MATHEMATICS OF COMPUTATION, to appear (January 1995).

ANALYSIS OF COPPERSMITH'S BLOCK WIEDEMANN ALGORITHM FOR THE PARALLEL SOLUTION OF SPARSE LINEAR SYSTEMS*

ERICH KALTOFEN

ABSTRACT. By using projections by a block of vectors in place of a single vector it is possible to parallelize the outer loop of iterative methods for solving sparse linear systems. We analyze such a scheme proposed by Coppersmith for Wiedemann's coordinate recurrence algorithm, which is based in part on the Krylov subspace approach. We prove that by use of certain randomizations on the input system the parallel speed up is roughly by the number of vectors in the blocks when using as many processors. Our analysis is valid for fields of entries that have sufficiently large cardinality. Our analysis also deals with an arising subproblem of solving a singular block Toeplitz system by use of the theory of Toeplitz-like matrices.

1 INTRODUCTION

The problem of solving larger unstructured sparse linear systems with exact arithmetic arises in computer algebra and number theory. For instance, sieve-based integer factoring algorithms can lead to systems of over 200,000 equations and variables with over 11 million nonzero entries that need to be solved over the Galois field of 2 elements (Lenstra et al. 1993). One way problems of such a large size are tackled is by structured Gaussian elimination that prevents fill-in at the early stages, resulting in much denser systems of smaller dimensions (see, e.g., LaMacchia and Odlyzko 1991 and Pomerance and Smith 1992), in the cited example about 72,000. An alternate approach is to use iterative methods. LaMacchia and Odlyzko (1991) use an exact arithmetic version of the conjugate gradient algorithm, and Wiedemann (1986) bases his method on finding linear relations in Krylov subspaces. Both approaches use the black-box model for sparse matrices, that is, they require a linear number of matrix times vector products and quadratically many field operations, both measured in the dimension of the system. Therefore, the iterative methods do not depend on certain structural properties of the sparse matrix, in contrast to structured Gaussian elimination.

1991 *Mathematics Subject Classification.* Primary 15A06; Secondary 65F10.

Key words and phrases. Sparse linear systems, parallel and distributed computation, randomized algorithms, finite fields, exact arithmetic, block Toeplitz matrices, Krylov subspaces.

*This material is based on work supported in part by the National Science Foundation under Grant No. CCR-90-06077 and under Grant No. CRR-93-19776.

Clearly, the iterative methods are subject to parallelization when considering the steps inside the outer iteration loop, i.e., the matrix times vector products (see Schwabe et al. 1993).

A more difficult task is the parallelization of the outer iterative loop in any of these algorithms. It seems natural to consider blocks of vectors instead of single vectors. Arithmetic is then performed on small square matrices in place of the individual field entries, and the length of the iteration should become a fraction of the original length, where the reduction should be proportional to the number of single column vectors in each block. Since the components of the blocks can be processed in parallel, the arithmetic necessary in each iteration can be carried out on a parallel system as fast as the unblocked iteration on a sequential computer. Therefore, the overall parallel speed-up should be roughly proportional to the number of vectors in each block. Coppersmith (1994) suggests such a modification to the Wiedemann approach. He presents a complete algorithm, a heuristic mathematical argument, and experimental results that the blocking has the desired effect. In this paper we will give a full analysis for a variant of Coppersmith's method in the case that the field of entries has sufficiently many elements. The specifics are as follows.

Wiedemann's (1986) algorithm for computing the N dimensional solution vector of a possibly singular system of N linear equations over a finite field $\mathbb{K}$ requires $\leq 3N$ multiplications of the coefficient matrix B by vectors and $O(N^2 \log N)$ additional arithmetic operations in the coefficient field. Only $O(N)$ additional field elements need to be stored. The method is randomized and computes first the sequence of field elements

$$a^{(i)} = u^{\mathrm{tr}} B^i v \in \mathbb{K} \quad \text{for } 0 \leq i \leq 2N-1,$$

where u and possibly v are vectors with random entries from $\mathbb{K}$. The key property is that this sequence is generated by a linear recursion that with high probability corresponds to the minimum polynomial of B, and which can be computed by the Berlekamp/Massey algorithm.

Coppersmith (1994) proposes to use simultaneously m random vectors for u and n random vectors for v. The sequence now is a sequence of $m \times n$ matrices

$$\boldsymbol{a}^{(i)} = \boldsymbol{x}^{\mathrm{tr}} B^i \boldsymbol{y} \in \mathbb{K}^{m \times n} \quad \text{where } \boldsymbol{x}^{\mathrm{tr}} \in \mathbb{K}^{m \times N} \text{ and } \boldsymbol{y} \in \mathbb{K}^{N \times n}.$$

Clearly, the individual entries in $\boldsymbol{a}^{(i)}$ can be computed independently and in parallel. Coppersmith then cleverly generalizes the Berlekamp/Massey algorithm needed to compute a linear recurrence that generates this sequence and observed experimentally that over the Galois field with 2 elements the linear recurrence is determined by the first $N/m + N/n + O(1)$ matrices $\boldsymbol{a}^{(i)}$. Thus the algorithm, when executed in a parallel/distributed setting performs much faster.

We prove that if the coefficient matrix satisfies a certain condition regarding the degree of its minimum polynomial then the algorithm is guaranteed to compute a solution with high probability. Moreover, that condition can be obtained by pre-conditioning the coefficient matrix by pre- and postmultiplying it with certain random structured matrices (Wiedemann 1986; Kaltofen and Saunders 1991). In all our estimates, we suppose that the field of entries is of sufficient cardinality. Furthermore, we give an alternate approach to computing the linear generator for the sequence of $\leq N/m + N/n + 2n/m + 1$ matrices $\boldsymbol{a}^{(i)}$. The problem can be cast as a homogeneous linear system of block Toeplitz structure, and the theory of Toeplitz-like matrices (Kailath et al. 1979) can be applied. We show that by randomization such singular systems can be solved by generalizations of the Levinson and Durbin algorithms (see Gohberg et al. 1986) or by generalizations of doubling methods that achieve even faster asymptotic speeds by use of FFT-based polynomial multiplication (Bitmead and Anderson 1980; Morf 1980). By use of block Toeplitz solvers we are also able to probabilistically and efficiently compute the rank of a singular sparse matrix. Lastly, we show that all of our algorithms can be carried out on parallel computers.

The key idea in our analysis is the observation that generically, i.e., when the projection blocks $\boldsymbol{x}$ and $\boldsymbol{y}$ are symbolic, the block method can be specialized to the original Wiedemann algorithm. From that specialization one then can prove that certain necessary rank conditions must hold generically. By the commonly used Schwartz/Zippel lemma those rank conditions will thus hold with high probability for random blocks.

Our results also impact the sequential complexity of sparse linear system solving. Suppose, e.g., that B is non-singular, and that $\epsilon > 0$. Using blocking, one can find a solution vector $x = B^{-1}b$, where $b \in \mathbb{K}^N$, by $(1+\epsilon)N + O(1)$ multiplications of B times vectors, and $O(N^2 \log N \operatorname{loglog} N)$ arithmetic operations in $\mathbb{K}$, needing $O(N)$ additional storage for field elements. The algorithm chooses $O(N)$ random field elements and is successful with probability $1 - O(N^2)/(\operatorname{card}(\mathbb{K}) - 1)$, where $\operatorname{card}(\mathbb{K})$ denotes the cardinality of the field $\mathbb{K}$. Here the constants implied by the big-O notation grow with $1/\epsilon$. Note that it had been an open problem how to speed the Wiedemann method such that no more than cN, where $c < 2$, matrix times vector products would be needed.

Notation

We write $\mathbb{K}^N$ for the set of column vectors over $\mathbb{K}$, and $\mathbf{0}^N$ for the N dimensional zero column vector; $\mathbf{0}^{N\times M}$ is the $N \times M$ zero matrix. By I_N we denote the $N \times N$ identity matrix, and by $e_i^{(N)}$ we denote the N dimensional i^{th} unit vector, that is a vector with 1 in the i^{th} coordinate and 0 everywhere else. Matrices are indicated by capital letters, e.g., A, B, G, V, W, and by bold lower case letters, e.g., $\boldsymbol{a}$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, while single column vectors are written as lower case symbols, e.g., b, v, w, etc. Matrices written in script font or bold lower case Greek, such as $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{A}$, $\mathcal{K}$, $\mathcal{T}$, $\boldsymbol{\alpha}$, have entries that contain indeterminates. Vector and matrix transposition is indicated by superscript $^{\text{tr}}$. Individual entries in matrices or vectors are selected by square brackets; for instance, $A[1,1]$ denotes the left upper entry in the matrix A. We indicate a block matrix whose entries are matrices or vectors by vertical and horizontal strokes, such as $\left[\begin{array}{c|c} \boldsymbol{a} & \boldsymbol{a}'' \\ \hline \boldsymbol{a}' & \boldsymbol{a}''' \end{array}\right]$.

2. Randomizations

Several randomization techniques for linear and polynomial algebra have been advanced in the past, which we will make use of below. We collect the needed results in this section.

Theorem 1. *Let $F(x_1, \ldots, x_\nu)$ be a non-zero ν-variate polynomial over an integral domain and let S be a subset of that domain. Then the probability of avoiding the zeros of F while evaluating in S is bounded as follows:*

$$\operatorname{Prob}\Big(F(s_1, \ldots, s_\nu) \neq 0 \mid s_j \in S \text{ for all } 1 \leq j \leq \nu\Big) \geq 1 - \frac{\deg F}{\operatorname{card} S}.$$

Here $\deg(F)$ denotes the total degree of F, i.e., the maximum of all term exponent sums and $\operatorname{card}(S)$ denotes the cardinality of S. The theorem in the above form was given by Schwartz (1980). Somewhat different versions are due to DeMillo and Lipton (1978) and Zippel (1979; 1993, §12).

In the following, we consider an $N \times N$ singular matrix A with entries from a field. By A_i we shall denote the leading $i \times i$ principal submatrix, i.e., the $i \times i$ submatrix located in the left upper corner of A, where $1 \leq i \leq N$. We say that A has *generic rank profile* (cf. Delsarte et al. 1985) if A_j is non-singular for all $1 \leq j \leq \operatorname{rank} A$. In such a case, no search for non-zero pivot elements would have to be performed during triangularization by Gaussian elimination. The following is Theorem 2 of Kaltofen and Saunders (1991).

Theorem 2. *For an $N \times N$ matrix A of rank r consider the matrix product $\widetilde{A} = VAW$ with*

$$V = \begin{bmatrix} 1 & v_2 & v_3 & \ldots & v_N \\ & 1 & v_2 & \ldots & v_{N-1} \\ & & 1 & \ddots & \vdots \\ & & & \ddots & v_2 \\ & \mathbf{0} & & & 1 \end{bmatrix} \quad \text{and} \quad W = \begin{bmatrix} 1 & & & & \\ w_2 & 1 & & \mathbf{0} & \\ w_3 & w_2 & 1 & & \\ \vdots & & \ddots & \ddots & \\ w_N & w_{N-1} & \ldots & w_2 & 1 \end{bmatrix},$$

where the elements of the unit upper triangular Toeplitz matrix V and the elements of the unit lower triangular Toeplitz matrix W are randomly and uniformly selected from a subset S of the field of entries. Then $\widetilde{A}$ has generic rank profile with probability no less than $1 - r(r+1)/\operatorname{card}(S)$. Furthermore, if A is non-singular, the product AW has generic rank profile with probability at least $1 - \frac{1}{2}(N-1)N/\operatorname{card}(S)$.

