City University of New York (CUNY)
CUNY Academic Works

Computer Science Technical Reports | CUNY Academic Works

2004

TR-2004015: Superfast Algorithms for Singular Toeplitz-like Matrices

Victor Y. Pan

How does access to this work benefit you? Let us know!

More information about this work at: https://academicworks.cuny.edu/gc_cs_tr/251
Discover additional works at: https://academicworks.cuny.edu

This work is made publicly available by the City University of New York (CUNY).
Contact: AcademicWorks@cuny.edu

Superfast Algorithms for Singular Toeplitz-like Matrices *

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

September 13, 2004

Abstract

We apply the superfast divide-and-conquer MBA algorithm to possibly singular $n \times n$ Toeplitz-like integer input matrices and extend it to computations in the ring of integers modulo a power of a random prime. We choose the power which barely fits the size of a computer word; this saves word operations in the subsequent lifting steps. We extend our early techniques for avoiding degeneration while preserving the Toeplitz structure. Our resulting algorithm supports nearly optimal randomized bit cost estimates for the solution of possibly singular Toeplitz and Toeplitz-like linear systems of equations, various related fundamental matrix computations (rank, null space) as well as computing the univariate polynomial gcd and resultant, Padé approximation, and rational interpolation where all input values are integers.

0 Introduction

0.1 Background and our progress

Matrices with the structure of Toeplitz type are ubiquitous in computations in sciences, engineering, and signal processing (see, e.g., Kailath and Sayed (editors) 1999 [KS99], Pan 2000 [P00, Section 1.1], Pan 2001 [P01], and the bibliography therein). In computer algebra, some of the most fundamental problems amount to or can be reduced to solving possibly singular but consistent Toeplitz systems of linear equations $M\mathbf{x} = \mathbf{b}$. Algorithms in Pan 2002 [P02], [P02a], and Pan et al. [PMRWa] apply to nonsingular integer input and rely on Hensel's lifting. We accelerate these algorithms and extend them to the

*Supported by NSF Grant CCR 9732206 and PSC CUNY Awards 65393-0034 and 66437-0035

singular case by combining them with a variant of the superfast divide-and-conquer MBA algorithm due to Morf 1974, 1980 and Bitmead and Anderson 1980 [M74], [M80], [BA80], (also see [P01, Chapter 5]). This algorithm applies to the structured matrices having small displacement rank (see [KS99] and [P01] on this fundamental concept); the Toeplitz input is a special case.

0.2 The complexity of Toeplitz and Toeplitz-like computations modulo a prime power

Hereafter, log stands for $\log_2$, $m(n)$ is the arithmetic cost of multiplying two polynomials of degree n, and $\mu(d)$ is the bit operation cost of multiplying two integers in the range from -2^d to 2^d. We have

$$2n-1 \leq m(n) \leq \min\{c_{class}n^2, c_k n^{\log 3}, (c_{ck}n\log n)\log\log n\}, \quad (0.1)$$
$$2d-1 \leq \mu(d) \leq \min\{C_{class}d^2, C_k d^{\log 3}, (C_{ss}d\log d)\log\log d\}, \quad (0.2)$$

where $\log 3 = 1.5849625\ldots$, $c_{class} < c_k < c_{ck}$, $C_{class} < C_k < C_{ss}$, c_{class}, C_{class}, c_k, C_k, c_{ck} and C_{ss} are constants, and the abbreviations "class", "k", "ck", and "ss" refer to the "classical", "Karatsuba's", "Cantor and Kaltofen's", and "Schönhage and Strassen's" algorithms, respectively (see, e.g., Bernstein, 2003 [B03], von zur Cathen and Gerhard 2003 [GG03]). Hereafter $V = (v_{i,j})_{i,j}$ denotes the matrix with the (i,j)th entries $v_{i,j}$, $V^T = (v_{j,i})_{i,j}$ is its transpose, and $|V| = 1 + \max_{i,j}|v_{i,j}|$. The vector $\mathbf{v} = (v_i)_i$ is a special case of a matrix V.

Given two positive integers n and w, a positive ϵ, a random prime p such that $\log p = O(\log(n\log|M|))$, an integer vector $\mathbf{b}$, and a Toeplitz-like $n \times n$ integer matrix M of rank ρ, our solution modulo p^w of the a consistent linear system $M\mathbf{x} = \mathbf{b}$ requires generating $O(n\log\frac{n}{\epsilon})$ random bits and performing $A = O(m(n) + m(p)\log p)$ arithmetic operations with the precision of $\lceil w\log p\rceil$ bits provided that with a probability of at most ϵ we may output failure or erroneously conclude that the system is inconsistent. Clearly, the A arithmetic operations are the word operations if $\log p$ is within the length of a computer word. The computations amount to

$$B = O((m(n) + m(\rho)\log\rho)\mu(w\log p)) \quad (0.3)$$

bit operations. (0.3) exemplifies our general format of $O(a\mu(b))$ where $O(a)$ bounds the arithmetic cost and $O(b)$ bounds the bit precision of computing; we have $a = m(n) + m(\rho)\log\rho$ and $b = w\log p$ in (0.3).

The bound (03) covers the cost of the certification of the correctness of the output and thus the consistency of the system $M\mathbf{x} = \mathbf{b}$; the failure of our algorithm only implies the inconsistency of the system with a probability within $1-\epsilon$. We state this bound for a prime p randomly sampled in a sufficiently large interval, but it also holds for a large fixed prime p and a random Toeplitz integer matrix M due to an estimate in Daykin 1960 [D60].

