mathbf{r}^{(i)}$ and $\mathbf{u}^{(i)}$ for all i are relaced with block vectors.*

Next we extend initialization Algorithm 6.1 and its analysis assuming that q and s are some unknown powers p^u and p^v of a fixed prime p.

9.2 Initialization via iterative refinement: introductory comments

Our initialization algorithm either fails (which rarely occurs on the average input) or yields matrix Q that satisfies equation (9.1) for $q = p^u$ and $s = p^v$ equal to the powers of a fixed prime p where the sum $b = u+v$ exceeds the order of p in $\det M$. We can ensure this property if we choose $m = p^b$ exceeding the bound in Fact 2.4. This bound, however, tends to be overly pessimistic. Choosing b based on this bound, we would excessively increase the overall computational cost bound for the average input. Thus we choose the exponent b dynamically, first testing more moderate values. If this does not work, we reapply the algorithm for larger powers $m = p^b$ within a fixed tolerance value.

In the next subsection we describe our basic algorithm that for a fixed vector $\mathbf{f}$ applies iterative refinement to yield a vector $\mathbf{v}$ that satisfies the equation $M\mathbf{v} = q\mathbf{f} \mod (qs)$. The algorithm uses $O((n\log(|M| + p^b) + \log|\mathbf{f}|)m_M)$ ops.

Then, given a prime p, its power $m = p^b$, and a displacement generator $(G, H) = ((\mathbf{g}_i)_{i=1}^r, (\mathbf{h}_i)_{i=1}^r))$ of length r for a matrix M, we compute a displacement generator of length r for a matrix Q satisfying equation (9.1). First we apply our basic algorithm $2r$ times (at $2r$-fold cost) to solve $2r$ linear systems of equations $-M\tilde{\mathbf{u}}_i = q_i\mathbf{g}_i \mod m$, $M^T\tilde{\mathbf{v}}_i = q_{i+r}\mathbf{h}_i \mod m$ for $i = 1, \dots, r$, where $q_i = p^{u_i}$ and $q_{i+r} = p^{v_i}$ for $i = 1, \dots, r$ are integer powers of p defined by our basic algorithm (and depending on the matrix M and the vectors $\mathbf{g}_i$ and $\mathbf{h}_i$). Then (at a negligible cost) compute the integers $q = \max_{i=1}^{2r} q_i$ and $s = m/q$ and the vectors $\mathbf{u}_i = \tilde{\mathbf{u}}_i q/q_i$ and $\mathbf{v}_i = \tilde{\mathbf{v}}_i q/q_{i+r}$ such that $M\mathbf{u}_i = q\mathbf{g}_i \mod m$ and $M\mathbf{v}_i = q\mathbf{h}_i \mod m$ for $i = 1, \dots, r$. This defines a generator $(U, V) = ((\mathbf{u}_i)_{i=1}^r, (\mathbf{v}_i)_{i=1}^r)$ of length r for the displacement UV^T of the matrix Q satisfying (9.1).

9.3 The basic initialization algorithm

Algorithm 9.2. *The basic initialization algorithm.*
INPUT: *a nonsingular matrix $M \in \mathbb{Z}^{n\times n}$, a vector $\mathbf{f} \in \mathbb{Z}^n$, a prime p, and two positive integers b and t such that $m = p^b \geq 2^{t+2}|M|$.*
OUTPUT: *either FAILURE if $\delta((M^{-1}\mathbf{f})_j) \bmod p^b = 0$ for some j, $j = 1, \dots, n$, or two integers q and s, both the powers of p and such that $qs = p^b$, and the vector $\mathbf{z}$ such that $M\mathbf{z} = q\mathbf{f} \mod (qs)$.*
INITIALIZATION: *as in Algorithm 6.1.*
COMPUTATIONS:
Stages 1 and 2 are as in Algorithm 6.1.
Stage 3. Compute the integer $v = \max_j \mathrm{ord}_p(\delta((M_0^{-1}\mathbf{f})_j))$. If $v < b$, output the integers $q = p^v$ and $s = p^{b-v} = m/q$; compute and output the vector $\mathbf{z} = qM_0^{-1}\mathbf{f} \mod (qs)$ (so that $M\mathbf{z} \bmod (qs) = M_0\mathbf{z} \bmod (qs) = q\mathbf{f} \mod (qs)$). Otherwise output FAILURE.

In virtue of Theorems 2.8 and 6.1, Algorithm 9.2 is unlikely to fail if p is a random prime from a moderately large range (even where $b = 1$). According to

[pw08] (see the beginning of Section 9), the failure is also unlikely in the case of a fixed p and random $n \times n$ structured integer matrix M provided $p^b \gg n$.

If the algorithm outputs FAILURE, one can apply some heuristic recipes from [pw08] or reapply the algorithm for b equal to a greater power of the same prime p or for p replaced with a distinct basic prime.

