2002

TR-2002002: Can We Optimize Toeplitz/Hankel Computations? II. Singular Toeplitz/Hankel-like Case

V. Y. Pan

Can We Optimize Toeplitz/Hankel Computations ? II. Singular Toeplitz/Hankel-like Case

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

May 26, 2002

Abstract

In Part I, under the bit operation cost model we achieved nearly optimal randomized solution of nonsingular Toeplitz/Hankel linear system of equations based on Hensel's lifting. In Part II, we extend these results to the singular Toeplitz/Hankel-like case based on the MBA divide-and conquer algorithm and its combination with Hensel's lifting. We specify randomization and estimate the error/failure probability and the bit operation (Boolean) cost of some fundamental computations (such as rank and null space computation and solving consistent but singular linear systems).

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

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

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

1 Introduction

In Part I of this paper, we presented nearly optimal algorithms for solving a nonsingular Toeplitz/Hankel linear system of equations. This is a highly important problem with numerous applications to computations in sciences, engineering and communication. In many applications, however, one has to deal with singular Toeplitz/Hankel matrices or, more generally, matrices with structures of Toeplitz/Hankel type. For example Toeplitz/Hankel matrices associated with the polynomial gcd computation and Padé approximation become singular where the input polynomials have nonconstant gcd. Sylvester, subresultant, and other block matrices with Toeplitz blocks are examples of nonToeplitz matrices with structure of Toeplitz type. In this part of the paper, we cover the extension of the results of Part I to a generally singular Toeplitz/Hankel-like input.

Toeplitz/Hankel-like extension is immediate by means of the celebrated technique of the displacement representation of structured matrices (cf. [KS99], [P01]), and the bibliography therein.

To handle a singular input, we employ the MBA divide-and conquer algorithm of Morf 1974, 1980 and Bitmead and Anderson 1980, which computes the recursive triangular factorization of Toeplitz/Hankel-like matrices. This algorithm is well known (see [M74],[M80],[BA80] and the bibliography and further study in [P01]), but to decrease its computational precision and Boolean (bit operation) cost, we perform it modulo a smaller prime p. Unsuccessful choices of p may destroy the algorithm, but we estimate that this may only occur with a low probability when we choose p at random in an appropriately fixed range. The probability and bit cost analysis based on detailed elaboration of the algorithm are our main subjects. In particular, representation of the algorithm with a binary tree enables us to analyse its performance for a singular input and to specify the choice of the basic prime p. In this way, we arrive at nearly linear (that is, nearly optimal) randomized bit cost bounds. They apply to problems of computing the rank of the input matrix M and a nonsingular submatrix of the maximum size in the randomly preconditioned matrix UML, testing consistency of a linear system $M\mathbf{x} = \mathbf{b}$, and computing modulo p its solution (if the system is consistent) and a nontrivial vector in the null space of a singular matrix M. For the two latter problems the bit cost bounds cover verifying correctness of the output in $\mathbf{Z}_p$; in other cases erroneous output is possible but with a small controlled probability. The MBA output in $\mathbf{Z}_p$ can serve as the basis for application

of Hensel's lifting of Part I to yield nearly optimal algorithms for singular Toeplitz/Hankel-like computations.

Some elements of our analysis appeared earlier in [K95],[P00],[PZ00], and [P01, Chapter 5] but under the distinct models of computing (that is, PRAM in [P00] and arithmetic models in [K95], [PZ00] and [P01]), and nowhere to the comparable extent of elaboration. In particular, the bit cost bounds in all these works are inferior to ours.

We organize our presentation as follows. In Sections 2–4, we recall some definitions and basic facts for computations with Toeplitz/Hankel-like matrices. In Section 5, we elaborate upon the divide-and-conquer algorithm for recursive triangular factorization of nonsingular matrices, recall its applications to solving consistent linear suystems and computing the null space of a matrix, and estimate the arithmetic cost. In Sections 6 and 7, we specify the choice of a random prime p and estimate the error/failure probability and the bit cost where the computation is performed modulo p. In Section 8, we comment on the extension to computing the rational output values based on application of Hensel's lifting of Part I.

2 Definitions and basic facts

We begin with definitions and basic facts, partly repeating those of Part I.

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

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

Definition 2.3. $|M|$ *denotes the column norm of* M, $|M| = ||M||_1 = \max_j \sum_i |m_{i,j}|$ *for* $M = (m_{i,j})$.

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

Definition 2.5. *For a matrix M, let $M^{(k)}$ denote its $k \times k$* leading principal *(northwestern) submatrix. (Hereafter we use the abbreviation* l.p.s.*) M has* generic rank profile *if $M^{(k)}$ are nonsingular matrices for $k = 1, \dots, \rho$ where $\rho = \mathrm{rank}(M)$. A nonsingular matrix having generic rank profile is called* strongly nonsingular. *$v_S \le 2n^2 - n$ and i_S arithmetic operations are sufficient to multiply a given $n \times n$ matrix S by a vector and to invert it, respectively.*

To each arithmetic operation performed with d-bit precision, we asign the cost of $O(\mu(d))$ bit operations (hereafter log stands for $\log_2$),

$$\mu(d) \le C_{class} d^2, \mu(d) \le C_k d^{\log\ 3}, \mu(d) \le (C_{ss} d\ \log\ d) \log\ \log\ d, \qquad (2.1)$$

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

$m(n)$ is the arithmetic cost of multiplying two polynomials of degree n,

$$n \le m(n) \le (c_* n \log\ n) \log\log\ n \qquad (2.2)$$

3 Displacement representation of structured matrices

This paper specializes to the structure of Toeplitz/Hankel type but it is more convenient to state definitions in a more general form.

Definition 3.1. Displacement operators L of Stein and Sylvester types *map a matrix M into its displacements: $L(M) = \nabla_{A,B}(M) = AM - MB$ (a Stein type operator) or $L(M) = \Delta_{A,B}(M) = M - AMB$ (a Sylvester type operator). A and B are called* operator matrices, *$r = \mathrm{rank}(L(M))$ is called the L-*rank *of M or* the displacement rank *of M. If*

$$L(M) = GH^T,\ G = (\mathbf{g}_1, \dots, \mathbf{g}_\ell) \text{ and } H = (\mathbf{h}_1, \dots, \mathbf{h}_\ell) \qquad (3.1)$$

*are $l \times n$ matrices and M and $L(M)$ are $n \times n$ matrices, then the matrix pair (G, H) is called a (nonunique) L-*generator *(or* displacement generator*) of length l for $M, l \ge r$. If M is an $m \times n$ matrix and l is small relatively to m and n (as we specify, if $l = O(1)$ as $m + n \to \infty$), then M is said to have L-*structure *or to be an L-*structured matrix.

Theorem 3.2. *[P01, Theorem 4.6.4]. For a given L-generator (G, H) of length l for a matrix M having L-rank $r \le l$, it is sufficient to use $O(l^2 n)$ arithmetic operations to compute an L-generator (G_1, H_1) of length r for the matrix M.*

The next results are immediately verified.