0.3 Combination with Hensel's lifting and nearly optimal complexity bounds over the rationals

The results of our computations modulo p^w can be applied to initialize the generalized Hensel lifting algorithm in [PMRWa] followed by a rational number reconstruction algorithm based on the algorithms in [GG03], Wang and Pan 2003 [WP03], 2004 [PW04], or Monahan 2004 [M04]. To keep the precision of computing within the length λ of a computer word, we ought to let $p^w \leq 2^\lambda$. In this case only the reconstruction stage requires a higher precision of computing. To save lifting steps and word operations, we follow the recipe of the *saturated initialization* in [PMRWa] and choose $w = \lceil \lambda / \log p \rceil$ for a fixed prime p.

Under the Boolean (bit-operation) model of computing we nearly reach optimality for the solution of an integer Toeplitz linear system. Namely for this solution, our algorithm uses $O(nm(n)\mu(\log n) + n\mu(n \log n))$ bit operation. This is within the factor of $m(n)\frac{\mu(n)}{n^2}$ from the information lower bound of $n^2 \log n$; the latter bound follows because each n values representing the output may have up to $n \log n$ bits. Indeed, these values are the ratios of the pairs of determinants, whereas the determinants can be as large as $n^{\frac{n}{2}}$ because Hadamard's bound is generally sharp.

0.4 Extensions

Besides solving a linear system $M\mathbf{x} = \mathbf{b}$, the extended MBA algorithm probabilistically computes the determinant of M, the rank of M, a vector from the null space of M and a shortest generator for a matrix whose columns form a basis for the null space of M. The bit cost bound of (0.3) applies. Furthermore, the known techniques (see, e.g., [P01]) enable extension of the algorithm and its nearly optimal cost estimates to computing the univariate polynomial gcd, lcm, and resultant as well as Padé approximation and rational interpolation. The complexity estimates for system solving and computing a vector from the null space and the determinant are of the so called Las Vegas type, that is, they cover the correctness verification, but not so for the other listed problems, which are thus of the so called Monte Carlo type. In particular our algorithm outputs a lower bound on the rank of M which equals the rank with a probability of at least $1 - \epsilon$.

0.5 Comparison with related works

The algorithm in [PMRWa] supports the same complexity estimates as in this paper for the average integer Toeplitz matrix. For the worst case Toeplitz input and for a Toeplitz-like input, however, the estimates in [PMRWa] increase by the factor of n and thus exceed ours by this factor. The growth occurs at the initialization stage of solving the linear system $M\mathbf{x} = \mathbf{b}$ modulo p^w, $w \log p \leq \lambda$, and is avoided when we apply our version of the MBA algorithm modulo p^w.
The original works [M74],[M80], [BA80] have introduced the MBA algorithm

for the computations in the rational, real and complex fields. The lifting approach requires its implementation in the ring $\mathbb{Z}_{p^w}$ of integers modulo a prime power, and applications to polynomial computations require the coverage of a singular input. Randomized preconditioning in Kaltofen and Saunders 1991 [KS91] enables us to avoid the singularity issue, but one still should extend to the computations modulo p^w the algorithm for the compression of the displacement generators. This algorithm must be recursively applied in the MBA algorithm to yield the claimed overall complexity bounds. In Kaltofen 1995 [K95, Appendix] this problem is handled with introducing the order of $n \log^2 n$ random bits, but one may rather easily save the factor of $\log n$ bits [P01, Section 5.7]. Furthermore, in Pan 1992, [P92, Proposition A.6] the problem was solved deterministically over polynomial rings, and in [P01, Section 4.6.2] the latter deterministic solution was extended to any field. Our present paper carries the deterministic extension over to the ring of integers modulo p^w. We also clarify and elaborate upon the entire MBA construction in this ring, which is required to support the implementation of the algorithm in [PMRWa] since so far this has not been done clearly enough.

0.6 Organization of the paper

In the next three sections, we recall some background on structured matrices, Toeplitz/Hankel-like matrices, and Hensel's lifting. In Sections 4–6, we elaborate upon the MBA algorithm for recursive factorization of a matrix and related matrix computations. In Sections 8, we bound the precision of the computations. In Section 9, we cover the issues of pivoting and randomization versus degeneration. We estimate the arithmetic and Boolean (bit operation) complexity in Sections 7 and 10. In the Appendix we recall some relevant properties of displacement generators and demonstrate the BCRF trees by examples.

1 Displacement representation of structured matrices

We specialize our study to the matrix structure of Toeplitz type but next state some basic results for more general matrix structures (cf. [P01]).

Definition 1.1. Displacement operators L of the Stein and Sylvester types *map an $n \times n$ matrix M into its displacements $L(M) = \nabla_{A,B}(M) = AM - MB$ and $L(M) = \Delta_{A,B}(M) = M - AMB$, respectively. (The operators ∇ and Δ are closely related to one another (cf. Theorem A.2).) A and B are $n \times n$* operator matrices, *$r = \operatorname{rank} L(M)$ is the L-*rank *or the* displacement rank *of M. If*

$$L(M) = GH^T, \tag{1.1}$$

$$G = (\mathbf{g}_1, \ldots, \mathbf{g}_l) \quad \textit{and} \quad H = (\mathbf{h}_1, \ldots, \mathbf{h}_l) \tag{1.2}$$

*are $l \times n$ matrices, then the matrix pair (G, H) (nonunique for a fixed $L(M)$) is an L-*generator *(or a* displacement generator*) of length l for M, $l \geq r$. (G, H)*

is also a generator *of length l for $L(M)$. If M is an $m \times n$ matrix and l is small relatively to* $\min\{m, n\}$*, then M is said to have the L-structure or to be an L-structured matrix.*

Theorem 1.1. *[P01, Theorem 4.6.4]. Given an L-generator (G_1, H_1) of length l for a matrix M having L-rank $r \le l$, it is sufficient to use $O(l^2 n)$ arithmetic (field) operations to compute an L-generator (G, H) of length r for the matrix M.*

The theorem is supported in any field by the algorithm in [P01, Section 4.6.2].
We also need the following simple results.