9.4 The computational precision and the Boolean cost estimates

According to the argument in Section 6.4, we can extend the asymptotic complexity estimates in Section 5 to Algorithm 9.2 provided $b \log p = O(\log \gamma)$. The estimates grow proportionally to $\mu(b)$, where b exceeds the value $v = \max_j \operatorname{ord}_p(\delta((M_0^{-1}\mathbf{f})_j))$.

This value has the order of $n \log_p \gamma$ for a fixed prime p, e.g., for $p = 2$, and for the worst case input matrix M. According to the analysis and tests in [pw08], however (see the beginning of Section 9), $v = O(\log_p \gamma)$ for reasonably large integers b and for the average integer matrices M with the displacement structures. So we can extend the cost bounds of Section 5 to the initialization for the average input.

10 Matrix inversion via Newton's and generalized Newton's lifting

Newton's lifting is a well known classical counterpart of Hensel's [y76], [g84]. In [mc79] both techniques were simultaneously adapted to matrix computations, for which they have similar power. Next we recall Newton's lifting for matrix inversion, specify it to the case of structured input matrices, analyze it, estimating its complexity, propose its generalization (in particular its binary version), and compare its performance with Hensel's.

Given an integer $s > 1$ and a pair of matrices M and $Q^{(0)}$ such that $MQ^{(0)} = I \bmod s$, Newton's lifting algorithm recursively computes the matrices $Q^{(i)} = Q^{(i-1)}(2I - MQ^{(i-1)}) \bmod (s^{2^i})$, $i = 1, 2, \dots$ (see [mc79], [p01, Chapter 7]). One immediately verifies that the matrix equation $Q^{(i)} = Q^{(i-1)}(2I - MQ^{(i-1)})$ implies that $I - MQ^{(i)} = (I - MQ^{(i-1)})^2$, and so the iterates $Q^{(i)}$ satisfy $I - MQ^{(i)} = 0 \bmod s^{2^i}$ for all i, thus rapidly improving the initial approximate matrix inverse $Q^{(0)}$. The algorithm has a celebrated numerical counterpart (see, e.g., [ps91]).

Then again we cannot satisfy the initial assumption $MQ^{(0)} = I \bmod s$ if $\det M = 0 \bmod s$, which motivates shifting to the generalized Newton's lifting, which begins with a matrix $Q^{(0)}$ and recursively computes the matrices $Q^{(1)}, Q^{(2)}, \dots$ such that

$$MQ^{(0)} = qI \bmod (qs),\ qQ^{(i)} = Q^{(i-1)}(2qI - MQ^{(i-1)}) \bmod (qs^{2^i}), \quad (10.1)$$

$i = 1, 2, \ldots, h$. Then we deduce that $q^{2^i-1}(qI - MQ^{(i)}) = (q^{2^{i-1}-1}(qI - MQ^{(i-1)}))^2 = (qI - MQ^{(0)})^{2^i} = 0 \mod (qs)^{2^i}$ and therefore $qI - MQ^{(i)} = 0 \mod (qs^{2^i})$. For $q = 1$, we come back to Newton's lifting for matrix inversion.

Surely, our initialization recipes for Hensel's and generalized Hensel's lifting can be extended to Newton's and generalized Newton's.

Every generalized Newton's step (10.1) is essentially reduced to performing $n \times n$ matrix multiplication twice. For general matrices, this operation is expensive, although it is substantially accelerated on multiprocessors.

For matrices M and $Q^{(i)}$ having consistent displacement structures, we dramatically simplify multiplication by performing it in terms of the associated short displacement generators. Unlike the case of Hensel's lifting, *one must modify Newton's classical lifting* to exploit the matrix structure. Here our respective modifications where we assume a displacement operator L and rely on Corollary 2.1:

$$G^{(i+1)} = -Q^{(i)}(2I - MQ^{(i)})G, \quad H^{(i+1)T} = H^T Q^{(i)}(2I - MQ^{(i)})$$

where $L = \nabla_{A,B}$,

$$G^{(i+1)} = BQ^{(i)}(2I - MQ^{(i)})G, \quad H^{(i+1)T} = H^T B^{-1} Q^{(i)}(2I - MQ^{(i)})$$

where $L = \Delta_{A,B}$ and the matrix B is nonsingular, and

$$G^{(i+1)} = Q^{(i)}(2I - MQ^{(i)})A^{-1}G, \quad H^{(i+1)T} = H^T Q^{(i)}(2I - MQ^{(i)})A$$

where $L = \Delta_{A,B}$ and the matrix A is nonsingular.