Theorem 3.3. *[P01, Theorem 1.3.1]. If an operator matrix A is non-singular, then $\nabla_{A,B} = A\Delta_{A^{-1},B}$. If an operator matrix B is non-singular, then $\nabla_{A,B} = -\Delta_{A,B^{-1}}B$.*

Theorem 3.4. *Let M be a non-singular matrix, then $\nabla_{A,B}(M^{-1}) = -M^{-1}\nabla_{B,A}(M)M^{-1}$. Furthermore, if A is a non-singular matrix, then $\Delta_{A,B}(M^{-1}) = AM^{-1}\Delta_{B,A}(M)A^{-1}M^{-1}$, and if B is a non-singular matrix, then $\Delta_{A,B}(M^{-1}) = M^{-1}B^{-1}\Delta_{B,A}(M)M^{-1}B$.*

Theorem 3.5. *For two pairs of scalars a and b and matrices M and N of the same size and any linear operator L (in particular, for $L = \nabla_{A,B}$ and $L = \Delta_{A,B}$ for any pair of matrices A and B), we have $L(aM + bN) = aL(M) + bL(N)$.*

Theorem 3.6. *[P01, Theorem 1.5.4]. For a 5-tuple $\{A, B, C, M, N\}$ of matrices of compatible sizes, we have*

$$\nabla_{A,C}(MN) = \nabla_{A,B}(M)N + M\nabla_{B,C}(N)$$

and

$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N + AM\nabla_{B,C}(N).$$

Furthermore, if B is a non-singular matrix, then

$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N + AMB\Delta_{B^{-1},C}(N),$$

and if C is a non-singular matrix, then

$$\Delta_{A,C}(MN) = \Delta_{A,B}(M)N - AMB\Delta_{B,C^{-1}}(N)C.$$

Based on the results in this subsection, we may represent L-structured matrices M in compressed form via $(m+n)l$ entries of their short L-generators (rather than via mn entries of M) and operate with them according to the following flowchart:

$$COMPRESS \longrightarrow OPERATE \longrightarrow DECOMPRESS$$

Theorems 3.2 – 3.6 support the OPERATE stage, where in addition to saving the memory space, we may usually dramatically decrease computational time (see the next two subsections). Theorem 4.8 in the next section specifies the DECOMPRESS stage for the most popular class of structured matrices with Toeplitz/Hankel-like structure.

4 Toeplitz-like and Hankel-like matrices

Definition 4.1. *For any scalar f and vector $\boldsymbol{v} = (v_i)_{i=0}^{n-1}$ we define the $n \times n$ unit f-circulant matrix $Z_f = (z_{i,j}), z_{i,i-1} = 1, i = 2, \dots, n; z_{1,n} = f, z_{i,j} = 0$ otherwise, and the f-circulant matrix*

$$Z_f(\boldsymbol{v}) = \sum_{i=0}^{n-1} v_i Z_i^f$$

with the first column $\boldsymbol{v}$. (Note that $Z_f^n = fI$.) Hereafter we write $Z_0 = Z, Z_0(\boldsymbol{v}) = Z(\boldsymbol{v})$. The reflection matrix $J = (j_{g,h}), j_{g,n+1-g} = 0$, for $g = 0, \dots, n-1, j_{g,h} = 0$ for $h + g \neq n$, reverses any vector $\boldsymbol{v} = (v_i)_{i=0}^{n-1}$, that is, $J\boldsymbol{v} = (v_{n-i-1})_{i=0}^{n-1}, J^2 = I$.

Definition 4.2. *A matrix $T = (t_{i,j})$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$ for every pair of its entries $t_{i,j}$ and $t_{i+1,j+1}$. $H = (h_{i,j})$ is a Hankel matrix if $h_{i,j} = h_{i-1,j+1}$ for every pair of its entries $h_{i,j}$ and $h_{i-1,j+1}$.*

Fact 4.3. *Any f-circular matrix $Z_f(\boldsymbol{v})$ is a Toeplitz matrix; $Z(\boldsymbol{v})$ is a lower triangular Toeplitz matrix.*

Theorem 4.4. *TJ and JT are Hankel matrices if T is a Toeplitz matrix, and HJ and JH are Toeplitz matrices if H is a Hankel matrix.*

Theorem 4.5. *For any pair of distinct scalars e and f and any Toeplitz matrix T, there exist nonunique pairs $(Z_e(\boldsymbol{u}), Z_f(\boldsymbol{v}))$ and $(Z(\boldsymbol{w}), Z(\boldsymbol{x}))$ such that $T = Z_e(\boldsymbol{u}) + Z_f(\boldsymbol{v}) = Z(\boldsymbol{w}) + Z^T(\boldsymbol{x})$.*

Theorem 4.6. *Given an $m \times n$ Toeplitz or Hankel matrix, its multiplication by a vector is a subproblem of multiplication of two polynomials of degrees $m + n - 2$ and $n - 1$, whose coefficients are given by the entries of the input matrix and vector, respectively.*

Definition 4.7. *An $m \times n$ matrix M is* Toeplitz-like *if $r = \mathrm{rank}(L(M))$ is small relatively to $m+n$ (we specify that $r = O(1)$ as $m+n \to \infty$), where $L = \nabla_{Z_e, Z_f}, L = \nabla_{Z_e^T, Z_f^T}, L = \Delta_{Z_e, Z_f^T}, L = \Delta_{Z_e^T, Z_f}$ for a fixed pair of scalars e and f, and if M is given with its L-generator of length $l = O(r)$. (Any Toeplitz matrix T satisfies these requirements for $r \le l \le 2$.) A matrix M is* Hankel-like *if MJ or JM is Toeplitz-like.*

Theorem 4.8. *[P01, Examples 4.4.1 and 4.4.2]. Let (3.1) hold. Then*

$$(1-ef)M = \sum_{j=1}^{l} Z_e(\mathbf{g}_j) Z_f^T(\mathbf{h}_j) \quad \text{if} \quad L = \Delta_{Z_e, Z_f^T}, ef \neq 1,$$

$$(1-ef)M = \sum_{j=1}^{l} Z_e^T(J\mathbf{g}_j) Z_f(J\mathbf{h}_j) \quad \text{if} \quad L = \Delta_{Z_e^T, Z_f}, ef \neq 1,$$

$$(e-f)M = \sum_{j=1}^{l} Z_e(\mathbf{g}_j) Z_f(J\mathbf{h}_j) \quad \text{if} \quad L = \nabla_{Z_e, Z_f}, e \neq f,$$

$$(e-f)M = \sum_{j=1}^{l} Z_e^T(J\mathbf{g}_j) Z_f^T(\mathbf{h}_j) \quad \text{if} \quad L = \nabla_{Z_e^T, Z_f^T}, e \neq f.$$

Theorems 4.6 and 4.8 together imply the following corollary:

Corollary 4.9. *An $n \times n$ Toeplitz-like or Hankel-like matrix M given with its displacement generator of length l can be multiplied by a vector by using $v_M = O(m(n)l)$ field or ring operations for $m(n)$ in (2.2).*

The bound l in Definition 4.7 can be reduced to r due to Theorem 3.2. The next theorem enables us to choose any pair of e and f in Definition 4.7. In this paper it is sufficient to consider e and f in the set $\{-1, 0, 1\}$.

Theorem 4.10. *For any four-tuple of scalars (a, b, e, f) and any matrix M, the matrices $\Delta_{A(a),B(b)}(M) - \Delta_{A(e),B(f)}(M)$ and $\nabla_{A(a),B(b)}(M) - \nabla_{A(e),B(f)}(M)$ have rank of at most two, provided that $A(u) = Z_u, B(v) = Z_v$; $A(u) = Z_u, B(v) = Z_v^T$; $A(u) = Z_u^T, B(v) = Z_v$, or $A(u) = Z_u^T, B(v) = Z_v^T$*

Proof. Combine the relations $\Delta_{A,B}(M) - \Delta_{E,F}(M) = EM(F-B) + (E-A)MB$; $\nabla_{A,B}(M) - \nabla_{E,F}(M) = (A-E)M + M(F-B)$;

$$\mathrm{rank}(A-E) \le 1 \text{ if } A = Z_a, E = Z_e \text{ or if } A = Z_a^T, E = Z_e^T;$$

$$\mathrm{rank}(F-B) \le 1 \text{ if } B = Z_b, F = Z_f \text{ or if } B = Z_b^T, E = Z_f^T.$$

□

The next results extend the displacement representation of a Toeplitz/Hankel-like matrix to its blocks.

