Parallel Complexity of Computations with General and Toeplitz-like Matrices Filled with Integers and Extensions

Victor Y. Pan *
Department of Mathematics and Computer Science
Lehman College, City University of New York
Bronx, NY 10468
Internet: VPAN@LCVAX.LEHMAN.CUNY.EDU

Abstract[1]

Computations with Toeplitz and Toeplitz-like matrices are fundamental for many areas of algebraic and numerical computing. The list of computational problems reducible to Toeplitz and Toeplitz-like computations includes, in particular, the evaluation of the gcd, the lcm, and the resultant of two polynomials, computing Padé approximation and the Berlekamp-Massey recurrence coefficients, as well as numerous problems reducible to these. Transition to Toeplitz and Toeplitz-like computations is currently the basis for the design of the fastest known parallel (RNC) algorithms for these computational problems.

Our main result is in contructing nearly optimal randomized parallel algorithms for Toeplitz and Toeplitz-like computations and, consequently, for numerous related computational problems (including the computational problems listed above), where all the input values are integers and all the output values are computed exactly. This includes randomized parallel algorithms for computing the rank, the determinant, and a basis for the null-space of an $n \times n$ Toeplitz or Toeplitz-like matrix A filled with integers, as well as a solution $\mathbf{x}$ to a linear system $A\mathbf{x} = \mathbf{f}$ if the system is consistent. Our algorithms use $O((\log n)\log(n \log \|A\|))$ parallel time and $O(n \log n)$ processors, each capable of performing (in unit time) an arithmetic operation, a comparision, or rounding a rational number to a closest integer. The cost bounds cover the cost of the verification of the correctness of the output. The computations by these algorithms can be performed with the precision of $O(n \log \|A\|)$ bits, which matches the precision required in order to represent the output, except for the rank computation, where

*Supported by NSF Grants CCR 9020690 and CCR 9625344 and PSC CUNY Award No. 666327.

[1]The results of this paper have been presented at the ACM-SIAM Workshop "Mathematics of Numerical Analysis: Real Numbers Algorithms", Park City, Utah, 1995 (cf. [P96c]).

the precision of the computation decreases. The algorithms involve either a single random parameter or at most $2n-1$ ones.

The cited processor bounds are less by roughly factor n than ones supported by the known algorithms that run in polylogarithmic arithmetic time and do not use rounding to the closest integers.

Technically, we first devise new algorithms supporting our old nearly optimal complexity estimates for parallel computations with general matrices filled with integers. Then we decrease dramatically, by roughly factor $n^{1.376}$, the processor bounds required in these algorithms in the case where the input matrix is Toeplitz-like. Our algorithms exploit and combine some new techniques (which may be of independent interest, e.g. in the study of parallel and sequential computation of recursive factorization of integer matrices) as well as our earlier techniques of variable diagonal (relating to each other several known algebraic and numerical methods), stream contraction, and the truncation of displacement generators in Toeplitz-like computations; our development and application of these techniques may be of independent interest.

Key words: parallel algorithms, randomized algorithms, Toeplitz matrix computations, Toeplitz-like matrices, polynomial gcd, displacement rank, computational complexity, block Gauss-Jordan decomposition, p-adic lifting, Newton-Hensel's lifting.

1991 Mathematics Subject Classification: 68Q22, 68Q25, 68Q40, 65Y20, 47B35, 65F30.

1 Introduction

1.1 Toeplitz and Toeplitz-like matrices and some applications.

The fast version of Euclidean algorithm [AHU74], [BGY80] computes the greatest common divisor (gcd) of two polynomials of degrees at most n by using $O(n(\log n)^2)$ arithmetic or field operations (that is, additions, subtractions, multiplications and divisions), but to yield substantial parallel acceleration, one has to reduce the problem to the solution of the associated (possibly singular) Toeplitz-like (resultant or subresultant) or Toeplitz linear system of $O(n)$ equations, $T\mathbf{x} = \mathbf{f}$ [BGH82], [G84], [BP94]. (The concepts of Toeplitz and Toeplitz-like matrices are well known (see [KKM79], [CKL-A87], [BP94], pages 47-48, 138-141, 148-151), but for reader's convenience, we recall their definitions below (also see Definitions 2.8 and 13.1 in sections 2 and 13).

The gcd computation is only one (though celebrated) example of various major problems of algebraic and numerical computing whose solution is reduced to solving Toeplitz or Toeplitz-like linear systems of equations. The list of such problems includes the computation of the resultant, the Sturm and subresultant sequences, and the least common multiple (lcm) for a pair of univariate polynomials ([BT71], [BGY80], [BP94], sections 2.8-2.10), as well as the shift register synthesis and linear recurrence computation [Be68], [Ma75], inverse scattering [BK87], adaptive filtering [K74], [H91], modelling of stationary and nonstationary processes [KAGKA89], [KVM78], [K87], [L-AK84], [L-AKC84], numerical computations for Markov chains [Ste94], Padé approximation of an analytic function [BGY80], polynomial interpolation and multipoint evaluaton [PSLT93], [PZHY97], solution of partial differential and integral equations [Bun85], [C47/48], [KLM78], [KVM78], parallel computations with general matrices over an arbitrary field of constants [P91], [P92], [KP91], [KP92], approximating polynomial zeros [P95], [P96a], [P97], and the solution of polynomial systems of equations [EP97], [MP98], [BMP98].

Furthermore, the general reduction techniques of [P90] enable us to extend the algorithms available for Toeplitz and Toeplitz-like computations to computations with some other major classes of structured matrices, such as Cauchy-like and Vandermonde-like matrices, also highly important in many areas of computing [H95], [GKO95], [OP98].

The design of new effective algorithms for parallel solution of Toeplitz and Toeplitz-like linear systems will be our major goal. For reader's convenience, we will next briefly recall the definitions and some basic properties of Toeplitz and Toeplitz-like matrices (see section 13 and Definition 2.8 of section 2 for more details).

$T = (t_{i,j})$ is an $n \times n$ *Toeplitz matrix* if

$$t_{i,j} = t_{i+1,j+1} \quad \text{for} \quad i, j = 0, 1, \ldots, n-2, \tag{1.1}$$

that is, if the entries of T are invariant in their shifts into the diagonal direction. Toeplitz matrices are easy to store, since such an $n \times n$ matrix is fully represented by the $2n-1$ entries of its first column (or row, respectively) and its last column (or row). Multiplication of an $n \times n$ Toeplitz matrix by a vector can be reduced to three FFTs (e.g. via its reduction to polynomial multiplication modulo x^{2n-1} (see [BP94], page 133)) and can be performed by using $O(n \log n)$ arithmetic operations. Hereafter, arithmetic operations, as well as comparisons of pairs of rational numbers and the rounding of a rational number to a closest integer, are referred to as *ops*.

Due to the structural properties of Toeplitz matrices, one may solve a nonsingular Toeplitz linear system of n equations by using $O(n(\log n)^2)$ ops [BGY80], [Morf80], [BA80], [Mu81], [dH87], [AGr88], [K95]. (Note that we would need storage space $n^2 + n$ and $2n^2 - n$ ops to multiply a general matrix by a vector and order of n^d ops with $d > 2$ to solve a general nonsingular linear system of n equations.)

The above properties are extended to the class of $n \times n$ *Toeplitz-like matrices*, that is, ones represented in the form

$$T = \sum_{i=1}^{\ell} L_i U_i, \tag{1.2}$$

where L_i and U_i^T are $n \times n$ lower triangular Toeplitz matrices, U_i^T is the transpose of U_i, and ℓ is bounded by a fixed constant, $\ell = O(1)$. (Note that any $n \times n$ matrix can be represented in the form (1.2) for $\ell \leq n$.) It suffices to store the $2\ell n$ entries of the first columns of L_i and U_i^T for $i = 1, \dots, \ell$ in order to represent T. These 2ℓ columns form a pair of $n \times \ell$ matrices called a *displacement generator* of T of length ℓ.

Representation (1.2) for a Toeplitz-like matrix enables us to manipulate with $O(n)$ entries of its displacement generator (rather then with its n^2 entries). Furthermore, we may immediately multiply a matrix T of (1.2) by a vector by using $O(\ell n \log n)$ ops, also we may solve a linear system $T\mathbf{x} = \mathbf{f}$ in $O(\ell^2 n (\log n)^2)$ ops if T is nonsingular [Morf80], [BA80], [Mu81].

We have $\ell \leq 2$ in (1.2) for Toeplitz matrices, their inverses and resultant matrices, $\ell \leq g+h$ for $g \times h$ block matrices with Toeplitz blocks. Furthermore, the transposition of a matrix leaves ℓ invariant, whereas ℓ may grow only slowly in multiplication and addition/subtraction of pairs of matrices and stays unchange or grows only nominally in the inversion of a nonsingular matrix (see section 13).

1.2 NC and RNC solutions (some background).

Due to their reduction to Toeplitz/Toeplitz-like linear systems, several computational problems listed in the previous section, including the gcd, lcm, and resultant computation and Padé approximation, can be solved by using $O(n(\log n)^{d_1})$ ops, where n is the input size, and $d_1 \leq 3$. (We may need to allow $d_1 = 3$ in order to handle singular Toeplitz/Toeplitz-like linear systems; we reduce their solution to computing the rank of the coefficient matrix and to the subsequent solution of a nonsingular Toeplitz-like linear system.) Like Euclidean algorithm, however, such solution algorithms require order of n parallel steps.

The known alternative algorithms yield NC or randomized NC (RNC) solutions of all the cited computational problems [BGH82], [G84], [BP94], that is, yield their solution by using $t(n) = O((\log n)^c)$ time and $p(n) = O(n^d)$ arithmetic processors, for two fixed constants c and d, under the customary EREW PRAM arithmetic model of parallel computing [KR90], [J92]. (Alternatively, we may define the NC and RNC solutions as the families of arithmetic, Boolean or arithmetic-Boolean circuits for the above problems having depths $O((\log n)^c)$ and sizes $O(n^d)$ for two fixed constants c and d [G86].) Indeed, NC/RNC solution of a linear system of n equations can be computed over any field of constants [Cs76], [Be84], [Ch85], [KP91], [KP92]. These algorithms, however, leave open the important problem of processor efficiency of (R)NC Toeplitz and Toeplitz-like computations, that is, of having the ratios $p(n)/T_+(n)$ or even $p(n)/T_-(n)$ at the level $O((\log n)^{c_1})$ for a constant c_1, where $T_+(n)$ and $T_-(n)$ denote the record upper and lower bounds on the sequential time of the solution, respectively. Indeed, on the one hand, we have already cited the bound $T_+(n) = O(n(\log n)^3)$, and clearly, $T_-(n) \geq n$. On the other hand, $p(n)$ has order n^d for $d > 3$ in [Cs76], [Be84], [Ch85] and for $d > 2$ in [KP91], [KP92], for solving general linear systems in NC/RNC, whereas $p(n)$ has order n^d for $d \geq 2$ for the known NC/RNC Toeplitz/Toeplitz-like solvers over any field of constants [P92], [KP94], [P96], [P96b]. To yield NC/RNC and processor efficiency, we must decrease d to the optimal level 1. An approach toward this goal was outlined in [BP94], page 357, incorporating various nontrivial techniques developed earlier for computations with general and dense structured matrices [Morf80], [BA80], [P85], [P87], [P92], [P92b], [P93], [P93a], and our objective in the present paper is to show in detail how this can be done, under certain assumptions on the model of computing.

1.3 The model of computing.

Our main assumption is that the input consists of integers (for the reduction from a real input, one may use binary or decimal chopping followed by scaling) and that rounding a rational number to a closest integer, as well as an arithmetic operation or comparison of two rationals, are allowed as unit cost operations. The bit-precision of these computations will be bounded at the optimal level of the output precision, so that we achieve solution at a low Boolean cost.

Stating our estimates for the computational cost, we will let $O_A(t,p)$ and $O_B(t,p)$ denote the simultaneous bounds $O(t)$ on the parallel time and $O(p)$ on the number of arithmetic or Boolean processors, respectively. We will routinely decrease the processor bounds slightly, by exploiting the B-*principle* of parallel computing, which is a variant of Brent's principle and according to which $O(s)$ time-steps of a single arithmetic or Boolean processor may simulate a single time-step of s arithmetic or, respectively, Boolean processors [KR90], [PP95]. According to the B-principle, the bound $O_A(t,kp)$ implies the bound $O_A(tk,p)$, and similarly $O_B(t,kp)$ implies the bound $O_B(tk,p)$, for a parameter $k \geq 1$. (For $p = 1$, we arrive at sequential time bounds $O_A(tk,1)$ and $O_B(tk,1)$.) By applying the well-known technique based on the B-principle, one may slow down the computations at the stages requiring too many processors. In many cases this increases the time bound only by a constant factor but more substantially decreases the processor bound; a celebrated example is the summation of n values, where application of the B-principle decreases the asymptotic cost bound from $O_A(\log n, n)$ to $O_A(\log n, n/\log n)$ (cf. [Q94], pages 44–46; [BP94], pages 297–298). (The converse trade-off of time and processor bounds is not generally possilble, but for almost all matrix computations that we consider and, more generally, for any task of the evaluation of a set of multivariate polynomials, one may always transform an NC/RNC algorithm into one using $O(\log^2 n)$ time and $O(n^d)$ arithmetic processors for some finite but generally quite large constant d [VSBR83], [MRK88]. We will not use the latter result as we are concerned about processor bounds.)

Remark 1.1 *Our algorithms for Toeplitz/Toeplitz-like computations are essentially reduced to computing the convolutions (which can be performed via FFT, assuming that the 2^h-th roots of 1 are available) and the inner products of pairs of vectors. These basic operations (for vectors of a dimension n) can be performed at the cost*

$O_A(\log n, n)$ and $O_A(\log n, n/\log n)$, respectively, under both EREW PRAM model and more realistic models such as hypercube, butterfly and shuffle-exchange processor arrays [Le92], [Q94]. Thus, it is possible to implement our algorithms efficiently assuming the latter models.

1.4 Our main results.

The algorithms of this paper extend our previous work on parallel computations with general matrices [P85], [P87], [P93a], [BP94] (cf. also [PR91], [P92a], [P93b], [PR93]) by means of incorporation of some techniques developed in [P90], [P92], [P92b], [P93], [P93a] for computations with Toeplitz and Toeplitz-like matrices. As a result, we arrive at some the new RNC algorithms for the most fundamental computations with the latter classes of matrices filled with integers (such as the computation of their ranks, null-spaces and determinants and solving linear systems of equations). These algorithms yield optimal (up to polylogarithmic factors) time and processor bounds, which improves by factor n the processor bounds of the known RNC algorithms. By using the known reduction to Toeplitz and Toeplitz-like computations, we also extend our results to yield similar nearly optimal upper bounds on the time and processor complexity (also achieving order of n improvement versus the known RNC algorithms) for many other related computations (e.g. the computation of polynomial gcd and lcm and Padé approximation) where the input values are integers.

We will emulate the historic line, by first treating the case of general matrices and then improving the algorithms in the Toeplitz/Toeplitz-like case. We will start with recalling the record parallel complexity bounds of [P85], [P87] for computations with a general $n \times n$ input matrix; we will give their alternative derivation. Stating these bounds in Theorem 1.1 below, we will use the value ω satisfying $2 \leq \omega < 2.376$ and such that a pair of $n \times n$ matrices can be multiplied at the arithmetic cost $O_A(\log n, n^\omega)$. We note that the magnitudes of det A and the integer entries of adj $A = A^{-1}$ det A can be as large as $||A||^n$ or $||A||^{n-1}$, which means the output precision of an order of $n \log ||A||$. We ensure that the precision of the computations by our algorithms does not exceed this level. Furthermore, we compute the rank of A by using even a lower precision, which enables some decrease of the Boolean cost of the computation of the rank.

Technically, we will largely follow the cited outline of [BP94] and combine a variety of the known techniques, in particular ones developed in [P85], [P87], [P92b], [P93], [P93a], and some new ones (such as the combination of primal and dual recursive decompositions of an integer matrix with the objective to bound the magnitude of the intermediate and output values). The required combination of all these techniques is highly nontrivial and never was presented in either complete or accessible form.

The main purpose of our paper is to give such a presentation or, formally speaking, to give complete and accessible proofs of the two theorems below. The first of them only handles the case of general integer input matrices (in this case our parallel complexity results repeat ones of [P85], [P87], except for the presently improved Boolean cost of the rank computation), but we use distinct alternative proof, which should be technically interesting in its own right and is fully used in our subsequent relatively simpler extension to the Toeplitz/Toeplitz-like case, handled by our second theorem.

Theorem 1.1 *Let A be a $k \times h$ matrix and let $\mathbf{f}$ be an h-dimensional vector, both matrix and vector filled with integers that range from -2^a to 2^a for some $a > 1$. Let $k + h = O(n)$. Then, with an error probability of at most n^{-c} for a fixed positive constant c, one may compute r, the rank of A, at a randomized computational cost bounded by $O_A((\log n)\log(n \log a), n^{\omega})$ and by $O_B((\log n)(\log(n \log a))^2 \log\log(n \log a), n^{\omega+1}\log(na))$. Furthermore, one may compute the determinant of A and, if A is nonsingular, then also the inverse of A and the solution to a linear system $A\mathbf{x} = \mathbf{f}$, all of them at a randomized computational cost bounded by $O_A((\log n)\log(na), n^{\omega})$ and by $O_B((\log n)(\log(na))^2 \log\log(na), (\log n)(a + \log n)n^{\omega+1}/\log(na))$. If A is an $n \times n$ singular matrix, the latter bounds also apply to the computation of $n - r$ basis vectors of a null-space of A and a solution $\mathbf{x}$ to a linear system $A\mathbf{x} = \mathbf{f}$ provided that this system is consistent. The same cost bounds apply to testing correctness of the computed value of $r=$rank A as well as of all other output values. In these computations, $2n-1$ random parameters are used for computing rank A, $\det A$, and the null-space of A, and a single random parameter is used for all other tasks including the computation of $|\det A|$. The above complexity estimates do not cover the cost of generation of the random parameters.*

In the case of a $k \times h$ Toeplitz or Toeplitz-like input matrix (defined in section 1.1 and also in sections 2 (Definition 2.8) and 13 (Definition 13.1)), an extension of our approach yields much smaller (by factor $n^{\omega+1}/\log n$) upper bounds on the processor complexity of the same computations (with no increase of the asymptotic time-bounds):

Theorem 1.2 *Under the assumptions of Theorem 1.1, let the input matrix A be a Toeplitz matrix or a Toeplitz-like matrix. Then all the processor complexity estimates of Theorem 1.1 can be decreased by factor $n^{\omega-1}/\log n$, preserving the time bounds, to yield the randomized parallel complexity bounds $O_A((\log n)\log(n \log a), n \log n)$, $O_B((\log n)(\log(n \log a))^2 \log\log(n \log a), (n^2 \log n)\log(na))$ and $O_A((\log n)\log(na), n \log n)$, $O_B((\log n)(\log(na))^2 \log\log(na), (a+\log n)(n \log n)^2/\log(na))$, respectively. Here, the inverse of A and the basis matrix for the null-space of A are assumed to be output in the form of their displacement generators.*

We refer the reader to Remark 12.2 of section 12 on possible minor refinement of the estimates of both theorems.

Due to substantial economization of computational resources in our algorithms for Toeplitz/Toeplitz-like computations, they may become practically efficient provided that they are supported by subroutines for multiprecision parallel computations with integers and polynomials and by the development of the interface between algebraic and numerical computing, both required in our algorithms. Such a development is motivated by various potential benefits, our algorithms is but one of many examples. The practical implementation of our algorithms for general $n \times n$ matrices faces a harder problem of the storage of n^2 long integers in the computer memory (versus $2n-1$ in the Toeplitz case), and this task becomes practically infeasible at some point as n increases.

Our algorithms do not improve the known sequential algorithms for Toeplitz and Toeplitz-like computations [BGY80], [Morf80], [BA80], [Mu81], [dH87], which run in nearly optimal arithmetic time of $O(n \log^2 n)$, but some of our techniques may be of practical and theoretical interest for sequential computations too. In particular, our Toeplitz-Newton's iteration techniques are effective for rapid practical improvement of approximate solution of Toeplitz and Toeplitz-like linear systems of equations [PBRZ99], and our study of integral version of recursive decomposition as well as our bounds on the growth of the auxiliary integers (particularly, of the auxiliary determinants) is a natural but nontrivial extension of the Bareis version of Gaussian elemination. Even our simple idea of the precision decrease in the randomized computation of the rank (by performing the computation modulo a fixed prime) leads to a substantial decrease of the sequential Boolean time bounds (for both general and Toeplitz/Toeplitz-like matrices). The latter trick also applies to the closely related problem of the computation of the degree of the gcd and lcm of two polynomials with integer coefficients.

1.5 Extensions.

We already cited [BGY80], [G84], [P92], [P96b], and [BP94] on the reduction of the computation of the gcd, the lcm, and the resultant of two polynomials as well as Padé approximation of a formal power series or of a polynomial – to the computation of the rank of a Toeplitz/Toeplitz-like matrix and solving a nonsingular Toeplitz/Toeplitz-like linear system of equations. The integrality of the input can be preserved in this reduction, and the input size may grow by a factor of at most 2. Therefore, *the computational complexity estimates of Theorem 1.2 are immediately extended to the listed problems* of the gcd, lcm, resultant and Padé computations (assuming the restrictions on the size and integrality of the input), as well as to various computational problems reducible to the latter ones.

Furthermore, we refer the reader to [P90], on the general techniques that immediately enable extension of our results of Theorem 1.2 to computations with Cauchy-like and Vandermonde-like input matrices, to [BP93], [BP94] on the extension to the case of matrices represented as the sums of Hankel-like and Toeplitz-like matrices, and to [BGY80], [BP94], [P96b], [PSLT93], [PZHY97] and other references cited in the beginning of this paper on various applications of the computations with Toeplitz-like and other structured matrices (see also Remark 14.1 in section 14).