In the case of a Toeplitz matrix $T = (t_{k-j})_{k,j} = M/q$, we can represent the approximate inverses $Q^{(i)} = qM^{-1} \mod (qs^{2^i})$ in $\mathbb{Z}_{qs^{2^i}}$, $i = 0, 1, \ldots$, with their $n \times 2$ generators $Q^{(i)}(\mathbf{e}_1, \mathbf{t}) = (Q^{(i)}\mathbf{e}_1, Q^{(i)}\mathbf{t})$ where the vector $\mathbf{t}$ is defined in Theorem 2.6. Then iteration (10.1) takes the following form,

$$qQ^{(i)}(\mathbf{e}_1, \mathbf{t}) = Q^{(i-1)}(2qI - MQ^{(i-1)})(\mathbf{e}_1, \mathbf{t}) \mod (qs^{2^i}), \tag{10.2}$$

$i = 1, 2, \ldots$, provided we are given the generators $M(\mathbf{e}_1, \mathbf{t})$ and $Q^{(0)}(\mathbf{e}_1, \mathbf{t})$ for the matrices M and $Q^{(0)}$ such that $MQ^{(0)}(\mathbf{e}_1, \mathbf{t}) = q(\mathbf{e}_1, \mathbf{t}) \mod (qs)$. Every iteration step is reduced essentially to multiplication of the matrix M by the $n \times 2$ matrix $Q^{(i-1)}(\mathbf{e}_1, \mathbf{t})$ and of the matrix $Q^{(i-1)}$ by the resulting $n \times 2$ matrix. This takes only $O(m(n))$ ops (see Theorems 2.5 and 2.6).

For $q = 1$ the iteration process (10.2) for Toeplitz matrix M takes the form

$$Q^{(i)}(\mathbf{e}_1, \mathbf{t}) = Q^{(i-1)}(2I - MQ^{(i-1)})(\mathbf{e}_1, \mathbf{t}), \quad i = 1, 2, \ldots. \tag{10.3}$$

More generally, we recall Corollary 2.1 and reduce the inversion of a matrix M with a displacement rank r to solving $2r$ linear systems of equations with the coefficient matrix M. Then every generalized Newton's lifting step amounts to multiplication of the matrix $Q^{(i-1)}(2I - MQ^{(i-1)})$ by $2r$ vectors. Here $Q^{(i-1)}$

denotes the approximate inverse reconstructed from the $2r$ respective vectors computed in the previous iteration, except that at the initial step, these vectors are given by the columns of the displacement generators G and H of the matrix $Q^{(0)}$.

Let us compare the performance of Newton's and Hensel's lifting. Every Newton's as well as generalized Newton's lifting step roughly doubles the precision of computing and the number of correct bits per an output value in an iteration step, while Hensel's and generalized Hensel's lifting keep the precision of computing bounded by a fixed tolerance and add about the same number of correct bits per an output value in every step. As a result, in h steps generalized Newton's lifting produces the solution modulo qs^{2^h}, which generalized Hensel's lifting yields in 2^h steps, but computations with extended precision are generally required already in a relatively small number of Newton's steps.

Let us supply some specific estimates. Assume a Toeplitz matrix M and generalized Newton's lifting. Then the initial approximate inverse $Q^{(0)} = M^{-1} \mod (qs)$ is lifted to $M^{-1} \mod (qs^{2^h})$ in h lifting steps, each performing $O(m(n))$ ops. For $q = 1$ this covers the standard (structured) Newton's lifting.

Since the precision of computing is doubled in each lifting step, the overall Boolean (bit-operation) complexity of Newton's lifting is dominated by the cost of the last lifting step, bounded by $O(m(n)\mu(l_{\text{out}}))$ where l_{out} denotes the required output precision. In the case of an $n \times n$ Toeplitz-like input matrix M given with its displacement generator of a length r the overall complexity increases to $O(r^2 m(n)\mu(l_{\text{out}}))$.

In comparison with Hensel's lifting, this means the Boolean cost increase by the factor $r\mu(l_{\text{out}})/(h\mu(l_{\text{out}}/h))$, in spite of the more rapid progress in Newton's lifting. The analysis seems to give upper hand to Hensel's lifting even where the positive integer r is small, e.g., in the case of Toeplitz inputs, where $r \leq 2$.

Our conclusion is different under the word model, however. In its typical choice, the initial prime p is by far less than the word length λ. Therefore the first steps of both Newton's and Hensel's liftings are performed by using about the same number of word operations, and the much more rapid progress of Newton's lifting is a clear advantage. Namely the initial steps of Newton's lifting can save a significant number of word operations wherever the ratio $\lceil \log(2qs\gamma) \rceil / \lambda$ is small.

This suggests that the most effective policy is to apply Newton's lifting initially and to continue this application as long as its output precision qs^{2^i} stays within the word length λ. If this length is exceeded, one should reduce the output approximate inverse modulo qs^{2^j} for $j = \lceil \log_2 \log_s(\lambda/q) \rceil$ and then shift to Hensel's lifting.

Finally Newton's lifting is much more friendly than Hensel's to parallel implementation. By parallelizing Newton's steps, one can yield dramatic acceleration, although at the expense of using more processors.