As we will see later, it is often useful to work with the matrix $\widetilde{A}$, which has generic rank profile, instead of with A. The following technique from Kaltofen and Saunders (1991, Theorem 4) shows how to find random solutions to linear systems.

Proposition 1. *Let A be an $N \times N$ matrix of rank r, and suppose that the leading $r \times r$ principal submatrix A_r is non-singular. Then for a random column vector y with coordinates from the field of entries, the vector*

$$x = \left[\begin{array}{c} A_r^{-1} b' \\ \hline \mathbf{0}^{N-r} \end{array}\right] - y,$$

is a random solution to $Ax = b$, where the vector b' consists of the first r coordinates of $b + Ay$.

The rank of a matrix can be related to the degree of the minimum polynomial of a certain product matrix by the following theorem (see Kaltofen and Saunders 1991, Lemma 2).

Theorem 3. *Let A be an $N \times N$ singular matrix that has generic rank profile, let r be the rank of A, and let G be a diagonal matrix where each entry on the diagonal has been uniformly selected from a subset S of the field of entries with $0 \notin S$. Then the minimum polynomial of AG has degree $1 + r$ with probability at least $1 - \frac{1}{2}r(r+1)/\mathrm{card}(S)$.*

3. Linearly generated sequences

We now discuss some basic facts about linearly generated sequences of elements in a vector space V over the field $\mathbb{K}$. A sequence

$$\{a_i\}_{i=0}^{\infty}, \quad \text{where } a_i \in \mathsf{V},$$

is *linearly generated* over $\mathbb{K}$ if there exist $c_0, c_1, \ldots, c_N \in \mathbb{K}$, $N \geq 0$, $c_L \neq 0$ for some L with $0 \leq L \leq N$, such that

$$\forall j \geq 0\colon c_0 a_j + \cdots + c_N a_{j+N} = 0. \tag{1}$$

The polynomial $c_0 + c_1\lambda + \cdots + c_N\lambda^N$ is called a *generating polynomial* for $\{a_i\}_{i=0}^{\infty}$. The set of all generating polynomials for $\{a_i\}_{i=0}^{\infty}$ together with the zero polynomial forms an ideal in $\mathbb{K}[\lambda]$. The unique polynomial generating that ideal, normalized to have leading coefficient 1, is called the *minimum polynomial* of a linearly generated sequence $\{a_i\}_{i=0}^{\infty}$. Every generating polynomial is divisible by the minimum polynomial.

Let W be also a vector space over $\mathbb{K}$, and let $\Phi\colon \mathsf{V} \longrightarrow \mathsf{W}$ be a linear map from V to W. Then the sequence $\{\Phi(a_i)\}_{i=0}^{\infty}$ is also linearly generated by a minimum polynomial that divides the minimum generating polynomial of $\{a_i\}_{i=0}^{\infty}$. Let $B \in \mathbb{K}^{N\times N}$ be a square matrix over a field. The sequence of $N \times N$ matrices $\{B^i\}_{i=0}^{\infty}$ is linearly generated, and its minimum polynomial is the minimum polynomial of B, which will be denoted by f^B. For any column vector $b \in \mathbb{K}^N$ the vector sequence $\{B^i b\}_{i=0}^{\infty}$, where $B^i b \in \mathbb{K}^N$, is also linearly generated by f^B. However, its minimum polynomial, denoted by $f^{B,b}$, can be a proper divisor of f^B. For any row vector $u^{\mathrm{tr}} \in \mathbb{K}^{1\times N}$ the sequence $\{u^{\mathrm{tr}} B^i b\}_{i=0}^{\infty}$, where $u^{\mathrm{tr}} B^i b \in \mathbb{K}$, is linearly generated as well, and its minimum polynomial, denoted by $f_u^{B,b}$, is again a divisor of $f^{B,b}$.

Wiedemann's method is based on the fact that for random vectors u and b one gets $f_u^{B,b} = f^B$ with probability bounded away from 0. Below we shall give a new proof of this fact in the restricted case where the field has sufficiently many elements. But first we need to establish equality of the minimum polynomials for "generic" projections.

Proposition 2. *Let $B \in \mathbb{K}^{N\times N}$ and let*

$$\upsilon = \begin{bmatrix} \upsilon_1 \\ \vdots \\ \upsilon_N \end{bmatrix} \quad \text{and} \quad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}$$

be vectors whose entries are indeterminates. Then the sequence $\{\upsilon^{\mathrm{tr}} B^i \beta\}_{i=0}^{\infty}$ over the transcendental extension field $\mathbb{K}(\upsilon_1, \ldots, \upsilon_N, \beta_1, \ldots, \beta_N)$ is linearly generated by f^B.

Proof. Since $f_v^{B,\beta}$ is a monic factor of f^B, the coefficients of $f_v^{B,\beta}$ are all elements in the ground field $\mathbb{K}$. Let e_ν denote the ν^{th} unit vector. By specializing v to e_j and β to e_k, the sequence specializes to $\{(B^i)[j,k]\}_{i=0}^\infty$, which is also linearly generated by $f_v^{B,\beta}$. Therefore $f_v^{B,\beta}$ linearly generates $\{B^i\}_{i=0}^\infty$ and must be equal to f^B. ⊠

The minimum generator for a sequence $\{a_i\}_{i=0}^\infty$ of field elements $a_i \in \mathbb{K}$ can be computed by the Berlekamp/Massey algorithm (Massey 1969). This algorithm will determine the minimum polynomial $f^{(\min)}$ of such a sequence from the first $2M$ elements, where $M = \deg(f^{(\min)})$. If more elements are given, the computed minimum polynomial cannot change. Therefore, we have the following lemma.

Theorem 4. *Suppose $\{a_i\}_{i=0}^\infty$, where $a_i \in \mathbb{K}$, is linearly generated by the minimum polynomial $f^{(\min)}$. Let $M = \deg(f^{(\min)})$ and let $M' \geq M$. Suppose a polynomial g with $\deg(g) \leq M'$ linearly generates a sequence*

$$\{a_0, a_1, \ldots, a_{2M'-1}, a'_{2M'}, a'_{2M'+1}, \ldots\}$$

whose first $2M'$ elements agree with $\{a_i\}_{i=0}^\infty$. Then $a'_i = a_i$ for all $i \geq 2M'$ and g is a polynomial multiple of $f^{(\min)}$.

Now reconsider the linear equations arising from (1) when $a_i \in \mathbb{K}$. Setting $c_N = 1$, we can solve the first N equations for c_k, where $0 \leq k < N$. By Theorem 4 any monic polynomial whose coefficient vector is a solution to this system must linearly generate the entire sequence. Therefore, the arising $N \times N$ coefficient matrix, which is a Toeplitz matrix, is non-singular if N is exactly the degree of the minimum polynomial, and is singular if N is larger than that degree, because then there are more than one such polynomial. We finally can establish the following fact.

Theorem 5. *Let $B \in \mathbb{K}^{N\times N}$, let $b \in \mathbb{K}^N$ and suppose that the coordinates of the vector u are selected uniformly randomly from a subset S of $\mathbb{K}$. Then* $\mathrm{Prob}(f^{B,b} = f_u^{B,b}) \geq 1 - \deg(f^{B,b})/\mathrm{card}(S) \geq 1 - N/\mathrm{card}(S)$.

Proof. Let v be the vector with the indeterminate coordinates $v_1, \ldots, v_N$. As in Proposition 2 we have $f^{B,b} = f_v^{B,b}$. Therefore, for $M = \deg(f^{B,b})$ the $M \times M$ coefficient matrix arising from the linear recurrence given by the minimum polynomial $f_v^{B,b}$ of $\{v^{\mathrm{tr}} B^i b\}_{i=0}^\infty$ is a non-singular Toeplitz matrix over $\mathbb{K}(v_1, \ldots, v_N)$. Let Δ be the determinant of that matrix. Then for any vector $u \in \mathbb{K}^N$ with $\Delta(u_1, \ldots, u_N) \neq 0$ the sequence $\{u^{\mathrm{tr}} B^i b\}_{i=0}^\infty$ cannot be generated by a smaller minimum polynomial, because otherwise the corresponding coefficient matrix would have to be singular as stated just before. By Theorem 1 a zero of Δ is avoided with probability no less than $1 - \deg(\Delta)/\mathrm{card}(S) \geq 1 - M/\mathrm{card}(S)$. ⊠

4. Coppersmith's block Wiedemann algorithm

In order to prepare for later discussion, we first give a particular variant of Wiedemann's (1986, §III, first paragraph) coordinate recurrence method for solving a homogeneous linear system. This variant already accounts for some changes necessitated by the later block version. Let $B \in \mathbb{K}^{N\times N}$ be a singular matrix, where $\mathbb{K}$ is a finite field; we seek a non-zero vector $w \in \mathbb{K}^N$ such that $Bw = \mathbf{0}$.

Algorithm *Homogeneous Wiedemann*

Step W1: *Pick random vectors $u \in \mathbb{K}^N$ and $v \in \mathbb{K}^N$. For any integers $M' \geq M \geq N$, compute*

$$b = Bv, \quad a^{(i)} = u^{\mathrm{tr}} B^i b, \quad 0 \leq i \leq M + M' - 1.$$

(The letters u and b now agree with the ones in Wiedemann's paper.) This requires at least $2M$ multiplications of B by vectors.

Step W2: *Compute a non-zero solution to the linear homogeneous $M' \times (M+1)$ Toeplitz system*

$$\begin{bmatrix} a^{(M)} & \dots & a^{(1)} & a^{(0)} \\ a^{(M+1)} & a^{(M)} & a^{(2)} & a^{(1)} \\ \vdots & \ddots & & \vdots \\ a^{(M+M'-1)} & \dots & & a^{(M'-1)} \end{bmatrix} \begin{bmatrix} c^{(M)} \\ c^{(M-1)} \\ \vdots \\ c^{(0)} \end{bmatrix} = \mathbf{0}^{M'}.$$

Define the generating polynomial

$$f(\lambda) = c^{(L)}\lambda^L + c^{(L+1)}\lambda^{L+1} + \cdots + c^{(M)}\lambda^{(M)},$$

where $c^{(\ell)} = 0$ for all $0 \leq \ell < L \leq M$ and $c^{(L)} \neq 0$. Such a polynomial can be determined, e.g., by the Berlekamp/Massey algorithm, which then requires, for $M' = M = N$, $O(N^2)$ arithmetic operations in $\mathbb{K}$. Here we introduce unnecessary generality for the later analysis of the block Wiedemann method. Note that

$$u^{\mathrm{tr}} B^j f(B) b = 0 \quad \text{for all } 0 \leq j \leq M - 1,$$

which implies by Theorem 4 that $f(\lambda)$ is a polynomial multiple of $f_u^{B,b}(\lambda)$. Furthermore, by Theorem 5 with probability no less than $1 - N/\mathrm{card}(\mathbb{K})$, $f(\lambda)$ is a polynomial multiple of the polynomial $f^{B,b}(\lambda)$, i.e.,

$$c^{(L)} B^L b + c^{(L+1)} B^{L+1} b + \cdots + c^{(M)} B^M b = \mathbf{0}. \tag{2}$$

Step W3: *Compute*

$$\widehat{w} = c^{(L)} v + c^{(L+1)} Bv + \cdots + c^{(M)} B^{M-L} v.$$

This requires at most $M - L$ additional multiplications of B times a vector. One may argue as follows that $\widehat{w} \neq \mathbf{0}^N$ with probability at least $1 - 1/\mathrm{card}(\mathbb{K})$ (Coppersmith 1994): for $v' = v + w_0$, where $w_0 \in \mathrm{kernel}(B)$, the vector $b = Bv'$ and hence the sequence $a^{(i)}$ does not change. However,

$$\begin{aligned} \widehat{w}' &= c^{(L)} v' + c^{(L+1)} Bv' + \cdots + c^{(M)} B^{M-L} v' \\ &= \widehat{w} + c^{(L)} w_0. \end{aligned}$$

Therefore in the set of vectors $v + \mathrm{kernel}(B)$, at most one vector can produce $\widehat{w}' = \mathbf{0}$. Note that the solution $c^{(0)}, \ldots, c^{(M)}$ is computed without any information on w_0.