Among possible extensions of Theorem 1.1, consider the case where the integer matrix A is symmetric positive definite, sparse, and associated with an $s(n)$-separatable graph given with its $s(n)$-separator family (cf. [LRT79], [P93b], [PR93]). If such a matrix A is well-conditioned (even if its entries are not integers but any real numbers), then, at the arithmetic cost $O_A((\log n)^3, (s(n))^\omega / \log n)$, the parallel algorithm of [P93b], [PR93] numerically computes both recursive factorization of such a matrix and its determinant, as well as a solution $\mathbf{x} = A^{-1}\mathbf{f}$ to a linear system $A\mathbf{x} = \mathbf{f}$ (if $\det A \neq 0$). Numerical approximation is involved in this algorithm at the auxiliary stages of matrix inversions, where a parallel algorithm of [PR89] is applied. If A is filled with integers, then this stage can be performed exactly, by using the algorithms of [P85], [P87]. Then, the exact recursive factorization of A, $\det A$, and $A^{-1}\mathbf{f}$ can be computed at the arithmetic cost $O_A((\log n)^3, (s(n))^\omega + n)$. By employing the algorithm of this paper for recursive decomposition and inversion of a general integer matrix, one may improve the latter bounds a little, to yield $O_A((\log n)^2, (s(n))^\omega + n)$.

The results of Theorems 1.1 and 1.2 can be further extended to various other matrix computations, by using the known reduction techniques of [BP94], [P96], [P96b]. For demonstration, consider the computation of the characteristic polynomial $c_A(x) = \det(xI - A)$ of the above sparse $n \times n$ matrix A. Such a polynomial has degree n. We may first concurrently compute $c_A(x)$ at $n+1$ distinct points $x_0, \ldots, x_n$ and then obtain its coefficients by interpolation. If the chosen values of x_i are larger than $n\|A\|$, then the matrices $x_i I - A$ are positive definite, and we may compute $c_A(x_i)$ for $i = 0, 1, \ldots, n$, at the overall computational cost $O_A((\log n)^2, ((s(n))^\omega + n)n)$. These bounds dominate the cost of the subsequent interpolation producing the polynomial $\det(xI - A)$.

As another example, the algorithms of [BP94], page 357, for Padé approximation and polynomial gcd have been used in [P95] and [P96a] in order to obtain the record parallel arithmetic complexity estimates for approximating polynomial zeros. Our present improvement of these results of [BP94] in Theorem 1.2 immediatley implies the respective minor improvement of the results of [P95] and [P96a]:

Corollary 1.1 *Given a positive b and the cofficients of an n-th degree monic polynomial with zeros $z_1, \ldots, z_n$ satisfying $\max_i |z_i| \leq 1$, one may compute approximations $z_1^*, \ldots, z_n^*$ to $z_1, \ldots, z_n$ satisfying $|z_i^* - z_i| < 2^{-b}$, $i = 1, \ldots, n$; the computation is randomized; its arithmetic cost is bounded by $O_A((\log n)^3((\log n)^2 + \log(b+2)), \frac{n}{\log n})$.*

1.6 Outline of the method.

A major ingredient of our approach is the *variable diagonal method* of [P85], [P87], which combines several algebraic and numerical techniques to yield effective parallel inversion of a matrix A filled with integers. The method includes *Newton's iteration*, which effectively solves the latter problem provided that a good *initial approximation* to A^{-1} is available. Such an approximation is not available, however, for a general integer matrix A. The recipe of [P85], [P87] is to invert at first the auxiliary matrix $F = V - apI$, where $V = A \bmod p$, I is the identity matrix of an appropriate size, p is a prime, $p \geq n$, and a is a sufficiently large integer. (We follow this recipe and show that it suffices to choose $a = 10pn^2$ in our case.) Then the matrix $-I/(ap)$ is a good initial approximation to F^{-1}, which we rapidly improve by Newton's iteration, until F^{-1} is approximated closely enough. Since F is an integer matrix, $\det F$ and the entries of adj $F = (\det F)F^{-1}$ are integers, which can be recovered by rounding their approximations within absolute errors less than $1/2$. This gives us $(\det A) \bmod p$, (adj A) mod p, and $A^{-1} \bmod p$. Then the algebraic technique of *p-adic (Newton-Hensel's) lifting* is applied. In ℓ the matrix steps, for a sufficiently large ℓ, the matrix $A^{-1} \bmod p$ is lifted to $A^{-1} \bmod p^L$, $L = 2^l$. Then the lifting of $A^{-1} \bmod p$ is extended to lifting similarly (det A) mod p and (adj A) mod p. Finally, det A and adj A are easily recovered from $(\det A) \bmod p^L$ and (adj A) mod p^L.

The remaining ingredient is the approximation of $\det F$. In [P85], [P87], this is achieved as a by-product of solving the more general task of computing $\det(xI - F)$, the characteristic polynomial of F. In the present paper, we employ a more routine approach, based on the computation of *recursive* (block) *decomposition* (RD) of F (cf. [St69], [Morf74], [Morf80], [BA80], [P87]) or, equivalently, on the computation of nested *Schur's complements*, also called *Gauss transforms* (cf. [C74], [F64]). A single recursive step of this approach is the decomposition of the input matrix (represented as a 2×2 block matrix) into the product of a block diagonal matrix and two block triangular matrices (see equation (2.3) in section 2). Such a decomposition can be obtained by block Gauss-Jordan elimination and can be reduced to a few matrix multiplications (their parallel implementation is simple) and inversions (they are made simple by Newton's iteration, since good initial approximations are given by matrices $-I/(ap)$). As in [P85], [P87], random choice of a large prime p in a fixed large interval enables us to avoid degeneration and singularities (with a high probability).

An important point, as in [P85], [P87], is that in spite of computing all the matrix inverses approximately, we finally recover them exactly (as well as all the other matrices involved in the RDs) by exploiting the representation of their entries as the ratios of integers. To emplasize this point, we called the resulting RDs the *integral RDs* (*IRDs*). The only remaining nontrivial problem in the computation of the IRD of F and det F is *to bound the magnitudes* of the integers involved. In the present paper, this problem is solved based on the computation of the *dual RD*, that is, the RD of F^{-1}. (The celebrated techniques of [B68] do not suffice, and their extension to our case is nontrivial not just because we deal with recursive block decomposition, rather than with the more customary Gaussian elimination, but also because we need to control the magnitudes of the determinants of the matrices involved in the RD, which is a much harder problem, and we use the dual RD in order to solve it.)

As soon as the IRDs of F and F^{-1} are available, we obtain the RDs modulo p of A and A^{-1}. Now, we apply the techniques of p-adic (Newton-Hensel's) lifting not only to $A^{-1} \bmod p$ but to the entire RDs modulo p of A and A^{-1}, in order to obtain the RDs modulo p^L of A and A^{-1} for $L = 2^l$ and a sufficiently large l. $(\det A)$ mod p^L is recovered from such an RD of A. As det A is an integer, we recover easily det A and then the matrix adj $A = A^{-1}/\det A$, whose entries are integers.

This approach only gives us an alternative derivation of the estimates of [P85], [P87] for parallel complexity of some fundamental computations with general matrices. The new algorithms, however (unlike ones of [P85], [P87]), have an advantage of allowing their effective extension to Toeplitz/Toeplitz-like case. Indeed, manipulation with displacement generators, rather than with matrices themselves, enables the decrease of the processor complexity of the RNC algorithms outlined above to the optimal level linear in n.

The nontrivial problem in such a Toeplitz/Toeplitz-like extension is the control of the length of the displacement generators in the process of Newton's iteration. (Uncontrolled growth of the length would immediately imply the growth of the processor bounds by factor $n^{\omega-1}/\log n$ for $\omega > 2.375$.) We solve this problem by applying two techniques of *truncation of generators (TG)*, which we borrow from [P92] and [P92b], [P93], [P93a], respectively.

The above outline was essentially given in [BP94]. Presently, we also add the technique of *stream contraction* specified in section 10 (and, essentially, being the *pipelining* of the two processes of RD and Newton's iteration) borrowed from [PR91]. Stream contraction enables additional acceleration of our algorithms by factor $\log n$. (Using the technique of stream contraction for the acceleration of Toeplitz-like computations was also proposed in [R95], though the algorithms of [R95] did not give any improvement of the processor bounds in the Toeplitz /Toeplitz-like case versus the much larger bounds known in the case of general input matrices (see our Remarks 6.1, 11.1 and 14.2 in sections 6, 11 and 14 and our similar comments in [P96b])).

1.7 Organization of the paper.

The order of our presentation will slightly differ from the one outlined above. After some preliminaries in section 2, we will introduce the recursive decomposition (RD) and extended RD (ERD) of a matrix in section 3. In section 4, we define the integral RD (IRD) and show the transition from RD to IRD. We recall an algorithm for approximate matrix inversion via Newton's iteration in section 5 and apply it in order to approximate the RD and the IRD of an integer matrix in section 6. We estimate the errors and parallel complexity of these computations in sections 7-9. We apply pipelining (stream contraction) to achieve acceleration by factor $\log n$ in section 10, extend the results of sections 6 and 10 to computing the ERD modulo a fixed prime in section 11, and use p-adic lifting to recover (from the ERD) the inverse, the determinant, the rank, and the null-space of an integer matrix (thus proving Theorem 1.1) in section 12. In section 13, we recall some known definitions and properties for computations with Toeplitz and Toeplitz-like matrices. In section 14, we apply these properties to improve the results of section 12 in the Toeplitz and Toeplitz-like cases (thus proving Theorem 1.2). Section 15 is left for a brief discussion.

Acknowledgements Detailed and thoughtful comments by a referee and by a reviewer helped me a great deal to improve my original draft and to make it more accessible for the reader. The request by the area editor Joachim von zur Gathen to incorporate the appendix into the body of the paper also served the same goal.

2 Some Definitions and Auxiliary Results for Matrix Computations

We will next recall some customary definitions and well-known basic properties of general matrices.

Definition 2.1 (matrix notation). *I and 0 denote the* identity *and* null *matrices, of appropriate sizes. W^T is the* transpose *of a matrix or vector W.* diag $(w_i)_{i=0}^{n-1} =$ diag $(w_0, \dots, w_{n-1})$ *is the* diagonal matrix *whose diagonal is filled with $w_0, \dots, w_{n-1}$; $D(W) =$* diag $(W) =$ diag $(w_{i,i})_{i=0}^{n-1}$ *for a matrix $W = (w_{i,j})$.* rank W *is the* rank *of W.* det W *is the* determinant *of a square matrix W.* adj W *is the* adjoint (adjugate) matrix *of W, equal to W^{-1}* det W *for a nonsingular matrix W.*

In our error analysis, we will use the customary vector and matrix norms [GL89/96].

Definition 2.2 (vector and matrix norms) $\| \mathbf{v} \| = \| \mathbf{v} \|_1 = \sum_i |v_i|$, $\| \mathbf{v} \|_2 = \left(\sum_i v_i^2 \right)^{1/2}$, *for a real vector $\mathbf{v} = (v_i)$. $\| W \|_g = \max_{\|\mathbf{v}\|_g = 1} \| W\mathbf{v} \|_g$, $g = 1, 2$; $\| W \| = \| W \|_1$, for a matrix W.*

Proposition 2.1 (norm bounds). $\| W \| = \| W \|_1 = \max_j \sum_i |w_{i,j}|$, *for a matrix $W = (w_{i,j})$. Furthermore, if W is a $k \times k$ matrix and V is its submatrix, then*

$$||V||_g \leq ||W||_g, \; g = 1, 2;$$

$$||W||/k^{1/2} \leq ||W||_2 \leq ||W||k^{1/2}.$$

Proof. To prove the bound $||V||_g \leq ||W||_g$, note that $||W\mathbf{w}||_g \geq ||V\mathbf{v}||_g$ if $\mathbf{v}$ is a subvector of $\mathbf{w}$ and if $\mathbf{w}$ has zero components corresponding to the columns of W that are not in V. Other claimed relations can be found in [GL89/96]. □

We will also use the following known fact (cf. [GL89/96] or [BP94]):

Proposition 2.2 (bounds on the determinant and the entries of the adjoint matrix). *Let W be a $k \times k$ matrix. Then $|\det W| \leq (\| W \|_g)^k$, and furthermore, $|v| \leq (\| W \|_g)^{k-1}$ for every entry v of adj W, where $g = 1, 2$.*

Definition 2.3 (column-diagonally dominant (c.-d.d.) matrices). $d(W) = \| WD^{-1}(W) - I \|$. *A matrix W is* column-diagonally dominant *(hereafter, we will use the abbreviation* c.-d.d.*) if $d(W) < 1$.*

Definition 2.4 (leading principal submatrix (l.p.s.) and its Schur complement). *For a $k \times k$ matrix W, let $W^{(q)}$ denote its $q \times q$ northwestern or* leading principal submatrix *(hereafter, we will write* l.p.s.*), formed by the intersection of the first q rows and the first q columns of W, $q = 1, 2, \ldots, k$. If B is a nonsingular l.p.s. of W and if*

$$W = \begin{pmatrix} B & C \\ E & G \end{pmatrix}, \tag{2.1}$$

then the matrix

$$S = S(W, B) = G - EB^{-1}C \tag{2.2}$$

is called the Schur complement *of B in W.*

Schur complement S of (2.2) can be obtained by Gaussian or block Gaussian elimination applied to the matrix W, provided that the elimination process can be carried out (cf. [GL89/96], P3.22, page 103). In particular, it is easily verified that for $k > q$ the Schur complement of $B = W^{(q)}$ in a $k \times k$ matrix W is the $(k-q) \times (k-q)$ matrix obtained from W in q steps of Gaussian elimination (without pivoting) provided that these steps can be carried out (with no division by 0). The latter assumption holds, in particular, if W is a c.-d.d. matrix.

By applying block Gauss-Jordan elimination to the 2×2 block matrix W of (2.1), with a nonsingular block B, we obtain the following decomposition, which will be fundamental for our study:

$$W = \begin{pmatrix} I & 0 \\ EB^{-1} & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & B^{-1}C \\ 0 & I \end{pmatrix}. \tag{2.3}$$

If a matrix W is nonsingular, then (2.3) implies that the matrix S is also nonsingular. By inverting the matrices on both sides of (2.3), we obtain that

$$\tilde{W} = W^{-1} = \begin{pmatrix} I & -B^{-1}C \\ 0 & I \end{pmatrix} \begin{pmatrix} B^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -EB^{-1} & I \end{pmatrix}. \tag{2.4}$$

(2.4) immediately implies

Proposition 2.3 *Under (2.1)–(2.4), the matrix S^{-1} is the trailing principal (that is, southeastern) submatrix of $\tilde{W} = W^{-1}$.*

Our algorithms will rely on the decompositions of (2.3), (2.4), recursively applied to the matrices B, B^{-1}, S and S^{-1}, which we will call *recursive decompositions (RDs).* The next definitions and results will cover the nonsingularity properties required for the existence of such recursive extension of (2.3), (2.4) and some other relevant properties of l.p.s.'s and Schur complements (the s.p.d. matrices will be used only at the very end of sections 12).

Definition 2.5 (strongly nonsingular matrices). *A matrix W is* strongly nonsingular *if all its leading principal submatrices are nonsingular.*

Definition 2.6 (symmetric positive definite (s.p.d.) matrices). *A real matrix M is* symmetric positive definite (s.p.d.) *if it can be represented as the product AA^T for a nonsingular matrix A.*

The next proposition (cf. [BP94], exercise 4c, page 212) extends strong nonsingularity and the s.p.d. property to an l.p.s. and a Schur complement.

Proposition 2.4 *If a matrix W of (2.1) is strongly nonsingular (respectively, if W is s.p.d.), then so are its every l.p.s., including the matrix B of (2.1), and the Schur complement S of B, defined by (2.2).*

Corollary 2.1 *Any s.p.d. matrix is strongly nonsingular.*

By recursively applying block Gaussian elimination at first to the block matrix W of (2.1) with $B = W^{(r)}$ and then to $W^{(r)}$, with the *l.p.* (that is, leading principal or northwestern) block $W^{(q)}$, $q < r$, we obtain

Proposition 2.5 (transitivity of Schur's complementation). *If $r > q$ and if $W^{(r)}$ and $W^{(q)}$ are nonsingular matrices, then $S(W, W^{(q)}) = S(S(W, W^{(r)}), S(W, W^{(r)})^{(r-q)})$.*

We also easily deduce

Proposition 2.6 (transitivity of the c.-d.d. property). *If $d(W) < 1$ for a matrix W of (2.1), then B, S, and W are nonsingular matrices, $d(B) \leq d(W) < 1$, $d(S) \leq d(W) < 1$, and (2.3)–(2.4) hold.*

The following result, together with Propositions 2.2 and 2.5, will be basic for our bounds on the values involved in recursive decompositions.

Proposition 2.7 *Assuming (2.1)–(2.2), every entry of the matrix S det B is a subdeterminant (that is, the determinant of a submatrix) of the matrix W.*

Proof. Let $B = W^{(q)}$. Consider the l.p. (that is, northwestern) entry $s_{0,0}$ of S. By Proposition 2.5, it is the Schur complement of B in the submatrix $W^{(q+1)}$ of W. By Proposition 2.3, $s_{0,0}^{-1} = \det B / \det W^{(q+1)}$. Therefore, $s_{0,0} \det B = \det W^{(q+1)}$, which proves the proposition for $s_{0,0}$. To extend this result to any entry $s_{i,j}$ of S, interchange the i-th and 0-th rows and the j-th and 0-th columns of S and the respective pairs of rows and columns of W. □

Clearly, the matrix adj $B = B^{-1}$ det B is filled with integers if W is. Proposition 2.7 (or alternatively equation (2.2)) implies the similar property of the matrix S det B. We summarize these observations for future references.

Proposition 2.8 (integrality of adjonts and scaled Schur complements). *If a matrix W of (2.1) is filled with integers, then so are the matrices* adj $B = B^{-1}\det B$ *and* S $\det B$.

Recursive application of the next result enables us to recover $\det W$ from recursive decomposition of W.

Proposition 2.9 (factorization of the determinants and submatrices implied by matrix decomposition). *Matrix equation (2.3) implies that* $\det W = (\det B) \det S$ *and, furthermore, that*

$$W^{(t)} = \begin{pmatrix} I & 0 \\ EB^{-1} & I \end{pmatrix}^{(t)} \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix}^{(t)} \begin{pmatrix} I & B^{-1}C \\ 0 & I \end{pmatrix}^{(t)}, \quad \textit{for } t = 1, \ldots, k \ ,$$

and $\det W^{(t)} = (\det B) \det S^{(t-q)}$, *for* $t = q+1, \ldots, k$, *provided that* W *is a* $k \times k$ *matrix.*

In section 12, we will also use the following definitions and known results:

Definition 2.7 (the null-space of a matrix) (cf. [GL89/96] or [BP94]). *The null-space* $\mathbf{N}(A)$ *of a matrix* A *is the linear space formed by all vectors* $\mathbf{x}$ *satisfying the vector equation* $A\mathbf{x} = \mathbf{0}$.

Fact 2.1 *Two vector equations,* $A\mathbf{x} = \mathbf{f}$ *and* $A\mathbf{y} = \mathbf{f}$, *together imply that* $\mathbf{x} - \mathbf{y} \in \mathbf{N}(A)$ *or, equivalently, any solution* $\mathbf{x}$ *to a consistent linear system* $A\mathbf{x} = \mathbf{f}$ *can be represented in the form* $\mathbf{x} = \mathbf{x}_0 + \mathbf{z}$, *where* $\mathbf{x}_0$ *is a fixed specific solution and* $\mathbf{z} \in \mathbf{N}(A)$.

Toeplitz matrix computations will be studied in sections 13 and 14, but also the proposition below involves Toeplitz matrices and is needed in section 12.

Definition 2.8 (Toeplitz matrices). $T = (t_{i,j})$ *is a* $k \times k$ *Toeplitz matrix if* $t_{i+1,j+1} = t_{i,j}$ *for* $i, j = 0, 1, \ldots, k-2$ *(cf. (1.1)), that is, if the entries of* T *are invariant in their shifts in the diagonal direction. (Such a matrix is defined by its two columns (or rows) – the first one and the last one.) A square lower triangular Toeplitz matrix is defined by its first column* $\mathbf{u}$ *and is denoted* $L(\mathbf{u})$. $Z = Z_k = (z_{i,j})$ *is a* $k \times k$ *lower triangular Toeplitz matrix with the first column* $(0, 1, 0, \ldots, 0)^T$, *so that* $z_{i+1,i} = 1$ *for* $i = 0, 1, \ldots, k-2$, $z_{i,j} = 0$ *if* $i \neq j+1$.

Proposition 2.10 *[KS91] (cf. Lemmas 1.5.1 and 2.13.1 of [BP94]). Let* S *be a fixed finite set of cardinality* $|S|$. *Let* A, L, *and* U *be* $n \times n$ *matrices, rank* $A = r$, L *and* U^T *being unit lower triangular Toeplitz matrices, each defined by the* $n-1$ *entries of its first columns. Let these entries be chosen from* S *at random, independently of each other, under the uniform probability distribution on* S. *Then the matrix* $(UAL)^{(r)}$ *is strongly nonsingular with a probability at least* $1 - (r+1)r/|S|$.

3 Recursive Decomposition (RD) and Extended RD (ERD) of a C.-D.D. Matrix

With minor deviation from the order of our outine of section 1.6 but in accordance with section 1.7, we will next study the RD, then, in section 4, the integral RD (IRD) and in section 5, matrix inversion.

Hereafter, for convenience, let log stand for $\log_2$, let $n = 2^h$ for an integer $h = \log n$, and let V be a fixed $n \times n$ c.-d.d. matrix.