11 Comparison of lifting with the divide–and–conquer MBA algorithm

For the inversion of a structured strongly nonsingular matrix $M = \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix}$ one can readily adapt the divide-and-conquer MBA algorithm from [m80], [ba80]. (One can produce a strongly nonsingular matrix with a probability near one by applying randomized preprocessing in Theorem 7.2 to a nonsingular matrix.) The algorithm recursively employs the following factorization,

$$M^{-1} = \begin{pmatrix} I & -M_{00}^{-1}M_{01} \\ O & I \end{pmatrix} \begin{pmatrix} M_{00}^{-1} & O \\ O & S^{-1} \end{pmatrix} \begin{pmatrix} I & O \\ -M_{10}M_{00}^{-1} & I \end{pmatrix}. \tag{11.1}$$

The matrix

$$S = S(M_{00}, M) = S^{(k)}(M) = M_{11} - M_{10}M_{00}^{-1}M_{01} \tag{11.2}$$

is called the *Schur complement* of the block M_{00} in the matrix M and the *Gauss transform* of M (cf. [gl96]).

Let $I(k)$, $M(k)$, and $A(k)$ denote the number of ops required to invert a $k \times k$ matrix, to multiply a pair of $k \times k$ matrices, and to add or subtract them, respectively, where clearly $A(k) \le M(k)$.

Remark 11.1. *We recall that* $M(k) \le c_{\text{immense}}k^{2.376}$ *(this bound in [cw90] holds for an immense constant* c_{immense} *and is the smallest known bound in the case of immense integers* k*) and* $M(k) \le ck^{2.776}$ *(this bound in [lps92] holds for a moderate constant* c *and is the smallest known bound in the case where* $20 \le k \le 10^{20}$*). The latter bound relies on the technique of trilinear aggregation (due to [p72]; cf. also [p84], [lps92], and [k04]), which was an ingredient of the algorithm in [cw90] as well and was the first nontrivial application of the* tensor decompositions *for the design of effective algorithms for fundamental matrix computations.*

The above factorization implies that $I(2k) \le 2I(k) + 6M(k) + 2A(k)$ for all positive integers k. Recursively $I(n) = O(M(n)\log n)$, and if $M(n)$ is of the order cn^{1+a} for two positive constants a and c, then $I(n) = O(M(n))$.

Next assume a Toeplitz or Toeplitz-like matrix M and estimate the cost of its inversion based on the above recursive factorization. We can multiply a pair of $n \times n$ Toeplitz or Toeplitz-like matrices in $O(m(n))$ ops, and we are tempted to substitute this bound as $M(n)$ to arrive at $n \times n$ Toeplitz-like inversion in $O(m(n)\log n)$ ops.

To substantiate this estimate, however, we must extend the input Toeplitz structure to all the auxiliary matrices involved into the recursive factorization. We first recall that the inverse of a Schur complement is a trailing block of the inverse and then that the displacement rank r of a matrix is preserved in the transition to its inverse (see Corollary 2.1) and grows by at most one in the transition to the blocks.

This covers the matrices M_{00} and S in the initial factorization and then recursively covers all auxiliary leading blocks and Schur complements in the recursive factorization because the leading blocks as well as their Schur complements in a Schur complement S are Schur complements in the original matrix M (cf. [p01, Chapter 5] or [pmr10]). Consequently Fact 2.6 implies the bound $O(r)$ on the displacement rank of all auxiliary matrices involved into the recursive factorization.

This is not yet enough to support the desired cost estimate because we need to bound the length of the displacement generators in the representation of all these matrices, and not just their displacement ranks. The desired extension of the bound from the ranks to the lengths is obtained based on the following result (valid over any field).

Theorem 11.1. *Suppose we are given a pair of $n \times k$ matrices G and H such that the $n \times n$ matrix $L = GH^T$ has a rank $r \le k$. Then it is sufficient to perform $O(nM(k)/k)$ ops to compute a pair of $n \times r$ matrices G_r and H_r such that $L = G_r H_r^T$ where $M(k)$ is the arithmetic complexity of the $k \times k$ matrix multiplication.*

Proof. See [p01, Section 4.6.2]. □

The resulting arithmetic cost estimate for solving a nonsingular Toeplitz-like linear system of n equations turns into the bound $O(r^2 m(n) \log n)$ in the case of a THVC linear system whose coefficient matrix is defined with its displacement generators of a length r. The bound is extended from the case of Toeplitz structure to the case of Vandermonde and Cauchy structures by means of displacement transformations in Remark 2.1.

Even though technically the MBA algorithm is completely different from lifting, it supports the complexity bound that only by factor $r \log n$ exceeds ours under the randomized Boolean (bit operation) cost model (see [pmr10]). Recently this factor was decreased to $O((M(r)/r^2) \log n)$ in [bjs08].

Furthermore the MBA algorithm enables us to decrease by the factor n/r the randomized Monte Carlo bound on the cost of lifting initialization in $\mathbb{Z}_p$ for a reasonably large random prime p (see Theorem 2.8). Consequently one can decrease by factors $\log n$, $\log N$, or $\log s$ the cost bounds in Corollaries 7.2 and 8.1 and Theorems 8.1 and 8.2, respectively.