Suppose now that $\widehat{w} \neq \mathbf{0}^N$. *Finally, determine the first integer i such that $B^i\widehat{w} = \mathbf{0}^N$ and return $w = B^{i-1}\widehat{w}$.* By (2), this should happen, with high probability, for an integer $i \leq L+1$. At most $L+1$ more multiplications of B by a vector are required. $\square$

Let $m, n < N$. Coppersmith's (1994) block version essentially uses

$$\begin{array}{lll} \boldsymbol{x}^{\mathrm{tr}} \in \mathbb{K}^{m\times N} & \text{in place of} & u^{\mathrm{tr}}, \\ \boldsymbol{z} \in \mathbb{K}^{N\times n} & \text{in place of} & v, \quad \text{and} \\ \boldsymbol{y} = B\boldsymbol{z} \in \mathbb{K}^{N\times n} & \text{in place of} & b = Bv. \end{array}$$

(The letters B, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$, agree with the ones in Coppersmith's paper; we use bold type to indicate their block nature.) Thus the sequence consists of the $m \times n$ matrices

$$\boldsymbol{a}^{(i)} = \boldsymbol{x}^{\mathrm{tr}} B^i \boldsymbol{y} \in \mathbb{K}^{m\times n}, \quad 0 \leq i.$$

(Coppersmith further transposes these matrices.) The main point is that a non-trivial linear dependence of the type (2) can be found from roughly $N/m + N/n$ sequence elements $\boldsymbol{a}^{(i)}$. A brief description of a variant of the block Wiedemann algorithm follows:

Algorithm *Block Wiedemann*

Step C1: *Pick random vectors $x_1, \ldots, x_m$, $z_1, \ldots, z_n \in \mathbb{K}^N$.* Let

$$\boldsymbol{x}^{\mathrm{tr}} = \begin{bmatrix} x_1^{\mathrm{tr}} \\ \hline \vdots \\ \hline x_m^{\mathrm{tr}} \end{bmatrix}, \qquad \boldsymbol{y} = B \cdot [\, z_1 \mid \ldots \mid z_n \,].$$

Compute

$$\boldsymbol{a}^{(i)} = \boldsymbol{x}^{\mathrm{tr}} B^i \boldsymbol{y}, \quad \textit{for all } 0 \leq i < \frac{N}{m} + \frac{N}{n} + \frac{2n}{m} + 1.$$

This requires less than

$$\left(1 + \frac{n}{m}\right) N + \frac{2n^2}{m} + 2n \tag{3}$$

multiplications of B times a vector. However, for every y_ν, the ν^{th} columns of the sequence matrices $\boldsymbol{a}^{(i)}$, namely $\boldsymbol{x}^{\mathrm{tr}} B^i y_\nu$, can be computed simultaneously, yielding a coarse-grain parallelization. Alternately, one may for each i perform the products $B \cdot (B^{i-1} y_\nu)$ in parallel, as Coppersmith does, which is finer grain and requires synchronization for each i. For instance, if the matrix is embedded on the distributed memory of a processor network (Schwabe et al. 1993), the entire block $B^{i-1}\boldsymbol{y}$ can be communicated through the network. Note that computing the products $\boldsymbol{x}^{\mathrm{tr}} \cdot (B^i y_\nu)$ for all ν and i requires some additional $O(\,(m+n)\,N^2\,)$ arithmetic operations in $\mathbb{K}$, if done sequentially.

Step C2: *Let $D = \lceil N/n \rceil$, $S = n(D+1)$, $E = \lceil S/m \rceil$, and let $R = mE$. Compute a non-zero solution to the linear homogeneous $R \times S$ linear system (of block Toeplitz structure)*

$$\left[\begin{array}{c|c|c|c|c} \boldsymbol{a}^{(D)} & \dots & & \boldsymbol{a}^{(1)} & \boldsymbol{a}^{(0)} \\ \hline \boldsymbol{a}^{(D+1)} & \boldsymbol{a}^{(D)} & & \boldsymbol{a}^{(2)} & \boldsymbol{a}^{(1)} \\ \hline \vdots & & \ddots & & \vdots \\ \hline \boldsymbol{a}^{(D+E-1)} & \dots & & & \boldsymbol{a}^{(E-1)} \end{array}\right] \left[\begin{array}{c} c^{(D)} \\ \hline c^{(D-1)} \\ \hline \vdots \\ \hline c^{(0)} \end{array}\right] = \mathbf{0}^R, \tag{4}$$

where $c^{(i)} \in \mathbb{K}^n$ for all $0 \le i \le D$. The dimensions are bounded as follows: $N + n \le S < N + 2n$, $S \le R$, and $E < N/m + 2n/m + 1$. Therefore we have

$$D + E < \frac{N}{n} + 1 + \frac{N}{m} + \frac{2n}{m} + 1,$$

which determines the length of the sequence $\boldsymbol{a}^{(i)}$. Define the generating polynomial with (right side) vector coefficients

$$f(\lambda) = \lambda^L \boldsymbol{y} c^{(L)} + \lambda^{L+1} \boldsymbol{y} c^{(L+1)} + \dots + \lambda^D \boldsymbol{y} c^{(D)},$$

where $c^{(\ell)} = \mathbf{0}^n$ for all $0 \le \ell < L \le D$ and $c^{(L)} \ne \mathbf{0}^n$. Coppersmith in his paper computes such a non-zero vector polynomial by his generalization of the Berlekamp/Massey algorithm to polynomials with matrix coefficients. In any case, we have $\boldsymbol{x}^{\mathrm{tr}} B^j f(B) = \mathbf{0}^m$ for all $0 \le j \le E - 1$. As we will argue later, with high probability the projections by $\boldsymbol{x}^{\mathrm{tr}}$ do not introduce any additional linear dependence, so that

$$f(B) = B^L \boldsymbol{y} c^{(L)} + B^{L+1} \boldsymbol{y} c^{(L+1)} + \dots + B^D \boldsymbol{y} c^{(D)} = \mathbf{0}^N. \tag{5}$$

Step C3: *Compute*

$$\widehat{w} = \boldsymbol{z} c^{(L)} + B \boldsymbol{z} c^{(L+1)} + \dots + B^{D-L} \boldsymbol{z} c^{(D)}.$$

This requires at most $D - L$ additional multiplications of B times a vector (using a Horner evaluation scheme). One may argue as in Step C3 above that $\widehat{w} \ne \mathbf{0}^N$ with probability at least $1 - 1/\mathrm{card}(\mathbb{K})$ (see also proof of Theorem 6 in §5). Suppose now that $\widehat{w} \ne \mathbf{0}^N$. *Finally, determine the first integer i such that $B^i \widehat{w} = \mathbf{0}^N$ and return $w = B^{i-1} \widehat{w}$.* By (5), this should happen, with high probability, for an integer $i \le L + 1$. At most $L + 1$ more multiplications of B by a vector are required. Altogether, this step performs

$$D + 1 < \frac{N}{n} + 2 \tag{6}$$

multiplications of B by a vector. An additional $O(N^2)$ arithmetic operations in $\mathbb{K}$ are required to compute $\boldsymbol{z} c^{(i)}$ for $L \le i \le D$ and to add the $D - L + 1$ vectors in the Horner scheme. Note that with n processors the parallel number of matrix-vector multiplication can be reduced (see Appendix B). $\square$

Coppersmith's paper raises two distinct problems with the block Wiedemann method.

1. The efficient computation of a non-trivial solution to (4). He proposes a clever generalization of the Berlekamp/Massey algorithm to linearly generated sequences of matrices. Although one can define the notion of a minimum generator, a proof that the algorithm produces it has so far eluded us. However, we may proceed directly by computing a non-trivial solution of our system by either a method for Toeplitz-like matrices or by the Wiedemann algorithm itself and by using a fast polynomial (over $\mathbb{K}$) multiplication algorithm (see §6).

2. The probabilistic analysis, in particular the fact that with high probability the polynomial found, $f(\lambda)$, satisfies (5). We will show this to be true at least in the case that the minimum polynomial f^B of the coefficient matrix B has degree $\deg(f^B) = \operatorname{rank}(B) + 1$. Fortunately, by the randomizations of the Theorems 2 and 3 this condition can be enforced for any matrix B. Let us consider, e.g., solving a non-singular system $x = A^{-1}b$. We then can randomize $\widetilde{A} = AWG$ where W is random unit lower triangular Toeplitz matrix and G is random diagonal matrix, and execute the block Wiedemann method on the $(N+1) \times (N+1)$ matrix

$$B = \left[\begin{array}{c|c} \widetilde{A} & b \\ \hline \mathbf{0}^{1\times N} & 0 \end{array}\right].$$

Note that $\widetilde{A}$ has with high probability N distinct eigenvalues, and multiplications of $\widetilde{A}$ by vectors are inexpensive because W is Toeplitz.

5. Probabilistic analysis

We now justify Coppersmith's block version of the Wiedemann algorithm. We will prove the following theorem.

Theorem 6. *Let $\mathbb{K}$ be a finite field, and let $B \in \mathbb{K}^{N\times N}$ be a singular matrix whose minimal polynomial f^B has degree $\deg(f^B) = \operatorname{rank}(B) + 1$. Suppose that the vector blocks $\boldsymbol{x}^{\mathrm{tr}} \in \mathbb{K}^{m\times N}$ and $\boldsymbol{z} \in \mathbb{K}^{N\times n}$ are chosen at random and that $\widehat{w} \in \mathbb{K}^N$ is computed by the block Wiedemann algorithm of §3. Then with probability no less than $1 - (2\operatorname{rank}(B) + 1)/\operatorname{card}(\mathbb{K}) \geq 1 - (2N-1)/\operatorname{card}(\mathbb{K})$ we have $\widehat{w} \neq \mathbf{0}^N$ and $B^{L+1}\widehat{w} = \mathbf{0}^N$ for some integer $L \leq \lceil N/n \rceil$.*

The key property for the algorithm to succeed is equation (5). We will prove (5) first if the entries in $\boldsymbol{x}$ and $\boldsymbol{z}$ are indeterminates $\xi_{\iota,\mu}$ and $\zeta_{\iota,\nu}$, where $1 \leq \mu \leq m$, $1 \leq \nu \leq n$, and $1 \leq \iota \leq N$. In this case, the algorithm is performed over the rational function field over $\mathbb{K}$,

$$\mathbb{L} = \mathbb{K}(\xi_{1,1}, \ldots, \xi_{N,m}, \zeta_{1,1}, \ldots, \zeta_{N,n}).$$

In order to distinguish when the algorithm is performed over $\mathbb{K}$ and when over $\mathbb{L}$, we will write $\mathcal{X}$ and $\mathcal{Z}$ for the undetermined $\boldsymbol{x}$ and $\boldsymbol{z}$ and

$$\mathcal{Y} = B\mathcal{Z} \in \mathbb{L}^{N\times n}, \quad \mathcal{A}^{(i)} = \mathcal{X}^{\mathrm{tr}} B^i \mathcal{Y} \in \mathbb{L}^{m\times n}.$$

The equation (5) is equivalent to the solution vector c of (4) satisfying the following block Krylov system:

$$[\, B^{D+1}\boldsymbol{z} \mid \dots \mid B^2\boldsymbol{z} \mid B\boldsymbol{z}\,] \begin{bmatrix} c^{(D)} \\ \hline c^{(D-1)} \\ \hline \vdots \\ \hline c^{(0)} \end{bmatrix} = 0. \tag{7}$$

Clearly, any solution of (7) also solves (4). We first state that generically, i.e., over $\mathbb{L}$, no other solutions to (4) exists. We will prove this fact later using Proposition 4 stated below.