We will define a *recursive decomposition (RD)* of such a matrix $W = V$ based on its representation in the form (2.3) for $q = n/2$. We will first apply (2.3) to $W = V$ and then, recursively, to $W = B$ and $W = S$, and so on, though in fact, we will mostly care about the diagonal blocks V, $V_0 = B$, $V_1 = S$, and so on, which we will identify with the nodes of a binary tree, T. Similarly, we define the *dual RD* of $\tilde{W} = W^{-1}$ based on the recursive application of (2.4) to $\tilde{W} = B^{-1}$ and $\tilde{W} = S^{-1}$. (We will study such a dual RD in section 11 and will use it in section 12.)

The node of T associated with a binary strings α of length $|\alpha|$ is a $k \times k$ matrix, denoted by V_α, where $k = n/2^{|\alpha|}$. The root of the tree is the $n \times n$ matrix $V = V_\Lambda$, associated with the empty string Λ. For a string β of length less than h, we let $\beta 0$ and $\beta 1$ denote the two strings obtained by appending 0 and 1 to β, respectively. (We use the two characters α and β to distingush between the two classes of binary strings – of length at most h and less than h, respectively.) We will assume that the matrix equations (2.1)–(2.4) are satisfied for $W = V_\beta$, $B = V_{\beta 0}$, $S = V_{\beta 1}$, and for any binary string β of length $|\beta| < h$. The resulting RD of the matrix V continues up to the level h, where it reaches its leaves-matrices $V_\alpha = (v_\alpha)$ of size 1×1, where $|\alpha| = h$, and where v_α denotes the single entry of V_α.

Proposition 3.1 *All the nodes V_α of the tree T are c.-d.d. matrices; moreover, $d(V_\alpha) \le d(V) < 1$ for all binary strings α.*

Proof. Recall that V is a c.-d.d. matrix and recursively apply Proposition 2.6. □

Let us formalize the computation of the RD of V, by extending the notation (2.1)–(2.2) to V_α. We will write

$$V_\beta = \begin{pmatrix} B_\beta & C_\beta \\ E_\beta & G_\beta \end{pmatrix} , \tag{3.1}$$

$$V_{\beta 0} = B_\beta , \quad V_{\beta 1} = S_\beta = G_\beta - E_\beta B_\beta^{-1} C_\beta , \tag{3.2}$$

for all binary strings β of length less than h.

Then, computation of the RD of a c.-d.d. matrix V amounts to recursive computation of $V_{\beta 1}$ of (3.2) for all binary strings β of length increasing from 0 to $h-1$. As a by-product, the computation produces the matrices $V_{\beta 0}^{-1} = B_\beta^{-1}$ for all binary strings

β of length less than h. By appending these inverse matrices $V_{\beta 0}^{-1}$ to the nodes $V_{\beta 0}$ of the tree T (and, consequently, to the RD of V), we arrive at the *extended RD* of V, which we will abbreviate as the ERD of V. The set of the matrices $V_{\beta 0}^{-1}$ will be called the *extending set of the RD of* V.

The RD and the ERD of any matrix V can be defined as long as all the involved nodes-matrices $V_{\beta 0}$ for all the binary strings β of length less than h are nonsingular. Recursive application of Proposition 2.4 and Corollary 2.1 yields

Proposition 3.2 *There exists the RD and the ERD of any strongly nonsingular (in particular, of any s.p.d.) matrix.*

Remark 3.1 *As soon as we have the RD of a c.-d.d. matrix V, we immediately obtain the RD of V^{-1} based on recursive application of (2.4). Having such an RD available, we may compute the solution $\mathbf{x} = V^{-1}\mathbf{f}$ of a linear system $V\mathbf{x} = \mathbf{f}$, at a lower computational cost $O_A((\log n)^2, n^2/(\log n)^2)$.*

4 Integral Recursive Decomposition (IRD) of a C.-D.D. Matrix Filled with Integers

Suppose that the input matrix V is filled with integers. Then, for all binary strings α, the matrices V_α are filled with rationals, and there exist integer multipliers m_α such that $m_\alpha V_\alpha$ are integer matrices. We will next specify a particular choice of such integer multipliers m_α.

We will use the notation $W^{(q)}$ of Definition 2.4 and the following definition:

Definition 4.1 $(\alpha)_2$ *denotes the binary value represented by a binary string α (of length at most h). $\alpha(q)$ denotes the binary string that represents a nonnegative binary value q, so that $(\alpha(q))_2 = q$. $H(\alpha)$ denotes $2^{h-|\alpha|} = n/2^{|\alpha|}$. $Q(\alpha)$ denotes $(\alpha)_2 H(\alpha)$.*

By applying Proposition 2.5, we obtain

Proposition 4.1 *Let a binary string α end with bit 1 and have length at most h. Then the matrix V_α is the Schur complement of $V^{(Q(\alpha))}$ in $V^{(Q(\alpha)+H(\alpha))}$.*

By combining this proposition with Proposition 2.7, we obtain

Proposition 4.2 *For any binary string $\alpha = \beta 1 \gamma$ of length at most h with γ being a string of zeros, let*

$$m_\alpha = \det V^{(Q(\beta 1))}. \tag{4.1}$$

Then the entries of the matrix $V_\alpha m_\alpha$ are the subdeterminants (that is, the determinants of some submatrices) of the matrix V. In particular, if V is filled with integers, then so are the matrices $V_\alpha m_\alpha$ for all $\alpha, |\alpha| \le h$.

Now we replace the matrix $V_{\alpha\gamma}$ by the pair of the scalar $m_\alpha = m_{\alpha\gamma}$ and the matrix $m_\alpha V_{\alpha\gamma}$, in the binary tree T, for every pair of binary strings α and γ such that α ends with 1, $|\alpha| + |\gamma| \leq h$, and γ consists only of zeros. This gives us the *integral recursive decomposition* (IRD) of a c.-d.d. integer matrix V. By definition, we will also include into the IRD of V the two sets, $\{\det V^{(k)},\ k = 1, \ldots, n\}$ and $\{\det V_\alpha, |\alpha| \leq h\}$, of the determinants associated to the RD of V. Clearly, having the IRD of V available, we may immediately compute the RD of V. Later in this section, we will specify a simple transition from the RD to the IRD. In section 12, we will also compute a dual IRD, by similarly extending the dual RD.

By combining Propositions 2.2 and 4.2, we obtain that for all binary strings $\alpha,\ |\alpha| \leq h$, we have

$$|m_\alpha| \leq \| V \|^n, \ ||m_\alpha V_\alpha|| \leq n||V||^n. \tag{4.2}$$

For a c.-d.d. integer matrix V given together with its RD, we will seek its IRD. We recall that v_α for $|\alpha| = h$ denotes the single entry of the 1×1 leaf-matrix V_α of the tree T of the RD. Recursive application of Propositions 2.5 and 2.9 immediately yields the two following results:

Proposition 4.3 *For every binary string α of length at most h, we have*

$$\det\ V_\alpha = \prod_\beta v_\beta\ ,$$

where $\prod_\beta$ denotes the product in all binary strings β of length h that have α as their prefix, that is, the associated nodes V_β are both leaves of the tree T and descendants of the node V_α in the tree T.

Proposition 4.4 $\det\ V^{(q)} = \prod_{(\alpha)_2 < q} v_\alpha$, *where $\prod_{(\alpha)_2<q}$ denotes the product in all binary strings α of length h for which $(\alpha)_2 < q$.*

By applying the well-known parallel prefix algorithm [EG88], [KR90], we deduce the following result from Propositions 4.1–4.4 and 2.8:

Corollary 4.1 *Given the RD of a c.-d.d. $n \times n$ matrix V filled with integers, one may compute the IRD of V at the cost $O_A(\log n, n/\log n)$.*

Due to the latter result and to matrix equations (2.1)–(2.3), the computation of the IRD of V can be reduced to a sequence of multiplications, inversions, and subtractions of integer matrices, and we obtain the following parallel arithmetic complexity estimates:

$$t_{\mathrm{IRD}}(n) \leq t_I(n/2) + t_{\mathrm{IRD}}(n/2) + 2t_M(n/2) + 1, \tag{4.3}$$

$$p_{\mathrm{IRD}}(n) \leq \max\{p_I(n/2), 2p_{\mathrm{IRD}}(n/2), 2p_M(n/2), n^2\}, \tag{4.4}$$

where $O_A(t_{IRD}(k), p_{IRD}(k)), O_A(t_I(k), p_I(k))$ and $O_A(t_M(k), p_M(k))$ denote the time and processor bounds for the computation of the IRD, the inverse and the product of $k \times k$ matrices, respectively, where

$$(O_A(t_M(k), p_M(k)) = O_A(\log k, k^\omega), \quad 2 \leq \omega < 2.376 \ , \tag{4.5}$$

[BP94], and where we also use the obvious complexity bounds $O_A(t_S(k), p_S(k)) = O_A(1, k^2)$ for the subtraction of $k \times k$ matrices.

As the computation of the IRD is our major goal, relations (4.3)-(4.5) motivate the next subject of our study, that is, matrix inversion.

5 Approximate Matrix Inversion via Newton's Iteration

Algorithm 5.1, *Newton's iteration for approximate matrix inversion.*

Input: a nonsingular $k \times k$ matrix B, two positive scalars b and c, $b > c$, and a matrix X_0 (a rough initial approximation to $-B^{-1}$) such that

$$c = -\log \parallel BX_0 + I \parallel \ . \tag{5.1}$$

Output: a matrix X such that

$$\parallel BX - I \parallel \leq 2^{-b} \tag{5.2}$$

and, consequently,

$$\parallel X - B^{-1} \parallel \leq 2^{-b} \parallel B^{-1} \parallel \ . \tag{5.3}$$

Computations:

1. Compute
$$g = \lceil \log(b/c) \rceil \ . \tag{5.4}$$

2. Recursively compute the matrices
$$X_i = X_{i-1}(2I + BX_{i-1}) \ , \quad i = 1, \ldots, g \ . \tag{5.5}$$

3. Output the matrix $X = -X_g$.

To prove *correctness* of Algorithm 5.1, deduce from (5.5) that

$$I + BX_i = (I + BX_{i-1})^2 \ , \quad i = 1, 2, \ldots, g \ ,$$

and consequently,

$$I + BX_i = (I + BX_0)^{2^i} \ ,$$
$$\| I + BX_i \| \leq \| I + BX_0 \|^{2^i} \ , \tag{5.6}$$

for $i = 1, 2, \ldots, g$. In particular, for $i = g$, we have

$$\| I + BX_g \| \leq \| I + BX_0 \|^{2^g} = 2^{-c2^g} \leq 2^{-b} \ ,$$

due to (5.1) and (5.4). This gives us (5.2) and (5.3), for $X = -X_g$.

To estimate the overall *computational cost* of performing Algorithm 5.1, observe that the i-th step (5.5) amounts to two matrix multiplications and to adding, at the cost $O_A(1, k)$, the matrix $2I$ to the matrix BX_{i-1}. Summarizing, we obtain the following result:

Proposition 5.1 *For g of (5.4), g steps (5.5) of Newton's iteration, performed at the overall cost $O_A(gt_M(k), p_M(k)) = O_A(g \log k, k^\omega)$, for ω of (4.5), $2 \leq \omega < 2.376$, suffice in order to compute a matrix $X = -X_g$ satisfying (5.2) and (5.3).*

Remark 5.1 *Proposition 5.1 enables us to estimate the complexity of approximate matrix inversion in terms of a scalar b and a matrix X_0. Their choice depends on the input matrix B and is critical for estimating the approximation errors and the number of iterations. We will elaborate this choice in the next sections. Here are some preliminary comments on the choice of X_0. If B is a c.-d.d. matrix, so that $d(B) < 1$, then (5.1) holds for $X_0 = -D^{-1}(B)$ and $c = -\log d(B)$, that is, $||BX_0 + I|| \leq d(B)$. this gives us a good policy for the choice of X_0 and c over the class of c.-d.d. matrices B. In this paper, however, we will only need to invert the c.-d.d. matrices B that are close to the scaled identity matrices $-mI$ for a fixed large integer m. The inverse of such a matrix B is well approximated by the scaled identity matrices $X_0 = -I/m$ satisfying*

$$||BX_0 + I|| < 1/(5n^2), \ \ c > 2 \log n. \tag{5.7}$$

In fact our algorithms and complexity estimates will remain valid under some assumptions that are weaker than (5.7). Say, the bound

$$c = -\log ||BX_0 + I|| > \theta > 0 \tag{5.8}$$

for a fixed constant θ would suffice. The choice of $X_0 = D^{-1}(B)$ also satisfies the error bound of (5.7), but $X_0 = -I/m$ is a Toeplitz matrix (see Definition 2.8), which will be a crucially important advantage for the proof of Theorem 1.2 in sections 13 and 14.

6 Approximate RD of an Integer Matrix and Its Extension to the Exact Evaluation of the IRD

Algorithm 5.1 is intended as matrix inversion block in the algorithms of sections 3–4 for computing the ERD, IRD, and the associated determinants of a c.-d.d. integer matrix V. Then, the matrices V_α and the values of det $V^{(q)}$ and det V_α for all q and α are computed approximately, even where we still perform all arithmetic operations over the rationals, with infinite precision and no errors. Our next goal is to yield the exact IRD, assuming that we fixed a sufficiently large b and defined X_0 and c according to Remark 5.1.

Let us write $\tilde{V}_\alpha$, $\tilde{\det}\ V^{(q)}$, and $\tilde{\det}\ V_\alpha$ for the computed approximations to V_α, det $V^{(q)}$, and det V_α, respectively. Then we extend the relations (3.1), (3.2), (5.1), (5.4), and (5.5) by writing

$$\tilde{V}_\beta = \begin{pmatrix} \tilde{B}_\beta & \tilde{C}_\beta \\ \tilde{E}_\beta & \tilde{G}_\beta \end{pmatrix} , \tag{6.1}$$

$$\tilde{V}_{\beta 0} = \tilde{B}_\beta \ , \qquad \tilde{V}_{\beta 1} = \tilde{G}_\beta - \tilde{E}_\beta X_\beta \tilde{C}_\beta \ , \tag{6.2}$$

for all binary strings β of length less than h, where the policy of defining $X_{\beta,0}$ (in accordance with Remark 5.1) will be specified later on, and where

$$c(\beta) = -\log \parallel \tilde{B}_\beta X_{\beta,0} + I \parallel \ , \tag{6.3}$$

$$g(\beta) = \log(b/c(\beta)) \ , \tag{6.4}$$

$$X_{\beta,i} = X_{\beta,i-1}(2I + \tilde{B}_\beta X_{\beta,i-1}) \ , \qquad i = 1, \ldots, g(\beta) \ , \tag{6.5}$$

$$X_\beta = -X_{\beta,g(\beta)}, \tag{6.6}$$

and I is the identity matrix of an appropriate size.

Algorithm 6.1, *approximating the RD of a c.-d.d. matrix.*

Input: a positive integer h, $n = 2^h$, a positive b, positive integers $g(\alpha)$ for all binary strings α of length at most h, and an $n \times n$ c.-d.d. matrix $V = \tilde{V}_\Lambda$.

Output: a set of matrices $\tilde{V}_\alpha$ for all binary strings α of length at most h, satisfying (6.1)–(6.6), and the values $\tilde{\det}\ V_\alpha$ for all α and $\tilde{\det}\ V^{(q)}$, $q = 1, \ldots, n$, defined according to Proposition 4.4 with det and v replaced by $\tilde{\det}$ and $\tilde{v}$, respectively.

Computations: Starting with $V = \tilde{V}_\Lambda$ for the empty string Λ, recursively apply equations (6.1)–(6.6) and a fixed policy of defining $X_{\beta,0}$, in order to compute $\tilde{V}_{\beta 0}$ and $\tilde{V}_{\beta 1}$ for all binary strings β of length less than h. (For binary strings β of length $h-1$, the matrices $\tilde{V}_{\beta 0}$ and $\tilde{V}_{\beta 1}$ have size 1×1, so $\tilde{V}_{\beta 0}$ is inverted immediately, and then the matrix $\tilde{V}_{\beta 1}$ is computed based on (2.1) and (2.2) for $W = \tilde{V}_\beta$.) Finally, compute $\tilde{\det}\ V_\alpha$ and $\tilde{\det}\ V^{(q)}$ for all α and q by applying Propositions 4.3 and 4.4, under the above modification of the notation.

Correctness of the algorithm is immediately verified, provided that the value b is chosen sufficiently large so that all matrices $\tilde{B}_\beta$ – which approximate the c.-d.d.

matrices B_β – still have the property of being c.-d.d. and that the matrices $X_0 = X_{\beta,0}$ are chosen satisfying (5.8) for $B = \tilde{B}_\beta$ for all binary strings β. We note that (5.6) and (6.4) together imply that $||I + \tilde{B}_\alpha X_\alpha|| \leq 2^{-b}$ and, consequently,

$$||X_\alpha - \tilde{B}_\alpha^{-1}|| \leq 2^{-b}||\tilde{B}_\alpha^{-1}||, \tag{6.7}$$

which extends (5.3).

The computational cost is bounded by $O_A((\log n)^2 g, n^\omega)$, for $g = \max_{|\beta|<h} g(\beta)$ and for ω of (4.5). (Recursively apply (4.3)-(4.5) and Proposition 5.1.) A desired upper bound on g (and, consequently, on the parallel time) will be ensured by (5.8), (6.3), (6.4), and appropriate choice of b. Such a choice and its analysis will be shown in the next sections. $g(\beta)$ will in fact be independent of β, that is, we will choose $g(\beta) = g$ for all β.

Next, for a c.-d.d. matrix V filled with integers, we will apply Algorithm 6.1 for a sufficiently large b, and then, we will apply the techniques of *integer rounding* (compare [P85], [P87], and [BP94], page 252) to extend the resulting approximate RD of V to the evaluation of the IRD of V. To yield this extension, we will choose b in Algorithm 6.1 sufficiently large to ensure the following bounds:

$$|\tilde{\det}\, V^{(q)} - \det V^{(q)}| < 1/2\ , \quad q = 1, \ldots, n\ , \tag{6.8}$$

$$\| \tilde{V}_{\beta 1} - V_{\beta 1} \| < \| V \|^{-n} /2\ , \text{ for all binary strings } \beta, \ \ |\beta| < h\ , \tag{6.9}$$

where $\tilde{\det}\, V^{(q)}$ and $\tilde{V}_{\beta 1}$ denote the approximations to $\det V^{(q)}$ and $V_{\beta 1}$, respectively, computed by Algorithm 6.1, for the fixed value of b.

Under (6.8) and (6.9), we recover the IRD of V as follows:

Algorithm 6.2.

Input: a set $\{\tilde{\det}\, V^{(q)},\ q = 1, \ldots, n\}$ of approximations to $\det V^{(q)}$ for all q and an approximate RD of an $n \times n$ matrix V filled with integers, such that (6.8) and (6.9) hold.

Output: the IRD of V.

Computations:

1. Round the values $\det \tilde{V}^{(q)}$ to the closest integers; output the resulting integer values of $\det V^{(q)}$, $q = 1, \ldots, n$.

2. Compute the matrices $\tilde{W}_{\beta 1} = \tilde{V}_{\beta 1} \det V^{(Q(\beta 1))}$ and round their entries to the closest integers; output the resulting integer matrices $W_{\beta 1} = V_{\beta 1} \det V^{(Q(\beta 1)}$, for all binary strings β of length less than h.

3. For all binary strings β of length less than h and all binary strings γ filled with zero bits and satisfying $|\beta 1 \gamma| \leq h$, output the matrices $W_{\beta 1 \gamma} = V_{\beta 1 \gamma} \det V^{(Q(\beta 1))}$ (cf. (4.1) and Definition 4.1).

Correctness of Algorithm 6.2 follows since, clearly, the values $\det V^{(q)}$ are integers for all q and since the matrices $W_{\beta 1} = V_{\beta 1} \det V^{(Q(\beta 1))}$ are filled with integers for all β (due to Proposition 4.2). The *computational cost* of performing the algorithm is bounded by $O_A(1, n^2)$.

Remark 6.1 *Instead of choosing the multipliers $m_{\beta 1}$ based on Proposition 4.2, one may follow the more straightforward recipe of [R95] and recursively define m_β by using induction on $|\beta|$ and by writing $m_{\beta 1} = m_\beta$ det $V_{\beta 0}$. Then, however, the order of* $\log|m_\beta|$ *grows from $|\beta|$ (compare our bounds (4.2)) to $|\beta|^2$, and the bit-precision and the bit-complexity of the computations grow by the extra factor n. (The statement of Proposition 5.1 of [R95] is false, its proof relies on an erroneous claim that if a matrix mA is filled with integers, then so is the matrix m* adj *A; this claim is false, say, for $m = 3$ and the matrix $A =$ diag* $(1/3, \ldots, 1/3)$.)