Unlike Algorithm 6.1 the MBA algorithm can be applied over any field of constants that supports strong nonsingularity of the input. Realistically we fulfill this requirement by involving random parameters from a set of large cardinality. Namely suppose we apply the algorithm over the rationals and wish to bound the precision of computing, say by $cn \log(n||M||)$ for a positive constant c. Then the standard way is to apply the algorithm modulo sufficiently many random primes and then to recover the rational output by means of the Chinese Remainder Algorithm.

Unlike the case of lifting, it is not sufficient to work with a single random prime p, but one needs the order of n primes, each of the order $n \log(n||M||)$. Moreover application of the algorithm modulo a large prime power p^b readily

leads to degeneration. In particular the MBA algorithm does not work modulo the powers of two.

Whereas lifting involves only two matrices M and $Q = M^{-1} \mod s$, the MBA algorithm generally processes over $2n$ matrices of various sizes, which can be viewed as a certain implementation advantage of lifting.

12 Concluding remarks

We applied numerical iterative refinement to initialize Hensel's symbolic lifting of the solutions of linear systems of equations as well as Newton's lifting of matrix inverses where the input values are integers. We proposed four variants of Newton's structured lifting, that is Newton's lifting applied to structured matrices and exploiting structure to perform the computations faster.

In the case of Toeplitz, Hankel and other structured input matrices the resulting Hensel's and Newton's lifting algorithms support nearly optimal Boolean complexity estimates. We unified the algorithms for the four most popular classes of structured matrices.

Our algorithms are most valuable in the case of the inputs for which numerical computations with double precision cannot produce the output with required accuracy. This includes some important classes of structured matrices (see [b85], [gi88], [t94]).

Our combination of symbolic lifting with numerical iterative refinement is an example of successful symbolic-numerical algorithms (see [tcs04], [snc07], [snc07a], [tcs08], [snc09] on this subject area). Searching for further examples of this kind one can try to devise effective symbolic counterparts to various other iterative numerical algorithms for linear systems of equations (cf. [epy98]).

We extended our algorithms and our nearly optimal bound on the Boolean complexity to Berlekamp–Massey's reconstruction of a linear recurrence from its values and to computing the gcd, lcm, and Padé approximation for univariate polynomials.

Furthermore we generalized both Hensel's and Newton's liftings by allowing to begin them with basic reduction modulo a nonprime, e.g., modulo a power of two, which enabled us to perform all our algorithms in binary base.

Appendix

A Randomized reconstruction of rational solution

By using the Las Vegas randomization, we decrease the bound in Theorem 4.1 by the factor of $\log d$ provided $\mu(d) = O(d^{\log_2 3})$ or $\mu(d) = O((d \log d) \log\log d)$ and $\rho(d)$ is bounded in (2.1). Empirical evidence shows further progress with some heuristics.

A.1 Randomized reconstruction algorithm

Write

$$\delta = \mathrm{lcm}_j\, \delta(x_j), \quad 1 \le j \le n \tag{A.1}$$

(for $\delta(y)$ in Definition 2.1), that is δ is the least common multiple of the denominators in all rational coordinates x_j of the solution $\mathbf{x} = (x_j)_j$ to the system $M\mathbf{x} = q\mathbf{b}$.

Algorithm A.1. *Randomized reconstruction of rational solution.*

INPUT: *The same as in Algorithm 3.1 and in addition a positive $\varepsilon < 1$, an integer $h = 1 + \lfloor \log_s(2n(\alpha\sqrt{n})^{2n-1}\eta\beta)\rfloor$ of equation (4.2), and the vector $\mathbf{x}^{(h)} = (x_i^{(h)})_{i=1}^n = qM^{-1}\mathbf{f} \bmod (qs^h)$.*

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

$$M\mathbf{y} = \delta q\mathbf{f}. \tag{A.2}$$

INITIALIZATION: *Compute*

$$K = 2\lceil \log(1/\varepsilon)\rceil, \tag{A.3}$$

$$\eta = \lceil 6 + 2n\log(n\alpha)\rceil \tag{A.4}$$

for α and β in (3.2). Then sample K pseudo random vectors

$$\mathbf{c}_k = (c_{jk})_{j=1}^n \in \mathbb{Z}_\eta^n, \quad k = 1, \ldots, K. \tag{A.5}$$

COMPUTATIONS:

1. *Compute the K integers $w_k = \mathbf{c}_k^T\mathbf{x}^{(h)} = \sum_{j=1}^n c_{jk}x_j^{(h)}$, $k = 1, \ldots, K$.*
2. *Recover a unique set of the pairs of coprime integers ν_k and δ_k such that*

$$(\nu_k/\delta_k) \bmod (qs^h) = w_k,\ 1 \le 2\delta_k|\nu_k| \le qs^h, \quad 2|\nu_k| < qs^h, \tag{A.6}$$

   for $k = 1, \ldots, K$.