Theorem 1.2.

a) $\nabla_{A,B}(M^{-1}) = -M^{-1}\nabla_{B,A}(M)M^{-1}$ *for a nonsingular M;*

b) $L(aM + bN) = aL(M) + bL(N)$ *for scalars a and b, matrices M and N, and any linear operator L, e.g., $L = \nabla_{A,B}$, $L = \Delta_{A,B}$, for any pair of matrices A and B;*

c) [P01, Theorem 1.5.4]: $\nabla_{A,C}(MN) = \nabla_{A,B}(M)N + M\nabla_{B,C}(N)$.

We may represent L-structured $m \times n$ matrices M in compressed form via $(m + n)l$ entries of their short L-generators and operate with these matrices according to the following flowchart:

$$\text{COMPRESS} \longrightarrow \text{OPERATE} \longrightarrow \text{DECOMPRESS}$$

(cf. [P01]).

Equation (2.1) is an example of compression. With Theorems 1.1 and 1.2, we dramatically decrease the computational time and memory space in the OPERATE stage (also see the next two sections). On the DECOMPRESS stage, see our Theorem 2.3, [P01, Sections 4.4–4.6], and Pan and Wang 2003 [PW03].

2 Toeplitz and Toeplitz-like matrices

Definition 2.1. *$T = (t_{i,j})_{i,j}$ is a Toeplitz matrix if $t_{i,j} = t_{i+1,j+1}$ for every pair of its entries $t_{i,j}$ and $t_{i+1,j+1}$. Especially, for any scalar f and vector $\mathbf{v} = (v_i)_{i=0}^{n-1}$, define the $n \times n$ unit f-circulant matrix $Z_f = (z_{i,j})_{i,j=0}^{n-1}$, $z_{i,i-1} = 1$, $i = 2, \dots, n$; $z_{1,n} = f$, $z_{i,j} = 0$ for other pairs (i, j), and define the f-circulant matrix $Z_f(\mathbf{v}) = \sum_{i=0}^{n-1} v_i Z_f^i$ with the first column $\mathbf{v}$. (Note that $Z_f^n = fI$ and that every matrix $Z_0(\mathbf{v})$ is lower triangular.)*

Theorem 2.1. *For any pair of distinct scalars e and f and any Toeplitz matrix T, there exist nonunique pairs $(Z_e(\mathbf{u}), Z_f(\mathbf{v}))$ and $(Z_0(\mathbf{w}), Z_0(\mathbf{x}))$ such that $T = Z_e(\mathbf{u}) + Z_f(\mathbf{v}) = Z_0(\mathbf{w}) + Z_0^T(\mathbf{x})$.*

Theorem 2.2. *Multiplication of a $k \times n$ Toeplitz matrix by a vector is a subproblem of multiplication of two polynomials of degrees $k+n-2$ and $n-1$; the coefficients are the entries of the input matrix and vector, respectively.*

Definition 2.2. *An $m \times n$ matrix M is* Toeplitz-like *if $r = \operatorname{rank} L(M)$ is small relatively to $\min\{m, n\}$ and if M is given with its L-generator of length $l = O(r)$, where $L = \nabla_{Z_e, Z_f}$, $L = \nabla_{Z_e^T, Z_f^T}$, $L = \Delta_{Z_e, Z_f^T}$, or $L = \Delta_{Z_e^T, Z_f}$ for a fixed pair of scalars e and f.*

For example, here is the displacement of a Toeplitz matrix $T = (t_{i-j})_{i,j=0}^{n-1}$ with $l \le 2$ [P01, page 120]:

$$\nabla_{Z_e, Z_f}(T) = Z_e T - T Z_f = (Z_e \mathbf{t}_- - f\mathbf{t})\mathbf{e}_{n-1}^T + \mathbf{e}_0 (e J\mathbf{t} - Z_f^T \mathbf{t}_-)^T \tag{2.1}$$

for $\mathbf{t} = (t_i)_{i=0}^{n-1}$, $\mathbf{t}_- = (t_{-i})_{i=0}^{n-1}$, and any pair of scalars e and f. Note that

$$|\nabla_{Z_e, Z_f}(M)| \le (\max\{1, |e|\} + \max\{1, |f|\})|M|. \tag{2.2}$$

It is sufficient for us to use e and f in the set $\{-1, 0, 1\}$ (see Theorem A.1).

Theorem 2.3. *[P01, Example 4.4.2]. Let (1.1) and (1.2) hold for $L = \nabla_{Z_e, Z_f}$, $e \ne f$. Then*

$$(e-f)M = \sum_{j=1}^{l} Z_e(\mathbf{g}_j) Z_f(J\mathbf{h}_j).$$

Theorems 1.1, 2.2, and 2.3 together imply the following corollary.

Corollary 2.1. *An $n \times n$ matrix M of displacement rank r given with its displacement generator of length l for $L = \nabla_{Z_e, z_f}$ can be multiplied by a vector by using $O(m(n)l)$ ring operations or alternatively $O(m(n)r + l^2 n)$ field operations for $m(n)$ in (0.1).*

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

Theorem 2.4. *Let $P_0 = \begin{pmatrix} I_l & 0 \\ 0 & 0 \end{pmatrix}$ and $P_1 = \begin{pmatrix} 0 & 0 \\ 0 & I_l \end{pmatrix}$ be $n \times n$ matrices. Then $Z_f P_0 - P_0 Z_f = f\mathbf{e}_{n-1}\mathbf{e}_0^T - \mathbf{e}_l \mathbf{e}_{l+1}^T$, $Z_f P_1 - P_1 Z_f = \mathbf{e}_{n-l-1}\mathbf{e}_{n-l} - f\mathbf{e}_{n-1}\mathbf{e}_0$ for any real f.*

Corollary 2.2. *Define P_0 and P_1 as in Theorem 2.4. Write $L = \nabla_{Z_e, Z_f}$,*
$\delta = \delta_{L,i,j}(M) = L(P_i M P_j) - P_i L(M) P_j = (Z_e P_i - P_i Z_f) M P_j - P_i M (P_j Z_f - Z_f P_i)$ for $i, j \in \{0, 1\}$. Then rank $\delta \le 4$.