Proposition 3. *Suppose that the minimum polynomial f^B of B has the degree $\deg(f^B) = \min\{N, \operatorname{rank}(B) + 1\}$. Then for $D = \lceil N/n \rceil$ and $E = \lceil n(D+1)/m \rceil$ we have the rank equalities*

$$\begin{aligned} &\operatorname{rank}\left(\left[\begin{array}{c|c|c|c|c} \mathcal{A}^{(D)} & \dots & & \mathcal{A}^{(1)} & \mathcal{A}^{(0)} \\ \hline \mathcal{A}^{(D+1)} & \mathcal{A}^{(D)} & & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \\ \hline \vdots & & \ddots & & \vdots \\ \hline \mathcal{A}^{(D+E-1)} & \dots & & & \mathcal{A}^{(E-1)} \end{array}\right]\right) \\ &= \operatorname{rank}([\, B^D\mathcal{Y} \mid B^{D-1}\mathcal{Y} \mid \dots \mid B\mathcal{Y} \mid \mathcal{Y}\,]) \\ &= \operatorname{rank}(B). \end{aligned}$$

The proof of this proposition is based on its validity for $m = n = 1$, which we shall first prove. In that case we will denote our generic sequence by

$$\boldsymbol{\alpha}^{(i)} = \mathcal{A}_{1,1}^{(i)} = \mathcal{X}_1^{\mathrm{tr}} B^{i+1} \mathcal{Z}_1 \in \mathbb{L}, \quad \text{for } i \geq 0.$$

Proposition 4. *Let $M' \geq M \geq N$. Define*

$$\mathcal{T} = \begin{bmatrix} \boldsymbol{\alpha}^{(M)} & \dots & & \boldsymbol{\alpha}^{(1)} & \boldsymbol{\alpha}^{(0)} \\ \boldsymbol{\alpha}^{(M+1)} & \boldsymbol{\alpha}^{(M)} & & \boldsymbol{\alpha}^{(2)} & \boldsymbol{\alpha}^{(1)} \\ \vdots & & \ddots & & \vdots \\ \boldsymbol{\alpha}^{(M+M'-1)} & \dots & & & \boldsymbol{\alpha}^{(M'-1)} \end{bmatrix} \in \mathbb{L}^{M' \times (M+1)}$$

and

$$\mathcal{K} = [\, B^M \mathcal{Z}_1 \mid \dots \mid B^2 \mathcal{Z}_1 \mid B\mathcal{Z}_1\,] \in \mathbb{L}^{N \times M}.$$

Then

$$\operatorname{rank}(\mathcal{T}) = \operatorname{rank}(\mathcal{K}) = \begin{cases} \deg(f^B) & \textit{if } B \textit{ is non-singular,} \\ \deg(f^B) - 1 & \textit{if } B \textit{ is singular.} \end{cases}$$

Proof. Since $f^B(\lambda)$ linearly generates the sequence $\{B^i \mathcal{Z}_1\}_{i=0}^{\infty}$, where $B^i\mathcal{Z}_1 \in \mathbb{L}^N$, we must have the rank inequality

$$\operatorname{rank}([\, B^M \mathcal{Z}_1 \mid \ldots \mid B\mathcal{Z}_1 \mid \mathcal{Z}_1 \,]) \le \deg(f^B)$$

for any vector $\mathcal{Z}_1$ and any integer $M \ge N$. Moreover, by Proposition 2 of §3 the linear image $\{\mathcal{X}^{\mathrm{tr}} B^i \mathcal{Z}_1\}_{i=0}^{\infty}$ is minimally generated by f^B, so the minimum generator $f^{B,\mathcal{Z}_1}$ cannot be a polynomial of lesser degree, hence

$$\operatorname{rank}([\, B^N \mathcal{Z}_1 \mid \ldots \mid B\mathcal{Z}_1 \mid \mathcal{Z}_1 \,]) = \deg(f^B).$$

If B is a non-singular matrix, the minimum generating polynomial of the sequence $\{B^i \mathcal{Y}_1\}_{i=0}^{\infty} = \{B^{i+1}\mathcal{Z}_1\}_{i=0}^{\infty}$ does not change, while for singular B the minimum generating polynomial is $f^B(\lambda)/\lambda$, thus the rank of $\mathcal{K}$ drops by 1. We define this polynomial as

$$f_-^B(\lambda) = \begin{cases} f^B(\lambda) & \text{if } B \text{ is non-singular,} \\ f^B(\lambda)/\lambda & \text{if } B \text{ is singular.} \end{cases}$$

So far, we have shown that $\operatorname{rank}(\mathcal{K}) = \deg(f_-^B)$.

Second, we need to prove that $\operatorname{rank}(\mathcal{T}) = \operatorname{rank}(\mathcal{K})$. As before we can argue by making use of Proposition 2 that f_-^B is the minimum generating polynomial of the sequence $\{\boldsymbol{\alpha}^{(i)}\}_{i=0}^{\infty}$. We finally show that $\operatorname{rank}(\mathcal{T}) = \deg(f_-^B)$. Consider any non-zero solution $\gamma \in \mathbb{L}^{M+1}$ of

$$\mathcal{T}\gamma = \begin{bmatrix} \boldsymbol{\alpha}^{(M)} & \ldots & \boldsymbol{\alpha}^{(1)} & \boldsymbol{\alpha}^{(0)} \\ \boldsymbol{\alpha}^{(M+1)} & \boldsymbol{\alpha}^{(M)} & \boldsymbol{\alpha}^{(2)} & \boldsymbol{\alpha}^{(1)} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{\alpha}^{(M+M'-1)} & \ldots & & \boldsymbol{\alpha}^{(M'-1)} \end{bmatrix} \begin{bmatrix} \gamma^{(M)} \\ \gamma^{(M-1)} \\ \vdots \\ \gamma^{(0)} \end{bmatrix} = \mathbf{0}^{M'}. \tag{8}$$

Then for all $j = 0, \ldots, M-1 \le M'-1$

$$\boldsymbol{\alpha}^{(M+j)}\gamma^{(M)} + \cdots + \boldsymbol{\alpha}^{(j)}\gamma^{(0)} = 0,$$

hence by Theorem 4 of §3 the polynomial

$$\varphi(\lambda) = \gamma^{(M)}\lambda^M + \cdots + \gamma^{(1)}\lambda + \gamma^{(0)} \in \mathbb{L}[\lambda]$$

generates the entire sequence $\{\boldsymbol{\alpha}^{(i)}\}_{i=0}^{\infty}$. This implies that f_-^B divides φ, so φ is in the linear span over $\mathbb{L}$ of

$$f_-^B(\lambda), \lambda f_-^B(\lambda), \lambda^2 f_-^B(\lambda), \ldots, \lambda^\delta f_-^B(\lambda), \quad \text{where } \delta = M - \deg(f_-^B). \tag{9}$$

Also, any coefficient vector of a polynomial in the linear span of (9) solves (8), since any such polynomial generates $\{\boldsymbol{\alpha}^{(i)}\}_{i=0}^{\infty}$. Therefore the dimension of the kernel of $\mathcal{T}$ is equal to the dimension of the space generated by (9), which is $M + 1 - \deg(f_-^B)$, thus the rank of $\mathcal{T}$,

$$M + 1 - (\text{the dimension of the kernel of } \mathcal{T}),$$

is equal to $\deg(f_-^B)$. ⊠

Proof of Proposition 3. Consider the specialization

$$\mathcal{Z}' = [\,\mathcal{Z}_1 \mid \underbrace{B^{D+1}\mathcal{Z}_1}_{\mathcal{Z}_2'} \mid \underbrace{B^{2(D+1)}\mathcal{Z}_1}_{\mathcal{Z}_3'} \mid \ldots \mid \underbrace{B^{(n-1)(D+1)}\mathcal{Z}_1}_{\mathcal{Z}_n'}\,].$$

Then the set of columns in

$$[\,B^D\mathcal{Z}' \mid B^{D-1}\mathcal{Z}' \mid \ldots \mid \mathcal{Z}'\,]$$

is equal to

$$\{\mathcal{Z}_1, B\mathcal{Z}_1, B^2\mathcal{Z}_1, \ldots, B^{n(D+1)-1}\mathcal{Z}_1\}.$$

Since $n(D+1) > N$, this set has rank equal $\deg(f^B)$, as is argued in the proof of Proposition 2. Therefore the "more generic" matrix

$$[\,B^D\mathcal{Z} \mid B^{D-1}\mathcal{Z} \mid \ldots \mid \mathcal{Z}\,]$$

has rank $\geq \deg(f^B)$. Now define

$$\mathcal{K}^{\boxplus} = [\,B^D\mathcal{Y} \mid B^{D-1}\mathcal{Y} \mid \ldots \mid \mathcal{Y}] = B \cdot [\,B^D\mathcal{Z} \mid B^{D-1}\mathcal{Z} \mid \ldots \mid \mathcal{Z}\,],$$

which thus satisfies $\operatorname{rank}(\mathcal{K}^{\boxplus}) \leq \operatorname{rank}(B)$. If B is non-singular, the matrix $\mathcal{K}^{\boxplus}$ actually has full rank N, since by assumption $\deg(f^B) = N$. From Proposition 4 we further get for a singular B that

$$\deg(f^B) - 1 = \operatorname{rank}([\,B^{D+1}\mathcal{Z}' \mid B^D\mathcal{Z}' \mid \ldots \mid B\mathcal{Z}'\,]) \leq \operatorname{rank}(\mathcal{K}^{\boxplus}),$$

hence $\deg(f^B)-1 \leq \operatorname{rank}(\mathcal{K}^{\boxplus}) \leq \operatorname{rank}(B)$, which implies by the assumption of the theorem that $\operatorname{rank}(\mathcal{K}^{\boxplus}) = \operatorname{rank}(B)$. Note that if B is singular and $\deg(f^B) = N$ we automatically have $\operatorname{rank}(B) = N - 1 = \deg(f^B) - 1$.