3. *Compute the least common multiple of the denominators*

$$\delta_{lcd} = \mathrm{lcm}_k\, \delta_k, \quad 1 \le k \le K. \tag{A.7}$$

4. *Compute the integer vector $\mathbf{y} = (y_j)_{j=1}^n$ such that $\mathbf{y} \bmod (qs^h) = \delta_{lcd}\mathbf{x}^{(h)}$ and $2|y_j| < qs^h$ for all j. If $M\mathbf{y} = q\delta_{lcd}\mathbf{f}$, output $\mathbf{y}$ and $\delta = \delta_{lcd}$; otherwise output FAILURE.*

A.2 Correctness proof

Combining equations (4.2), (A.4), and (A.5) with Fact 2.4 implies (A.6). Now, correctness of Algorithm A.1 is implied by the following simple result.

Theorem A.1. *δ_{lcd} in (A.7) divides δ in (A.1). Furthermore,*

$$\text{Probability}(\delta_{lcd} \neq \delta) \leq \varepsilon.$$

Theorem A.1 is deduced similarly to Theorem 2.1 in [egv00] based on equations (4.2), (A.3)–(A.7), and the following lemma.

Lemma A.1. *For a prime p, integers K in (A.3), k such that $1 \leq k \leq K$, δ in (A.1), η in (A.4), and δ_k in (A.6), we have* $\text{Probability}(\text{ord}_p(\delta_k) < \text{ord}_p(\delta))$ *equal to $1/\eta$ for $p \geq \eta$ and to $\lfloor \eta/p \rfloor / \eta \leq 1/p$ for $p \leq \eta$.*

Proof. Let $l = \text{ord}_p(\delta) = \max_j \text{ord}_p(\delta(x_j))$ for $1 \leq j \leq n$. W.l.o.g., let $l = \text{ord}_p(\delta(x_1))$ 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 $\mathbf{x} = M^{-1}\mathbf{f}$, $l \geq h$, and a, b, u, and v are four integers coprime with p. Clearly, $\text{ord}_p(\delta_k)$ for δ_k in (A.6) never exceeds l; it equals l if and only if $cub - avp^{l-h}$ is coprime with p. The probability bound follows because the integers ub and p are coprime and because c is a random element in the ring $\mathbb{Z}_\eta$. □

A.3 The bit-complexity of randomized reconstruction of rational solution

Let us first estimate the bit complexity of performing Algorithm A.1 in terms of $l = O(n \log \gamma + \log q)$ in (9.2), m_S in Definition 2.4, $\mu(d)$ in Fact 2.1, $\rho(d)$ in (2.1), and K in (A.3). We need the following auxiliary result.

Lemma A.2. *Let j and k be positive integer parameters, $j \to \infty$. Then $O(\mu(j)k)$ bit operations are sufficient to multiply two positive integers u and v such that $u < 2^j$ and $v < 2^{j+k}$.*

Proof. Represent v as $\sum_{i=0}^{k-1} v_i 2^{ij}, 0 \leq v_i < 2^j$ for all i. Compute the products $w_i = uv_i$ for $i = 0, 1, \ldots, k-1$. This takes $O(\mu(j)k)$ bit operations. Now compute the sum $uv = \sum_{i=0}^{k-1} w_i 2^{ij}$. This takes $O(jk)$ bit operations. □

Algorithm A.1 involves $O(Kn\mu(l))$ bit operations at Stage 1; $O(K\rho(l))$ at Stage 2; $O(K\mu(l) \log l)$ at Stage 3, and $O(n\mu(l))$, $O(n\mu(\log \beta)l/\log \beta)$, and $O(m_M \mu(\log \gamma)l/\log \gamma)$ for computing the vectors $\delta_{lcd}\mathbf{x}^{(h)}$, $q\delta_{lcd}\mathbf{f}$, and $M\mathbf{y}$ at Stage 4, respectively. (The two latter bounds are deduced based on Lemma A.2.) Summarizing, we obtain the following estimates.

Theorem A.2. *Algorithm A.1 generates nK random elements in $\mathbb{Z}_\eta$ for η in (A.4) and $K = 2\lceil \log(1/\epsilon) \rceil$ in (A.3). It either fails (this occurs with a probability of at most ϵ) or computes the scalar δ of equation (A.1) and the solution $\mathbf{y}$ to linear system (A.2). The algorithm involves*

$$B_1 = O(Kn\mu(l) + K\rho(l) + m_M\mu(\log\gamma)l/\log\gamma)$$

bit operations for $l = O(n\log\gamma + \log q)$ in (9.2), $\rho(d)$ in (2.1), γ in (3.2), m_S in Definition 2.4, and $\mu(d)$ in Fact 2.1; it involves $o(B_1)$ bit operations for generating nK pseudo random elements in $\mathbb{Z}_\eta$.