3 Generalized Hensel's lifting for a linear system of equations and its initialization

Let us briefly recall the lifting algorithm in [P02, P02a, PMRWa]. Its input consists of three integers h, q and s, $h > 0, q > 0, s > 1$, a vector $\mathbf{b} \in \mathbb{Z}^n$, and two matrices $M \in \mathbb{Z}^{n\times n}$ and $Q = qM^{-1} \bmod (qs)$ satisfying

$$MQ \bmod (qs) = qI. \tag{3.1}$$

In h lifting steps, the vector $\mathbf{x}^{(h)} = (qM^{-1}\mathbf{b}) \bmod (qs^h)$ is output. Each step essentially amounts to multiplying the matrices M and Q by a vector with the precision of $O(\log \max\{qs, \gamma\})$ bits for $\gamma = 2n|M| + |\mathbf{b}|$. For a sufficiently large h in $O(n \log_s \gamma)$, the rational vector $x = M^{-1}\mathbf{b}$ is recovered from $\mathbf{x}^{(h)}$. For a Toeplitz input matrix M, the overall bit operation cost of lifting is bounded by

$$B = O(m(n)n\mu(d_1) \log_s \gamma), d_1 = \log \max\{qs, \gamma\} \tag{3.2}$$

for $m(n)$ in (0.1), $\mu(d)$ in (0.2). This also covers the cost of randomized Las Vegas recovery of $\mathbf{x}$ from $\mathbf{x}^{(h)}$.

The algorithms in [PMRWa] initialize lifting by computing a matrix Q and two integers q and s such that (3.1) holds where q and s are the powers of a fixed integer $p > 1$. These algorithms involve more than n^2 arithmetic operations with the precision of over n bits, that is, more than n^3 bit operations for the worst case input matrix M, but according to the estimates and experiments in [PMRWa] the precision and the bit cost are less than that by the factor for the average integer Toeplitz matrix M. In our present paper we dramatically accelerate the initialization for the matrix M in the worst case. We compute Q satisfying (3.1) for $q = 1$ and $s = p^w$ by performing $O(m(n) \log n)$ arithmetic operations with the precision of $\log s$ bits, that is, $O((m(n) \log n)\mu(\log s))$ bit operations.

Technically, our algorithm extends the divide-and-conquer MBA algorithm, which recursively factorizes a preconditioned input matrix and computes and inverts its nonsingular submatrix of the maximal rank.

4 Recursive block triangular factorization of a strongly nonsingular matrix

The reader may first assume computations with rationals, although we are going to specify these computations in the rings modulo a prime power later on.

Definition 4.1. *Let $M^{(k)}$ denote the $k \times k$ leading principal (that is, northwestern) submatrix of a matrix M. M has generic rank profile if $M^{(k)}$ are nonsingular matrices for $k = 1, \dots, \rho$ where $\rho = \operatorname{rank} M$. A nonsingular matrix having generic rank profile is called strongly nonsingular. Define the* Schur complement *of the nonsingular $k \times k$ block $M_{00} = M^{(k)}$ in a general 2×2 block*

matrix

$$M = \begin{pmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{pmatrix} \tag{4.1}$$

as follows:

$$S = S(M, M_{00}) = S^{(k)}(M) = M_{11} - M_{10}M_{00}^{-1}M_{01}. \tag{4.2}$$

Applying the block Gauss–Jordan elimination to M, obtain the well-known block triangular factorization

$$M = \begin{pmatrix} I & 0 \\ M_{10}M_{00}^{-1} & I \end{pmatrix} \begin{pmatrix} M_{00} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & M_{00}^{-1}M_{01} \\ 0 & I \end{pmatrix}. \tag{4.3}$$

$$\det M = (\det M_{00}) \det S. \tag{4.4}$$

If M is nonsingular, then (4.1) implies that S is nonsingular, and we obtain

$$M^{-1} = \begin{pmatrix} I & -M_{00}^{-1}M_{01} \\ 0 & I \end{pmatrix} \begin{pmatrix} M_{00}^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -M_{10}M_{00}^{-1} & I \end{pmatrix}. \tag{4.5}$$

(4.3) implies the following theorem.

Theorem 4.1. *If M is nonsingular, the S^{-1} is the southwestern block of M^{-1}.*

Theorem 4.2. *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 4.1 and 4.2 a) together imply the following corollary.

Corollary 4.1. *If an $n \times n$ matrix M is strongly nonsingular, then so are all Schur complements $S^{(l)}(M^{(k)})$, $l = 1, \dots, k-1$; $k = 2, \dots, n$.*

Due to this corollary, we may recursively extend factorizations (4.3) and (4.5) to the matrices M_{00} and S provided M is strongly nonsingular. Choose $M_{00} = M^{(k)}$ with $k = \lceil n/2 \rceil$, apply factorizations (4.3) and (4.5) recursively to M_{00} and S, and stop when you arrive at 1-by-1 matrices. This defines the *balanced complete recursive (block triangular) factorization* of a strongly nonsingular matrix M and its inverse (we use the abbreviation BCRF). Let us associate the BCRF of M with a binary tree $T_{n,n}$ as follows. M is the root with the left child M_{00} and the right child S. Extend (4.3) and (4.5) to every internal node N of the tree. Identify the two children of N in the tree as its leading principal block of the half-size (the left child) and the Schur complement of this block in N (the right child). This extends the assignment of the children M_{00} and S for $N = M$. The leaves of the tree are 1-by-1 matrices. We refer the reader to Appendix B for demonstration.