7 Errors of the Approximation of the RD and the Transition from the RD to the IRD for a C.-D.D. Matrix

We are going to implement the next step of the outline of section 1.6 by specifying a c.-d.d. matrix V, whose IRD will give us the IRD of A modulo a prime p. We recall that, according to Definition 2.1, we write I to denote the identity matrices of appropriate sizes. We will next specify (in terms of n and $||V||$) a choice of the input parameter b of Algorithm 6.1 that will enable us to satisfy the relations (6.8) and (6.9), where V is an $n \times n$ matrix of the following class:

$$V = F - mI \ , \tag{7.1}$$

F is an $n \times n$ matrix filled with non-negative integers that are less than a fixed prime $p \geq n$ (we will work with $F = A \bmod p$ for an input matrix A), and

$$m = 10p^2n^2 \ . \tag{7.2}$$

Remark 7.1 *The choice of a larger m would have made V more strongly diagonally dominant (which is what we would like to have) but would have involved larger integers, which would have increased the Boolean cost of the resulting computations, so we choose only a moderately large m. In fact, our construction allows us to choose even a little smaller m.*

Next, let us prove that the entries of the matrices V of this class and of all matrices V_α of their RDs satisfy the following rough estimate, which will suffice for our purpose:

Proposition 7.1 *The entries of the matrices $V_\alpha + mI$ lie in the range between $-1/2$ and $p - 1/2$, for all binary strings α of length at most $h = \log n$.*

Proof. By the definition of the matrix V, the entries of the matrix $F = V + mI$ range from 0 to $p-1$. By Proposition 4.1, in suffices to prove that the entries of every Schur complement S of an l.p.s. $B = V^{(q)}$ in V range from $-1/2$ to $p - 1/2$. Since $S = G - EB^{-1}C$ (assuming (2.1) for $W = V$) and since the entries of the submatrix $G + mI$ of F range from 0 to $p-1$, it suffices to prove that the entries of the matrix $EB^{-1}C$ range between $-1/2$ and $1/2$. Since the matrices $F^{(q)} = B + mI$, C and E are submatrices of F, their entries also range from 0 to $p-1$. Therefore,

$$||C|| < (p-1)n, ||E|| < (p-1)n,$$

$$-mB^{-1} = (I - F^{(q)}/m)^{-1} = \sum_{i=0}^{\infty}(F^{(q)}/m)^i,$$

$$||B^{-1}|| \leq (1/m)(1/(1-a)),\ \ a = ||F^{(q)}||/m < (p-1)n/m < 0.03.$$

Consequently, $||B^{-1}|| < 2/m$, $||EB^{-1}C|| \leq 2(p-1)^2n^2/m < 1/2$. □

Hereafter, we will write

$$w = m + (p-1)n. \tag{7.3}$$

Here are three corollaries of Proposition 7.1, the first and the third of them are immediate.

Corollary 7.1 *Let $|\alpha| = h$, so that $V_\alpha = (v_\alpha)$ is a 1×1 matrix, and let w be defined by (7.3). Then we have*

$$|v_\alpha| < m + p.$$

Corollary 7.2 $\| V_{\beta 0}^{-1} \| < 1.1/m$, *for all binary strings β of length at most $h-1$.*

Proof. Write $F_{\beta 0} = V_{\beta 0} + mI$. Due to Proposition 7.1, we have $\| F_{\beta 0} \| < (p-1)n < m/(10np)$. On the other hand,

$$V_{\beta 0}^{-1} = \frac{1}{m}(I - F_{\beta 0}/m)^{-1} = \left(\frac{1}{m}\right)\sum_{i=0}^{\infty}(\frac{F_{\beta 0}}{m})^i \ .$$

Therefore,

$$\| V_{\beta 0}^{-1} \| \leq \left(\frac{1}{m}\right)\sum_{i=0}^{\infty}(\frac{\| F_{\beta 0} \|}{m})^i < \left(\frac{1}{m}\right)\sum_{i=0}^{\infty}\frac{1}{(10np)^i} = \frac{10np}{(10np-1)m} < \frac{1.1}{m} \ . \quad \square$$

Corollary 7.3 $2^{c_\beta} = ||V_{\beta 0}X_{\beta,0} + I|| < 1/(10pn) \leq 1/(10n^2)$ *for $X_{\beta,0} = -I/m$ and for all binary string β of length at most $h-1$.*

Hereafter, we will assume that $n > 1$ and that the matrices $X_{\beta,0}$ for all β are chosen as in Corollary 7.3, so that the relations (5.8) and even (5.7) hold. Our next task is to estimate the desired range for b, which would enable us to recover the IRD. In this section we will prove the following basic proposition.

Proposition 7.2 *Under (7.1)–(7.3), both requirements (6.8) and (6.9) are satisfied if the matrices $V_{\beta 1}$ for all binary strings β of length less than h are approximated by Algorithm 6.1 within an error norm bound*

$$\sigma = 0.5/w^n \ . \tag{7.4}$$

Proof. We deduce from (7.1) and (7.3) that

$$\parallel V \parallel \leq m + (p-1)(n-1) < w.$$

Therefore, we will satisfy (6.9) if we approximate the matrices $V_{\beta 1}$ within the error norm bound (7.4). To prove that (6.8) is satisfied too, we need the next lemma.

Lemma 7.1 *The requirement (6.8) is satisfied if the values v_α, for all binary strings α of length h, are approximated within an error bound*

$$\delta \leq \overline{w}^{1-n}/(2n+2), \ \ \overline{w} = m + p < w. \tag{7.5}$$

Proof of the lemma. By the virtue of Proposition 4.4, det $V^{(q)}$ is the product of exactly q values v_α, for α denoting binary strings of length h. Under the assumptions of Lemma 7.1, the maximum error of computing det $V^{(q)}$ may only increase if we assume that $q = n$, $v_\alpha = \overline{w}$, and the approximations to v_α equal $\overline{w} + \delta$ for all α. Then, det $V = \overline{w}^n$ is approximated by $(\overline{w} + \delta)^n$, with an approximation error

$$E = (\overline{w} + \delta)^n - \overline{w}^n = \overline{w}^n((1 + (\delta/\overline{w}))^n - 1) = \overline{w}^{n-1}\delta \sum_{i=1}^{n} (\delta/\overline{w})^{i-1} \begin{pmatrix} n \\ i \end{pmatrix}.$$

We have $\begin{pmatrix} n \\ 1 \end{pmatrix} = n$, $\begin{pmatrix} n \\ i \end{pmatrix} < 2^n$ for all i, and $\delta/w < 1$. Therefore,
$E < (n + (\delta/\overline{w})(n-1)2^n)\delta\overline{w}^{n-1}$.

(7.2) and (7.5) together imply that

$$\overline{w} > (n-1)2^n\delta \ ,$$

and we may rewrite our bound on E as follows:

$$E < (n+1)\delta\overline{w}^{n-1} \ .$$

Substitute (7.5) and obtain that $E < 1/2$. □

To complete the proof of Proposition 7.2, we observe that

$$\sigma = 0.5/w^n < w^{1-n}/(2n+2) < \overline{w}^{1-n}/(2n+2)$$

(compare (7.5)), and the values v_α for all binary strings α of length h (except for the string $\alpha(0)$ consisting of h zeros) are among the entries of the matrices $V_{\beta 1}$ for $|\beta| \leq h - 1$. $v_{\alpha(0)}$ is an entry of V and is known exactly without any computation. Therefore, the assumptions of Lemma 7.1 and, consequently, the requirement (6.8) is satisfied too. □

8 Estimating the Error Accumulation and the Precision of the Aproximation of the Matrix Inverse

In this section, we will extend Proposition 7.2 by estimating the parameter b of Algorithm 6.1 (which expresses the precision of the matrix inversion) to ensure (7.4) and, consequently, (6.8) and (6.9).

Proposition 8.1 *Under some choice of $b = O(n \log\ p)$, the bounds (6.8) and (6.9) can be satisfied in all applications of Newton's iteration (6.5) within Algorithm 6.1, which is in turn applied to approximate the RD of a matrix V satisfying (7.1) and (7.2).*

The remainder of this section is devoted to the proof of Proposition 8.1. Due to the bound (5.6), it is actually quite clear that we would ensure (7.4) if we choose sufficiently large values $b = O(n \log p)$, $g(\alpha)$ of (6.4) and m of (7.1) but we will deduce (7.4) already for m of (7.2) and $g(\alpha) = O(\log(n \log p))$ for all α. As usually in the proofs involving error analysis, some tedious estimates are required. The idea of our proof is to condense estimating the error propagation into a single step, which will allow its recursive extension to cover all the nodes V_α of the tree representing the RD. This basic step of our analysis will be given in the form of Proposition 8.2. (We will first give some preliminaries, then will state and prove this proposition and then will show that its conclusion enable us to extend recursively its assumptions (and, consequently, its conclusion too) and thus to extend the error estimates recursively to all the descendants of the current node V_α of the tree.)

Proof of Proposition 8.1. Consider a path in the tree T from the root V to a leaf $V_\alpha = (v_\alpha)$, $|\alpha| = h$. Algorithm 6.1 follows such a path, by recursively proceeding from matrices V_β to $V_{\beta 0}$ and $V_{\beta 1}$. For given V_β and $V_{\beta 0}$, the algorithm approximates the matrices $V_{\beta 0}^{-1}$ within the error norm bounds $2^{-b} \parallel V_{\beta 0}^{-1} \parallel$ and then extends such an approximation to approximating $V_{\beta 1}$. The errors of the computed approximation to V_β are accumulated in computation of all descendands of V_β along the paths in T. We need to estimate the resulting overall errors in all the output matrices along all such paths, assuming that b is large though of order $O(n \log\ p)$.

We will next analyze a single recursive step along such a path, that is, we will first bound the matrix $\Delta(V_\beta)$ of the initial errors of the approximation of $W = V_\beta$, and then will estimate the propagated errors of the approximation of $S = V_{\beta 1}$ caused by the combined errors due to the initial ones, given by $\Delta(V_\beta)$, and ones of Newton's iterates for the inversion of $V_{\beta 0}$.

Hereafter, we will write $\tilde{M}$ to denote the approximations to matrices M computed by Algorithm 6.1, for M denoting V_α (for any binary string α), a submatrix of V_α, or any other auxiliary matrix involved. We will also write

$$\Delta(M) = \tilde{M} - M \ . \tag{8.1}$$

We will first estimate $\| \Delta(V_{\alpha 1}) \|$ in terms of $\| \Delta(V_\alpha) \|$. For convenience, we write $W = V_\alpha$, $S = V_{\alpha 1}$, recall (2.1), (2.2), and estimate the error propagation in the transition from W to B and S. From Proposition 7.1 and Corollary 7.2, we obtain that

$$\max\{\| B \|, \| C \|, \| E \|, \| G \|\} \leq \| W \| \leq w \ , \tag{8.2}$$

$$\| B^{-1} \| < 1.1/m \ . \tag{8.3}$$

We also write (cf. (8.1)):

$$\tilde{W} = \begin{pmatrix} \tilde{B} & \tilde{C} \\ \tilde{E} & \tilde{G} \end{pmatrix}, \quad \Delta(W) = \begin{pmatrix} \Delta(B) & \Delta(C) \\ \Delta(E) & \Delta(G) \end{pmatrix} ,$$

$$\tilde{S} = \tilde{G} - \tilde{E} L \tilde{C} \ ,$$

where L denotes the computed approximation to $\tilde{B}^{-1}$. Then, clearly,

$$\max\{\| \tilde{B} \|, \| \tilde{C} \|, \| \tilde{E} \|, \| \tilde{G} \|\} \leq \| \tilde{W} \|, \tag{8.4}$$

$$\max\{\| \Delta(B) \|, \| \Delta(C) \|, \| \Delta(E) \|, \| \Delta(G) \|\} \leq \| \Delta(W) \| \ . \tag{8.5}$$

We will assume that the errors of the approximations obtained via Algorithm 6.1 are sufficiently small so that the following inequalities hold (also cf. the bound $||B^{-1}|| < 0.03$ obtained in the proof of Proposition 7.1):

$$\| L - \tilde{B}^{-1} \| \leq \nu \| \tilde{B}^{-1} \|, \quad \nu \leq 1/(4000m^2), \tag{8.6}$$

$$\| \tilde{B}^{-1} \| < 1.11/m, \quad \| L \| \leq (1+\nu) \| \tilde{B}^{-1} \| < 1.11(1+\nu)/m. \tag{8.7}$$

Remark 8.1 *The inequalities of (8.6) are reconciled with (5.3) for $\nu \leq 2^{-b}$, L replacing X, and $\tilde{B}$ replacing B.*

In the next proposition we will bound approximation errors for $V_{\beta 0}$ and $V_{\beta 1}$ in terms of a single positive parameter $\Delta = \Delta(\beta)$ defined by the errors of the approximation of V_β and by the parameter ν, which is in turned defined by the error exponent b of Newton's iteration for the approximation of $\tilde{B}^{-1}$.

Proposition 8.2 *Suppose that the inequalities (8.2)–(8.7) hold and that a positive $\Delta = \Delta(\beta)$ satisfies the following bounds:*

$$||\Delta(W)|| \leq \Delta, \ 40\nu m \leq \Delta < 0.01 \ w, \tag{8.8}$$

where $W = V_\beta$ and β is a binary string of length less than h. Then, we have

a) $\| \Delta(B) \| = \| \Delta(V_{\beta 0}) \| \leq \Delta$,
b) $\| \Delta(S) \| = \| \Delta(V_{\beta 1}) \| \leq 5\Delta$.

Proof. Part a) of the proposition follows immediately since $V_{\beta 0}$ is a submatrix of V_β. To deduce part b), we will use the following bound (implied by (8.2) and (8.8)):

$$\| \tilde{W} \| < 1.01\ w \ , \tag{8.9}$$

as well as the next proposition:

Proposition 8.3 *For any 4-tuple of $k \times k$ matrices, X, $\tilde{X}$, Y and $\tilde{Y}$, we have*
a) $\| \Delta(X \pm Y) \| \leq \| \Delta(X) \| + \| \Delta(Y) \|$,
b) $\| \Delta(XY) \| \leq \| \Delta(X) \| \ \| Y \| + \| \Delta(Y) \| \ \| X \| + \| \Delta(X) \| \ \| \Delta(Y) \|$,
and if X and $\tilde{X}$ are nonsingular matrices, then also
c) $\| \Delta(X^{-1}) \| \leq \| X^{-1} \| \ \| \tilde{X}^{-1} \| \ \| \Delta(X) \|$.

Proof. The parts a)-c) follow immediatley from the next simple equations:
a) $\Delta(X \pm Y) = \Delta(X) \pm \Delta(Y)$,
b) $\Delta(XY) = \Delta(X)Y + X\Delta(Y) + \Delta(X)\Delta(Y)$,
c) $\Delta(X^{-1}) = X^{-1}\Delta(X)\tilde{X}^{-1} = -\tilde{X}^{-1}\Delta(X)X^{-1}$. □

Since $S = G - EB^{-1}C$ under (2.2), we will next recursively extend the bound $||\Delta W||$ on the error norms of G, E, B and C to yield some bounds on the error norms of B^{-1}, EB^{-1}, $EB^{-1}C$ and S.

We first apply part c) of the latter proposition for $X = B$ and $\Delta(X^{-1}) = \tilde{B}^{-1} - B^{-1}$ and obtain that

$$\| \Delta(B^{-1}) \| \leq \| B^{-1} \| \ \| \tilde{B}^{-1} \| \ \| \Delta(B) \| \ .$$

Substitute (8.3) and (8.7) into the latter bound and obtain that

$$\| \Delta(B^{-1}) \| < 1.221 \| \Delta(W) \| /m^2 \ .$$

Combine the relations (8.6) and (8.7) to obtain that $||L - \tilde{B}^{-1}|| \leq 1.11\nu/m \leq (1.11)\Delta/(4000m^3)$. Combine the latter bounds on the norms, recall (8.8) and deduce that

$$\begin{aligned} \| L - B^{-1} \| \leq \| \Delta(B^{-1}) \| + \| L - \tilde{B}^{-1} \| &< (1.221 + (1.11)/(4000m))\Delta/m^2 \\ &< 1.3\Delta/m^2 \ . \end{aligned} \tag{8.10}$$

Apply part b) of Proposition 8.3 for $X = E$, $Y = L$ and deduce that

$$\| \Delta(EL) \| \leq \| \Delta(E) \| \ \| L \| + \| L - B^{-1} \| \ (\| E \| + \| \Delta(E) \|) \ .$$

Recall from (7.2) and (7.3) that $w/m \leq 1.02$. By combining the two latter inequalities with our bounds on $\| L \|$, $\| L - B^{-1} \|$, $\| E \|$, and $\| \Delta(E) \|$ (see (8.4)–(8.10)), obtain that

$$\| \Delta(EL) \| \leq \left(\frac{1.11}{m}(1+\nu) + \left(\frac{1.3}{m^2}\right) 1.01w \right) \Delta \leq \frac{2.5}{m}\Delta \ . \tag{8.11}$$

Then again, we apply part (b) of Proposition 8.3, this time for $X = EL$, $Y = C$, and obtain that

$$\| \Delta(ELC) \| \leq \| \Delta(EL) \| \ \| \tilde{C} \| + \| \Delta(C) \| \ \| EL \| \ .$$

Substitute our previous estimates (8.2), (8.4)–(8.9), (8.11) into the latter inequality and deduce that

$$\| \Delta(ELC) \| \leq \left(\left(\frac{2.5}{m}\right) 1.01w + \frac{1.11}{m}(1+\nu)w \right) \Delta \leq 3.7\Delta w/m.$$

Now, since $w/m \leq 1.02$, we have

$$\| \Delta(ELC) \| \leq 4\Delta.$$

We obtain $\| \Delta(G) \| \leq \Delta$ from (8.5) and (8.8). By applying part a) of Proposition 8.3 for $X = G$, $Y = ELC$, we deduce that

$$\| \Delta(S) \| \leq \| \Delta(ELC) \| + \| \Delta(G) \| \leq 5\Delta,$$

which proves Proposition 8.2. □

Now, we observe that the assumptions of Proposition 8.2 are satisfied for $W = V$, $\tilde{W} = V$, $\Delta = 40\nu m$ and extend them to $W = V_\alpha$ for all α. The extension from $W = V_\beta$ to $W = V_{\beta 0}$ for any β is trivial. We will comment on the extension to $W = V_1$, which will be our sample for the extension from $W = V_\beta$ to $W = V_{\beta 1}$ for any β. We write $\tilde{B} = B = V_0$, $L = -X_{1,g}$, $X_{1,0} = -I/m$, $g \geq 4$ and define by (6.5) the matrices $X_{1,i}$ for all i. Now, observe that $||I + X_{1,0}W|| = ||F||/m$, $||I + X_{1,0}\tilde{W}|| \leq ||F||/m + \Delta/m \leq (\Delta + (p-1)n)/m < 1/(10pn) < 1/m^{1/2}$.

Therefore, (5.6) implies that

$$\| X_i + B^{-1} \| \leq \| B^{-1} \| /m^{2^{i-1}}, \quad i = 1, \ldots, g,$$

and consequently, since $L = -X_g$, for $g \geq 4$, we have

$$\| L - B^{-1} \| \leq \nu \| B^{-1} \|, \quad for \ \ \nu \leq 1/m^8 < 1/(4000m^2n^{12}),$$

thus satisfying (8.6). The remaining assumption (8.7) of Proposition 8.2 is also easily verified (by using (8.5) and (8.8) and by following the line of the proof of Corollary 7.2).

Now, we are ready to extend Proposition 8.2 recursively, which will give us the desired upper bound on $||\Delta(V_\alpha)||$ in terms of b. By applying this proposition recursively, we extend its assumptions to $W = V_0$, $W = V_1$, $\Delta = 40\nu m$. (In the subsequent recursive extension from $W = V_{\beta 1}$ to $W = V_\beta$ for any binary string β of length at most $h-1$, we will choose ν depending on α but satisfying (8.6) for all α.)

Apply the bounds of parts a) and b) of Proposition 8.2 recursively and obtain that

$$\| \Delta(W) \| \leq \Delta \sum_{i=0}^{|\alpha|} 5^i < n^3 \Delta,$$

for $W = V_\alpha$ and all α (with $|\alpha| \leq h = \log n$). Let us choose b that enables us to reconcile the initial choice of $\Delta = 40\nu m$ and the latter bound on $||\Delta(W)||$ with (8.6)–(8.8). Recall Remark 8.1, recall that the choice of g according to (6.4) implies the bound (6.7) on the output approximation to the inverse, substitute $\nu = 2^{-b}$, and obtain the desired estimate:

$$\| \Delta(V_\alpha) \| < (40mn^3\Delta)2^{-b} < 2^{2-b}w^2 n \; , \tag{8.12}$$

for all α.

Let us choose b of order $n \log p$ satisfying the bound

$$b \geq 3 + \log n + (n+2)\log w,$$

which is compatible with the choice of $b = \log(1/\nu)$ and with (8.6). Substitute this bound on b into the preceding upper bound on $\| \Delta(V_\alpha) \|$ and obtain that

$$\| \Delta(V_\alpha) \| < 0.5 \; w^{-n} \; ,$$

for all α. This satisfies the requirement (7.4) of Proposition 7.2 and completes the proof of Proposition 8.1. □

Remark 8.2 *By Corollary 7.3, $c(\beta) = O(\log n)$ for all binary strings β of length at most $h-1$. Furthermore, (6.4) and the above choice of b are compatible with the choice of $g = g(\beta)$ of order $\log b = \log(n \log p)$ for all β.*

9 Computations with Rounding-off: Estimates for the Finite Precision and Computational Cost

So far, we assumed the infinite precision of computing the RD and IRD by means of Algorithms 6.1 and 6.2. Next, we will show that this is not necessary for obtaining the result of Proposition 8.1, that is, we will prove

Proposition 9.1 *The estimates of Proposition 8.1 hold even if the computations by Algorithms 6.1 and 6.2 are performed with a precision of $\tilde{b}$ bits, for some $\tilde{b} = O(n \log\ p)$ provided that a single extra Newton's step (6.5) is performed in each application of Algorithm 6.1.*

Proof. Let us first assume the computations of Newton's step (6.5) with the infinite precision, but in the transition from $W = V_\alpha$ to $S = V_{\alpha 1}$ for all binary strings α, $|\alpha| < h$, let the computations be performed with rounding to the $\tilde{b}$-bit precision. Assuming B^{-1} available, the latter transition involves two multiplications and a subtraction of $k \times k$ matrices for $k = 2^{h-|\alpha|}$ (compare (2.2)). By applying the techniques of backward error analysis [W65], [BL80], we bound the norm of the matrix $\epsilon(S)$ of the errors of the approximation to S caused by rounding:

$$\| \epsilon(S) \| \leq n^{O(1)} 2^{-\tilde{b}} \| W \| (1 + \| L \| \| W \|) ,$$

for L denoting the computed approximation to B^{-1} (cf. (8.6), (8.7)). By applying the relations (8.2), (8.7), (7.2), (7.3), and Proposition 7.1, we obtain that

$$\| \epsilon(S) \| \leq m^{O(1)} 2^{-\tilde{b}} .$$

By choosing $\tilde{b}$ of order $n \log p$, we make $\| \epsilon(S) \|$ less than $2 \| \Delta(W) \|$ for $W = V_\alpha$ and for all α. This is less than 40 % of the upper bound that we have in part b) of Proposition 8.2. Combining both of these bounds gives us cumulative upper bound $7 \| \Delta(W) \|$, which shows the overall impact of the above rounding errors. This enables us to preserve the validity of the bound (8.12) (since $\sum_{i=0}^{h} 7^i \leq n^3$ for $h = \log n$) and, consequently, of the entire proof of Proposition 8.1.