B Some details for the overall computational cost of the solution with Hensel's lifting

Theorem B.1. *Assume a prime p, a vector $\mathbf{f} \in \mathbb{Z}^n$, and a nonsingular matrix $M \in \mathbb{Z}^{n\times n}$ with the structure of Toeplitz, Hankel, Vandermonde, or Cauchy type, having a displacement rank r and given with a displacement generator of a length in $O(r)$, and write $Q = M^{-1} \mod p$. Then we can compute the rational solution $\mathbf{x}$ to the linear system $M\mathbf{x} = \mathbf{f}$ by using single random parameter p and at the Las Vegas randomized bit–operation cost within the following bounds:*

i) $O(r^2 m(n)\mu(\log p)\log n)$ *at the initialization stage,*

ii) $O((m_Q\mu(\log p) + m_M\mu(\log(\gamma np)))h)$ *at the lifting stage;*

iii) $O(n\rho(l))$ *at the stage of Las Vegas randomized reconstruction of rational solution, which involves $n\lceil \log\frac{1}{\varepsilon}\rceil\lceil \log(6 + 2n\log(n\alpha))\rceil$ random bits and may fail with a probability of at most $\varepsilon > 0$.*

Here $m(n)$ and $\mu(d)$ are defined in Fact 2.1, $\rho(d)$ in Theorem 2.1, m_S in Definition 2.4, γ and α in (3.2), $l = O(n\log\gamma)$ in (4.1) for $q = 1$, and $h = O(n\log_p(\gamma n))$ in (4.2) for $s = p$.

Proof. The bounds B_L, B_R, and $B_L + B_R$ follow from our analysis in Sections 3, A.3, and 6, respectively. □

C On the History of the Method of Displacement Transformation

The Method of Displacement Transformation was proposed in [p89/90] and became widely recognized due to its application in [gko95], which besides [p89/90] cited the paper [h95] as the next publication appeared on this method. The displayed letter of 1991 from Georg Heinig to the present author can be of some interest for the history of this study.

Universität Leipzig
Sektion Mathematik
Hauptgebäude
Leipzig DDR-7010, Germany

Nov. 15, 1990

Dear Professor Rao,

Thank you very much for sending me your paper „On computations with dense structured matrices". It is very useful for us since we are also just dealing with fast algorithms for generalized Hilbert (we call them „Cauchy") and Vandermonde matrices. We would appreciate it very much if you continue to send us your papers. I am planning to write a book next year entitled „Fast algorithms for structured matrices and interpolation problems", which will appear in the OT-series of I. Gohberg. Therefore, any information from your side will be most useful for me.

With best regards
Sincerely,
Georg Heinig

P.S. I am now working with Leipzig University

References

[abm99] J. Abbott, M. Bronstein, T. Mulders. Fast Deterministic Computation of the Determinants of Dense Matrices, *Proc. Intern. Symp. Symbolic and Algebraic Comput. (ISSAC'99)*, 197–204, ACM Press, New York, 1999.

[b85] J. R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equations, *SIAM J. Sci. Stat. Comput.*, **6(2)**, 349–364, 1985.

[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.

[bjs08] A. Bostan, C.–P. Jeannerod, E. Schost, Solving structured linear systems with large displacement rank, *Theoretical Computer Science* **407 (1–3)**, 155–181, 2008.

[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.

[cw90] D. Coppersmith, S. Winograd, Matrix Multiplicaton via Arithmetic Progressions. *J. Symbolic Comput.*, **9(3)**, 251–280, 1990.

[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 Symp. on Foundations of Computer Science (FOCS'2000)*, 675–685, IEEE Computer Society Press, Los Alamitos, CA, 2000.

[epy98] I. Z. Emiris, V. Y. Pan, Y. Yu, Modular Arithmetic for Linear Algebra Computations in the Real Field, *J. of Symbolic Computation*, **21**, 1–17, 1998.

[f07] M. Fürer, Faster Integer Multiplication, *Proceedings of 39th Annual Symposium on Theory of Computing (STOC 2007)*, 57–66, ACM Press, New York, 2007.

[g84] J. von zur Gathen, Hensel and Newton Metods in Valuation Rings, *Math. of Computation*, **42(166)**, 802–824, 1984.

[gg03] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, UK, 2003 (second edition).

[gi88] W. Gautschi, G. Inglese, Lower Bounds for the Condition Number of Vandermonde Matrices, *Numerische Math.*, **52**, 241–250, 1988.

[gko95] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Mathematics of Computation*, **64**, 1557–1576, 1995.

[gl96] G. H. Golub, C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 1996 (third addition).

[go94] I. Gohberg, V. Olshevsky, Complexity of Multiplication with Vectors for Structured Matrices, *Linear Algebra and Its Applications*, **202**, 163–192, 1994.

[h79] G. Heinig, Beitrage zur spektraltheorie von Operatorbuschen und zur algebraischen Theorie von Toeplitzmatrizen, Dissertation **B**, *TH Karl-Marx-Stadt*, 1979.

[h95] G. Heinig, Inversion of Generalized Cauchy Matrices and the Other Classes of Structured Matrices, *Linear Algebra for Signal Processing, IMA Volume in Mathematics and Its Applications*, **69**, 95–114, Springer, 1995.