Algorithm 4.1. *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:*

Invert the leaves (1-by-1 matrices) directly, then recursively invert all other nodes based on factorization (4.5) *and its recursive extension. 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 compute and invert its right sibling based on (4.5) or its extension. Stop when the root M is inverted.*

Due to (4.4), we immediately extend Algorithm 4.1 to computing $\det M$.

5 Extension to matrices having generic rank profile

Algorithm 5.1. *The BCRF of a submatrix of a matrix having generic rank profile.*

INPUT: *a matrix M having generic rank profile.*

OUTPUT: *$\rho = \operatorname{rank} M$ and the BCRF of the submatrix $M^{(\rho)}$ and its inverse.*

COMPUTATIONS: *Proceed as in Algorithm 5.1 but with an additional provision in the case where a computed leaf $S^{(k+1,k)}$ equals zero; in this case 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 4.1).*

The BCRF tree $T(M) = T_{n,\rho}$ computed by using Algorithm 5.1 is a subtree of the BCRF tree $T_{n,n}$ associated with Algorithm 4.1. For $\rho < n$, the computation stops where a leaf equals 0. In this case we define the output subtree: first delete from $T_{n,n}$ the leaf $S^{(\rho+1,\rho)}$ together with all unprocessed leaves, that is, all leaves $S^{(i+1,i)}$ for $i \geq \rho$; then recursively delete every parent having no child. Simple techniques extend Algorithm 5.1 to computing the matrix rank and the null space and solving a singular linear system of equations [BP94, Section 2.2], [P01, Section 5.2], [P03, Section 6].

6 Towards the generic rank profile property

If a matrix M of rank ρ has generic rank profile, then, due to Corollary 4.1, we may safely compute the BCRF of the matrix $M^{(\rho)}$ and then invert this matrix over the rationals. Our next goal is to preprocess any matrix M to

turn it into a matrix having generic rank profile. If M is an $n \times n$ real nonsingular matrix, then we may compute the BCRF of any of the two strongly nonsingular matrices M^TM or MM^T and then compute and output $M^{-1} = (M^TM)^{-1}M^T = M^T(MM^T)^{-1}$ and $(\det M)^2 = \det(M^TM) = \det(MM^T)$. In fact we also have $\|W\|_2 \geq \|W^{(k)}\|_2$, $\|W\|_2 \geq \|S(W, W^{(k)})\|_2$ for all k if $W = M^TM$ or $W = MM^T$ (see [GL96]). The recipe does not work for singular matrices M such as $M = \left(\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, but various randomized preconditioners ([KS91], [P01, Sections 5.6 and 5.7], Chen et al. 2002 [CEKSTV02] and the bibliography therein) probabilistically ensure the generic rank profile property for any matrix M over any ring. Let us specify a preconditioner well suited for Toeplitz-like computation.

Theorem 6.1. *[KS91]. Let $M \in \mathbb{R}^{n\times n}$ for any ring $\mathbb{R}$. Let S be a finite set of cardinality $|S|$ in $\mathbb{R}$ 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 their first columns. 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 matrix UML has generic rank profile with a probability of at least $1-\bar{\epsilon}$, where $\bar{\epsilon} = \rho^2/|S|$, $\rho = \operatorname{rank} M \leq n$.*

For a fixed positive $\bar{\epsilon}$, we may ensure the above probabilistic bound $1-\bar{\epsilon}$ by choosing

$$S = \{z \in \mathbb{Z} : |z| \leq \lceil n^2/(2\bar{\epsilon})\rceil\}. \tag{6.1}$$

Under (6.1) we generate at most $(2n-2)\lceil \log(2\lceil n^2/(2\bar{\epsilon})\rceil)\rceil$ random bits and have

$$|UML| \leq n^2|U| \cdot |M| \cdot |L| \leq n^6|M|/(4\bar{\epsilon}^2). \tag{6.2}$$

7 The arithmetic cost estimates

Theorem 7.1. *Let M be a Toeplitz-like matrix given with its displacement generator of length r. Assume computations with rationals. Then the arithmetic cost of performing Algorithms 4.1, 5.1, and preconditioning in Theorem 6.1 is bounded as follows: $O(m(n)r^2 \log n)$ with $m(n)$ in (0.1) for Algorithm 4.1 (which also covers computing $\det M$); $O(m(\rho)r^2 \log \rho)$ with $\rho = \operatorname{rank} M$ for Algorithm 5.1; $O(m(n)r^2)$ for preconditioning M with M^T, and $O(m(n)r)$ for the Toeplitz preconditioning in Theorem 6.1.*

Proof. The displacement rank of all matrices in the BCRF of M, UML or M^TM is in $O(r)$ due to Theorem 4.1 and the results in Sections 1 and 2. By applying the algorithm that supports Theorem 1.1, we operate with their displacement generators of length $O(r)$. Now Theorem 7.1 follows from Corollary 2.1. □

Remark 7.1. *For smaller ρ and/or larger r, it can be faster to operate with the entries rather than displacement generators.*

8 Bounding the precision of computing

Assume that we are given a prime p, an integer $w \geq 1$, and a strongly nonsingular matrix M in $\mathbb{Z}_p$ and let us seek a matrix Q satisfying (3.1) for $q = 1$, $s = p^w$. We apply Algorithm 4.1 performing the computations in $\mathbb{Z}_{p^w}$. This means the precision $\lceil \log(p^w - 1) \rceil \leq \lceil w \log p \rceil$ and the bit operation complexity $A \log \mu(w \log p)$ where A denotes the arithmetic cost estimates in Theorem 7.1. Unless $w \log p$ exceeds the length λ of a computer word, A equals also the number of the word operations involved. Practically, one should choose

$$w = \lfloor \lambda / \log p \rfloor \tag{8.1}$$

to save the subsequent lifting steps and word operations. This policy was called *saturated initialization* in [PMRWa].