We will use a similar specialization for the columns of $\mathcal{X}$ to establish that the rank of

$$\mathcal{T}^{\boxplus} = \left[\begin{array}{c|c|c|c|c} \mathcal{A}^{(D)} & \ldots & & \mathcal{A}^{(1)} & \mathcal{A}^{(0)} \\ \hline \mathcal{A}^{(D+1)} & \mathcal{A}^{(D)} & & \mathcal{A}^{(2)} & \mathcal{A}^{(1)} \\ \hline \vdots & & \ddots & & \vdots \\ \hline \mathcal{A}^{(D+E-1)} & \ldots & & & \mathcal{A}^{(E-1)} \end{array}\right] \qquad (10)$$

agrees with the rank of $\mathcal{T} \in \mathbb{L}^{M' \times (M+1)}$ of Proposition 4 with the dimensions

$$M = n(D+1) - 1 = S - 1 \geq N \quad \text{and} \quad M' = mE = R > S - 1.$$

Consider the specialization $\mathcal{Z}'$ given above, and

$$\mathcal{X}' = [\,\mathcal{X}_1 \mid \underbrace{(B^{\text{tr}})^E\mathcal{X}_1}_{\mathcal{X}_2'} \mid \underbrace{(B^{\text{tr}})^{2E}\mathcal{X}_1}_{\mathcal{X}_3'} \mid \ldots \mid \underbrace{(B^{\text{tr}})^{(m-1)E}\mathcal{X}_1}_{\mathcal{X}_n'}\,].$$

Then with $\mathcal{A}'^{(i)} = \mathcal{X}'^{\mathrm{tr}} B^{i+1} \mathcal{Z}'$ there exist permutation matrices $P \in \{0,1\}^{R\times R}$ and $Q \in \{0,1\}^{S\times S}$ such that

$$PTQ = \left[\begin{array}{c|c|c|c|c}
\mathcal{A}'^{(D)} & \dots & & \mathcal{A}'^{(1)} & \mathcal{A}'^{(0)} \\ \hline
\mathcal{A}'^{(D+1)} & \mathcal{A}'^{(D)} & & \mathcal{A}'^{(2)} & \mathcal{A}'^{(1)} \\ \hline
\vdots & & \ddots & & \vdots \\ \hline
\mathcal{A}'^{(D+E-1)} & \dots & & & \mathcal{A}'^{(E-1)}
\end{array}\right].$$

The row and column permutations move the entry

$$\begin{aligned}\mathcal{A}'^{(D+I-J)}_{i,j} &= \mathcal{X}'^{\mathrm{tr}}_i B^{D+I-J} B \mathcal{Z}'_j \\ &= \mathcal{X}_1^{\mathrm{tr}} B^{(i-1)E} B^{D+1+I-J} B^{(j-1)(D+1)} \mathcal{Z}_1\end{aligned}$$

in the right side block Toeplitz matrix, which is in row $mI + i$, where $0 \le I < E$ and $1 \le i \le m$, and column $nJ + j$, where $0 \le J < D+1$ and $1 \le j \le n$, to row $E(i-1) + I + 1$ and column $(D+1)(n-j) + J + 1$ in $\mathcal{T}$, namely

$$\begin{aligned}\mathcal{T}_{E(i-1)+I+1,(D+1)(n-j)+J+1} &= \boldsymbol{\alpha}^{(\,M\ +\ E(i-1)+I+1\ -\ ((D+1)(n-j)+J+1)\,)} \\ &= \mathcal{X}_1^{\mathrm{tr}} B^{n(D+1)+E(i-1)+I+(D+1)(j-n)-J} \mathcal{Z}_1.\end{aligned}$$

Therefore, the rank of $\mathcal{T}^{\boxplus}$ is no less than the rank of $\mathcal{T}$ with the given dimensions, which by Proposition 4 and the assumptions is equal to $\deg(f^B) = N$ for non-singular B, and is equal to $\deg(f^B) - 1$ for singular B. Since the kernel of $\mathcal{K}^{\boxplus}$ is contained in the kernel of $\mathcal{T}^{\boxplus}$, the rank cannot be more. $\boxtimes$

Proof of Theorem 6. Let

$$\Delta(\xi_{1,1}, \dots, \xi_{N,m}, \zeta_{1,1}, \dots, \zeta_{N,n})$$

be a non-zero maximal minor of $\mathcal{T}^{\boxplus}$ in (10). Then for all matrices $\boldsymbol{x}$ and $\boldsymbol{z}$ with

$$\Delta(x_{1,1}, \dots, x_{N,m}, z_{1,1}, \dots, z_{N,n}) \neq 0$$

any solution to (4) must also solve (7), because the ranks of both coefficient matrices will be equal to $\deg(f^B) - 1$. Hence

$$B^{L+1}\boldsymbol{z}c^{(L)} + B^{L+2}\boldsymbol{z}c^{(L+1)} + \cdots + B^{D+1}\boldsymbol{z}c^{(D)} = B^{L+1}\widehat{w} = \mathbf{0}^N$$

for $0 \le L \le D$ such that $c^{(L)} \neq 0$ and $c^{(\ell)} = 0$ for all $0 \le \ell < L$. By Theorem 1 the probability of hitting a zero of Δ is no more than $\deg(\Delta)/\mathrm{card}(\mathbb{K}) \le 2\,\mathrm{rank}(B)/\mathrm{card}(\mathbb{K})$.

It remains to estimate the probability that $\widehat{w} \neq 0$. The argument, by Coppersmith, is as that for step W3. For a matrix $\boldsymbol{y} = B\boldsymbol{z} \in \mathbb{K}^{N\times n}$ consider the equivalence class

$$\{\boldsymbol{z}' \in \mathbb{K}^{N\times n} \mid \boldsymbol{y} = B\boldsymbol{z}' = B\boldsymbol{z}\} \tag{11}$$

of $\mathbb{K}^{N\times n}$. Then for each member in that class

$$\begin{aligned}\widehat{w}' &= z'c^{(L)} + Bz'c^{(L+1)} + \cdots + B^{(D-L)}z'c^{(D)} \\ &= \underbrace{zc^{(L)} + Bzc^{(L+1)} + \cdots + B^{(D-L)}zc^{(D)}}_{\widehat{w}} + (z'-z)c^{(L)},\end{aligned}$$

where

$$z' - z = [\, w_1 \mid w_2 \mid \ldots \mid w_n \,] \quad \text{with } Bw_\nu = \mathbf{0} \text{ for all } 1 \le \nu \le n.$$

Since, given any $c^{(L)} \in \mathbb{K}^n \setminus \{\mathbf{0}^n\}$, for randomly chosen w_ν in the kernel of B the linear combination

$$c_1^{(L)} w_1 + \cdots + c_n^{(L)} w_n$$

uniformly samples vectors in the kernel of B, at most a fraction of $1/\mathrm{card}(\mathbb{K})$ matrices in the set (11) can give $-\widehat{w}$ as that linear combination and thus lead to $\widehat{w}' = \mathbf{0}$. Therefore, the probability that $\widehat{w} = \mathbf{0}$ is no more than $1/\mathrm{card}(\mathbb{K})$.

Summing both estimates bounds the probability of failure. ⊠

6. Fast solution of Toeplitz-like singular systems

In §7 we will give the asymptotic analysis of the block Wiedmann method described in §4. There it will be necessary to efficiently compute a non-zero solution of the block-Toeplitz system (4). We know five different approaches to this subproblem. The first is the one given by Coppersmith (1994) based on a generalization of the Berlekamp/Massey algorithm to linearly generated sequences of matrices. Coppersmith's method requires $O((m+n)N^2)$ arithmetic operations in $\mathbb{K}$ and needs no randomization, but the correctness of the algorithm remains an open problem. The second method is presented in Kaltofen (1993, Proof of Theorem 2) and uses the Wiedemann method of §4 and a fast matrix times vector routine for the arising block-Toeplitz coefficient matrix. Randomization and fast FFT-based polynomial multiplication is necessary, and the arithmetic cost is $O((m+n)N^2 \log N \operatorname{loglog} N)$.

The last three methods are based on the theory of Toeplitz-like matrices (Kailath et al. 1979) and also require randomization. Without fast FFT-based polynomial multiplication one can achieve arithmetic complexity $O((m+n)N^2)$ using a generalization of the Levinson-Durbin algorithm to Toeplitz-like matrices (see Gohberg et al. 1986). The fourth method is the speeded counterpart of that approach using divide-and-conquer and fast FFT-based polynomial multiplication (Bitmead and Anderson 1980; Morf 1980). The number of arithmetic operations can then be bounded by $O((m+n)^2 N (\log N)^2 \operatorname{loglog} N)$. The fifth possibility is to adapt the processor-efficient parallel linear system solver by Kaltofen and Pan (1992) to Toeplitz-like matrices. In this section we shall describe and analyze the generalized Levinson-Durbin algorithm with emphasis to the singular case. The asymptotically faster divide-and-conquer algorithm is described in an appendix to this paper.

First, we need to introduce the notion of the displacement rank of a matrix, which applies not only to block-Toeplitz matrices but also to their inverses. We consider $N \times N$ matrices; define the lower-shift matrix

$$Z = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \mathbf{0} \\ & 1 & \ddots & & \\ & & \ddots & & \\ \mathbf{0} & & & 1 & 0 \end{bmatrix}$$

and define the matrix shift operators

$$\downarrow A = ZA \quad \text{and} \quad \Rsh A = AZ^{\mathrm{tr}}.$$

The matrix $\downarrow A$ is equal to A after being shifted down by one row, filling the first row by zeros, and the matrix $\Rsh A$ is equal to A after being shifted to the right by one column, filling the first column by zeros. Following Kailath et al. (1979), we define

$$\phi_+(A) = A - \downarrow(\Rsh A) = A - ZAZ^{\mathrm{tr}} \quad \text{and} \quad \alpha_+(A) = \operatorname{rank} \phi_+(A),$$

the latter being the *displacement rank of A with respect to the displacement operator ϕ_+*. The fundamental property is that, given 2α column vectors $y_1, \ldots, y_\alpha$ and $z_1, \ldots, z_\alpha$, the functional equation in the matrix X

$$X - \downarrow(\Rsh X) = \sum_{j=1}^{\alpha} y_j z_j^{\mathrm{tr}} \tag{12}$$

has the unique solution

$$X = \sum_{j=1}^{\alpha} L[\![y_j]\!]\, U[\![z_j^{\mathrm{tr}}]\!], \tag{13}$$

where $L[\![y]\!]$ denotes a lower-triangular Toeplitz matrix whose first column is y and $U[\![z^{\mathrm{tr}}]\!]$ denotes an upper triangular Toeplitz matrix whose first row is z^{tr}. Therefore a matrix of displacement rank α w.r.t. ϕ_+ is a sum of α products of lower and upper triangular Toeplitz matrices. We shall call the vectors $y_1, \ldots, y_\alpha$ and $z_1, \ldots, z_\alpha$ in

$$Y = \sum_{j=1}^{\alpha} y_j z_j^{\mathrm{tr}} = \underbrace{[\, y_1 \mid y_2 \mid \ldots \mid y_\alpha \,]}_{\boldsymbol{y}} \cdot \left.\begin{bmatrix} z_1^{\mathrm{tr}} \\ \hline z_2^{\mathrm{tr}} \\ \hline \vdots \\ \hline z_\alpha^{\mathrm{tr}} \end{bmatrix}\right\} \boldsymbol{z} \tag{14}$$

the *left* and *right generators* of the $N \times N$ matrix Y. The matrix Y is usually a displaced matrix $\phi_+(X)$. Furthermore, we shall call the representation (13) the *ΣLU representation* for X. That representation requires only the storage of $O(\alpha N)$ field elements. Clearly, one may derive a generator (14) for Y by choosing the vectors y_j to be α linearly independent columns of Y, and the entries in each column of the right factor matrix with the rows z_j^{tr} to be the linear combination that yields the corresponding column of Y.

We can now sketch the generalized Levinson-Durbin algorithm as described in Gohberg et al. (1986, §1.C).