[hr84] G. Heinig, K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators, Operator Theory*, **13**, Birkhäuser, 1984.

[k04] I. Kaporin, The Aggregation and Cancellation Techniques As a Practical Tool for Faster Matrix Multiplication, *Theoretical Computer Science*, **315**, **2–3**, 469–510, 2004.

[kkm79] T. Kailath, S. Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *Journal of Mathematical Analysis and Applications*, **68, 2**, 395–407, 1979.

[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.

[lps92] J. Laderman, V. Y. Pan, H. X. Sha, On Practical Algorithms for Accelerated Matrix Multiplication, *Linear Algebra and Its Applications*, **162–164**, 557–588, 1992.

[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.

[ms04] T. Mulders, A. Storjohann, Certified Dense Linear System Solving, *J. of Symbolic Computation*, **37(4)**, 485-510, 2004.

[p72] V. Y. Pan, On Schemes for the Evaluation of Products and Inverses of Matrices (in Russian), *Uspekhi Matematicheskikh Nauk*, **27**, **5 (167)**, 249-250, 1972.

[p84] V. Y. Pan, How Can We Speed up Matrix Multiplication? *SIAM Review,* **26**, **3**, 393–415, 1984.

[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.

[p89/90] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. of Computation*, **55(191)**, 179–190, 1990. Proceedings version in *Proc. ISSAC'89*, 34-42, ACM Press, New York, 1989.

[p96] V. Y. Pan, Parallel Computation of Polynomial GCD and Some Related Parallel Computations over Abstract Fields, *Theoretical Computer Science*, **162, 2**, 173–223, 1996.

[p01] V. Y. Pan, *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[pgmq] V. Y. Pan, D. Grady, B. Murphy, G. Qian, R. E. Rosholt, A. Ruslanov, Schur Aggregation for Linear Systems and Determinants, *Theoretical Computer Science*, *Special Issue on Symbolic–Numerical Algorithms* (D. A. Bini, V. Y. Pan, and J. Verschelde editors), **409** (**2**), 255–268, 2008.

[pmr10] V. Y. Pan, B. Murphy, R. E. Rosholt, Unified Nearly Optimal Algorithms for Structured Integer Matrices, in *Numerical Methods for Structured Matrices and Applications: Georg Heinig memorial volume*, *Operator Theory: Advances and Applications*, **199**, 359–375, Birkhaüser, Basel, 2010.

[pq10] V. Y. Pan, G. Qian, Randomized Preprocessing of Homogeneous Linear Systems of Equations, *Linear Algebra and Its Applications*, **432**, 3272–3318, 2010.

[pqa] V. Y. Pan, G. Qian, On Solving Linear System with Randomized Augmentation, Tech. Report TR 2010009, *Ph.D. Program in Computer Science, Graduate Center, the City University of New York*, 2010. Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=352

[ps91] V. Y. Pan, R. Schreiber, An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM Journal on Scientific and Statistical Computing*, **12, 5**, 1109–1131, 1991.

[pw08] V. Y. Pan, X. Wang, Degeneration of Integer Matrices Modulo an Integer, *Linear Algebra and Its Application*, **429**, 2113–2130, 2008.

[snc07] *Symbolic-Numeric Computation*, (Dongming Wang and Li-Hong Zhi, editors), Birkhäuser, Basel/Boston, 2007.

[snc07a] *Proceedings of the 2rd Intern. Workshop on Symbolic-Numeric Computation (SNC2007)*, July 2007, London, Ontario, Canada (J. Verschelde and S. Watt, eds.), ACM Press, New York, 2007.

[snc09] *Proceedings of the 3rd Intern. Workshop on Symbolic-Numeric Computation (SNC2009)*, August 2009, Kyoto, Japan (Hiroshi Kai and Hiroshi Sekigawa eds.), ACM Press, New York, 2009.

[t94] E. E. Tyrtyshnikov, How Bad Are Hankel Matrices? *Numerische Mathematik*, **67(2)**, 261–269, 1994.

[tcs04] *Theoretical Computer Science*, Special Issue on Algebraic and Numerical Algorithms (I. Z. Emiris, B. Mourrain, V. Y. Pan, editors), **315**, **2**–**3**, 307–672, 2004.

[tcs08] *Theoretical Computer Science*, Special Issue on Symbolic–Numerical Algorithms (D. A. Bini, V. Y. Pan, and J. Verschelde, editors), **409**, **2**, 155–331, 2008.

[w86] D. Wiedemann, Solving Sparse Linear Equations over Finite Fields, *IEEE Trans. Inf. Theory*, **IT-32**, 54–62, 1986.

[wp03] X. Wang, V. Y. Pan, Acceleration of Euclidean Algorithm and Rational Number Reconstruction, *SIAM J. on Computing*, **32(2)**, 548–556, 2003.

[y76] D. Y. Y. Yun, Algebraic Algorithms Using p-adic Constructions, *Proceedings of ACM Symposium on Symbolic and Algebraic Computation*, 248–259, ACM Press, New York, 1976.