9 Degeneration and randomization

Since M is a strongly nonsingular matrix, we only need to care about degeneration at the stages of the compression of displacement generators, at which we employ Theorem 1.1 and the algorithm supporting it. To explain how to avoid degeneration in these computations, we need the following definition.

Definition 9.1. *Let $m, u \in \mathbb{Z}$, $m > 1$. Then u has the order s in m if s is the maximal integer such that $m^{-s}U \in \mathbb{Z}$.*

Now, we avoid degeneration at the compression stage by always choosing the pivot that has the smallest order in p among all candidate entries. It is easy to verify that this indeed enables us to avoid divisions by the multiples of p, even under the partial pivoting, where we minimize the order in p among the entries in the same column of a matrix.

We may relax the assumption that the matrix M is strongly nonsingular by relying on the application of Algorithm 5.1 (instead of Algorithm 4.1) and Theorem 6.1, which holds in the ring $\mathbb{Z}_{p^w}$. This enable us to handle nonsingular rather than strongly nonsigular input matrices M.

The next results show that the transition from $\mathbb{Z}$ to $\mathbb{Z}_p$ is unlikely to cause degeneration where p in $O(n \log(n|M|))$ is random and large enough and also where p is fixed and p/n is large while M is random.

Theorem 9.1. *Fix $\epsilon > 0$. Suppose that w is a positive integer, $M \in \mathbb{Z}^{n \times n}$ is nonsingular, and a prime p is randomly sampled from the range $(y/20, y]$ under the uniform probability distribution in this range where $y = \frac{n\xi \ln |M|}{w\epsilon} \geq 114$, $\xi = \frac{16 \ln 114}{16 \ln 5.7 - \ln 114} = 16\alpha\beta = 3.278885\cdots$, $\alpha = \frac{ln114}{16ln5.7} = 0.17006750\cdots$, and $\beta = \frac{1}{1-\alpha} = 1.2049303$. Then we have*

$$P = \text{Probability}((\det M) \bmod p^w = 0) < \epsilon. \tag{9.1}$$

Proof. See Theorem 10.1 in [PMRWa]. □

Theorem 9.2. *For two integers p (a prime) and positive n, the fraction of exactly $1-\frac{1}{p}$ Toeplitz matrices in $\mathbb{Z}_p^{n\times n}$ are nonsingular, and the fraction of exactly $(1-\frac{1}{p})\ (1-\frac{1}{p}+\frac{1}{p^2})^{n-1}$ such matrices are strongly nonsingular.*

Proof. See [D60] and Theorem 10.5 in Kaltofen and Lobo 1996 [KL96]. □

Theorems 9.1 and 9.2 show that the transition from $\mathbb{Z}$ to $\mathbb{Z}_p$ is unlikely to cause the degeneration of a matrix M where p is a sufficiently large random prime of the order of $n\log|M|$ as well as where p is a fixed moderately large prime, and M is a random integer Toeplitz matrix.

The extension of the latter estimates to the case of Toeplitz-like matrices is an interesting open problem. If we detect degeneracy in the process of computing the BCRF, we may change the basic prime p or try the heuristic recipe in [PMRWa, Section 10.3], that is, first invert in $\mathbb{Z}_p$ a matrix $M+GH^T$ for random (or for random Toeplitz) $n\times i$ matrices G and H for a small i; then invert it in $\mathbb{Z}$, and finally extend this to inverting in $\mathbb{Z}$ the matrix M based on the Sherman–Morrison–Woodbury formula [GL96, page 50].

10 The overall bit-complexity estimates

Let us summarize the overall bit complexity estimates of computing the triple (g,k,Q). Choose a random prime p and consider the problems of inverting a Toeplitz-like matrix M in $\mathbb{Z}_p$ as well as the extensions to linear system solving and null space computations, based on the well known techniques (e.g., in [BP94, Section 2.2], [P01, Section 5.2] or [P03, Section 6]).

Preconditioning in Section 6 does not affect the overall asymptotic bit complexity bound except that $2n-2$ random parameters must be generated and we should replace $|M|$ by $|M|n^6/(4\bar{\epsilon}^2)$. By computing a shortest displacement generator for $WV-I$ and verifying that it vanishes, one may certify that the output matrix W is the inverse of the input matrix V. Here we assume that the matrices W and V are given with their short displacement generators. We combine Theorems 1.1, 1.2, 6.1 and 7.1, and exploit the cited extensions to computing the matrix rank and a vector in (or a generator for the basis for) the null space and to solving singular linear systems of equations.

The following theorem summarizes the resulting complexity estimates.

Theorem 10.1. *Suppose that we are given two positive numbers ϵ and $\bar{\epsilon}$ and a sufficiently large positive y and let a prime p be randomly sampled under the uniform probability distribution in the range $(y/20,y]$. Let w be a positive integer. Let a displacement generator (G,H) of length $r=O(1)$ be given in $\mathbb{Z}_p$ for an $n\times n$ matrix M. Let $p>1+n^2/(2\epsilon)$ (cf. (6.1)) and $(\xi/\epsilon)\rho\ln(|M|n^6)/(4\bar{\epsilon}^2))\ge 114$ for $\rho=\operatorname{rank}M$ and the constant ξ in Theorem 9.1 where $w=1$. Then it is sufficient to define two random unit*

lower triangular Toeplitz matrices L and U^T by generating $2n-2$ random integers in the range $(-\lceil n^2/(2\bar{\epsilon})\rceil, \lceil n^2/(2\bar{\epsilon})\rceil]$ of (6.1), and in addition to perform $O((m(n) + m(\rho)\log\rho)\mu(w\log p))$ bit operations for $m(n)$ in (0.1) and $\mu(d)$ in (0.2) in order to output either FAILURE with a probability of at most $\epsilon + \bar{\epsilon}$ or a shortest displacement generator in $\mathbb{Z}_{p^w}$ for $((UML)^{(\rho)})^{-1}$. The latter bit cost bounds also cover solving in $\mathbb{Z}_{p^w}$ a linear system $M\mathbf{x} = \mathbf{b}$ where $\log|\mathbf{b}| = O(n\log|M|)$ or determining that the system is inconsistent in $\mathbb{Z}_{p^w}$ as well as the cost of computing the rank, a vector from the null space of M, and a displacement generator for a matrix whose columns form a basis for the null space of M. The cost bounds do not cover the certification of the rank, the basis, and the inconsistency of the linear system.