Theorem 4.11. *Let*

$$P_0 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, P_1 = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix}.$$

Then $\mathrm{rank}(Z_f P_i - P_i Z_f) \le 2$ *for* $i = 0, 1$ *and any* f.

Corollary 4.12. $\mathrm{rank}(L(P_i M P_j) - P_i L(M) P_j) \le 4$ *for* $i, j \in \{0, 1\}; L = \nabla_{Z_e, Z_f}, L = \Delta_{Z_e, Z_f^T}, L = \nabla_{Z_e^T, Z_f^T},$ and $L = \Delta_{Z_e^T, Z_f^T}$.

Fact 4.11 and Corollary 4.12 have simple constructive proofs, that is, given $i, j \in \{0, 1\}$ and an L-generator for 2×2 block matrix M of length l, we immediately compute an L-generator of length of at most $l + 4$ for the block $P_i M P_j$.

5 Recursive block triangular factorization for general matrices

To extend the algorithms of Part I to computations with singular matrices M, we first apply the known divide-and-conquer algorithms that recursively factorize a preconditioned input matrix to compute its nonsingular submatrix of the maximal rank. Our next goal is to specify and study the latter algorithms. We perform these algorithms modulo a moderately small random prime to keep the precision and bit complexity down. As soon as the desired nonsingular submatrix is computed, we apply the lifting algorithms of Part I, to yield rational or integer output values in **Q** or **Z**.

5.1 The case of a strongly nonsingular input matrix

We begin our study with a general matrix M represented as a 2×2 block matrix,

$$M = \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix}, \tag{5.1}$$

with a nonsingular $k \times k$ block $M_{00} = M^{(k)}$. We define the *Schur complement* of M_{00} in M as follows:

$$S = S(M, M_{00}) = S^{(k)}(M) = M_{11} - M_{10}M_{00}^{-1}M_{01}. \tag{5.2}$$

Now, we apply the block Gauss-Jordan elimination to M and obtain the well-known block triangular factorization

$$M = \begin{pmatrix} I & 0 \\ M_{10}M_{00}^{-1} & I \end{pmatrix} \begin{pmatrix} M_{00} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & M_{00}^{-1}M_{01} \\ 0 & I \end{pmatrix}. \tag{5.3}$$

$$\det(M) = \det(M_{00}) \det(S). \tag{5.4}$$

If M is nonsingular, then (5.1) implies that S is nonsingular, and we obtain

$$M^{-1} = \begin{pmatrix} I & -M_{00}^{-1}M_{01} \\ 0 & I \end{pmatrix} \begin{pmatrix} M_{00}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -M_{10}M_{00}^{-1} & I \end{pmatrix}. \tag{5.5}$$

(5.3) implies the following theorem.

Theorem 5.1. *S^{-1} is the southwestern block of M^{-1}.*

Factorizations (5.3) and (5.5) can be recursively applied to the matrices M_{00} and S provided that the leading principal (northwestern) submatrices of M_{00} and S are nonsingular. Let us show that this is always the case where M is strongly nonsingular. We begin with a simple observation that Schur complementation and projection of a matrix into its northwestern blocks are transitive.

Theorem 5.2. *Suppose that $h > 0, k > l > 0$, and $M^{(k)}$ and $M^{(l)}$ are nonsingular matrices. Then*
a) $(S^{(l)}(M))^{(h)} = S^{(l)}(M^{(h+l)})$,
b) $S^{(k-l)}(S^{(l)}(M)) = S^{(k)}(M)$.

Theorems 5.1 and 5.2 together imply the following corollary.

Corollary 5.3. *If an $n \times n$ matrix M is strongly nonsingular, then so are all its Schur complements $S^{(k)}(M), k = 1, \ldots, n-1$.*

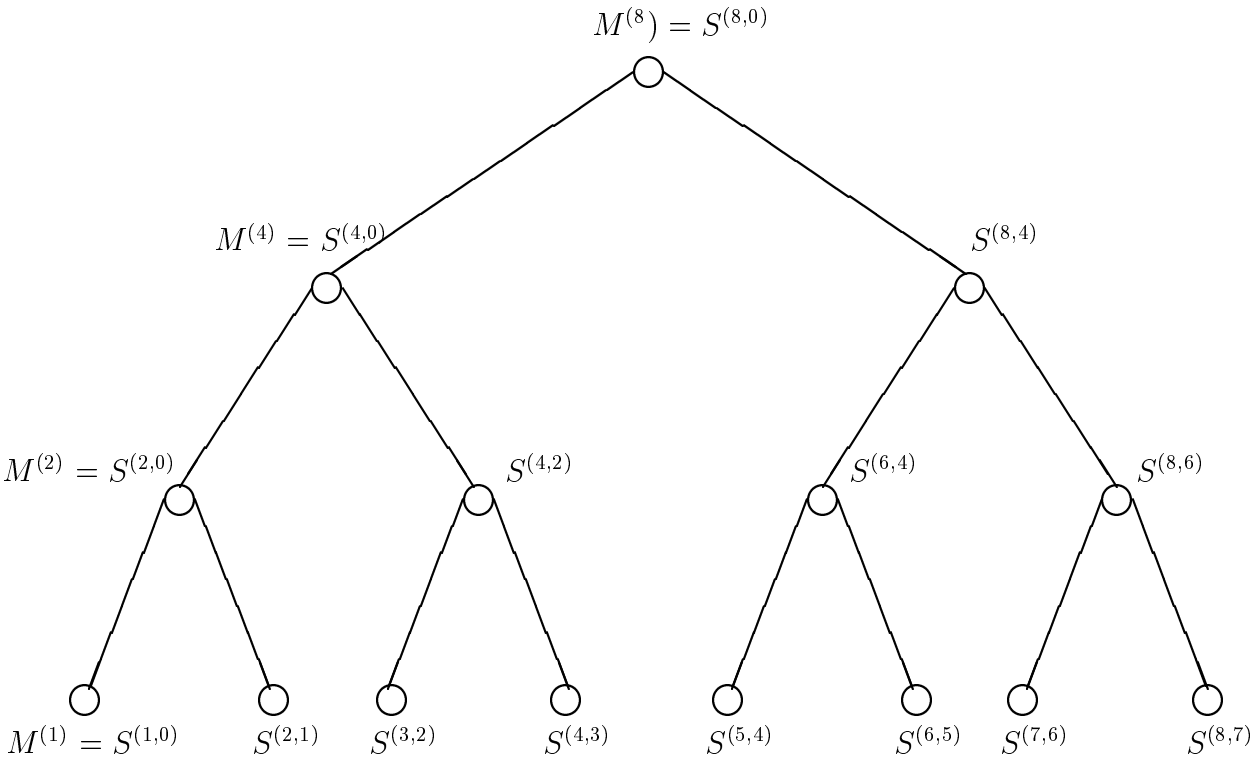

Figure 1: Generic BCRF tree $T_{8,8} = T(M)$ for an 8×8 matrix M.