It remains to estimate the impact of rounding to $\tilde{b}$-bit precision when we perform Newton's steps (6.5). Then again, we deal with two matrix multiplications (we ignore the errors caused by the simple addition step $2I + (WX_{i-1})$). By applying backward error analysis again, we estimate that

$$\| \epsilon(X_i) \| \leq n^{O(1)} 2^{-\tilde{b}} \| X_{i-1} \| (2 + \| W \| \| X_{i-1} \|) , \tag{9.1}$$

where $\epsilon(X_i)$ denotes the matrix of the errors of approximation of X_i due to rounding in performing iteration (6.5).

Our next goal is to prove the following bound,

$$\| X_{i-1} \| \leq 1.1(1 + 1/\tilde{m})/m < 1.21/m, \text{ for } i \geq 1, \tag{9.2}$$

ignoring for simplicity the terms of order $2^{-2\tilde{b}}$ or less.

We have from Corollary 7.3 that $\|I + BX_0\| < \frac{1}{\tilde{m}}$, for $\tilde{m} = 10pn \geq 20p$ and for $X_0 = -I/m$. Then we obtain from (5.6) that

$$\| I + BX_{i-1} \| \leq 1/\tilde{m}^{2^{i-1}}.$$

Therefore,

$$\| X_{i-1} + B^{-1} \| \leq \| B^{-1} \| / \tilde{m}^{2^{i-1}} .$$

Consequently,

$$\| X_{i-1} \| \leq \| B^{-1} \| \left(1 + \frac{1}{\tilde{m}^{2^{i-1}}}\right) ,$$

for $i = 1, 2, \ldots$. Substitute (8.3) and arrive at inequality (9.2).

Substitute bound (9.2) on $\| X_{i-1} \|$ and the bound $\| W \| \leq w$ of (8.2) into (9.1), recall (7.2) and (7.3), and obtain that

$$\| \epsilon(X_i) \| \leq (np)^{O(1)} 2^{-\tilde{b}} \quad \text{for all } i \ .$$

By choosing a sufficiently large $\tilde{b}$, though of order $n \log\ p$, we easily ensure that

$$\| \epsilon(X_i) \| < 2^{-b} / \| B \| \ .$$

Therefore,

$$\| I + B(X_i + \epsilon(X_i)) \| \leq \| I + BX_i \| + \| B \| \ \| \epsilon(X_i) \| \leq$$

$$\| I + BX_{i-1} \|^2 + 2^{-b} \ .$$

Since Newton's iteration (6.5) stops if $\| I + BX_{i-1} \| \leq 2^{-b}$, we may assume that $\| I + BX_{i-1} \| > 2^{-b}$, so that the rounding may at worst change (6.7) into the bound

$$\| I + B(X_i + \epsilon(X_i)) \| \leq 2 \ \| I + BX_i \| \leq 2^{\sigma} \ \| I + BX_0 \|^{2^i} < 2(2 \ \| I + BX_0 \|)^{2^i},$$

where $\sigma = \sum_{s=0}^{i} 2^s < 2^{i+1}$. Since $\| I + BX_0 \| \leq 1/m$, the impact of the rounding on the residual norm of the output approximation computed by Newton's iteration is more than compensated by a single extra step (5.5), (6.5) and this completes the proof of Proposition 9.1. □

Let us next summarize our current complexity extimates for the computation of the IRD, before we improve them slightly in the next section. Choose b and $\tilde{b}$ of order $n \log p$ and choose g of order $\log(n \log p)$, which is consistent with (6.4) under (5.7) or (5.8). Now, by combining the results of Corollaries 4.1 and 7.3 and Propositions 8.1 and 9.1 with Remark 8.2 and the estimates for the arithmetic parallel complexity of performing Algorithms 6.1 and 6.2 and by using the B-principle, obtain

Corollary 9.1 *Algorithm 6.2 computes the IRD of a matrix V satisfying (7.1) and (7.2) at the cost $O_A((\log n)^2 \log(n \log p), n^{\omega} / \log n)$ for ω of (4.5). Furthermore, $\tilde{b}$-bit precision suffices in these computations for some $\tilde{b}$ of order $n \log\ p$.*

10 Pipelined Computation of the IRD

Our next goal is a modification of Algorithm 7.1, which, as we claimed in section 1.6, will enable us to improve by factor $\log n$ the asymptotic time-complexity bounds of Corollary 9.1, without increasing the processor bound by more than a constant factor. To achieve this goal, we will incorporate into our construction the techniques of *pipelining* along the lines of [PR91] (where such techniques were called *stream contraction* and applied to computing the RD of a matrix over semirings of a certain class). Here are our informal underlying observations.

Alogorithm 6.1 is not fully efficient because it spends substantial time and work on refining the approximations to the inverses of the matrices $\tilde{B}_\beta = \tilde{V}_{\beta 0}$ (cf. (6.5)); this delays the subsequent use of such approximations in the inversion of the matrices $\tilde{V}_{\beta 10} = \tilde{B}_{\beta 1}$, which anyway starts with a much cruder approximation $-I/m$. Next, we will modify Algorithm 6.1. We will start the Newton process of the inversion of $\tilde{B}_{\beta 1}$ by relying on the available rough approximations to $\tilde{B}_\beta^{-1}$, and then we will recursively pump better approximations when the process progresses.

In other words, we are going to pipeline the recursion on α (decomposition) and the Newton one (inversion). To approximate the matrix $V_{\beta 1}$ of the RD and ERD, we will start using the intermediate approximations to the inverse $V_{\beta 0}^{-1}$, as soon as they are computed by Newton's iteration. We will update the resulting approximation to $V_{\beta 1}$ as soon as the approximation to $V_{\beta 0}^{-1}$ is refined, that is, in the process of the computation of the matrix $\tilde{V}_{\beta 1}$, we will keep refining every step of the computations as soon as we refine its input.

More precisely, we will initalize this process by fixing some natural g, to be specified later on. As before, α and β will denote binary strings, $|\alpha| \leq h$, $|\beta| < h$, γ will denote the unary strings consisting of zero bits. $u(\alpha)$ will denote the number of ones in a binary string α. t will denote integers in the range from t_0 to $g+h$. Here, $t_0 = u(\alpha)$ in (10.1) and (10.3) (where α is fixed), $t_0 = u(\beta)+1$ in (10.4)-(10.7) (where β is fixed), $t_0 = 0$ elsewhere, that is, in (10.2).

We will now define the following matrices whose subscripts α, β, γ and t range as specifed above:

$$V_{\alpha,t} = \begin{pmatrix} B_{\alpha,t} & C_{\alpha,t} \\ E_{\alpha,t} & G_{\alpha,t} \end{pmatrix}, \tag{10.1}$$

$$V_{\gamma,t} = V_\gamma \tag{10.2}$$

(cf. Definition 2.4),

$$V_{\alpha\gamma,t} = (V_{\alpha,t})^{(q)}, \text{ for } |\alpha\gamma| \leq h, \ q = 2^{h-|\alpha\gamma|}, \tag{10.3}$$

$$X_{\beta,t,0} = \{ \begin{array}{l} -I/m \text{ for } t = u(\beta)+1 \\ -X_{\beta,t-1} \text{ for } t > u(\beta)+1, \end{array} \tag{10.4}$$

$$X_{\beta,t,i+1} = X_{\beta,t,i}(2I + V_{\beta 0,t}X_{\beta,t,i}), \ i = 0,1,2,3,4, \tag{10.5}$$

$$X_{\beta,t} = -X_{\beta,t,4}, \tag{10.6}$$

$$V_{\beta 1,t+1} = G_{\beta,t} - E_{\beta,t}X_{\beta,t}C_{\beta,t}. \tag{10.7}$$

Now, are ready to specify our pipelined algorithm.

Algorithm 10.1, *stream contraction for approximating the RD.*

Input: natural g, h, and $n = 2^h$; an $n \times n$ matrix V.

Output: for all binary strings α of length at most h, matrices $V_{\alpha,g+u(\alpha)}$ satisfying the equations (10.1)–(10.7) and approximating the matrices V_α, respectively.

Computations:

Stage 0. Apply (10.2) for $t=0$ to define the matrices $V_{\gamma,0}$, $|\gamma|=0,\dots,h$.

Stage $\mathbf{t}$, $\mathbf{t=1,\dots,g+h}$.

Concurrently in all binary strings β of length less than h with $u(\beta)<t$, compute successively:

a) $X_{\beta,t,0}$, based on (10.4),

b) $X_{\beta,t,i+1}$ for $i=0,1,2,3,4$ based on (10.5),

c) $X_{\beta,t}$, by (10.6),

d) $V_{\beta 1,t+1}=G_{\beta,t}-E_{\beta,t}X_{\beta,t}C_{\beta,t}$, based on (10.1) and (10.7),

e) $V_{\beta 1\gamma,t+1}$, by (10.3) where $\alpha=\beta 1$.

These rules are complemented by the following

Stopping criterion: output the matrices $V_{\alpha,g+u(\alpha)}$ for all binary strings α of length at most h and cancel all the subsequent computations involving these matrices.

For reader's convenience, we will next list the matrices computed at stages 1, 2 and 3, letting $\gamma_0,\gamma_1,\gamma_2,\gamma_3$ denote unary strings filled with zeros.

Stage 1:

$X_{\gamma_0,1,0}=-I/m$,

$X_{\gamma_0,1,i+1}=X_{\gamma_0,1,i}(2I+V_{\gamma_0 0}X_{\gamma_0,1,i})$, $i=0,1,2,3,4$,

$X_{\gamma_0,1}=-X_{\gamma_0,1,4}$,

$V_{\gamma_0 1,2}=G_{\gamma_0}-E_{\gamma_0}X_{\gamma_0,1,}C_{\gamma_0}$, for $G_{\gamma_0},E_{\gamma_0},C_{\gamma_0}$ of (3.1), $|\gamma_0|<h$,

$V_{\gamma_0 1\gamma_1,2}=V^{(q)}_{\gamma_0 1,2}$, for $q=2^{h-1-|\gamma_0\gamma_1|}$, $|\gamma_0\gamma_1|<h$.

Stage 2: $X_{\gamma_0,2,0}=-X_{\gamma_0,1}$,

$X_{\gamma_0,2,i+1}=X_{\gamma_0,2,i}(2I+V_{\gamma_0 0,2}X_{\gamma_0,2,i})$, $i=0,1,2,3,4$,

$X_{\gamma_0,2}=-X_{\gamma_0,2,5}$,

$V_{\gamma_0 1,3}=G_{\gamma_0}-E_{\gamma_0}X_{\gamma_0,2}C_{\gamma_0}$, for $|\gamma_0|<h$,

$V_{\gamma_0 1\gamma_1,3}=V^{(q)}_{\gamma_0 1,3}$, for $q=2^{h-1-|\gamma_0\gamma_1|}$, $|\gamma_0\gamma_1|<h$,

$X_{\gamma_0 1\gamma_1,2,0}=-I/m$,

$X_{\gamma_0 1\gamma_1,2,i+1}=X_{\gamma_0 1\gamma_1,2,i}(2I+V_{\gamma_0 1\gamma_1 0,2}X_{\gamma_0 1\gamma_1,2,i})$, $i=0,1,2,3,4$,

$X_{\gamma_0 1\gamma_1,2}=-X_{\gamma_0 1\gamma_1,2,5}$,

$V_{\gamma_0 1\gamma_1 1,3}=G_{\gamma_0 1\gamma_1,2}-E_{\gamma_0 1\gamma_1,2}X_{\gamma_0 1\gamma_1,2}C_{\gamma_0 1\gamma_1,2}$, for $G_{\gamma_0 1\gamma_1,2},E_{\gamma_0 1\gamma_1,2},C_{\gamma_0 1\gamma_1,2}$ of (10.1), with $\alpha=\gamma_0 1\gamma_1, t=2, |\gamma_0\gamma_1|<h-1$,

$V_{\gamma_0 1\gamma_1 1\gamma_2,3}=V^{(q)}_{\gamma_0 1\gamma_1 1,3,}$, for $q=2^{h-2-|\gamma_0\gamma_1\gamma_2|}$, $|\gamma_0\gamma_1\gamma_2|<h-1$.

Stage 3: $X_{\gamma_0,3,0}=-X_{\gamma_0,2}$,

$X_{\gamma_0,3,i+1}=X_{\gamma_0,3,i}(2I+V_{\gamma_0 0}X_{\gamma_0,3,i})$, $i=0,1,2,3,4$,

$X_{\gamma_0,3}=-X_{\gamma_0,3,5}$,

$V_{\gamma_0 1,4}=G_{\gamma_0}-E_{\gamma_0}X_{\gamma_0,3}C_{\gamma_0}$, for $|\gamma_0|<h$,

$V_{\gamma_0 1\gamma_1,4}=V^{(q)}_{\gamma_0 1,4}$, for $q=2^{h-1-|\gamma_0\gamma_1|}$, $|\gamma_0\gamma_1|<h$,

$X_{\gamma_0 1\gamma_1,3,0}=-X_{\gamma_0 1\gamma_1,2}$,

$X_{\gamma_0 1\gamma_1,3,i+1}=X_{\gamma_0 1\gamma_1,3,i}(2I+V_{\gamma_0 1\gamma_1 0,3}X_{\gamma_0 1\gamma_1,3,i})$, $i=0,1,2,3,4$,

$X_{\gamma_0 1\gamma_1,3}=-X_{\gamma_0 1\gamma_1,3,5}$,

$V_{\gamma_0 1\gamma_1 1,4} = G_{\gamma_0 1\gamma_1,3} - E_{\gamma_0 1\gamma_1,3} X_{\gamma_0 1\gamma_1,3} C_{\gamma_0 1\gamma_1,3}$, for $G_{\gamma_0 1\gamma_1,3}, E_{\gamma_0 1\gamma_1,3}, C_{\gamma_0 1\gamma_1,3}$ of (10.1) with $\alpha = \gamma_0 1\gamma_1, t = 3, |\gamma_0\gamma_1| < h-1$,

$V_{\gamma_0 1\gamma_1 1\gamma_2,4} = V^{(q)}_{\gamma_0 1\gamma_1 1,4}$ for $q = 2^{h-2-|\gamma_0\gamma_1\gamma_2|}, |\gamma_0\gamma_1\gamma_2| < h-1$,

$X_{\gamma_0 1\gamma_1 1\gamma_2,3,0} = -I/m$,

$X_{\gamma_0 1\gamma_1 1\gamma_2,3,i+1} = X_{\gamma_0 1\gamma_1 1\gamma_2,3,i}(2I + V_{\gamma_0 1\gamma_1 1\gamma_2 0,3} X_{\gamma_0 1\gamma_1 1\gamma_2,3,i})$, $i = 0, 1, 2, 3, 4$,

$X_{\gamma_0 1\gamma_1 1\gamma_2,3} = -X_{\gamma_0 1\gamma_1 1\gamma_2,3,5}$,

$V_{\gamma_0 1\gamma_1 1\gamma_2 1,4} = G_{\gamma_0 1\gamma_1 1\gamma_2,3} - E_{\gamma_0 1\gamma_1 1\gamma_2,3} X_{\gamma_0 1\gamma_1 1\gamma_2,3} C_{\gamma_0 1\gamma_1 1\gamma_2,3}$, for $G_{\gamma_0 1\gamma_1 1\gamma_2,3}, E_{\gamma_0 1\gamma_1 1\gamma_2,3}, C_{\gamma_0 1\gamma_1 1\gamma_2,3}$ of (10.1), with $\alpha = \gamma_0 1\gamma_1 1\gamma_2$, $t = 3$, $|\gamma_0\gamma_1\gamma_2| < h-2$,

$V_{\gamma_0 1\gamma_1 1\gamma_2 1\gamma_3,4} = V^{(q)}_{\gamma_0 1\gamma_1 1\gamma_2 1,3}, q = 2^{h-3-|\gamma_0\gamma_1\gamma_2\gamma_3|}, |\gamma_0\gamma_1\gamma_2\gamma_3| < h-2$.

Correctness of Algorithm 10.1 is immediatley verified. It remains to specify the choice of natural g, which would satisfy the requirements of Proposition 7.2, and then to estimate the resulting *compuational cost*.

As in section 8, we assume infinite precision computations in Algorithm 10.1, but the same techniques of backward error analysis as in section 9 enable relatively simple transition to the case of computations with rounding to finite precision of order $n \log\ p$ bits.

The analysis of the approximation errors and of the computation precision given in section 8 is easily extended. In particular, the extension of Proposition 8.2 and its proof is immediate provided that in its statement $V_\beta, V_{\beta 0}$ and $V_{\beta 1}$ are replaced by $V_{\beta,t}$, $V_{\beta 0,t+1}$ and $V_{\beta 1,t+1}$, $t > u(\beta)$. Furthermore, the assumptions of this proposition are extended recursivley with each increase of t and the length $|\beta|$ by 1. Such an extension is analyzed as in section 9. (The transition from $V_{\beta,t}$ to $V_{\beta 1,t+1}$ involves five Newton's steps (10.5), versus four steps used in section 8; an extra step compensates us for the impact of the rounding errors; this suffices according to the analysis in section 9.) The small factor 5 of the error propagation bound of part b) of Propostion 8.2 (even when it increases to 7 due to the rounding errors) is immediately suppressed by Newton's steps (10.5). To accomodate the factors 5 or 7, we also should increase the upper bound on $||V^{-1}_{\beta 0,t} X_{\beta,t,0} + I||$ obtained in Corollary 7.3; the increase is from $1/(10n^2)$ to $1/(2n^2)$ or to $7/(10n^2)$, respectively. This, however, implies only a nominal increase of g, which we may set equal to

$$g = 1 + \lceil \log(b/(2\log n)) \rceil, \tag{10.8}$$

say. ($2\log n$ in the denominator replaces $\log(10n^2)$, which more than compensates us for the error propagation factors five or seven.) The computation of every $V_{\alpha,g+u(\alpha)}$ involves at least $5g$ Newton's steps (10.5), so that the output error norm bound 2^{-b} is guaranteed under (10.8). Therefore, to satisfy the requirement (7.4) of Proposition 7.2, it is sufficient to choose b of order $n \log p$. Then, by (10.8), we have

$$g = O(\log(n \log p)). \tag{10.9}$$

Now, let us estimate the *computational cost* of performing Algorithm 10.1.

For any t, Stage t amounts essentially to ten steps of multiplication of at most n/k pairs of $k \times k$ matrices, for $k = 2^\ell$, $\ell = 1, 2, \ldots, h$. All these multiplications for

for $k = 2^\ell$ and for all ℓ are performed concurrently. Their overall cost is bounded by $O_A(\log n, n^\omega)$, $2 \le \omega < 2.376$ (compare (4.5) and observe that $\sum_{\ell=1}^{h} 2^\ell (n/2^\ell)^\omega = O(n^\omega)$ for $\omega > 1$). Summarizing these bounds for all Stages t, $t = 1, \ldots, g+h$, we obtain

Proposition 10.1 *Algorithm 10.1 supports approximating the RD of a c.-d.d. $n \times n$ matrix V of (7.1) within the error norm bound 2^{-b}, at the overall cost $O_A((\log n)(\log n + g), n^\omega)$ for g of (10.8) and ω of (4.5).*

By combining Algorithms 10.1 and 6.2, summarizing the estimates for the computational cost of their performance, given in particular in (10.9) and Proposition 10.1, and extending the rounding error analysis applied in the proof of Proposition 9.1, we obtain

Corollary 10.1 *The IRD of a c.-d.d. $n \times n$ matrix V of (7.1) can be exactly computed at the computational cost $O_A((\log n)\log(n \log p), n^\omega)$, for ω of (4.5), $2 \le \omega < 2.376$; moreover, this computation can be performed by only involving operations with $\tilde{b}$-bit precision numbers for $\tilde{b} = O(n \log\ p)$.*

11 Computing Modulo a Fixed Prime of the ERD of an Integer Matrix

Our next goal is probabilistic extension of Corollary 10.1 from the class of matrices V of (7.1) to the class of all strongly nonsingular integer matrices A. In this section, we will compute the IRD and even the ERD of A modulo a fixed prime p; in the next section we will shift to the IRD of A.

Let $A = (a_{i,j})$ be a strongly nonsingular $n \times n$ matrix filled with integers $a_{i,j}$. Then by virtue of Proposition 3.2, there exists the RD of A. Let p be a fixed prime, let $0 \le f_{i,j} = a_{i,j} \bmod p < p$, for all i, j, and let

$$F = (f_{i,j}) = A \bmod p\ . \tag{11.1}$$

(Here and hereafter, we assume that $0 \le a \bmod p < p$ for any integer a.)

We will compute modulo p the ERD of the matrix F as an auxiliary stage of computing the ERD of A. At first, we should examine if there exists the ERD modulo p of F.