References

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

[B03] D. J. Bernstein, Fast Multiplication and Its Applications, preprint, 2003. Available from http://cr.yp.to/papers.html

[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.

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

[CEKSTV02] L.Chen, W. Eberly, E. Kaltofen, B. D. Saunders, W. J. Turner, G. Villard, Efficient Matrix Preconditioners for Black Box Linear Algebra, *Linear Algebra and Its Applications*, **343–344**, 119–146, 2002.

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

[D60] D. E. Daykin, Distribution of Bordered Persymmetric Matrices in a Finite Field, *J. Reine und Angewandte Math.,* **203**, 47–54, 1960.

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

[GG03] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge University Press, Cambridge, UK, 2003 (second edition).

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

[K95] E. Kaltofen, Analysis of Coppersmith's Block Wiedenmann Algorithm for the Parallel Solution of Sparse Linear Systems, *Math. of Computation*, **64, 210**, 777–806, 1995.

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

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

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

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

[M04] M. Monahan, Maximal Quatient Rational Reconstruction: an Almost Optimal Rational Reconstructions, *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC' 04)*, 243–249, ACM Press, New York, 2004.

[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.

[P90] V. Y. Pan, On Computations with Dense Structured Matrices, *Math. 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/New York, 2001.

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

[P02a] V. Y. Pan, Nearly Optimal Toeplitz/Hankel Computations, Technical Report TR 2002 017, *Ph.D. Program in Computer Science, Graduate Center of CUNY*, New York, 2002.

[P03] V. Y. Pan, Superfast Algorithms for Singular Toeplitz-Hankel-like Computations, Technical Report TR 2003 004 (revision of TR 2002 002), *Ph.D. Program in Computer Science, Graduate Center of CUNY*, New York, 2003.

[PMRWa] V. Y. Pan, B. Murphy, R. E. Rosholt, X. Wang, Toeplitz and Hankel Meet Hensel and Newton Nearly Optimal Algorithms and Their Practical Acceleration with Saturated Initialization, Technical Report TR 2004, *Ph.D. Program in Computer Science, Graduate Center of CUNY*, New York, 2004.

[PW03] V. Y. Pan, X. Wang, Inversion of Displacement Operators, *SIAM J. on Matrix Analysis and Applications*, **24, 3**, 660– 667, 2003.

[PW04] V. Y. Pan, X. Wang, On Rational Number Reconstruction and Approximation, *SIAM J. on Computing*, **33,2**, 502–503, 2004.

[W02] X. Wang, How Frequently Is the Matrix Nonsingular? Technical Report TR 2002 018, *Ph.D. Program in Computer Science, Graduate Center of CUNY*, New York, 2002.

[WP03] X. Wang, V. Y. Pan, Acceleration of Euclidean Algorithm and Rational Number Reconstruction, *SIAM Journal on Computing*, **32,2**, 548–556, 2003.

Appendix

A Some Properties of Displacement Operator

Theorem A.1. *For any 4-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 2, 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

$$\Delta_{A,B}(M) - \Delta_{E,F}(M) = EM(F - B) + (E - A)MB;$$
$$\nabla_{A,B}(M) - \nabla_{E,F}(M) = (A - E)M + M(F - B);$$
$$\text{rank}(A - E) \le 1 \text{ if } A = Z_a, E = Z_e \text{ or if } A = Z_a^T, E = Z_e^T;$$
$$\text{rank}(F - B) \le 1 \text{ if } B = Z_b, F = Z_f \text{ or if } B = Z_b^T, F = Z_f^T.$$

□

Theorem A.2. *[P01, Theorem 1.3.1]. If an operator matrix A is nonsingular, then $\nabla_{A,B}(M) = A\Delta_{A^{-1},B}(M)$. If an operator matrix B is nonsingular, then $\nabla_{A,B}(M) = -\Delta_{A,B^{-1}}(M)B$.*

B BCRF trees (Examples)

Figure 1 and 2 show two sample trees for 8×8 and 5×5 matrices M where we write $S^{(k,l)} = S^{(l)}(M^{(k)})$, $k \ge l$ (see Theorem 4.2 a) and $S^{(k,0)} = M^{(k)}$.

It is immediately verified that Algorithm 4.1 completely and correctly defines the computation of the BCRF. For example, given an 8×8 matrix M in Figure 1, it defines the following order (where $c(N)$ means "compute N", $i(N)$ means "invert N"):

$i(S^{(1,0)})$, $c(S^{(2,1)})$, $i(S^{(2,1)})$, $i(S^{(2,0)})$, $c(S^{(4,2)})$, $i(S^{(3,2)})$, $c(S^{(4,3)})$, $i(S^{(4,3)})$, $i(S^{(4,2)})$, $i(S^{(4,0)})$, $c(S^{(8,4)})$, $i(S^{(5,4)})$, $c(S^{(6,5)})$, $i(S^{(6,5)})$, $i(S^{(6,4)})$, $c(S^{(8,6)})$, $i(S^{(7,6)})$, $c(S^{(8,7)})$, $i(S^{(8,7)})$, $i(S^{(8,6)})$, $i(S^{(8,4)})$, $i(S^{(8,0)})$.

Figure 3 shows the BCRF tree for a singular 8×8 matrix of rank 5.