By applying factorizations (5.3) and (5.5) recursively to M_{00} and S, where we choose $M_{00} = M^{(k)}$ with $k = \lceil n/2 \rceil$, and stopping when we arrive at 1-by-1 matrices, we obtain *the balanced complete recursive (block triangular) factorization* (we abbreviate this as BCRF) of a strongly nonsingular matrix M and its inverse.
Let us associate the BCRF of M with a binary tree $T_{n,n}$: M is the root with two children M_{00} and S, recursive extension of (5.3) and (5.5) to every internal node N of the tree defines two diagonal blocks (which are M_{00} and S for $N = M$) represented as the two children of N in the tree. The leaves of the tree are one-by-one matrices.
Figure 1 and 2 show two sample trees for 8×8 and 5×5 matrices M where we write $S^{(k,l)} = S^{(l)}(M^{(k)}), k \geq l$ (see Theorem 5.2a) and $S^{(k,0)} = M^{(k)}$.

Algorithm 5.4. *The BCRF of a strongly nonsingular matrix.*
INPUT: a strongly nonsingular $n \times n$ matrix M.
OUTPUT: M^{-1} and the BCRF of M and M^{-1}.
COMPUTATIONS: Define the BCRF tree $T(M) = T_{n,n}$, then recursively compute and invert all matrices associated with its nodes according to the following rules.
1. The left child of every node N is available automatically as a leading principal block of the parent node.
2. Invert the leaves (one-by-one matrices) directly, then recursively invert all other nodes based on factorization (5.5) and its recursive extension, that

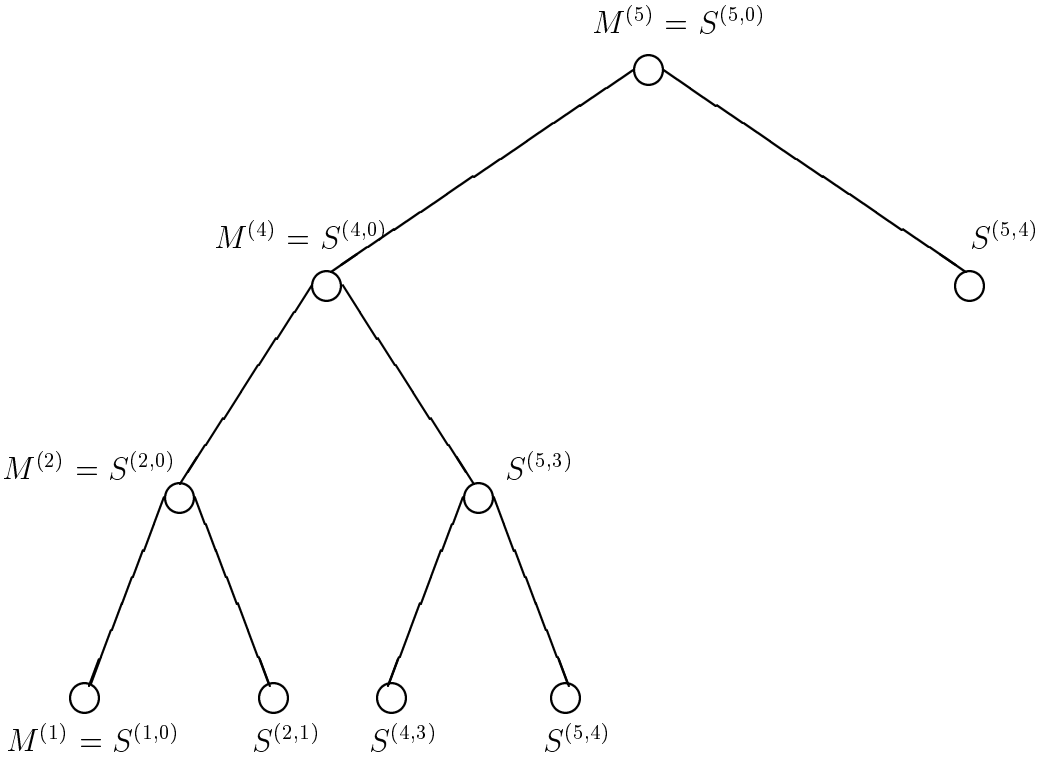

Figure 2: Generic BCRF tree $T_{5,5} = T(M)$ for a 5×5 matrix M.

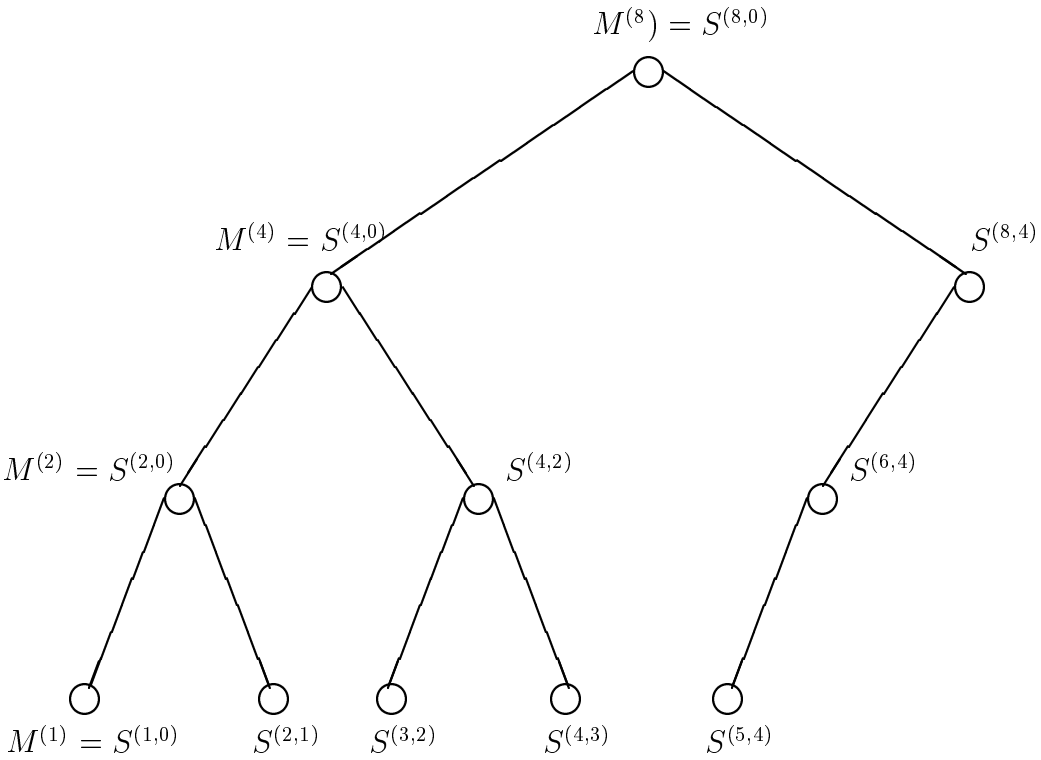

Figure 3: The BCRF tree $T(M) = T_{8,5}$ for Algorithm 5.4 applied to an 8×8 matrix M of rank 5 having generic rank profile.

is, in each recursive step first invert both children, immediately thereafter invert their parent. Inverting two siblings, proceed in the following order: invert the left sibling first, then immediately compute and invert its right sibling based on (5.2) or its extension.
3. Stop when the root M is inverted.

It is immediately verified that the latter rules completely and correctly define the computation of the BCRF.