Lemma 11.1 *[IR82]. Let $f(n)$ be a function defined on the set of positive integers such that $f(n) > 0$ and $\lim_{n\to\infty} f(n) = \infty$. Then there exist two positive constants C and n_0 such that, for any $n > n_0$, the interval*

$$J = \{p : f(n)/n < p < f(n)\} \tag{11.2}$$

contains at least $f(n)/(C \log f(n))$ distinct primes.

Lemma 11.2 *Let $f(n)$, $h_q(n)$ and $k_q(n)$, q=1, ..., Q, be some functions in n such that $h_q(n)$ are integer valued, $h_q(n) \neq 0$,*

$$0 < (h_q(n))^{1/k_q(n)} \leq f(n)/n, \ k_q(n) > 0, \ \lim_{n\to\infty} f(n) = \infty, \tag{11.3}$$

for $q = 1, \ldots, Q$. Let p be a random prime in the interval J of (11.2). Then for the positive constants C and n_0 of Lemma 11.1 and for any fixed $n > n_0$, we have $h_q(n) \neq 0 \bmod p$ for $q = 1, \ldots, Q$ with a probability at least $1-(CK(n)\log f(n))/f(n)$, where $K(n) = \sum_{q=1}^{Q} k_q(n)$.

Proof. Let $l_q(n)$ primes lying in the interval J divide $h_q(n)$. Then their product also divides $h_q(n)$ and, therefore, cannot exceed $h_q(n)$. As these primes lie in the interval J, each of them exceeds $f(n)/n$, and their product exceeds $(f(n)/n)^{l_q(n)}$. Hence, $(f(n)/n)^{l_q(n)} < h_q(n)$. Compare this inequality with the assumed bound $h_q(n) \leq (f(n)/n)^{k_q(n)}$ and obtain that $l_q(n) < k_q(n)$. This holds for all q. Therefore, the number of primes lying in J and dividing at least one of the integers $h_q(n)$ (for any q) is at most $\sum_{q=1}^{Q} l_q(n) < \sum_{q=1}^{Q} k(n) = K(n)$. Compare this number with the overall number of primes in J estimated in Lemma 11.1 and obtain the desired probability estimate. $\square$

Proposition 11.1 *Let $\rho > 2$ be a fixed scalar, let A be a strongly nonsingular $n \times n$ integer matrix, where $n > 1$, $\|A\| > 1$, and let p be a prime chosen randomly (under the uniform probability distribution) in the interval $J = \{p : \ n^{\rho-1} \log \parallel A \parallel < p < n^{\rho} \log \parallel A \parallel\}$. Then $p \geq n$, and the matrix F of (11.1) is strongly nonsingular modulo p with a probability at least $1 - P_{\rho,n}$, for $P_{\rho,n} < (n+1)Cn^{1-\rho}$ and for a positive constant C of Lemmas 11.1 and 11.2.*

Proof. Apply Lemma 11.2 for $f(n) = n^{\rho} \log ||A||$, $h_q(n) = |\det A^{(q)}|$, $Q = n$, and

$$k_q(n) = (q \log \ ||A||)/\log \ (n^{\rho-1} \log \ ||A||),$$

$q = 1, \ldots, n$. Recall from Proposition 2.2 that $|\det A^{(q)}| \leq ||A^{(q)}||^q \leq ||A||^q$, for all q, $q \leq n$, and deduce that (11.3) holds for all $q \leq n$. We immediately deduce that $K(n) = \sum_{q=1}^{n} k_q(n) = ((n+1)n \log ||A||)/(2 \log(n^{\rho-1} \log ||A||))$ and $(\log f(n))/f(n) = (\log(n^{\rho} \log ||A||))/(n^{\rho} \log ||A||)$. Substitute these expressions for $K(n)$ and $(\log f(n))/f(n)$ into Lemma 11.2 and obtain that $(\det A^{(q)}) \bmod p \neq 0$ for $q = 1, \ldots, n$ with a probability at least $1 - P_{\rho,n}$ where

$$P_{\rho,n} < \frac{(n+1)nC \log \ (n^{\rho} \log \ ||A||)}{2n^{\rho} \log \ (n^{\rho-1} \log \ ||A||)} = \frac{(n+1)C}{2n^{\rho-1}} \left(1 + \frac{\log \ n}{\log \ (n^{\rho-1} \log \ ||A||)}\right),$$

for all k. By assumption, we have $||A|| \geq 2$, $\rho > 2$, $n \geq 2$, and it follows that $\log(n^{\rho-1} \log ||A||) > \log \ n$. Combine this bound with the above bound on $P_{\rho,n}$ and obtain the claimed estimate of Proposition 11.1. $\square$

Now, we will assume that a prime p has been chosen in the interval J of Proposition 11.1 and the matrix F of (11.1) is strongly nonsingular modulo p and, therefore, possesses its ERD modulo p.

The next algorithm computes modulo p such an ERD, representing each auxiliary or output rational value as a pair of its numerator and denominator given as two integers reduced modulo p. (This enables us to avoid the costly stage of computing integer reciprocals modulo p.) The RD modulo p of A is computed already at Stage 1 of the algorithm. Subsequent stages yield the extending set of the RD modulo p via the computation of the dual RD modulo p (see the definitions of the extending set and the dual RD in section 3).

Algorithm 11.1, *computing the ERD modulo a fixed prime.*

Input: a prime p and a pair of strongly nonsingular $n \times n$ matrices A and $F = A \bmod p$ filled with integers.

Output: the (common) ERD modulo p of A and F.

Computations:

Stage 0. Compute $m = 10(np)^2$ and the c.-d.d. matrix $V = F - mI$ (cf. (7.1), (7.2)).

Stage 1. Compute modulo p the IRD of V by applying Algorithms 10.1 and 6.2. Then, compute modulo p the RD of V, by dividing modulo p all the computed matrices $m_\alpha V_\alpha$ of the IRD by the computed multipliers m_α for all α; represent the result of each divisions by a pair of an entry of $m_\alpha V_\alpha$ reduced modulo p and $m_\alpha \bmod p$. Output the computed RD modulo p of V, which is also the RD modulo p of $F = V \bmod p$.

Stage 2. Recall Proposition 4.4 and compute $\det V$.

Stage 3. Recall from Proposition 2.2 that $|\det V| \leq \| V \|^n$ and apply Newton's iteration (5.5) for $B = V$ in order to compute an approximation X to V^{-1} satisfying (5.3) for $B = V$ and for b satisfying

$$2^{-b}/m < \| V \|^{-n} /2.2 \ . \tag{11.4}$$

Then, compute the entries of the matrix $X \det V$ and round them to the closest integers, which gives us $\operatorname{adj} V$.

Stage 4. Compute the matrix $\hat{W} = (\operatorname{adj} V) \bmod p - mI$. Apply Algorithms 10.1 and 6.2 to compute modulo p the IRD of $\hat{W}$ (we will prove that this is the dual IRD modulo p of A, V and F). Then compute modulo p the matrices $\hat{W}_{\beta 0}$ of the RD of $\hat{W}$ for all binary strings β of length less than h. Output this set of matrices, to be denoted $\{(\hat{W}_{\beta 0}/\det V) \bmod p\}$. Their entries are the pairs of integers, each reduced modulo p; one integer of each pair is an entry of $\hat{W}_{\beta 0} \bmod p$ and another is $(\det V) \bmod p$. (This set of matrices defines the extending set $\{V_{\beta 0}^{-1} \bmod p\}$ of the RD modulo p of the input matrix F.)

To verify *correctness* of Algorithm 11.1, first extend Corollary 7.2 to obtain that $\| V^{-1} \| \leq 1.1/m$. Together with (11.4), this implies the bound

$$\| X \det V - \operatorname{adj} V \| < 1/2$$

for the matrix X computed at Stage 3 of Algorithm 11.1. Therefore, the rounding at this stage correctly defines adj V.

Furthermore, the matrices $\hat{W}_\alpha$ mod p (see Stage 4) represent the RD modulo p of adj V. Therefore, the set $\{(\hat{W}_\alpha/\det\ V) \bmod p\}$ represents the RD modulo p of V^{-1}. To complete the correctness proof, it remains to observe that the set of matrices $\{(\hat{W}_{\beta 0}/\det\ V) \bmod p, |\beta| < h\}$ is nothing else but the extending set $\{B_{\beta 0}^{-1} \bmod p, |\beta| < h\}$ of the (common) RD modulo p of the three matrices A, V and $F = V \bmod p = A \bmod p$. This follows from the next simple result:

Proposition 11.2 *Let $\{V_\alpha\}$ and $\{W_\alpha\}$ denote the RD and the dual RD of a pair of $n \times n$ matrices V and $W = V^{-1}$, respectively. Then, $V_\alpha^{-1} = W_\alpha$ for all binary strings α of length at most h.*

Proof. Compare (2.3) and (2.4) to obtain that $V_0^{-1} = W_0$, $V_1^{-1} = S^{-1} = W_1$. Recursively extend this observation to all binary strings α, to complete the proofs of both of Proposition 11.2 and, consequently, of the correctness of Algorithm 11.1. □

Similarly to deducing Corollary 10.1, we estimate the complexity of performing Algorithm 11.1. We arrive at

Proposition 11.3 *The ERD modulo a fixed prime p of an $n \times n$ matrix A filled with integers and strongly nonsingular modulo p (that is, such that $(\det A^{(q)}) \bmod p \neq 0$, for all q), as well as $\det A^{(q)} \bmod p$, for all q, can be computed at the cost $O_A((\log n)\log(n \log p),\ n^\omega)$ for ω of (4.5), $2 \leq \omega < 2.376$; moreover, this computation can be performed by computing with the $\tilde{b}$-bit precision operands for $\tilde{b} = O(n \log\ p)$.*

Remark 11.1 *One can be tempted to simplify Algorithm 11.1 and to compute modulo p the extending set $\{V_{\alpha 0}^{-1}\}$ of the RD of the matrix V via a more straightforward application of the techniques of sections 3–10. In particular, one may proceed by following the recipe of [R95]: first approximate the matrices $V_{\alpha 0}^{-1}$ closely enough, then multiply the approximations by appropriate integer multipliers M_α to arrive at approximations (within an error norm bounded by less than 1/2) to integer matrices $M_\alpha V_{\alpha 0}^{-1}$, and then recover the matrices $M_\alpha V_{\alpha 0}^{-1}$ via rounding and $V_{\alpha 0}^{-1}$ via divisions by M_α. The problem with this approach is in bounding the size of the multipliers M_α. We need to have $\log |M_\alpha| = \tilde{O}(n)$ in order to support the bit-precision bounds of Proposition 11.3, but if we follow the cited recipe, we would only reach the bounds of order $\tilde{O}(n^2)$ on $\log |M_\alpha|$, which would imply involving extra factor n in the bit-precision and the bit-complexity bounds. Here, the notation $\tilde{O}(s)$ should be read as $O(s \log^c s)$ for a constant c independent of s.*

12 p-adic Lifting of the ERDs and the Recovery of the Inverses, Determinants, and Ranks of Integer Matrices

In the previous section, we computed the ERD modulo p of an integer matrix A, which is strongly nonsingular modulo p. We will now compute its p-adic (Newton-Hensel's) lifting, that is, the ERD modulo p^{2^g} of A for a fixed natural $g \geq h = \log n$. We will achieve this by incorporating the known techniques [MC79] for p-adic lifting of matrix inverses into our Algorithm 10.1. In this application we will slightly simplify the algorithm by replacing the four steps of Newton's iteration of (10.4)–(10.6) by a single step of the computation of the matrix

$$X_{\beta,t} = X_{\beta,t,0}(2I - V_{\beta 0,t}X_{\beta,t,0}), \tag{12.1}$$

where

$$X_{\beta,t,0} = \{ \begin{array}{c} V_{\beta 0,t}^{-1} \bmod p \text{ for } t = u(\beta)+1, \\ X_{\beta,t-1} \text{ for } t > u(\beta)+1, \end{array} \tag{12.2}$$

and all matrices $V_{\beta 0,t}^{-1} \bmod p$ are supplied as an input to the p-adic lifting algorithm. (The latter expression for $X_{\beta,t,0}$ replaces (10.4).) The only other change versus Algorithm 10.1 is that all the arithmetic operations in (10.7) and (12.1) are perfomred modulo p^{2^s} for $s = t-1-u(\beta)$ and for $u(\beta)$ denoting (as in section 10) the number of ones in a binary string β.

Correctness of the resulting algorithm follows because equation (12.1) and the inductive assumption that $X_{\beta,t-1} = V_{\beta 0,t-1}^{-1} \bmod p^{2^s}$, $s = t-2-u(\beta)$, together imply that

$$(I - V_{\beta 0,t}X_{\beta,t} - (I - V_{\beta 0,t}X_{\beta,t,0})^2) \bmod p^{2^{s+1}} = 0 \ ,$$

and therefore,

$$X_{\beta,t} = V_{\beta 0,t}^{-1} \bmod p^{2^{s+1}} \tag{12.3}$$

(compare [MC79] or [BP94], Fact 3.3.1 on page 244).

The *arithmetic complexity* estimates $O_A((g+\log n)\log n, n^\omega)$ of Proposition 10.1 are extended to the case of the resulting algorithm (hereafter referred to as **Algorithm 12.1**), where g denotes a fixed natural input value, $g \geq h = \log n$.

We will keep assuming that p is a prime fixed in the interval J of Proposition 11.1, $||A|| > 1$, $n > 1$, and the matrix F of (11.1) is strongly nonsingular. Furthermore, hereafter we will assume that

$$g = 1 + \lfloor \log \frac{1 + n\log ||A||}{\log p} \rfloor \tag{12.4}$$

Then, we have

$$4||A||^{2n} \geq p^{2^g} > 2||A||^n. \tag{12.5}$$

Therefore, by the virtue of Proposition 2.2, the value $0.5p^{2^g}$ exceeds $|\det A|$ as well as the maximum absolute value of any entry of adj A. We observe that

$$q = \begin{cases} q \bmod p & \text{if } q \bmod p < 0.5q, \\ (q \bmod p) - p & \text{otherwise,} \end{cases}$$

provided that q is an integer and $2|q| < p$. These observations, Corollary 4.1, and relations (12.5) together enable us to recover det A from (det A) mod p^{2^g} and adj A from (adj A) mod p^{2^g}, as the p-adic lifting of the ERD is completed. Then, we may immediately compute $A^{-1} = (\text{adj } A)/\det A$, since A is a nonsingular matrix.

Remark 12.1 *We may control the computational precision at the last lifting stage (where the precision is the largest) simply by performing this stage modulo p^q where $q = \lfloor \log(2\|A\|^n) \rfloor + 1$, so that $2\|A\|^n \le p^q \le 2p\|A\|^n$.*

Summarizing the algorithms and the complexity estimates of this and the previous sections, we arrive at

Proposition 12.1 *Let A be a strongly nonsingular $n \times n$ matrix filled with integers. Let $n > 1$, let $\|A\| > 1$, and let p be a prime from the interval J of Proposition 11.1 for a fixed $\rho > 2$. Furthermore, let the matrix A be strongly nonsingular modulo p too. Then, one may compute A^{-1} and* det $A^{(k)}$*, $k = 1, 2, \dots, n$, in two stages that amount essentially to application of Algorithms 11.1 and 12.1, respectively, and are performed at the arithmetic cost bounded by $O_A((\log n)\log(n \log p), n^\omega)$, at the first stage (compare Proposition 11.3) and $O_A((\log n)\log(n \log ||A||), n^\omega)$, at the second stage, for ω of (4.5), $2 \le \omega < 2.376$.*

Assuming p chosen from the interval J of Proposition 11.1, we obtain that $\log p = O(\log(n \log ||A||))$, so that the overall arithmetic cost is dominated by the cost of the second stage:

Corollary 12.1 *Under the assumptions of Proposition 12.1, one may compute A^{-1} and* det $A^{(k)}$*, for $k = 1, 2, \dots, n$, at arithmetic cost $O_A((\log n)\log(n \log \|A\|), n^\omega)$, for ω of (4.5).*

Let us extend Proposition 12.1 and Corollary 12.1 to estimate at first the bit-precision and then the Boolean complexity of the same computations.

We immediately recall the bound $O(n \log\ p)$ on the bit-precision required in Algorithm 11.1, that is, at the first stage of the computations of Proposition 12.1. At the second stage (that is, essentially for Algorithm 12.1), we revisit the derivation of Proposition 10.1, where we estimated the complexity of the stage of numerical approximation of the RD and ERD of V, and recall or estimate again that this stage is essentially reduced to at most $g + h$ substages, for g of (12.4) and for $h = \log n$, such that the cost of performing each substage is dominated by the cost of ten steps of multiplication of at most n/k pairs of $k \times k$ matrices for $k = 2^\ell$ and $\ell = 0, 1, \ldots, h-1$. At the stage of the application of Algorithm 12.1, only four (instead of ten) steps are needed. At each of such four steps, all the matrix multiplications are performed concurrently, as in the case of the derivation of Proposition 10.1. Furthermore, at every step of Substage t of the second stage, $t = 1, \ldots, g+h$, at most $n/2^l$ pairs of matrices of the sizes $2^l \times 2^l$ are encountered for $l = n - |\beta| - 1, u(\beta) < t$. Such matrices are pairwise multiplied together modulo $p^{2^{t-u(\beta)}}$,

$$t - u(\beta) \leq \lambda(t) = \min \{t, g\}. \tag{12.6}$$

The above bounds on the modulo imply some bit-precision bounds since computation modulo ℓ can be performed with $2\lceil \log \ell \rceil$-bit precision. Furthermore, we recall the known estimates $O_B((\log k) \log\log k, k)$ for the Boolean complexity of performing an arithmetic operation modulo $2^k - 1$ (see [AHU74], [BP94], [CK91], [RT90]), which can be extended to our computations whenever we perform them with k-bit precision.

By combining the latter estimates with ones for the arithmetic cost and for the bit-precision of our computations, we bound the Boolean cost of performing Algorithm 11.1, that is, the first stage of the computations supporting Proposition 12.1 (cf. Corollary 10.1) by

$$O_B((\log n)(\log(n \log p))^2 \log\log(n \log p), n^{\omega+1} \log\ p),$$

and we bound the Boolean cost of performing the t-th stage of Algorithm 12.1 by

$$O_B((\log n)(\log(2^{\lambda(t)} \log p)) \log\log(2^{\lambda(t)} \log p), n^\omega 2^{\lambda(t)} \log p),\ t = 1, \ldots, g+h.$$

(Cf. (12.6) and recall that the t-th stage of Algorithm 12.1 is the t-th substage of the second stage of the computations of Proposition 12.1.)

By summarizing all these estimates, for p lying in the interval J of Proposition 11.1 and for g satisfying (12.4), (12.5), we estimate the Boolean complexity of our computations. To simplify the expressions for the resulting estimates, we write

$$A = (a_{i,j}), \quad a = \log \max_{i,j} |a_{i,j}|\ , \tag{12.7}$$

and obtain that $g = O(\log(na))$, $g+h = O(\log(na))$, $2^g \log p = O(na)$ for g of (12.4), $\log p = O(\log(na))$, $\log(n \log p) = O(\log(n \log a))$. Then, we rewrite our Boolean cost bounds as follows:

$$O_B((\log n)(\log(n \log a))^2 \log\log(n \log a), n^{\omega+1} \log(na))$$

for performing Algorithm 11.1,

$$O_B((\log n)(\log(na)) \log\log(na), n^{\omega+1} a)$$

for performing the t-th stage of Algorithm 12.1 for $t = g+1, \ldots, g+h$, where $\lambda(t,g) = g+1$, and

$$O_B((\log n)(t + \log\log(na))\log(t + \log\log(na)), n^{\omega}2^t \log p)$$

for performing the t-th stage of Algorithm 12.1 for $t = 1, \ldots, g$ where $\lambda(t,g) = t$. By applying the B-principle, we bound the overall cost of performing the first $g = O(\log(na))$ stages of Algorithm 12.1 by

$$O_B((\log n)(\log(na))^2 \log\log(na), n^{\omega+1}a/\log(na)),$$

and we bound the overall cost of performing its last $h = \log n$ stages by $O_B((\log n)^2(\log(na))\log\log(na), n^{\omega+1}a)$. Then again, we apply the B-principle, to yield the same parallel Boolean time bound, $O((\log n)(\log(na))^2 \log\log(na))$, in all the three estimates (for Algorithm 11.1, for the first g stages of Algorithm 12.1, and for its last h stages), which gives us the Boolean processor bounds $O((n^{\omega+1}(\log(n\log a))^2 \log\log(n\log a))/(\log(na)\log\log(na))) = O(n^{\omega+1}(\log(n\log a))^2/\log(na))$, $O(n^{\omega+1}a/\log(na))$, and $O((n^{\omega+1}a\log n)/\log(na))$, for these three groups of computations, respectively. We note that the sum of the three latter bounds gives us $O((\log n)(a + \log n)n^{\omega+1}/\log(na))$.

By using the Boolean cost bounds of Proposition 11.3 for computing $\det A^{(k)} \bmod p$, for all k, and by combining the cited Boolean time bound and the latter processor bound, we obtain

Proposition 12.2 *Under the assumptions of Proposition 12.1, one may compute the inverse matrix* $A^{-1} \bmod p$ *and* $\det A^{(k)} \bmod p$, $k = 1, \ldots, n$, *at the Boolean cost*

$$O_B((\log n)(\log(n\log a))^2 \log\log(n\log a), n^{\omega+1}\log(na)),$$

and one may compute the matrix A^{-1} *and* $\det A^{(k)}$, $k = 1, \ldots, n$, *at the Boolean cost* $O_B((\log n)(\log(na))^2 \log\log(na), (\log n)(a + \log n)n^{\omega+1}/\log(na))$ *for* ω *of (4.5) and* a *of (12.7).*

Remark 12.2 *Our choice of a prime* p *and our complexity estimates rely on the bounds of Proposition 2.2 on* $|\det W|$. *For a large class of matrices* W, *such bounds can be refined a little (e.g. by using Hadamard's upper bound on* $|\det A|$) *and so can our complexity estimates. Likewise, by expressing the estimates of Proposition 12.2 in terms of* $||A||$, *rather than* a, *one may obtain some slightly refined (though more complicated) estimates. Finally, our estimates for parallel Boolean cost can be slightly improved if, instead of the bounds* $O_B((\log k)\log\log k, k)$ *on the cost of an arithmetic operation, we will rely on the bounds* $O_B(\log k, k\log\log k)$, *which hold for the cost of an addition, a subtraction and a multiplication (see e.g. [BP94], p. 297). We may rely on the latter bound because the ops of the latter three classes are most numerous among all the ops in our algorithms. Similar observations apply to the estimates of Theorems 1.1 and 1.2.*

It remains to work out the strong nonsingularity issue in order to extend the complexity estimates of Corollary 12.1 and Proposition 12.2 to ones of Theorem 1.1. (Note that, in terms of a, the bounds of Corollary 12.1 turn into $O_A((\log n)\log(na)$, $n^\omega)$, as required in Theorem 1.1.)