Algorithm *Generalized Levinson-Durbin*

Input: Vectors $y_1, \ldots, y_\alpha$, $z_1, \ldots, z_\alpha$, and $b \in \mathbb{K}^N$ such that

$$A = \sum_{j=1}^{\alpha} L[\![y_j]\!] U[\![z_j^{\mathrm{tr}}]\!] \in \mathbb{K}^{N \times N}$$

has generic rank profile (see §2).

Output: An integer $r \le N$ and a vector $x' \in \mathbb{K}^r$ such that $r = \mathrm{rank}(A)$ and $A_r x' = b'$ where A_r is the largest non-singular leading principal submatrix of A, and $b' \in \mathbb{K}^r$ is the vector consisting of the first r coordinates of b.

Step L0: *Initialize* $A[1,1] \leftarrow \sum_{i=1}^{\alpha} y_i[1] \cdot z_i[1]$; $\gamma^{(1)} \leftarrow [\,1/A[1,1]\,]$; $\varphi_i^{(1)} \leftarrow [\,y_i[1]/A[1,1]\,]$ *for all* $i \leftarrow 1, \ldots, \alpha$; $\chi^{(1)} \leftarrow [\,b[1]/A[1,1]\,]$.

For $k \leftarrow 2, 3, \ldots$ *Do Steps L1–L5.*

Step L1: At this moment we have the $(k-1)$-dimensional vectors $\gamma^{(k-1)}$, $\chi^{(k-1)}$, and $\varphi_i^{(k-1)}$ for $1 \le i \le \alpha$ such that

$$A_{k-1}\gamma^{(k-1)} = e_{k-1}^{(k-1)}, \quad A_{k-1}\chi^{(k-1)} = b^{(k-1)}, \quad A_{k-1}\varphi_i^{(k-1)} = y_i^{(k-1)},$$

where $e_{k-1}^{(k-1)}$ is the $(k-1)^{\mathrm{st}}$ unit vector of $k-1$ dimensions, and $y_i^{(k-1)}$ and $b^{(k-1)}$ are vectors formed by the first $k-1$ coordinates of y_i and b, respectively. Steps L1–L5 compute the k-dimensional versions of the vectors γ, χ, and φ.

For $i \leftarrow 1, \ldots, \alpha$ *compute the scalars*

$$\mu_i^{(k)} \leftarrow -\sum_{j=1}^{k-1} z_i[j+1] \cdot \gamma^{(k-1)}[j].$$

As is easily verified from the ΣLU representation of the leading principal submatrix $A_k = \sum_{i=1}^{\alpha} L[\![y_i^{(k)}]\!] U[\![(z_i^{(k)})^{\mathrm{tr}}]\!]$, where $z_i^{(k)}$ is the vector formed by the first k coordinates of z_i, the $\mu_i^{(k)}$ satisfy

$$A_k \left[\begin{array}{c} 0 \\ \hline \gamma^{(k-1)} \end{array}\right] = e_k^{(k)} - \sum_{i=1}^{\alpha} \mu_i^{(k)} y_i^{(k)}. \tag{15}$$

Step L2: *Compute the* k^{th} *row of* A_k,

$$[\,A[k,1] \quad A[k,2] \quad \ldots \quad A[k,k]\,] \leftarrow$$
$$[\,0 \quad A[k-1,1] \quad \ldots \quad A[k-1,k-1]\,] + \sum_{i=1}^{\alpha} y_i[k]\,(z_i^{(k)})^{\mathrm{tr}}.$$

Step L3: *For* $i \leftarrow 1, \ldots, \alpha$ *compute the scalars*

$$\nu_i^{(k)} \leftarrow y_i[k] - \sum_{j=1}^{k-1} A[k,j] \cdot \varphi_i^{(k-1)}[j].$$

These scalars satisfy

$$\varphi_i^{(k)} = \left[\begin{array}{c} \varphi_i^{(k-1)} \\ \hline 0 \end{array}\right] + \nu_i^{(k)} \gamma^{(k)}, \quad \text{for all } 1 \le i \le \alpha. \tag{16}$$

Step L4: From (15) and (16) and the defining property for $\varphi_i^{(k)}$ we obtain the following equation for $\gamma^{(k)}$:

$$\gamma^{(k)} = \left[\frac{0}{\gamma^{(k-1)}}\right] + \sum_{i=1}^{\alpha} \mu_i^{(k)} \left[\frac{\varphi_i^{(k-1)}}{0}\right] + \left(\sum_{i=1}^{\alpha} \mu_i^{(k)} \nu_i^{(k)}\right) \gamma^{(k)}. \tag{17}$$

If $1 = \sum_{i=1}^{\alpha} \mu_i^{(k)} \nu_i^{(k)}$ *then return* $r \leftarrow k-1$ *and* $x' \leftarrow \chi^{(k-1)}$. In this case, A_k must be singular, as we will explain below. *Else set*

$$\gamma^{(k)} \leftarrow \frac{1}{1 - \sum_{i=1}^{\alpha} \mu_i^{(k)} \nu_i^{(k)}} \left(\left[\frac{0}{\gamma^{(k-1)}}\right] + \sum_{i=1}^{\alpha} \mu_i^{(k)} \left[\frac{\varphi_i^{(k-1)}}{0}\right] \right).$$

and compute all $\varphi_i^{(k)}$ *according to (16).*

Step L5: Similarly to (16), *compute the scalar* $\lambda^{(k)} \leftarrow b[k] - \sum_{j=1}^{k-1} A[k,j] \times \chi^{(k-1)}[j]$ *and the vector*

$$\chi^{(k)} \leftarrow \left[\frac{\chi^{(k-1)}}{0}\right] + \lambda^{(k)} \gamma^{(k)}. \qquad \square$$

Clearly, the above method requires no more than $O(\alpha N^2)$ arithmetic operations in the field of entries. The correctness of the algorithm hinges on the fact that a possible zero division in the computation of $\gamma^{(k)}$ indicates the singularity of the corresponding submatrix.

Lemma 1. *In the algorithm Generalized Levinson-Durbin,* $1 \neq \sum_{i=1}^{\alpha} \mu_i^{(\ell)} \nu_i^{(\ell)}$ *for all* $1 \leq \ell \leq k$ *if and only if* $A_1, \ldots, A_k$ *are not singular.*

Proof. The proof is by induction on k. Suppose now that $1 \neq \sum_{i=1}^{\alpha} \mu_i^{(k)} \nu_i^{(k)}$. Then by (17) a solution to $A_k \gamma^{(k)} = e_k^{(k)}$ can be found, and hence a solution exists to $A_k \chi' = c'$ for any k dimensional vector c' by the induction hypothesis and re-application of the algorithm. Therefore, A_k must be invertible. Second, we must proof that for non-singular $A_1, \ldots, A_k$ the division in Step L4 is indeed possible. Assume that $1 = \sum_{i=1}^{\alpha} \mu_i^{(k)} \nu_i^{(k)}$ and consider the last coordinate of (17): we must then have $\gamma^{(k-1)}[k-1] = 0$, which contradicts the non-singularity of A_{k-2} when considering solving $A_{k-1} \gamma^{(k-1)} = e_{k-1}^{(k-1)}$ by Cramer's rule. $\boxtimes$

We can now apply algorithm Generalized Levinson-Durbin to computing a random solution of (4) in §4. First, we rearrange the $R \times S$ coefficient matrix

$$A = \left[\begin{array}{c|c|c|c|c} \boldsymbol{a}^{(D)} & \ldots & & \boldsymbol{a}^{(1)} & \boldsymbol{a}^{(0)} \\ \hline \boldsymbol{a}^{(D+1)} & \boldsymbol{a}^{(D)} & & \boldsymbol{a}^{(2)} & \boldsymbol{a}^{(1)} \\ \hline \vdots & & \ddots & & \vdots \\ \hline \boldsymbol{a}^{(D+E-1)} & \ldots & & & \boldsymbol{a}^{(E-1)} \end{array}\right]$$

into an $m \times n$ block matrix such that each block is a Toeplitz matrix, using the permutation matrices P and Q of the proof of Proposition 3. However, by inspection of the proof of Proposition 3 we see that we may drop the rows in position

R, $R-m, \ldots, R-(R-S-1)m$ from A without affecting the probabilistic rank estimates, thus making a smaller coefficient matrix $A^{\square}$ of the square dimensions $S \times S$. Note the blocks are of dimensions $(R/m) \times (S/n)$, except in the last row of blocks, where the dimensions are $(R/m - R + S) \times (S/n)$. Briefly, the row permutation matrix P, which is now acting on $A^{\square}$, moves rows 1, $m+1$, $2m+1, \ldots$ into rows 1, 2, $3, \ldots$, and rows 2, $m+2$, $2m+2, \ldots$ into rows $(R/m)+1$, $(R/m)+2$, $(R/m)+3, \ldots$, etc.; the column permutation matrix Q moves columns 1, $n+1$, $2n+1, \ldots$ of $A^{\square}$ into columns 1, 2, $3, \ldots$, and columns 2, $n+2$, $2n+2, \ldots$ into columns $(S/n)+1$, $(S/n)+2$, $(S/n)+3, \ldots$, etc.

The matrix $PA^{\square}Q$ has displacement rank no more than $m+n$ w.r.t. ϕ_+, and a ΣLU representation can be easily computed from the (known) generators of $\phi_+(PA^{\square}Q)$. By pre- and post-multiplication by random unit upper and unit lower triangular Toeplitz matrices $V, W \in \mathbb{K}^{S\times S}$ we obtain a matrix

$$\widetilde{A} = V \cdot (PA^{\square}Q) \cdot W$$

of generic rank profile (see Theorem 2), that with high probability. Moreover, a ΣLU representation for $\widetilde{A}$ of length no more than $m+n+4$ can be computed in $O((m+n)S^2)$ arithmetic operations for the product by the usual product formulas for Toeplitz-like matrices (see Pan 1992, Proposition A.3; also Proposition 8 in the Appendix).

We now apply algorithm Generalized Levinson-Durbin to the ΣLU representation of $\widetilde{A}$ and a right side vector derived as in Proposition 1of §2. The algorithm returns an integer r that with high probability equals $\mathrm{rank}(\widetilde{A}) = \mathrm{rank}(B)$, and a solution to $\widetilde{A}_r^{-1}b'$. Using Proposition 1 we now can quickly obtain random vectors in the right null space of $\widetilde{A}$. Note that the $\gamma^{(k)}$ and $\varphi_i^{(k)}$ in the algorithm Generalized Levinson-Durbin need only be computed once, and then the system can be solved for new random b' in $O(N^2)$ arithmetic operations in $\mathbb{K}$.

7. Algorithms and their running times

The block Wiedemann method of §4 is used to solve both non-singular and singular sparse linear systems, i.e., linear systems with an efficient way to multiply the coefficient matrix by any vector. The method is randomized and can be executed sequentially or in parallel. Especially in the latter form the method becomes very efficient. We now present several variants of the block algorithm. One main point is that we are able to give both explicit expected running times and estimates on the success probability of the randomizations. We first deal with the sequential performance of the blocking. Parallel variants will be discussed below.