For example, given an 8×8 matrix M in Figure 1, they define the following order (where $c(N)$ means "compute N", $i(N)$ means "invert N"):
$i(S^{(1,0)}), c(S^{(2,1)}), i(S^{(2,1)}), i(S^{(2,0)}), c(S^{(4,2)}), i(S^{(3,2)}), c(S^{(4,3)}), i(S^{(4,3)}),$
$i(S^{(4,2)}), c(S^{(8,4)}), i(S^{(5,4)}), c(S^{(6,5)}), i(S^{(6,5)}), i(S^{(6,4)}), c(S^{(8,6)}), i(S^{(7,6)}),$
$c(S^{(8,7)}), i(S^{(8,7)}), i(S^{(8,6)}), i(S^{(8,4)}), i(S^{(8,0)}).$

5.2 Extension to general input matrices

We first assume input matrices having generic rank profile. Then we relax this assumption.

Algorithm 5.5. *The BCRF of a submatrix of a matrix having generic rank profile.*
INPUT: a matrix M having a generic rank profile.
OUTPUT: $\rho = \text{rank}(M)$ and the BCRF of $M^{(\rho)}$ and its inverse.
COMPUTATIONS: Proceed as in Algorithm 5.4 but stop as soon as a computed leaf $S^{(k+1,k)}$ turned out to be the (one-by-one) null matrix; then output $\rho = k$ and the BCRF of the matrices $M^{(\rho)}$ and its inverse and stop. If $S^{(k+1,k)} \neq 0$ for all $k \leq n-1$, then write $\rho = n$ and stop when M is inverted (as in Algorithm 5.4).

The BCRF tree $T(M) = T_{n,\rho}$ computed with Algorithm 5.5 is a subtree of the BCRF tree $T_{n,n}$ associated with Algorithm 5.4. For $\rho < n$, the subtree can be defined by deleting from $T_{n,n}$ the leaf $S^{(\rho+1,\rho)}$ where the algorithm stops, together with all unprocessed leaves, that is, all leaves $S^{(i+1,i)}$ for $i \geq \rho$, and then by recursively deleting every parent having no child. The BCRF tree $T_{8,5}$ is shown in Figure 3.

Correctness of Algorithm 5.5 is easily verified. Furthermore, we may relax the assumption that M has generic rank profile and apply Algorithm 5.5 to any square matrix M. In this case, the output of the algorithm and the

associated BCRF tree are the same as originally except that some nonnegative integer ρ_1 replaces $\rho = \text{rank}(M)$ where $\rho_1 \le \rho$; consequently, the BCRF tree $T_{n,\rho}$ is replaced by T_{n,ρ_1}.

Algorithm 5.6. *The BCRF of a submatrix of general matrix.*
INPUT: an $n \times n$ matrix M.
OUTPUT: a nonnegative integer ρ_1 such that the matrix $M^{(\rho_1)}$ is strongly nonsingular, so $\rho_1 \le \rho = \text{rank}(M)$; the BCRF of $M^{(\rho_1)}$ and its inverse.
COMPUTATIONS: Proceed as in Algorithm 5.5, computing ρ_1 instead of ρ; as soon as a null matrix $S^{(k+1,k)} = 0$ is computed for some k, output $k = \rho_1$ and the BCRF of $M^{(k)}$ and its inverse.

Correctness of this algorithm immediately follows from (5.5) for $M_{00} = M^{(k)}$.

5.3 Extension to computing the null space and solving a singular linear system of equations

Let us show two immediate applications of the BCRF. Assume that $\rho = \text{rank}(M)$ and $(M^{(\rho)})^{-1} = M_{00}^{-1}$ have been computed by means of Algorithms 5.4 - 5.6. Let us compute the null space of M and solve a linear system $M\mathbf{x} = \mathbf{b}$. Write $\mathbf{v}^{(h)}$ for the subvector made up of the first h components of a vector $\mathbf{v}$ and write $\mathbf{0}_h$ for the null vector of dimension h. General solution to a consistent linear system $M\mathbf{x} = \mathbf{b}$ is given by the vectors $\mathbf{x} = \mathbf{x}^{(0)} + \mathbf{z}$, where $\mathbf{x}^{(0)}$ is a fixed specific solution and $\mathbf{z}$ is any vector in the null space of M. Combining Algorithms 5.7 and 5.9 below supports computing such a general solution. Each of these algorithms is of interest in its own right as well.

Algorithm 5.7. *The null space.*
INPUT: a field $\mathbf{F}$, a matrix $M \in \mathbf{F}^{n\times n}, \rho = \text{rank}(M), \rho < n$; and the inverse of $M^{(\rho)}$.
OUTPUT: a basis for the null space of M.
COMPUTATIONS: Substitute $M_{00} = M^{(\rho)}$ into (5.1). Compute the matrix $K = -M_{00}^{-1}M_{01}$ and output the columns of the matrix

$$F = \begin{pmatrix} K \\ I_{n-p.} \end{pmatrix}$$

Algorithm 5.8. *A nontrivial vector in the null space.*
INPUT: as in Algorithm 5.4.
OUTPUT: a vector $\mathbf{v} \neq \mathbf{0}$ in the null space of M.
COMPUTATIONS: the same as in Algorithm 5.7 but restricted to a single selected column (e.g., the first column) of M_{01}, K and F.

Algorithm 5.9. *A linear system.*
INPUT: as in Algorithm 5.7, but complemented by a vector $\mathbf{b} \in \mathbf{F}^n$.
OUTPUT: either INCONSISTENT or a solution $\mathbf{x}$ to the linear system of equations $M\mathbf{x} = \mathbf{b}$.
COMPUTATIONS: Compute the vector $\mathbf{x}^{(\rho)} = (M^{(\rho)})^{-1}\mathbf{b}^{(\rho)}$. Substitute the vector $\mathbf{x} = (\mathbf{x}^{(\rho)T}, \mathbf{0}^T)^T$ into the linear system $M\mathbf{x} = \mathbf{b}$. If $M\mathbf{x} = \mathbf{b}$, output $\mathbf{x}$. Otherwise output INCONSISTENT.

5.4 How to yield the generic rank profile property

If M is an $n \times n$ integer (or more generally, real) nonsingular matrix, then we may compute the BCRF of any of the two strongly nonsingular matrices $M^T M$ or MM^T and then compute and output $M^{-1} = (M^T M)^{-1} M^T = M^T (MM^T)^{-1}$ and $(\det M)^2 = \det(M^T M) = \det(MM^T)$.
The recipe does not work for singular matrices M such as

$$M = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

but the randomized Toeplitz preconditioning in [KS91] probabilistically ensures the generic rank profile property for any M.

Theorem 5.10. *[KS91]. Let $M \in R^{n \times n}$ for any ring R. Let S be a finite set of cardinality $|S|$ in R or its extension. Let L and U^T be a pair of $n \times n$ unit lower triangular Toeplitz matrices, defined by the $2n-2$ subdiagonal entries of the pair of the first columns of L and U^T. Let these $2n-2$ entries be randomly and independently of each other selected from the set S under the uniform probability distribution on S. Then the preconditioned matrix UML has generic rank profile with a probability of at least $1 - \delta$, where $\delta = \rho^2/|S| \leq n^2/|S|, \rho = \mathrm{rank}(M) \leq n$.*

In our case $R = \mathbf{Z}$, so for a fixed positive δ, we may choose

$$S = \{z \in \mathbf{Z} : |z| \leq \lceil n^2/(2\delta) \rceil\} \tag{5.6}$$

to ensure generic rank profile property with a probability of at least $1-\delta$. In this choice of S we generate at most $(2n-2)\lceil \log(n^2/(2\delta))\rceil$ random bits and we have

$$|UML| \le |U| \cdot |M| \cdot |L| \le n^6|M|/(4\delta^2). \tag{5.7}$$

5.5 The case of a Toeplitz/Hankel-like input

If M is a Toeplitz/Hankel-like matrix given with its shortest displacement generator of length r, then in computations in Algorithms 5.4 - 5.9 all matrices can be represented with their shortest displacement generators of length $O(r)$ based on Theorems 3.2 – 3.6, 4.10, 4.11 and 5.1. Likewise, preconditioning by M^T or by U and L still keeps the length of the displacement generators in $O(r)$.