We will first assume that A is a nonsingular matrix. In this case, AA^T is an s.p.d. matrix and, consequently, a strongly nonsingular matrix, by Corollary 2.1. Consequently, AA^T is strongly nonsingular modulo p, with a probability $1 - P_{\rho,n}$ for $P_{\rho,n}$ bounded according to Proposition 11.1. Therefore, we may apply the results of this section to compute at first $(AA^T)^{-1}$ and then $A^{-1} = A^T(AA^T)^{-1}$ and $\mathbf{x} = A^{-1}\mathbf{f}$ satisfying $A\mathbf{x} = \mathbf{f}$. (Strong nonsingularity (modulo p) of AA^T is tested as a by-product of computing $(AA^T)^{-1}$.) We may also immediately compute $\det(AA^T) = (\det A)^2$, though this does not give us the sign of $\det A$. The matrix A is singular (that is, $\det\ A = 0$) if and only if application of the same approach to a matrix A requires to invert a singular matrix at some step.

Next, we will apply randomization to relax the assumptions about (strong) nonsingularity of A when we compute rank A and the sign of det A. Towards this goal, we fix $\rho > 2$, a sufficiently large finite set of integers, S, and two matrices U and L, as specified in Proposition 2.10, compute the matrix $\tilde{A} = UAL$ (cf. Remark 12.3 at the end of this section), fix a random prime p in the interval J of Proposition 11.1, and extend Algorithm 11.1 to compute $(\det\ \tilde{A}^{(k)}) \bmod p$, for $k = 1, \dots, n$, and $r(p) = \max\{k,\ (\det\ \tilde{A}^{(k)}) \bmod p \neq 0\}$. Let us write $\tilde{r} = \max\{k,\ \det\ \tilde{A}^{(k)} \neq 0\}$, so that rank $A \geq \tilde{r} \geq r(p)$. Furthermore, $\tilde{r} =$ rank A with a probability at least $P_r = 1 - (\tilde{r}+1)\tilde{r}/|S|$ (due to Proposition 2.10), and $\tilde{r} = r(p)$, with a probability $1 - P_{\rho,n}$, estimated in Proposition 11.1. Thus, we output $r(p)$ as rank A and arrive at the estimate of Theorem 1.1 for the randomized cost of computing rank A. (Note that in this case, the computations modulo p suffice; thus, in our computation of rank A, we omit the p-adic lifting stage and rely on the first Boolean cost estimate of Proposition 12.2.)

Let us extend this technique to the computation of the sing of det A. If $r(p) < n$, then (det A) $\bmod p = 0$, and we output $\det A = 0$, which is correct with a probability at least $1 - P_{\rho,n}$. Otherwise, that is, if $r(p) = n$, then we have $n \geq$ rank $A \geq r(p) = n$, that is, A is nonsingular. Furthermore, by using the randomization based on Proposition 2.10, we may compute $\det A = \det(UAL)$, because UAL is strongly nonsingular, with a probability at least $1-(n+1)n/|S|$ if A is nonsingular. By letting $|S| = n^4$, say, and by applying Propositions 11.3 and 12.2 to the matrix UAL, we arrive at the desired algorithm for $\det A$, supporting Theorem 1.1.

Now, assume that $r(p) < n$ and that the $r(p) \times r(p)$ leading principal submatrix $B = \tilde{A}^{(r(p))}$ of $\tilde{A}$ is nonsingular. Let us write $\tilde{A} = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$, $G = \begin{pmatrix} I & -B^{-1}C \\ 0 & I \end{pmatrix}$, and observe that $\tilde{A}G = \begin{pmatrix} B & 0 \\ D & Q \end{pmatrix}$, where $Q = 0$ if and only if $r(p) =$ rank A. (Compare [KP91] and [BP94], pp. 110 and 333.) This gives us an algorithm for verification whether $r(p)$=rank A (at the cost within the asymptotic cost bounds of Theorem 1.1). If so, then the $n-r$ columns of the matrix $LG\begin{pmatrix} 0 \\ I \end{pmatrix}$, where I denotes the $(n-r)\times(n-r)$ identity matrix, give us a basis for the null-space, $\mathbf{N}(A)$, of A (compare Definition 2.7). We recall from Fact 2.1 that if there exists a solution $\mathbf{x}$ to a linear system $A\mathbf{x} = \mathbf{f}$, then it can be represented as $\mathbf{x} = \mathbf{x}_0 + \mathbf{z}$, $\mathbf{x}_0$ being a fixed specific solution and $\mathbf{z}$ being a vector from $\mathbf{N}(A)$.

Let $\mathbf{g}$ be the r-dimensional prefix-subvector of $\mathbf{f}$, made by the first r components of $\mathbf{f}$. Let $\mathbf{y} = B^{-1}\mathbf{g}$ be the solution to the nonsingular system $B\mathbf{y} = \mathbf{g}$. Then, a specific solution $\mathbf{x}_0$ to the system $UA\mathbf{x} = U\mathbf{f}$ is given by $\mathbf{x}_0 = LG\begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}$ if the latter linear system is consistent, and we have $UA\mathbf{x_0} \neq U\mathbf{f}$ otherwise. This completes our proof of Theorem 1.1. $\square$

Remark 12.3 *Our computations supporting Theorem 1.1 include some $n \times n$ matrix multiplications (of A by A^T, L and U). Their cost bound is dominated by the complexity bounds of Theorem 1.1, and a similar argument applies to yield the extension of this theorem to Theorem 1.2 to be shown in section 14 (cf. Proposition 14.1). The increase of the matrix norm in the transition from A to AA^T and $\tilde{A} = UAL$ may cause the increase only by a constant factor in the estimate for the precision of the computations and their Boolean complexity (if we choose, say, $S = \{1, 2, \dots, |S|\}$ and $|S| = n^{O(1)}$).*

13 Some Definitions and Auxiliary Results on Computations with Structured Matrices

Our next goal is to show that the computational cost of our algorithms supporting Theorem 1.1 decreases dramatically, to the level of the estimates of Theorem 1.2, provided that the input matrix has Toeplitz-like structure. In this section we will recall some definitions and some simple and/or well-known facts on Toeplitz-like matrices, which we will use in the next section towars the stated goal (cf. equations (1.1)and (1.2) of section 1.1, Definition 2.8, and [BP94], [CKL-A87], [KKM79], [P92]).

Proposition 13.1 *The product of a $k \times k$ Toeplitz matrix (cf. Definition 2.8) and a vector of dimension k can be computed at the cost $O_A(\log k, k)$ (via reduction to three FFTs, each on $O(k)$ points, or to convolution of two vectors of dimension $O(k)$).*

Definition 13.1 *For a $k \times k$ matrix A and for the matrix Z of Definition 2.8 , write $F_+(A) = A - ZAZ^T$, $F_-(A) = A - Z^TAZ$. If $F(A) = GH^T$ for a pair of $k \times \ell$ matrices G and H and for $F = F_+$ or $F = F_-$, then the pair of G, H is called an F-*generator *of A of length ℓ. (Note that, in this case, the pair H, G is an F-generator of A^T of the same length.) The minimum length ℓ of an F-*generator *of A, for fixed*

A and F, is called the F-rank *of A, is denoted by $r_F(A)$, and is equal to* rank $F(A)$. *A $k \times k$ matrix A is called a* Toeplitz-like matrix *if it is given with its F-generator (for $F = F_+$ or $F = F_-$) having a length bounded by a constant independent of k. F-generators and F-ranks, for both $F = F_+$ and $F = F_-$, are also called* displacement generators *and* displacement ranks *(following the original definitions of [KKM79]).*

Proposition 13.2 *$r_F(T) \leq 2$ if T is a Toeplitz matrix, and $r_F(T) \leq 1$ if T is a triangular Toeplitz matrix, for $F = F_+$ and $F = F_-$. In particular, $r_F(I) = 1$.*

The correlation to (1.2) is given by the following result.

Proposition 13.3 *G, H is an F_+-generator (respectively, F_--generator) of A having a length ℓ, $G = (\mathbf{g_1}, \ldots, \mathbf{g}_\ell)$, $H = (\mathbf{h_1}, \ldots, \mathbf{h}_\ell)$, if and only if $A = \sum_{s=1}^{\ell} L(\mathbf{g_s})\ L^T(\mathbf{h_s})$ (respectively, if and only if $A = \sum_{s=1}^{\ell} L^T(\mathbf{g_s})\ L(\mathbf{h_s})$).*

Based on the latter results, we will operate with the F-generators of Toeplitz-like matrices, rather than with the matrices themselves. Such a representation is memory space efficient and also enables us to use less sequential time and fewer processors in Toeplitz-like computations, due to the following corollary (cf. Propositions 13.1 and 13.3):

Corollary 13.1 *The product of a $k \times k$ Toeplitz-like matrix by a vector of dimension k can be computed at the cost $O_A(\log k, k)$.*

The next result gives us more specific estimates – the cost bound of Toeplitz-like matrix multiplication is proportional to the square of the sum of the lengths of the F-generators of the input matrices, and such a length is roughly doubled in a matrix addition or multiplication.

Proposition 13.4 *Given F-generators, G_A, H_A of length ℓ_A and G_B, H_B of length ℓ_B, of $k \times k$ matrices A and B, respectively (for $F = F_+$ or $F = F_-$), one may compute an F-generator G_{AB}, H_{AB} of AB of length at most $\ell_A + \ell_B + 1$ at the cost $O_A(\log k, (\ell_A + \ell_B)^2 k)$, whereas an F-generator of $A + B$ of length at most $\ell_A + \ell_B$ is immediately available cost-free.*

In view of the latter results, we will study various bounds on the F-ranks and the length of F-generators, in particular regarding the matrices involved in the RD and Newton's iteration with Toeplitz-like input.

Proposition 13.5 *a) $r_{F+}(A) \leq r_{F-}(A) + 2$, $r_{F-}(A) \leq r_{F+}(A) + 2$ for any matrix A. Furthermore, an F_+-generator (respectively, F_--generator) of a length ℓ for any matrix A can be immediately transformed (at the cost $O_A(\log n, n)$ of performing $O(1)$ convolutions or FFTs) into an F_--generator (respectively, F_+-generator) of length at most $\ell + 2$ for A.*

b) If A is nonsingular, then $r_{F+}(A^{-1}) = r_{F-}(A)$.

The next result is immediately verified (compare Definition 2.4).

Proposition 13.6 *Let $GH^T = F_+(W)$ for a $k \times k$ matrix W. Then $(GH^T)^{(i)} = F_+(W^{(i)})$ for $i = 1, 2, \ldots, k$; furthermore, $r_{F_+}(C) \leq r_{F_+}(W)+1$, $r_{F_+}(E) \leq r_{F_+}(W)+1$, under (2.1), and $r_{F_+}(T) \leq r_{F_+}(W)+2$ for any submatrix T of W formed by contiguous sets of row and columns of W.*

It follows that $r_{F_+}(B) \leq r_{F_+}(W)$, under (2.1).

We observe similar relations for trailing principal submatrices and the operator F_-. By Proposition 2.3, S^{-1} is a trailing principal submatrix of W^{-1}. Therefore, $r_{F_-}(S^{-1}) \leq r_{F_-}(W^{-1})$. By applying Proposition 13.5 b) for $A = S$ and $A = W$, we obtain that $r_{F_+}(S) \leq r_{F_+}(W)$.

Proposition 13.7 *Let (2.1) and (2.2) hold, where B, S, and W are nonsingular matrices. Then* $\max\{r_{F_+}(B), r_{F_+}(S)\} \leq r_{F_+}(W)$.

By applying the latter proposition recursively, we bound the F_+-rank throughout the RD.

Corollary 13.2 *Let V_α be a matrix of the RD of a matrix A. Then, $r_{F_+}(V_\alpha) \leq r_{F_+}(A)$.*

So far, we have no tools yet to counter the growth of the length of the F-generators in the process of Newton's iteration. Developing such tools (which we call the techniques for *the truncation of a generator (TG))* is our next task. Namely, we will next (in Proposition 13.8) show how to compute a shorter F-generator of a matrix having small F-rank but given with its longer F-generator. This is our first technique of TG. It will be used to refine p-adic (Newton-Hensel's) lifting to bound the length of the F-generators of the matrices involved there. We will prove easily, based on Propositions 13.5, and 13.7, that such matrices have small F-rank if so has the input matrix. For Newton's iteration of Algorithm 5.1, such a property does not hold, and the F-rank of the computed approximations to the Toeplitz-like inverses may grow quite rapidly. These approximations, however, always have matrices with small F-rank nearby, and we will periodically shift to the latter matrices and then restart Newton's process. Our tool for such a shift will be Algorithm 13.1 (see [PBRZ99] on some alternative tools).

Proposition 13.8 *Let an F-generator of a $k \times k$ matrix A of length ℓ (for $F = F_+$ or $F = F_-$) and an upper bound $r^* < l$ on the F-rank $r_F(A)$ be given. Then an F-generator of A of length at most r^* can be computed at the cost $O_A(l,\ kl)$.*

Proof. Apply the proof of Proposition A.6 of [P92] or the solution of Problem 2.11 of [BP94], pp. 111-112. Verify that all the computations (including the computation of the LSP factorization or, alternatively, PLU factorization) can be performed at the claimed overall cost. □

Let us next show the promised alternative algorithm for controlling the length of F-generators of matrices involved in Newton's process. The algorithm relies on the *SVD truncation of F-generator*, which is our second TG technique.

Algorithm 13.1 [P92b], [P93], [P93a].

Input: $F = F_+$ or $F = F_-$, an F-generator G, H of a $k \times k$ matrix A of length l, and a natural $r' < l$.

Output: an F-generator G', H' of a $k \times k$ matrix A' of length at most r' such that

$$\|A' - A\|_2 \leq 2(1 + 2(r_F(A) - r')k) \min_Y \|Y - A\|_2 \ , \tag{13.1}$$

where the minimum is over all $k \times k$ matrices Y of F-rank at most r'.

Computations:

Stage 1. Compute the singular value decomposition (SVD) of the matrix $GH^T = F(A)$, that is, compute a pair U and V of unitary $k \times l$ matrices and an $l \times l$ diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_l)$ for positive $\sigma_1, \ldots, \sigma_l$ satisfying

$$GH^T = F(A) = U\Sigma V^T.$$

Stage 2. Compute and output an F-generator G', H' of A' of length at most r' as follows:

$$G' = U\Sigma_{r'}, \quad H' = VI_{r',l},$$

where $\Sigma_{r'} = \mathrm{diag}(\sigma_1, \ldots, \sigma_{r'}, 0, \ldots, 0)$ and $I_{r',l} = \mathrm{diag}(1, \ldots, 1, 0, \ldots, 0)$ are $l \times l$ matrices of rank r'.

On the correctness proof of this algorithm and on the bound $O_A(\log k, k/\log k)$, for $l = O(1)$, on the *computational cost* of its performance, see [P92b], [P93], [P93a].

Remark 13.1 *Bound (13.1) is proved in [P92b], [P93], [P93a], based on approximate computation of the SVD at Stage 1 of the algorithm. Any improvement of the approximation of the SVD would decrease the factor 2 of (13.1) which turns into 1 if the SVD is computed exactly.*

Remark 13.2 *If $r' \geq r_F(A)$, then (13.1) implies that $\|A' - A\|_2 = \min_Y \|Y - A\|_2 = 0$, and then Algorithm 13.1 is an alternative to the algorithm supporting Proposition 13.8, except that the latter algorithm is rational (it can be performed with no errors over the rational), whereas Algorithm 13.1 has a nonrational, though numerically stable stage of computing the SVD. This suggests that the algorithm supporting Proposition 13.8 should be applied in Algorithm 12.1, at the p-adic lifting stage, whereas Algorithm 13.1 is a better candidate to use in numerical applications of Algorithm 11.1, performed with rounding.*

14 Improvement of the Algorithms for the ERD, IRD, Inverse, Determinant, and Rank in the Toeplitz and Toeplitz-like Cases

Let us apply the techniques and the results of the previous section to re-examine the computation of the ERD and IRD of a strongly nonsingular $n \times n$ matrix A filled with integers in the case where A is a Toeplitz or Toeplitz-like matrix given with its F_+-generator G, H of length $r = r_{F_+}(A) = O(1)$.

We recall that $r_{F_+}(V_\alpha) \leq r$ for all matrices V_α of the RD of A (compare Corollary 13.2), and we will apply either the algorithm supporting Proposition 13.8 or Algorithm 13.1 in order to decrease (to a level at most r) the length of the computed F-generators of these matrices, in all cases where this length exceeds r. Likewise, we will obtain from Propositions 13.4– 13.6 that the computation of $V_{\beta 1,t+1}$ according to (10.1)–(10.7) only involves matrices whose F-ranks are bounded from above by $3r + r_{F_+}(X_{\beta,t}) + 6$.

According to our analysis, the matrix $X_{\beta,t}$ approximates $B_{\beta 0}^{-1}$ for all binary strings β of length at most $h-1$, and since $r_{F_+}(B_{\beta 0}) \leq r$, we have $r_{F_-}(B_{\beta 0}^{-1}) \leq r$, $r_{F_+}(B_{\beta 0}^{-1}) \leq r+2$ (compare Proposition 13.5). We will apply Algorithm 13.1 in order to compute an F_+-generator of length at most $r+2$ for a matrix $X'_{\beta,t}$ approximating $X_{\beta,t}$ and, therefore, also $V_{\beta 0}^{-1}$. (The approximation of $V_{\beta 0}^{-1}$ by $X'_{\beta,t}$ deteriorates slightly, versus the approximation by $X_{\beta,t}$, but since $X'_{\beta,t}$ still closely approxiamtes the matrix $V_{\beta 0}$, we more than compensate ourselves for such a deterioration by performing an extra Newton step in (10.5).) Then, all matrices involved in the computation of the ERD and the IRD of A will be represented by their F_+-generators of length $O(r)$.

Similar argument is applied to the computation of the p-adic lifting of the ERD of A, except that this argument is simplified since (12.3) and Proposition 13.5 together imply that

$$r_{F_+}(X_{\beta,t+1} \bmod p^{2^{t-u(\beta)}}) \leq r_{F_-}(X_{\beta,t+1} \bmod p^{2^{t-u(\beta)}}) + 2 =$$

$$r_{F_+}(B_{\beta 0}^{-1} \bmod p^{2^{t-u(\beta)}}) + 2 \leq r + 2.$$

Thus, to keep the length of the associated F_+-generators bounded, we just apply the rational algorithm that supports Proposition 13.8, instead of applying Algorithm 13.1. In fact, we may also apply other alternative techniques for bounding the length of an F-generator of $X_{\alpha,i+1}$; such techniques may relies on using distinct operators F, such as $F^+(A) = AZ - ZA$ (see [BP94], p. 189) or ones using some f-circulant matrices instead of Z (see [PBRZ99]).

Finally, it is easily verified (cf. [P96b]) that the computation (of section 12) of a basis for the null-space of A also involves only matrices represented by their F_+-generators of length $O(r)$, for a matrix A given with its F_+-generator of length r.

Let us now turn to estimating the computational cost, in the case of Toeplitz or Toeplitz-like input. There are two new features versus the case of a general integer input matrix A:

1) Performing every matrix multipication, we operate with F_+-generators of Toeplitz-like matrices involved in these multiplications and apply Propositions 13.3, 13.4 and Corollary 13.1.

2) Some of these matrix multiplicaitons are followed by the applicaiton of the algorithms supporting Proposition 13.8 or Algorithm 13.1.

The manipulation with the F_+-generators enables us to decrease the arithmetic processor bound of Corollary 12.1 from n^ω to $n \log n$, because concurrent multiplications of $O(2^t)$ pairs of $(n/2^t) \times (n/2^t)$ Toeplitz-like matrices, for $t = 1, \ldots, h$, $h = \log n$, are performed at the overall cost bounded by $O_A(\log n, n \log n)$ (versus $O_A(\log n, n^\omega)$ in the case of general integer input matrices). The estimated overall cost of the required computations (of A^{-1}, det A and so on) is dominated by the estimated cost of all Toeplitz-like matrix multiplications involved, because, according to section 13, the estimated cost of such a multiplication dominates the estimated cost of the application of both Algorithm 13.1 and the algorithm supporting Proposition 13.8.

Summarizing, we obtain the following result:

Proposition 14.1 *If the $n \times n$ input Toeplitz-like matrix A is strongly nonsingular and is filled with integers, then one may modify the randomized computation of its ERD and IRD according to the algorithms of sections 6–12 in order to perform all these computations at the overall cost $O_A((\log n) \log(n \log \|A\|), n \log n)$.*

The cost bounds of Proposition 14.1 are immediatley extended to the solution of all the computational problems listed in Theorem 1.1 where now we assume a Toeplitz-like input matrix A and represent its inverse or the basis matrix for its null-space by their short F-generators. (Verifying the correctness of the computation of the rank and the inverse, we should also deal with short F-generators and use Proposition 13.8 to avoid processing n^2 entries of $n \times n$ matrices, which would have required order of n^2 ops.)