Theorem 7. *Let $B \in \mathbb{K}^{N\times N}$ be a singular matrix and $1 \le m, n \le N$. Then one can compute the rank of B and a solution vector $w \in \mathbb{K}^N \setminus \{\mathbf{0}\}$ with $Bw = \mathbf{0}$ in no more than*

$$\left\lfloor \left(1 + \frac{n}{m} + \frac{1}{n}\right)N + \frac{2n^2}{m} + 2n + 2 \right\rfloor$$

multiplications of B times a vector in $\mathbb{K}^N$, and an additional

$$O\left((m+n)N^2 + \left(1 + \frac{n}{m}\right)N^2 \log N \operatorname{loglog} N\right)$$

arithmetic operations in $\mathbb{K}$. *The algorithm selects no more than* $(m+n+6)N+6n-4$ *random elements in* $\mathbb{K}$ *and computes the correct rank and succeeds to produce a solution with probability no less than* $1-2(N+n)^2/(\mathrm{card}(\mathbb{K})-1)$. *The algorithm requires an additional* $O((m+n)N)$ *amount of storage for field elements in* $\mathbb{K}$.

Proof. Consider the pre-conditioned matrix $\widetilde{B} = VBWG$, where $V \in \mathbb{K}^{N\times N}$ is a random unit upper triangular Toeplitz matrix, $W \in \mathbb{K}^{N\times N}$ is a random unit lower triangular Toeplitz matrix, and $G \in \mathbb{K}^{N\times N}$ is a random non-singular diagonal matrix. Then by Theorems 2 and 3 with probability of at least $1-\frac{3}{2}(N-1)N/(\mathrm{card}(\mathbb{K})-1)$ for the minimum polynomial $f^{\widetilde{B}}$ of $\widetilde{B}$ we have $\deg(f^{\widetilde{B}}) = \mathrm{rank}(\widetilde{B})+1$. Note that the cardinality of $\mathbb{K}$ in the estimate is decreased by 1 since the diagonal entries of G must be non-zero. Also, for a vector $b \in \mathbb{K}^N$ the product $\widetilde{B}b$ can be computed by one multiplication of B by a vector, and an additional $O(N \log N \operatorname{loglog} N)$ arithmetic operations in $\mathbb{K}$. This is because the product of V or W by an arbitrary vector can be reduced to polynomial multiplication, which over arbitrary fields is doable in $O(N \log N \operatorname{loglog} N)$ (Cantor and Kaltofen 1991). We remark that by using matrices corresponding to Beneš networks in place of V and W (Wiedemann 1986, §V) one can reduce the arithmetic complexity by the $\operatorname{loglog} N$ factor at the cost of requiring $O(N \log N)$ random field elements.

Now, the matrix $\widetilde{B}$ satisfies the assumptions of Theorem 6, and we can find a solution $\widetilde{w} \neq \mathbf{0}$ to $\widetilde{B}\widetilde{w} = \mathbf{0}$. Thus $w = VG\widetilde{w} \neq \mathbf{0}$ solves $Bw = \mathbf{0}$. By (3) and (6) of §4 the method multiplies $\widetilde{B}$ by a vector no more than

$$\Bigl(1+\frac{n}{m}+\frac{1}{n}\Bigr)N+\frac{2n^2}{m}+2n+2$$

many times. The extra work in terms of arithmetic operations in $\mathbb{K}$ is $O((m+n)N^2)$ plus the work it takes to solve (4), which by the results of §6 is doable in $O((m+n)N^2)$ additional arithmetic operations.

Let us suppose that $\widetilde{B}$ has the properties required by Theorem 6. The probabilistic analysis depends an the algorithm used for solving (4), whose $S\times S$ coefficient matrix we denote by A. In order to obtain a non-zero solution to $Ac = \mathbf{0}$, the additional randomization $\widetilde{A} = VAW$ discussed at the end of §6 must make the matrix $\widetilde{A}$ of generic rank profile. This is true for random V and W with probability $\geq 1-\frac{1}{2}(N+2n-1)(N+2n)/\mathrm{card}(\mathbb{K})$, since $S \leq N+2n$ (see Step C2 in §4). Furthermore, the vector selected by the randomization of Proposition 1 must be non-zero. This will happen with probability at least $1-1/\mathrm{card}(\mathbb{K})$. Note that the computation of r and $\widetilde{A}_r^{-1}b'$ when accomplished by the generalized Levinson-Durbin algorithm needs no further randomization. Moreover, by Proposition 3 and the proof of Theorem 6 the computed rank $r = \mathrm{rank}(A)$ is equal to $\mathrm{rank}(\widetilde{B}) = \mathrm{rank}(B)$, provided $\widetilde{A}$ and $\widetilde{B}$ have the properties stated above. Once a non-zero vector c is found, which can of course be verified on the spot, the estimates of Theorem 6 become valid. Therefore, the overall probability of success is at least

$$1-\Bigl(\frac{3}{2}\,\frac{(N-1)N}{\mathrm{card}(\mathbb{K})-1}+\frac{1}{2}\,\frac{(N+2n-1)(N+2n)}{\mathrm{card}\,\mathbb{K}}+\frac{1}{\mathrm{card}\,\mathbb{K}}\Bigr) \geq 1-\frac{2(N+n)^2}{\mathrm{card}(\mathbb{K})-1}.$$

The randomization for $\widetilde{B}$ requires $3N-2$ random non-zero field elements, that for $\widetilde{A}$ no more than $2N+4n-2$, and that for b of Proposition 1 no more than $N + 2n$. The block Wiedemann algorithm itself needs $(m + n)N$ random field elements, so the total is no more than the one stated above. Finally, the algorithm only needs to store $\boldsymbol{a}^{(i)}$ and the ΣLU representation of the matrix $\widetilde{A}$ and the ΣUL representation of the incremental inverses. ⊠

Theorem 7 can be employed to solve non-singular systems as outlined in the last paragraph of §4. We shall formulate the result not in terms of the block sizes m and n, but in terms of the quantity $\epsilon = (n/m) + (1/n)$. For suitable constant block sizes ϵ can be made arbitrarily close to 0. Thus we have the following sequential complexity result.

Corollary. *Let $B \in \mathbb{K}^{N\times N}$ be a non-singular matrix and let $\epsilon > 0$ be fixed. Then one can compute the solution vector $w = B^{-1}b$ with $b \in \mathbb{K}^N$ in no more than $(1 + \epsilon)N + O(1)$ multiplications of B times a vector in $\mathbb{K}^N$, and an additional $O(N^2 \log N \operatorname{loglog} N)$ arithmetic operations in $\mathbb{K}$. The algorithm selects $O(N)$ random elements in $\mathbb{K}$ and succeeds to produce the solution with probability no less than $1 - O(N^2)/(\operatorname{card}(\mathbb{K}) - 1)$. The algorithm requires an additional $O(N)$ amount of storage for field elements in $\mathbb{K}$. Note that here all big-O estimates depend on ϵ.*

Of course, the main application of blocking is to compute the sequence of matrices $\boldsymbol{a}^{(i)}$ in parallel. In order to make the statement of the next Theorem simpler, we suppose that m/n is constant and that we have n parallel processors.

Theorem 8. *Let $m, n > 0$ be integers $\leq N$ with $m/n = \Theta(1)$. Suppose there is a network of $n \leq N$ processors, each of which can independently multiply the matrix $B \in \mathbb{K}^{N\times N}$ by an arbitrary vector. Here B can either be stored in memory shared by all processors or be distributed over the local memories of the processors. Then a random solution to the linear system $Bw = b$, where $b \in \mathbb{K}^N$ is a vector in the space spanned by the columns of B, can be computed in no more than $2N/m + 4N/n + O(1)$ multiplications of B by vectors carried out simultaneously by each processor. Furthermore, each processor performs $O(N^2 \log N \operatorname{loglog} N)$ additional arithmetic operations in $\mathbb{K}$ and stores $O(N)$ intermediately computed field elements. As an extra intermediate substep the network of processors can solve a singular homogeneous system of displacement rank $m+n$ in $O(N^2)$ parallel time.*

Proof. The linear system $Bw = b$ under consideration is inhomogeneous and singular. We appeal to Proposition 1. The method first computes the rank r of B and then solves a non-singular $r \times r$ system, thus executing the block Wiedemann algorithm essentially twice. As in the proof of Theorem 7, before we can apply the block Wiedemann algorithm of §4, we must randomly pre-condition the matrix B to $\widetilde{B} = VBWG$. Now the product of $\widetilde{B}$ times a vector requires an additional $O(N \log N \operatorname{loglog} N)$ arithmetic steps in $\mathbb{K}$. In Step C1 of §4, each processor independently computes the sequence of column vectors $\{\boldsymbol{x}^{\mathrm{tr}}\widetilde{B}^i y_\nu\}$ for the ν^{th} columns y_ν of $\boldsymbol{y}$. These sequences have length $N/m+N/n+O(1)$. Finally, a single processor carries out the computations of Step C3, which require an additional $N/n + O(1)$ matrix times vector products. The work required to compute each $\boldsymbol{x}^{\mathrm{tr}} \cdot (\widetilde{B}^i y_\nu)$ in Step C1 and the single-processor work of Step C3 amounts to $O(N^2 \log N \operatorname{loglog} N)$ arithmetic operations in $\mathbb{K}$ for each processor.

The rank of the coefficient matrix of (4), which is equal to the rank of B with high probability, is computed in parallel by the methods described in §6.

That entails to first compute a ΣLU representation for $\widetilde{A}$ and then to apply algorithm Generalized Levinson-Durbin. By the well-known the product formula for Toeplitz-like matrices (Pan 1992, Proposition A.3; also Proposition 7 in the Appendix) the derivation of the generators for $\phi_+(\widetilde{A})$ can be distributed over the n processors, each performing $O(N^2)$ arithmetic. Finally, the work in steps L1, L2, L3, and L4 of the Generalized Levinson-Durbin algorithm can be divided among $n = \Theta(\alpha)$ processors, such that each processor performs $O(N)$ arithmetic operations. For instance, the sums of α vectors of k dimensions required in steps L2 and L4 can be computed in parallel by letting each processor compute the sum of $\lceil k/\alpha \rceil$ coordinates. Note that this scheme has a high communication cost, especially if the i^{th} processor locally has stored the vectors y_i, z_i, φ_i and the scalars μ_i and ν_i, and it may be more efficient to perform a binary tree addition, that at a cost of $O(N \log n)$.

Once the rank of B is known, we can apply Proposition 1 to the system $\widetilde{B}\tilde{x} = Vb$, from which we obtain the random solution $x = WG\tilde{x}$. In order to compute $\widetilde{B}_r^{-1}b'$ of Proposition 1, we execute the block Wiedemann algorithm on the $(r+1) \times (r+1)$ matrix

$$\left[\begin{array}{c|c} \widetilde{B}_r & b' \\ \hline \mathbf{0}^{1\times r} & 0 \end{array}\right],$$

following the ideas in the last paragraph of §4. Clearly, multiplying $\widetilde{B}_r$ by a vector can be accomplished in a single B times a vector product plus $O(N \log N \operatorname{loglog} N)$ field operations. ⊠

We remark that the above proof may be overly complex. A non-zero solution of the homogeneous linear system

$$(B - [\, c_1 b \mid c_2 b \mid \cdots \mid c_N b \,])\, w = \mathbf{0},$$

where $c_1, \ldots, c_N$ are random field elements, may yield

$$B\, \frac{1}{c_1 w[1] + \cdots + c_N w[N]} w = b,$$

provided the division is not by zero. However, since the block Wiedemann algorithm does not return a random vector in the null space of B, we cannot make any guarantees. Furthermore, Step C3 of the Block Wiedemann algorithm can be parallelized (see Appendix B) reducing the number of parallel matrix-vector multiplications.