5.6 Arithmetic Computational Cost Estimates

Suppose that n is a power of 2, let $F(n), I(n), M(n)$ and $A(n)$ be equal to the arithmetic cost for computing BCRF, matrix inversion, multiplication and addition/subtraction, respectively, assuming $n \times n$ input and nonsingularity whenever it comes to inversion. Hereafter, $M(n,q,r)$ denotes $n \times q$ by $q \times r$ (rectangular) matrix multiplication. Then

$$F(n) \le 2F(n/2) + 3M(n/2) + I(n/2) + A(n/2),$$

$$I(n) \le 2I(n/2) + 6M(n/2) + 2A(n/2).$$

By assuming strong nonsingularity of M and by applying these bounds recursively to $F(k)$ and $I(k)$ for $k = n/2, n/4, \ldots, 1$, we obtain that

$$F(n) = O(M(n)), I(n) = O(M(n)) \text{ if } M(n) \ge Cn^{\omega},$$

$$F(n) = O(M(n)\log\ n)I(n) = O(M(n)\log\ n) \text{ if } M(n) \le n\log^c\ n,$$

where c, C, and w are three constants independent of n, $C > 0, 3 \ge \omega > 1$. In the Toeplitz/Hankel case, these bounds are specialized to

$$F(n) = O(m(n)r^2\log\ n), I(n) = O(m(n)r^2\log\ n)$$

for Algorithm 5.4 and to the same bound but with ρ and ρ_1 replacing n for Algorithms 5.5 and 5.6, respectively. The latter bounds cover the

arithmetic cost of performing Algorithm 5.4 and also apply to computing $\det(M)$ due to (5.4). The same cost bounds with n replaced by ρ or ρ_1 apply to Algorithms 5.5 and 5.6, respectively. The arithmetic cost of preconditioning by the matrix M^T is $M(n)$, by the matrix U and L is in $O(nm(n))$ for $m(n)$ in (2.2) (see Section 5.2).
The arithmetic cost of performing Algorithms 5.7, 5.8 and 5.9 is bounded by $M(\rho, \rho, n-\rho), M(\rho, \rho, 1)$ and $M(n, \rho, 1)$, respectively.
For $n \times n$ Toeplitz/Hankel-like matrices M given with their shortest displacement generators of length of at most r and for $m(n)$ in (2.2), we have $M(n) = O(r^2 m(n))$, assuming all matrices represented by their shortest displacement generators of length $m^{O(r)}$. So for Algorithms 5.7, 5.8 and 5.9 in the Toeplitz/Hankel-like case we have

$$M(\rho, \rho, n-\rho) = O(m(n)r^2), M(\rho, \rho, 1) = O(m(\rho)r), M(n, \rho, 1) = O(rm(n)).$$

Let us summarize.

Theorem 5.11. *Let M be a Toeplitz/Hankel-like matrix given with its (shortest) displacement generator of length r. Then the arithmetic cost of performing Algorithms 5.4–5.9 and preconditioning is bounded as follows: $O(m(n)r^2 \log\ n)$ for Algorithm 5.4 (which also covers computing $\det(M)$); $O(m(\rho)r^2 \log\ \rho), \rho = \mathrm{rank}(M)$, for Algorithm 5.5; $O(m(\rho_1)r^2 \log\ \rho_1), \rho_1 \le \mathrm{rank}(M)$, for Algorithm 5.6; $O((n-\rho)m(\rho)r)$, for Algorithm 5.7; $O(m(\rho)r)$, for Algorithm 5.8; $O(m(n)r)$, for Algorithm 5.9; $O(m(n)r^2)$ for the preconditioning of M with M^T, and $O(m(n)r)$ for the Toeplitz preconditioning with U and L as in [KS91]; the latter bound does not cover the cost of generating the $(2n-2)\lceil \log\ (n^2/(2\delta))\rceil$ random bits involved. For Algorithm 5.7, the arithmetic cost bound can be turned into $O(m(n)r)$ if we represent the output matrix K with its short displacement generator.*

Remark 5.12. *For smaller ρ and/or larger r, it may still be faster to operate with the entries rather than displacement generators. E.g., the bounds of Theorem 5.11 could be restated as $O(\min\{m(n)r^2 \log\ n), n^\omega\})$ for Algorithm 5.4, $O(\min\{m(\rho)r^2 \log \rho, \rho^\omega\})$ for Algorithm 5.5, and so on. Furthermore, for $n \le 1, \rho \le 1$ and $\rho_1 \le 1$, the stated bounds degenerate, so $\log\ n, \log\ \rho$, and $\log\ \rho_1$ in Theorem 5.2 should be actually replaced by $\log\ (n+2), \log\ (\rho+2)$, and $\log\ (\rho_1+2)$, respectively. These and similar adjustments are left as an exercise for the reader.*

6 The choice of a random prime and the bit operation cost estimates

Performing Algorithms 5.4–5.9 modulo a smaller prime p keeps the precision and the Boolean cost low. The computation of the BCRF requires inversion of some matrices; they should not become singular in the reduction modulo p, that is, the reduction modulo p should not change the BCRF tree. By choosing random p in a proper range $(a, b]$, we ensure nonsingularity with a high probability even where b is in $n^{O(1)} \log |M|$. Our next goal is to specify the probabilistic estimates covering also possible errors/failure caused by the randomized preconditioning in Section 5.4. The estimates show a low probability of failure/errors even where random preconditioners U and L in Section 5.4 have entries in $n^{O(1)}$. We begin with a simple auxiliary result.

Theorem 6.1. *Suppose that Algorithms 5.4–5.9 and preconditioning are performed in* $\mathbf{Z}_p$ *for a random prime* p *from a fixed range* $(a, b]$,

$$a < p \le b.$$

Suppose the transition from the rational field $\mathbf{Q}$ *to* $\mathbf{Z}_p$ *(that is, reduction modulo* p *) throughout the algorithm does not change the BCRF tree of its input matrix. Then*
a) p *divides the differences between all pairs of values produced in* $\mathbf{Q}$ *and* $\mathbf{Z}_p$, *and*
b) the computation in $\mathbf{Z}_p$ *requires* $O(A\mu(\log\ p))$ *bit operations where* A *is the arithmetic cost in* $\mathbf{Q}$ *(see Theorem 5.11) and* $\mu(d)$ *is in 2.1.*

In this section we first complement Theorem 6.1 by estimating the probability that the transition from $\mathbf{Q}$ to $\mathbf{Z}_p$ changes the BCRF tree and thus causes the output error or failure of the algorithm. Then we extend this study to cover also the errors/failure caused by randomized preconditioning of Section 5.4. We study the Toeplitz/Hankel-like case and, furthermore, adopt the following realistic assumption.

Assumption 6.2. *An* $n \times n$ *integer matrix* M *is given with its shortest displacement generator* $(G/m, H)$, *where* G *and* H *are* $n \times r$ *integer matrices,* m *is a positive integer,*

$$\max\{|G|, |H|, m\} \le |M|^{\gamma r} n, \tag{6.1}$$

for a constant γ. For an integer Toeplitz/Hankel-like matrix M and operator matrices in Definition 4.7, with $e, f \in \{-1, 0, 1\}$, we have $m = 1, \gamma r = 1$.