To obtain a similar extension of the Boolean complexity bounds of Proposition 13.8 and Theorem 1.1, let us examine the precision of the computations by our algorithms simplified in the Toeplitz-like case. We recall that our Toeplitz-like computations can be ultimately reduced to vector convolutions (Propositions 13.1 and 13.4). Thus, we will bound the cost of our computations at the p-adic lifting stage based on the following estimate.

Proposition 14.2 *Given two vectors of dimension n filled with integers lying in the range from 0 to $2^k - 1$, the convolution of these vectors can be computed at the Boolean cost $O_B((\log(kn), kn \log\log(kn))$.*

Proof. The well-known *binary segmentation* techniques (see e.g. [BP94], section 3.9) reduces our convolution problem to the multiplication of two integers lying in the range from 0 to $2^{kn}-1$, and the known algorithms solve this task at the required cost. $\square$

The resulting Boolean cost bounds for performing the p-adic lifting stage will repeat ones of section 12, except that the Boolean (like arithmetic) processor bounds will decrease by factor $n^{\omega-1}/\log n$.

Let us show that this holds also for the Boolean cost of the rest of our computation.

When we approximate the ERD of an input Toeplitz-like matrix, we will effectively reduce the computations to performing FFTs (see Propositions 13.1 and 13.3) and will recall Corollary 3.4.1 of pp. 255–256 of [BP94], which shows a numerically stable implementation of FFT. We also recall that the known algorithms for the computation of the SVD of a matrix are numerically stable (see [GL89/96], [P93]). From these observations, we deduce that we may perform the computations with the same bit-precision (up to a constant factor independent of n), no matter whether we apply our original Algorithm 11.1 for an arbitrary $n \times n$ input matrix or its Toeplitz-like modification. Since in the latter case we use by factor $n^{\omega-1}/\log n$ fewer arithmetic processors, we will also use by factor $n^{\omega-1}/\log n$ fewer Boolean processors, thus replacing n^{ω} for ω of (4.5) by $n \log n$ in the Boolean cost estimates of section 12.

This enables us to extend Theorem 1.1 to arrive at Theorem 1.2. $\square$

Remark 14.1 *Inspection of our algorithms shows immediately that Proposition 14.1 and Theorem 1.2 can be extended to the case where the input matrix A is given with its F-generator of length r, provided that both time and processor bound increase by factor r. It is possilble to confine the cost increase to processor bound (increasing it by factor r^2). The only nontrivial stage is the decrease of the length of F-generators (cf. Proposition 13.8 and Algorithm 13.1). The algorithm supporting Proposition 13.8, however, can be modified by extending the probabilistic techniques of the proof of Theorem 1.1 (this would include, in particular, application of Proposition 2.10 using $n+r$ extra random parameters), whereas Algorithm 13.1 should be replaced by an alternative approach of [PBRZ99].*

Remark 14.2 *It may seem that Theorems 1.1 and 1.2 can be supported by a substatially simpler construction, and simplified construction has indeed been proposed in [R95]. Unfortunately, however, the construction of [R95] has no power for supporting the claimed results. In particular, the construction relies on the two "simplifying" recipes cited in our Remarks 6.1 and 11.1, and each of the recipes invalidates the resulting algorithm (see [P96c] for more details on these and some other of the many mishaps of [R95]). It is instructive, for getting better insight, to discuss two other major gaps of the construction of [R95] and of its analysis presented in [R95]. Both gaps are in area of Toeplitz-like computations, where [R95] becomes particularly prone to serious errors. In [R95], an algorithm of [BA80] is used in order to decrease the length of an F-generator of a matrix A to the level $r =$rank $F(A)$. Unlike our Algorithm 13.1 for the SVD truncation and our algorithm supporting Proposition 13.8, the algorithm of [BA80] only works if $(F(A))^{(r)}$, the $r \times r$ l.p.s of $F(A)$, is nonsingular.*

Furthermore, to support the algorithm of [R95], one must have matrix $(F(A))^{(r)}$ *well-conditioned. Actually, to salvage the algorithm of [R95] at this point, one would have had to use some techniques that are absent from [R95] and are substantially more advanced than ones used in [R95]. Likewise, some techniques are required to prevent the* F*-ranks of the computed approximations to* A^{-1} *from their disturbing growth (from the desired constant level to the level* n*) in less than* $\log n$ *Newton's steps, and then again, such techniques are absent from [R95] and are substantially more advanced than ones used in [R95]. The growth of the* F*-ranks immediately implies the growth by the extra factor* $n^{\omega-1}\log n$ *(for* ω *of (4.5)) of both arithmetic and Boolean processor complexity bounds, versus the ones claimed in [R95].*

15 Discussion.

Our paper leaves major open question of theoretical importance, whether the level of our parallel complexity estimates of Theorem 1.2 for Toeplitz and Toeplitz-like computations can be reached by means of purely algebraic approach, with using no rounding to the closest integers. This question is also of practical interest because the algorithms of this paper involve the exact computation of $\det A$ and, therefore, at some stage require to use the precision of computation of order $\log|\det A|$, which generally means the order of $n\log||A||$, even if we only need the output with a much lower precision. Historically, similar open problem had arisen for computations with general integer matrices, after appearance of [P85], [P87]. In that case (for general integer matrices), the subsequent works of [KP91], [KP92], [P91], [P92] gave us an alternative randomized algebraic solution that involved no rounding. Will this be eventually done also in the Toeplitz-like case or at least in the Toeplitz case?

References

[AGr88] G. S. Ammar, W. G. Gragg, Superfast Solution of Real Positive Definite Toeplitz Systems, *SIAM J. Matrix Anal. Appls.,* 9, 1, 61–76, 1988.

[AHU74] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Massachusetts, 1974.

[B68] E. H. Bareiss, Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination, *Math. Comput.,* 22, 565–578, 1968.

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

[Be68] E. R. Berlekamp, *Algebraic Coding Theory,* McGraw-Hill, New York, 1968.

[Be84] S. Berkowitz, On Computing the Determinant in Small Parallel Time Using a Small Number of Processors, *Inform. Proc. Lett.,* 18, 3, 147–150, 1984.

[BGH82] A. Borodin, J. von zur Gathen, J. Hopcroft, Fast Parallel Matrix and GCD Computation, *Information and Control*, 52, 3, 241–256, 1982.

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

[BK87] A. Bruckstein, T. Kailath, An Inverse Scattering Framework for Several Problems in Signal Processing, *IEEE ASSP Magazine,* 6-20, January 1987.

[BL80] D. Bini, G. Lotti, Stability of Fast Algorithms for Matrix Multiplication, *Numer. Math.,* 36, 63–72, 1980.

[BMP98] D. Bondyfalat, B. Mourrain, V. Y. Pan, Controlled Iterative Methods for Solving Polynomial Systems, *Proc. Annual ACM Intern. Symp. on Symb. and Algebraic Comp. (ISSAC 98)*, 252-259, ACM Press, New York, 1998.

[BP91] D. Bini, V. Y. Pan, Parallel Complexity of Tridiagonal Symmetric Eigenvalue Problem, *Proc. 2nd Ann. ACM-SIAM Symposium on Discrete Algorithms,* 384–393, ACM Press, New York, and SIAM Publications, Philadelphia, Pennsylvania, 1991.

[BP93] D. Bini, V. Y. Pan, Improved Parallel Computation with Toeplitz-like and Hankel-like Matrices, *Proc. of Annual ACM-SIGSAM Intern. Symp. on Symbolic and Agebraic Algorithms (ISSAC'93),* 193-200, ACM Press, New York (1993) and *Linear Algebra and Its Applications,* 188/189, 3-29, 1993.

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

[BT71] W. S. Brown, J. F. Traub, On Euclid's Algorithm and the Theory of Subresultants, *J. of ACM,* 18, 4 , 505–514, 1971.

[BT90] M. Ben-Or, P. Tiwari, Simple Algorithm for Approximating All Roots of a Polynomial with Real Roots, *J. of Complexity,* 6, 417–442, 1990.

[Bun85] J. R. Bunch, Stability of Methods for Solving Toeplitz Systems of Equations, *SIAM J. Sci. Stat. Comput.,* 6, 2, 349-364, 1985.

[C47/48] S. Chandrasekhar, On the Radiactive Equilibrium of a Stellar Atmosphere, *Astrophys. J.,* Pt. XXI, 106, 152-216, 1947; Pt. XXXII, 107, 48-72, 1948.

[C74] R. W. Cottle, Manifestation of the Schur Complement, *Linear Algebra and Its Applics.,* 8, 189-211, 1974.

[Ch85] A. L. Chistov, Fast Parallel Calculation of the Rank of Matrices over a Field of Arbitrary Characteristics, *Proc. FCT 85, Lecture Notes in Computer Science*, 199, 63–69, Springer, Berlin, 1985.

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

[CKL-A87] J. Chun, T. Kailath, H. Lev-Ari, Fast Parallel Algorithm for QR-factorization of Structured Matrices, *SIAM J. Sci. Stat. Comput.,* 8, 6, 899–913, 1987.

[Cs76] L. Csanky, Fast Parallel Matrix Inversion Algorithm, *SIAM J. Comput.,* 5, 4, 618–623, 1976.

[dH87] F. R. de Hoog, On the Solution of Toeplitz Systems, *Linear Algebra Appl.,* 88/89, 123–138, 1987.

[E67] J. Edmonds, Systems of Distinct Representatives and Linear Algebra, *J. Res. Nat. Bur. Stand.,* 71B(4), 241–245, 1967.

[EG88] D. Eppstein, Z. Galil, Parallel Algorithmic Techniques for Combinatorial Computation, *Annual Review of Computer Science,* 3, 233–283, 1988.

[EP97] I. Z. Emiris, V. Y. Pan, The Structure of Sparse Resultant Matrices, *Proc. Annual ACM International Symposium on Symbolic and Algebraic Computation* (ISSAC 97), 189-196, ACM Press, New York, 1997.

[F64] L. Fox, *An Introduction to Numerical Linear Algebra*, Oxford University Press, Oxford, England, 1964.

[G84] J. von zur Gathen, Parallel Algorithms for Algebraic Problems, *SIAM J. Comput.*, 13, 4, 802–824, 1984.

[G86] J. von zur Gathen, Parallel Arithmetic Computations: A Survey, *Proc. Math. Foundation of Computer Science, Lecture Notes in Computer Science*, 233, 93–112, Springer, Berlin, 1986.

[GKO95] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian Elimination with Partial Pivoting for Matrices with Displacement Structure, *Math. of Computation,* 64, 1557-1576, 1995.

[GL89/96] G. H. Golub, C. F. Van Loan, *Matrix Computations,* Johns Hopkins Univ. Press, Baltimore, Maryland, 1989 (second edition), 1996 (third edition).

[H91] S. Haykin, *Adaptive Filter Theory* , Prentice-Hall, Englewood Cliffs, New Jersey, 1991 (second edition).

[H95] G. Heinig, Inversion of Generalized Cauchy Matrices and Other Classes of Structured Matrices, *IMA Volume: Linear Algebra for Signal Processing in Math. and Its Applications,* 69, 95–114, Springer, 1995.

[IR82] K. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory*, Springer, Berlin, 1982.

[J92] J. Jà Jà, *An Introduction to Parallel Algorithms,* Addison-Wesley, Reading, Massachussetts, 1992.

[K74] T. Kailath, A View of Three Decades of Linear Filtering Theory, *IEEE Trans. Inform. Theory,* 20, 146–181, 1974.

[K87] T. Kailath, Signal Processing Applications of Some Moment Problems, *Proc. AMS Symp. in Applied Math.,* 37, 71–100, 1987.

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

[KAGKA89] R. King, M. Ahmadi, R. Gorgui-Naguib, A. Kwabwe, M. Azimi-Sadjadi, *Digital Filtering in One and Two Dimensions: Design and Applications,* Plenum Press, New York, 1989.

[KKM79] T. Kailath, S.-Y. Kung, M. Morf, Displacement Ranks of Matrices and Linear Equations, *J. Math. Anal. Appl.,* 68, 2, 395–407, 1979.

[KLM78] T. Kailath, L. Ljung, M. Morf, A New Approach to the Dertemination of Fredholm Resolvents of Nondisplacement Kernels, in *Topics in Functional Analysis* (I. Gohberg and M. Kac, eds.), 169-184, Academic Press, New York, 1978.

[KP91] E. Kaltofen, V. Y. Pan, Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, *Proc. 3rd Ann. ACM Symp. on Parallel Algorithms and Architectures,* 180-191, ACM Press, New York, 1991.

[KP92] E. Kaltofen, V. Y. Pan, Processor-Efficient Parallel Solution of Linear Systems II. The Positive Characteristic and Singular Cases, *Proc. of 33rd Ann. IEEE Symp. on Foundations of Computer Science,* 714–723, IEEE Computer Society Press, 1992.

[KP94] E. Kaltofen, V. Y. Pan, Parallel Solution of Toeplitz and Toeplitz-like Linear Systems over Fields of Small Positive Characteristic, *Proc. First Intern. Symposium on Parallel Symbolic Computation (PASCO '94)*, Linz (edited by Hoon Hong), Austria (Sept. 1994), *Lecture Notes in Computing,* **5**, 225-233, World Scientific Publishing Company, Singapore, 1994.

[KR90] R. Karp, V. Ramachandran, A Survey of Parallel Algorithms for Shared Memory Machines, *Handbook for Theoretical Computer Science* (J. van Leeuwen Editor), 869-941, North Holland, Amsterdam, 1990.

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

[KVM78] T. Kailath, A. Viera, M. Morf, Inverses of Toeplitz Operators, Innovations, and Orthogonal Polynomials, *SIAM Rev.,* 20, 1, 106–119, 1978.

[L-AK84] H. Lev-Ari, T. Kailath, Lattice Filter Parametrization and Modelling of Nonstationary Processes, *IEEE Trans. on Information Theory,* IT-30, 1, 2–16, 1984.

[L-AKC84] H. Lev-Ari, T. Kailath, J. Cioffi, Least Squares Adaptive Lattice and Transversal Filters; a Unified Geometrical Theory, *IEEE Trans. on Information Theory,* IT-30, 222–236, 1984.

[Le92] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees & Hypercubes,* Morgan Kaufmann Publishers, San Mateo, California, 1992.

[LRT79] R. J. Lipton, D. Rose, R. E. Tarjan, *Generalized Nested Dissection*, *SIAM J. Numer. Anal.,* **16, 2**, 346-358, 1979.

[Ma75] J. Makhoul, Linear Prediciton: A Tutorial Review, *Proc. IEEE,* 63, 4, 561–580, 1975.

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

[Morf74] M. Morf, Fast Algorithms for Multivariable Systems, Ph.D. Thesis, *Stanford University*, Stanford, California, 1974.

[Morf80] M. Morf, Doubling Algorithms for Toeplitz and Related Equations, *Proc. IEEE Internat. Conf. on ASSP*, 954–959, IEEE Computer Society Press, 1980.

[MP98] B. Mourrain, V. Y. Pan, Asymptotic Acceleration of Solving Multivariate Polynomial Systems of Equations, *ACM Symp. on Theory of Computing,* 488-496, ACM Press, New York, 1998.

[MRK88] G. L. Miller, V. Ramachandran, E. Kaltofen, Efficient Parallel Evaluation of Straight-line Code and Arithmetic Circuits, *SIAM J. Comput.*, 17, 4 , 687-695, 1988.

[Mu81] B. R. Musicus, Levinson and Fast Choleski Algorithms for Toeplitz and Almost Toeplitz Matrices, Internal Report, *Lab. of Electronics, M.I.T.,* Cambridge, Massachusetts, 1981.

[OP98] V. Olshevsky, V. Y. Pan, A Unified Superfast Algorithm for Boundary Rational Tangential Interpolation Problem and for Inversion and Factoriztion of Dense Structured Matrices, *Proc. 39th Annual IEEE Symposium on Foundations of Computer Science*, 192-201, IEEE Computer Society Press, 1998.

[P85] V. Y. Pan, Fast and Efficient Parallel Algorithms for the Exact Inversion of Integer Matrices, *Proc. 5th Ann. Conf. on Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science,* 206, 504–521, Springer Verlag, 1985.

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

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

[P91] V. Y. Pan, Complexity of Algorithms for Linear Systems of Equations, in *Computer Algorithms for Solving Linear Algebraic Equations (The State of the Art),* edited by E. Spedicato, *NATO ASI Series, Series F: Computer and Systems Sciences,* 77, 27-56, Springer, Berlin, 1991.

[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.

[P92a] V. Y. Pan, Complexity of Computations with Matrices and Polynomials, *SIAM Review,* 34, 2, 225–262, 1992.

[P92b] V. Y. Pan, Parallel Solution of Toeplitz-like Linear Systems, *J. of Complexity,* 8, 1–21, 1992.

[P93] V. Y. Pan, Decreasing the Displacement Rank of a Matrix, *SIAM J. on Matrix Analysis,* 14, 1, 118–121, 1993.

[P93a] V. Y. Pan, Concurrent Iterative Algorithm for Toeplitz-like Linear Systems, *IEEE Trans. on Parallel and Distributed Systems,* 4, 5, 592–600, 1993.

[P93b] V. Y. Pan, *Parallel Solution of Sparse Linear and Path Systems, in Synthesis of Parallel Algorithms* (J.H. Reif editor), Chapter 14, pp. 621–678, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[P95] V. Y. Pan, Optimal (up to Polylog Factors) Sequential and Parallel Algorithms for Approximating Complex Polynomial Zeros, *Proc. 27th Ann. ACM Symposium on Theory of Computing,* 741–750, ACM Press, New York, 1995.

[P96] V. Y. Pan, A New Approach to Parallel Computation of Polynomial GCD and to Related Parallel Computations over Abstract Fields, *Proc. Seventh Annual ACM-SIAM Symposium on Discrete Algorithms,* 518–527, ACM Press, New York, and SIAM Publications, Philadelphia, PA, 1996.

[P96a] V. Y. Pan, Optimal and Nearly Optimal Algorithms for Approximating Polynomial Zeros, *Computers & Math. (with Applications),* 31, 12, 97-138, 1996.

[P96b] V. Y. Pan, Parallel Computation of Polynomial GCD and Some Related Parallel Computations over Abstract Fields, *Theor. Computer Science*, 162, 2, 173-223, 1996.

[P96c] V. Y. Pan, Effective Parallel Computations with Toeplitz and Toeplitz-like Matrices Filled with Integers, *LAM: Proc. of AMS Summer Seminar on Math. of Numerical Analysis: Real Number Algorithms*, (Park City, Utah, July-August 1995), (J. Renegar, M. Shub, S. Smale editors), *Lectures in Applied Math.,* 32, 593-641, Amer. Math. Society Press, Providence, Rhode Island, 1996.

[P97] V. Y. Pan, Solving a Polynomial Equation: Some History and Recent Progress, *SIAM Review*, 39, 2, 187-220, 1997.

[PBRZ99] V. Y. Pan, S. Branham, R. Rosholt, A. Zheng, Newton's Iteration for Structured Matrices and Linear Systems of Equations, accepted for SIAM volume on *Fast Reliable Algorithms for Matrices with Structure,* SIAM Publications, Philadephia, 1999 (to appear).

[PP95] V. Y. Pan, F. P. Preparata, Work-Preserving Speed-up of Parallel Matrix Computations, *SIAM J. on Computing,* 24, 4, 811–821, 1995.

[PR89] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Dense Linear Systems, *Compuaters & Mathematics (with Applications)*, 17, 11, 1481-1491, 1989.

[PR91] V. Y. Pan, J. Reif, The Parallel Computation of the Minimum Cost Paths in Graphs by Stream Contraction, *Information Processing Letters,* 40, 79–83, 1991.

[PR93] V. Y. Pan, J. Reif, Fast and Efficient Parallel Solution of Sparse Linear Systems, *SIAM J. on Computing,* 22, 6, 1227-1250, 1993.

[PSLT93] V. Y. Pan, A. Sadikou, E. Landowne, O. Tiga, A New Approach to Fast Polynomial Interpolation and Multipoint Evaluation, *Computers & Math. (with Applications),* 25, 9, 25–30, 1993.

[PZHY97] V. Y. Pan, A. Zheng, X. Huang, Y. Yu, Fast Multipoints Polynomial Evaluation and Interpolarion via Computations with Structured Matrices, *Annals of Numer. Math.*, 4, 483-510, 1997.

[Q94] M.J. Quinn, *Parallel Computing: Theory and Practice,* McGraw-Hill, New York, 1994.

[R93] J. Reif, An $O(n \log^3 n)$ Algorithm for the Real Root Problem, *Proc. of 34rd Ann. IEEE Symp. on Foundations of Computer Science,* 626–635, IEEE Computer Society Press, 1993.

[R95] J. Reif, Work Efficient Parallel Solution of Toeplitz Systems and Polynomial GCD, *Proc. 27th Ann. ACM Symp. on Theory of Computing,* 751–761, ACM Press, NY, 1995.

[RT90] J. Reif, S.R. Tate, Optimal Size Division Circuits, *SIAM J. Comput.,* 19, 5, 912–925, 1990.

[St69] V. Strassen, Gaussian Elimination Is Not Optimal, *Numer. Math.,* 13, 354–356, 1969.

[Ste94] W. F. Stewart, *Introduction to the Numerical Solution of Markov Chains,* Princeton University Press, New Jersey, 1994.

[VSBR83] L. Valiant, S. Skyum, S. Berkowitz, C. Rackoff, Fast Parallel Computation of Polynomials Using Few Processors, *SIAM J. Comput.,* 12, 4, 641-644, 1983.

[W65] J. H. Wilkinson, *The Algebraic Eigenvalue Problem,* Clarendon Press, Oxford, 1965.