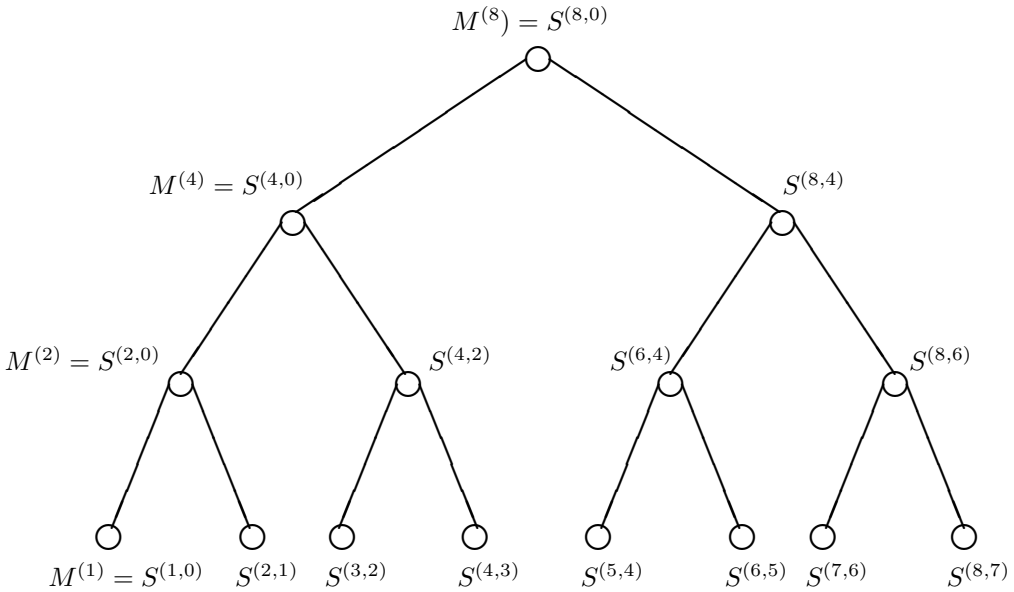

Figure 1: Generic BCRF tree $T_{8,8} = T(M)$ for an 8×8 matrix M.

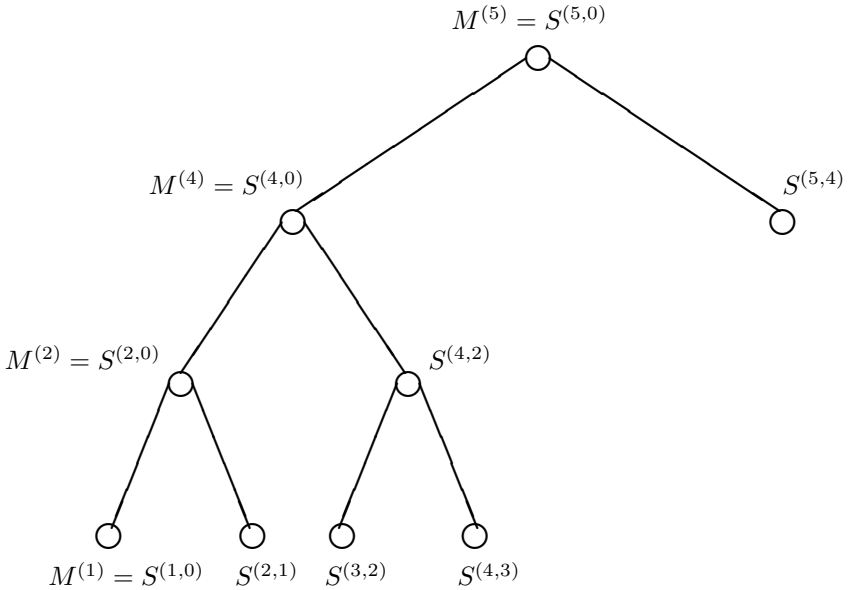

Figure 2: Generic BCRF tree $T_{5,5} = T(M)$ for a 5×5 matrix M.

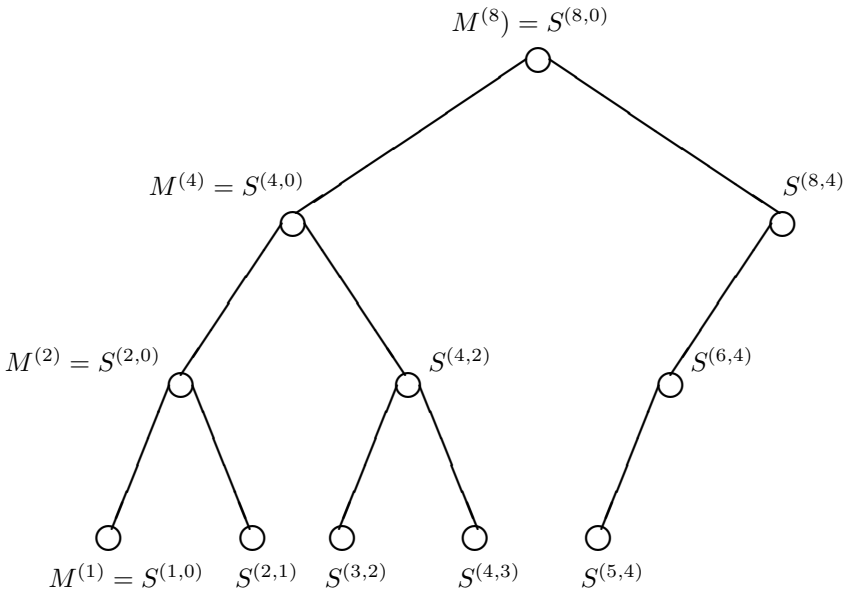

Figure 3: The BCRF tree $T(M) = T_{8,5}$ for Algorithm 4.1 applied to an 8×8 matrix M of rank 5 having generic rank profile.