The algorithm supporting Theorem 3.2 reduces the length of a rational displacement generator for M to its minimum and can be extended to yield the format $(G/m, H)$ with G, H, m satisfying (6.1).
We easily deduce the next result.

Theorem 6.3. *Under Assumption 6.2, let $\rho = \mathrm{rank}(M)$ and let M have generic rank profile. Then the BCRF tree $T_{n,\rho}$ for M in $\mathbf{Q}$ changes in the transition of the computations to $\mathbf{Z}_p$ for a prime p if and only if p divides*

$$D = m \prod_{i=1}^{\rho} |\det(M^{(i)})| \le m|M|^{(\rho+1)\rho/2}. \tag{6.2}$$

It remains to estimate the probability that p divides D. We begin with the following lemma.

Lemma 6.4. *Let a, b and H be positive functions in n, where H is an integral function, and $b \ge na > 1$. Let p be a random prime in the range $(a, b]$. Then*

$$\text{Probability}(p \text{ divides } H) \le (c \log\ H)(\log\ b)/(b \log\ a) \quad \text{for a constant } c.$$

Proof. Lemma 6.4 for $b = an$ is immediately implied by Corollary 7.8.2 in [P01], where $k(n) = \log_a H, f(n) = b, h(n) = H$ and $b = an$. The proof of the corollary is easily extended to the case where $b > an$.

□

Theorem 6.5. *Under the assumptions of Theorem 6.3, let $\rho_+ \ge \rho$, e.g., $\rho_+ = n$, and let p be a random prime in the range*

$$\rho_+^{u-1} \log\ |M^{(\rho_+)}| < p \le \rho_+^u \log\ |M^{(\rho_+)}| \tag{6.3}$$

for a real $u > 1$. Let D be defined by (6.2). Then

$$\text{Probability}(p \text{ divides } D) \le \hat{C}(\rho_+^2 + r)/\rho_+^u \tag{6.4}$$

for a constant $\hat{C}$.
Proof. Theorem follows from (6.1)–(6.3) and Lemma 6.4 for $H = D, a = \rho_+^{u-1} \log |M^{(\rho_+)}|$ and $b = \rho_+ a$.

□

To complete our probability estimates, let us relax the assumption that the matrix M has generic rank profile. If M is nonsingular, we simply apply Theorem 6.5 for $\rho_+ = \rho = n$ and M replaced by M^TM or MM^T. Otherwise we use randomized preconditioning in Theorem 5.10 for the set S in (5.6) and then apply Theorem 6.5 to the matrix M replaced by $\tilde{M} = UML$ and p satisfying

$$p > a > 2\left\lceil n^2/(2\delta)\right\rceil + 1 \geq (n^2/\delta) + 3. \tag{6.5}$$

Under (6.5) reduction modulo p does not changes the set S in (5.6), which thus retains all its $2\left\lceil n^2/(2\delta)\right\rceil + 1$ distinct elements. Then, by combining the above comments, the results of Section 5, and Theorem 6.5, we obtain the next randomized bit cost estimates. They are of Monte-Carlo type, that is, they do not cover the cost of verifying correctness of the output.

Theorem 6.6. *Under Assumption 6.2, let M be nonsingular. Let p be a random prime in the range (6.3) for $\rho_+ = n \geq r$ and M replaced by M^TM (or M^TM). Then the BCRF tree $T(M^TM)$ (or $T(M^TM)$, respectively) changes in the transition to computations in the field $\mathbf{Z}_p$ with a probability of at most Cn^{2-u} for a constant C; this probability is at most ϵ for $u \geq 2 + \log_n(C/\epsilon)$; in this case the random prime is represented with at most*

$$d_0 = \left\lceil \log_n((Cn^3/\epsilon)\log|M|)\right\rceil \tag{6.6}$$

bits, so $d_0 = O(\log\ n)$ if $\log\ |M| = O(n)$ and $\log\ (1/\epsilon) = O(\log\ n)$.

To simplify the estimates, let hereafter

$$r = O(\rho_+^2). \tag{6.7}$$

Theorem 6.7. *Under (6.7) and Assumption 6.2, let p be a random prime in the range (6.3) for $\rho_+ \geq \mathrm{rank}(M)$, for M replaced by $\tilde{M} = UML$, for U and L in Theorem 5.10 with S in (5.6), and for a satisfying (6.5). Then the BCRF tree $T(\tilde{M})$ changes in the transition of the computations from $\mathbf{Q}$ to $\mathbf{Z}_p$ with a probability of at most $\delta + C\rho_+^{2-u}$ for a constant C. This bound turns into at most $\delta + \epsilon$ if $u \geq 2 + \log_{\rho_+}(C/\epsilon)$. The latter inequality is compatible with (6.3),(6.5) and (5.7) if*

$$\epsilon \leq (C\rho_+\delta log\ |\tilde{M}|)/(n^2 + 3\delta) \leq C\rho_+\delta n^{-2}\log\ (n^6|M|/(4\delta^2)). \tag{6.8}$$

The computations defined by the BCRF tree require $2n-2$ parameters in the range (5.6) represented with a total of $(2n-2)\log(1+\lceil n^2/(2\delta)\rceil)$ bits besides a random prime p in the range (6.3) represented with at most

$$d_1 = \left\lceil \log((C\rho_+^2/\epsilon)\log\,(n^6|M|/(4\delta^2)))\right\rceil \tag{6.9}$$

bits.

7 Randomized bit complexity estimates

Due to the preceding section, we may confidently choose a random prime p in a proper range and then perform Algorithms 5.4–5.9 modulo p. This enables us to compute the rank ρ of M and a vector from its bull space, to test consistency of a linear system $M\mathbf{x}=\mathbf{b}$, and to compute modulo p a nonsingular submatrix $\tilde{M}^{(\rho)}$ of the maximum size $\rho\times\rho$ in a preconditioned matrix $\tilde{M}=UML$, together with its inverse $(M^{(\rho)})^{-1}$ modulo p. In all cases, we control the error probability and yield optimal bit complexity bounds (up to polylogarithmic factors). Next, we specify these bounds.
The preconditioning and the computation modulo p of the BCRF tree $T(\tilde{M})$ are performed with the precision of $\lceil\log\,p\rceil$ bits, where $\log\,p\le d_1$ in (6.9). If $\log\,|M|=O(n), \log\,(1/\delta)=O(\log\,n), \log\,(1/\epsilon)=O(\log\,n)$, then $d_1=O(\log\,n)$ in (6.9) and $\epsilon=O((\rho_+\delta\log\,n)/n^2)$ in (6.8).
By combining Theorems 4.11 and 6.7 with Algorithms 5.5 and 5.9, we deduce the next corollary.

Corollary 7.1. *Suppose Assumption 6.2 holds. Let $\mu(d)$ be defined in (2.1) and $m(n)$ in (2.2). Let $\rho_+\ge\mathrm{rank}(M)$. Let positive δ and ϵ satisfy (6.8). Then*
a) it is sufficient to generate a prime p in the range (6.3) represented with d_0 random bits for d_0 in (6.6) and in addition to perform $O((r^2m(n)\log\,n)\mu(d_0))$ bit operations in order to test with an error probability of at most ϵ if the input matrix is singular;
b) it is sufficient to generate $d_1+1+(2n-2)\log\,\lceil n^2/(2\delta)\rceil$ random bits for d_1 in (6.9) and in addition to perform

$$\beta_0=O((m(n)+rm(\rho)\log\,\rho)r\mu(d_1)\log(1/\epsilon)) \tag{7.1}$$

bit operations to compute $\rho=\mathrm{rank}(M)$ with an error probability of at most $\delta+\epsilon$ and

c) the bit cost and error probability bounds of parts a) and b) apply to computing modulo p shortest displacement generators for all nodes in the BCRF tree for the matrix $\tilde{M}^{(\rho)}$*, where* $\rho = n, \tilde{M} = M^T M$ *or* $\tilde{M} = MM^T$ *in part a) and* $\rho \le n, \tilde{M} = UML$ *in part b).*

The output of part c) is not our final goal. At the end of this section it is used as a basis for p-adic lifting. On the contrary, parts a) and b) supply randomized cost bounds for the properties of M independent of the choice of random p.
Next, based on the BCRF of $\tilde{M}$, we probabilistically test another p-independent property, that is, the consistency of the linear systems

$$\tilde{M}\mathbf{y} = U\mathbf{b} \quad \text{and} \quad ML\mathbf{y} = \mathbf{b},$$

which are equivalent to the system $M\mathbf{x} = \mathbf{b}$ for $\mathbf{x} = L\mathbf{y}$. Transition from $ML\mathbf{y} = \mathbf{b}$ to $ML\mathbf{y} - \mathbf{b} \mod p = \mathbf{0}$ for a random p in $(a, b]$ may cause erroneous labeling of the system inconsistent with a probability of at most $P(a, b) = n/(\phi(b) - \phi(a))$, where the numerator n corresponds to the n equations in the system and where $\phi(x)$ is Euler's totient function, $c_- \le (\phi(x) \log\ x)/x \le c_+$ for two constants c_- and c_+, so

$$P(a, b) \le cnd_1 2^{-d_1} \quad \text{for} \quad d_1 \quad \text{in} \quad (6.9) \quad \text{and a constant} \quad c. \tag{7.2}$$

Corollary 7.2. *The randomized asymptotic bound on the bit cost in Corollary 7.1b) also applies to computing modulo p the solution vector* $\mathbf{x} = \mathbf{x}^{(\rho)}$ *to the linear system* $M\mathbf{x} = \mathbf{b}$ *or detecting its inconsistency. Versus Corollary 7.1, the bound on the error probability in claiming the system inconsistent grows by at most* $P(a, b)$ *for* $P(a, b)$ *in (7.2).*

Remark 7.3. *Typically, we only have the upper bound* $\rho_+ = n$ *on* $\rho = \text{rank}(M)$ *until we compute* ρ*. We may, however, compute* ρ *probabilistically in an iterative search process that begins with applying Corollary 7.1b) (based on Algorithm 5.5) for a smaller candidate value* $\rho_+ = \rho_+^{(0)}$ *and then recursively repeats this step with doubling* ρ_+ *until the output value of* ρ *does not exceed the current* ρ_+*. In this process the bit operation cost bound increases by at most the factor of* $\beta = \left\lceil \log\ (2\rho/\rho_+^{(0)}) \right\rceil$*, denoting the number of recursive steps, whereas the number of random bits is bounded by* $(2n - 2) \log\ \lceil n^2/(2\delta) \rceil + d_2$ *for*

$$d_2 = \left\lceil \log\ ((C\rho^{0.5 \log\ (2\rho)}/\epsilon) \log\ (n^6 |M|/(4\delta^2))) \right\rceil \tag{7.3}$$

(d_2 equals d_1 for $\rho_+^4 = \rho^{\log\ (2\rho)}$). Instead of computing a single random prime in the range (6.3) for $\rho_+ = n$, the adaptive process computes β primes in the range (6.3) for $\rho_+^{(i)} = \rho_+^{(0)} 2^i < 2\rho, i = 0, 1, \ldots, \beta - 1$.

Based on Corollary 7.1c), we compute the matrix $Q_\rho = (\tilde{M}^{(\rho)}) \mod \rho$. Then application of Hensel's lifting Algorithm 3.1 in Part I with M replaced by $\tilde{M}^{(\rho)}$, $\mathbf{b}$ by $\mathbf{b}^{(\rho)}$, and Q by Q_ρ produces a solution $\mathbf{x}$ to the linear system $M\mathbf{x} = \mathbf{b}$ (see Algorithm 5.9). Likewise, we compute a nontrivial vector in the null space of M (see Algorithm 5.8). By applying the latter algorithm l times for $l \le n - \rho$, we compute l linearly independent vectors in the null space. For $l = n - \rho$, they form a basis in the null space (see Algorithm 5.7). By checking if $S(\tilde{M}, \tilde{M}^{(\rho)})$ is the null matrix, we verify that indeed $\rho = \mathrm{rank}(M)$. Let us summarize the bit cost estimates.

Theorem 7.4. *Under the assumptions of Corollaries 7.1 and 7.2, their randomized cost bounds can be extended to the problems of*
a) solving a consistent linear system $M\mathbf{x} = \mathbf{b}$ for $\mathbf{b}$ satisfying the weak bound $\log|\mathbf{b}| = O(n \log|M|)$, and
b) computing h linearly independent vectors in the null space of M for $h \le n - \rho$ (the vectors form a basis for the null space if $h = n - \rho$); the extension does not change the bounds on the number of random bits and the error probability, whereas the bit operation bound β_0 in (7.1) increases by

$$\beta_1 = O(r\rho m(\rho) \log\ (n|M|/\delta)\mu(\log\ p)/\log\ p)$$

for part a), and by

$$\beta_2 = O(r\ h\rho\ m(\rho)\ \log(n|M|/\delta))\mu(\log\ p)/\log\ p),$$

for part b), where $\mu(d)$ is defined in (2.1), $m(n)$ is in (2.2), and p is a random prime in the range (6.3) (as in Theorem 6.7). The cost bounds do not cover correctness verification. If
$\log\ |M| = O(n), \log\ (1/\delta) = O(\log\ n), \log\ (1/\epsilon) = O(\log\ n)$, then
$\log\ p = O(\log\ n), d_1 = O(\log\ n)$,

$$\beta_1 = O(\rho r m(\rho)\mu(\log\ n)),$$

$$\beta_2 = O(\rho h\ m(\rho)\mu(\log\ n)).$$

8 Extension to yielding rational output

The BCRF can be recomputed modulo sufficiently many random primes, and then the output values of Algorithms 5.4–5.9 as well as other values produced by the BCRF (e.g., $\det(M^{(\rho)})$ equal to the product of all nonzero diagonal entries in the BCRF) can be recovered by means of Chinese remaindering. This requires an extra factor of roughly $n \log |M|$ random bits. We may alternatively stay with a single random prime by applying Hensel's lifting in Part I, wherever the goal is the inversion of a matrix whose inverse modulo p has already been computed. The bit cost bounds in Part I apply. By lifting the entire BCRF, we obtain $\det(M)$ and then the integer matrix $\mathrm{adj}(M) = M^{-1} \det(M)$, thus avoiding the rational number reconstruction. The bit cost of computing the entire BCRF in this way has been estimated in [P00]. It is slightly (by a logarithmic factor) higher than the bit cost in our Part I for inverting a nonsingular Toeplitz/Hankel-like matrix (including both lifting and recovery).

References

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

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

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

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

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

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

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

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

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

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

[K95] E. Kaltofen, Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems, *Math. Comput.*, **62 (210)**, 777-806, 1995.

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

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

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

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

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

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

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

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

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

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

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

[P02] V. Y. Pan, Can We Optimize Toeplitz/Hankel Computations ?, preliminary report, 2002.

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

[PZ00] V. Y. Pan, A. Zheng, Superfast Algorithms for Cauchy-like Matrix Computations and Extensions, *Linear Algebra and Its Applications*, **310**, 83–108, 2000.