8. Conclusion

Our main contribution in this paper is to give a theoretical basis for the block generalization of the Wiedemann method. We could prove our algorithm for sufficiently large fields and by using certain pre-conditioning of the input matrix. The algorithm can still be valid without the assumptions on the degree of the minimum polynomial. For instance, the matrices arising in our polynomial factorization algorithm (Kaltofen and Lobo 1994) clearly have (degree of minimum polynomial) $<$ (rank $-$ 1), while the algorithm still produces a solution. Coppersmith observes, however, that for certain "pathological" cases the straight-forward algorithm might fail to compute a solution. Also, Coppersmith's application of factorization of integers leads to sparse systems over the field $\mathbb{K} = \mathbb{F}_2$ of 2 elements. In that situation, Proposition 3 could be relaxed. If the rank of (4) were one or two less than the rank of (7), with probability 1/2 or 1/4 we still would find a solution to (7). For very large finite fields such a rank deficiency would make the problem quite infeasible. Nonetheless, the case of very small finite fields remains to be resolved, which is an important open problem. We remark that if we have blocking by single vectors, the probability of success of the block Wiedemann algorithm is $o(1)$ for $\mathbb{K} = \mathbb{F}_2$ (see Wiedemann 1986, Proposition 3).

Our algorithms are formulated for finite fields only, but it is not difficult to extend them to fields such as the rational numbers and functions by the use of Chinese remaindering, interpolation, and p-adic lifting (McClellan 1973, Moenck and Carter 1979).

Further problems left unresolved regard the solution of the block-Toeplitz system arising in the course of the block algorithm. We have shown how by randomization the system can be brought into a regular form, namely of generic rank profile, and how then it can be solved very fast. It may be possible to avoid the condition that the coefficient matrix be of generic rank profile (cf. Delsarte et al. 1985) thus avoiding randomization. Coppersmith's generalized Berlekamp/Massey approach needs no randomizations either, but it remains open how to prove the algorithm correct or how to speed it by use of FFT-based polynomial multiplication (cf. Brent et al. 1980, §8). More importantly, it is unclear to us if the asymptotically faster divide-and-conquer algorithm by Bitmead-Anderson/Morf can in practice outperform the generalized Levinson-Durbin algorithm.

With Austin Lobo we have implemented several versions of the block Wiedemann algorithm in the programming language C for $\mathbb{K} = \mathbb{F}_p$ and executed it using simultaneously on a network of 8 Sun Sparc 2 computers, each rated 28.5MIPS. For $p = 32749$, for instance, we can solve a 20,000×20,000 system with 1.32 million non-zero entries on 4 computers in about 60 CPU hours. Our implementation also covers the case of small coefficient fields. E.g., for $p = 2$ we can solve a 100,000×100,000 system with 10.3 million non-zero entries on three 28.5MIPS-computers in about 54 CPU hours. In that case we use "double blocking," where each of the computers processes blocks of 32 vectors by 32 bit logic. The details of this experiment are published in Díaz et al. (1993). We have also applied the method to the problem of factoring polynomials of degree 10,000 and more over finite fields (Kaltofen and Lobo 1994). In that application, the coefficient matrix has a true black box representation with a fast function for the matrix-times-vector product.

Appendix A

By use of FFT-based polynomial arithmetic and a divide-and-conquer strategy, the block-Toeplitz system (4) of §4 can be solved in $O((m+n)^2 N (\log N)^2 \times \log\log N)$ arithmetic operations. We shall describe and analyze a version of the Bitmead-Anderson/Morf method suitable for such singular inputs. This approach will need to reduce a ΣLU representation of §6 for a matrix X to one with a minimum number of terms under the sum (13). Here, we solve this problem by randomization. Consider that we are given $\beta \geq \alpha$ generators for a matrix $Y = \phi_+(X)$,

$$Y = \hat{\boldsymbol{y}} \cdot \hat{\boldsymbol{z}}^{\mathrm{tr}}, \quad \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}} \in \mathbb{K}^{N\times\beta},$$

and we wish to determine the displacement rank $\alpha = \alpha_+(X)$ and a ΣLU representation of length α for X. We pick random matrices V and W as in Theorem 2 of §2. Then the matrix $\widetilde{Y} = VYW$ has, with high probability, generic rank profile. Since $\mathrm{rank}(Y) = \mathrm{rank}(\widetilde{Y})$, every column to the right of the first α columns of $\widetilde{Y}$ is a linear combination of the first α columns. These linear combinations determine generators for $\widetilde{Y}$; namely, $\widetilde{Y} = \tilde{\boldsymbol{y}} \cdot \tilde{\boldsymbol{z}}^{\mathrm{tr}}$, where $\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \in \mathbb{K}^{N\times\alpha}$. Here $\tilde{\boldsymbol{y}}$ are the first α columns of $\widetilde{Y}$ and each column in $\tilde{\boldsymbol{z}}^{\mathrm{tr}} = [\,I_\alpha \mid \ldots]$ corresponds to the linear combination yielding the column of $\widetilde{Y}$ in the same position. The minimum-length generators for Y are then obtained as $Y = (V^{-1}\tilde{\boldsymbol{y}}) \cdot (\tilde{\boldsymbol{z}}^{\mathrm{tr}} W^{-1})$. The running time of this method is stated in the next proposition.

Proposition 5. *From a ΣLU representation of $X \in \mathbb{K}^{N\times N}$ of length β, namely $X = \sum_{k=1}^{\beta} L[\![\hat{y}_k]\!]\, U[\![\hat{z}_k^{\mathrm{tr}}]\!]$, one can compute in $O(\alpha\beta N + \beta N \log N \operatorname{loglog} N)$ arithmetic steps in $\mathbb{K}$ a ΣLU representation $X = \sum_{j=1}^{\alpha} L[\![y_j]\!]\, U[\![z_j^{\mathrm{tr}}]\!]$, where $\alpha = \mathrm{rank}\,\phi_+(X)$ is minimum. The algorithm is randomized and requires $2N-2$ uniformly sampled elements from a set $S \subset \mathbb{K}$; it returns with probability no less than $1 - \alpha(\alpha+1)/\mathrm{card}(S)$ a correct result.*

Proof. First we note that the multiplication of an N dimensional row vector by a triangular $N \times N$ Toeplitz matrix or its inverse can be carried out in $O(N \times \log N \operatorname{loglog} N)$ arithmetic operations by use of asymptotically fast polynomial multiplication or power series inversion. Thus, we may compute $V\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}^{\mathrm{tr}} W$ in $O(\beta N \log N \operatorname{loglog} N)$ arithmetic operations. Next, one can incrementally compute and invert the leading principal submatrices $\widetilde{Y}_i$ of $\widetilde{Y}$ until a singular $\widetilde{Y}_{\alpha+1}$ is found. This costs $O(\alpha^2\beta)$ arithmetic operations in $\mathbb{K}$. Then the first α rows and columns of $\widetilde{Y}$ are found from $V\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}^{\mathrm{tr}} W$ in $O(\alpha\beta N)$ steps. Using $\widetilde{Y}_\alpha^{-1}$ we can also complete $\tilde{\boldsymbol{z}}^{\mathrm{tr}}$ in $O(\alpha^2 N)$ arithmetic steps. Finally, we have to pre- and post-multiply the generators of $\widetilde{Y}$ with V^{-1} and W^{-1}, costing $O(\alpha N \log N \operatorname{loglog} N)$ arithmetic steps in $\mathbb{K}$. ⊠

The main property of matrices of small displacement rank is that their inverses also have small displacement rank. Clearly, the inverse of a Toeplitz matrix is not Toeplitz but, as we will see, it is Toeplitz-like. However, the displacement operator ϕ_+ does not directly apply to the inverse; instead, a dual operator is used, which we now introduce. Consider the shift operators

$$\upharpoonleft A = Z^{\mathrm{tr}} A \quad \text{and} \quad \urcorner A = AZ.$$

The matrix $\upharpoonleft A$ is equal to A after being shifted up by one row, filling the last row by zeros, and the matrix $\urcorner A$ is equal to A after being shifted to the left by one column, filling the last column by zeros. Now define

$$\phi_-(A) = A - \upharpoonleft(\urcorner A) = A - Z^{\mathrm{tr}} A Z \quad \text{and} \quad \alpha_-(A) = \mathrm{rank}\,\phi_-(A),$$

the latter being the *displacement rank with respect to the displacement operator* ϕ_-. By transposition along the anti-diagonal of the matrix X in (12), one obtains a dual to the ΣLU representation; namely,

$$X - \upharpoonleft(\urcorner X) = \sum_{k=1}^{\bar{\alpha}} \bar{y}_k \bar{z}_k^{\mathrm{tr}} \iff X = \sum_{k=1}^{\bar{\alpha}} U[\![(\bar{y}_k^{\mathrm{rev}})^{\mathrm{tr}}]\!] L[\![\bar{z}_k^{\mathrm{rev}}]\!], \tag{18}$$

where z^{rev} is the reverse of a vector z; that is,

$$z^{\mathrm{rev}} = J \cdot z \quad \text{with} \quad J = \begin{bmatrix} 0 & \ldots & 0 & 1 \\ 0 & \ldots & 1 & 0 \\ \vdots & \iddots & & \vdots \\ 1 & & 0 & 0 \end{bmatrix} \in \mathbb{K}^{N \times N}.$$

We will call the right side of (18) the ΣUL representation of X. There is an explicit formula for converting a ΣLU representation to a ΣUL representation (cf. Bitmead and Anderson 1980, Lemma 5), which we will need later:

$$L[\![y]\!]U[\![z^{\mathrm{tr}}]\!] = IL[\![\hat{z}]\!] + U[\![\hat{y}^{\mathrm{tr}}]\!]I - U[\![(ZJy)^{\mathrm{tr}}]\!]L[\![ZJz]\!], \quad \text{for } y, z \in \mathbb{K}^N, \tag{19}$$

where $\hat{z}^{\mathrm{tr}}$ is the reversed last row of $L[\![y]\!]U[\![z^{\mathrm{tr}}]\!]$, and $\hat{y}$ the reversed last column of $L[\![y]\!]U[\![z^{\mathrm{tr}}]\!]$ but with the first entry set to 0. Note that I is the $N \times N$ identity matrix. From (19) and the dual formula

$$U[\![z^{\mathrm{tr}}]\!]L[\![y]\!] = L[\![\check{y}]\!]I + IU[\![\check{z}^{\mathrm{tr}}]\!] - L[\![ZJz]\!]U[\![(ZJy)^{\mathrm{tr}}]\!], \quad \text{for } y, z \in \mathbb{K}^N, \tag{20}$$

where $\check{z}^{\mathrm{tr}}$ is the first row of $U[\![z^{\mathrm{tr}}]\!]L[\![y]\!]$, and $\check{y}$ is the first column of $U[\![z^{\mathrm{tr}}]\!]L[\![y]\!]$ with its first entry set to 0, we conclude that for any square matrix A the inequalities $-2 \le \alpha_+(A) - \alpha_-(A) \le 2$ must hold. Moreover, the conversions (19) and (20) can be carried out in $O(N \log N \operatorname{loglog} N)$ arithmetic operations in $\mathbb{K}$. We finally can formulate the closure property with respect to matrix inversion.

Proposition 6. *For any nonsingular matrix $A \in \mathbb{K}^{N \times N}$ we have $\alpha_+(A) = \alpha_-(A^{-1})$ and $\alpha_-(A) = \alpha_+(A^{-1})$.*

An elegant proof of this property is found in (Pan 1992, Proposition A.4).

At task is to compute the ΣUL representation for the inverse of a non-singular matrix A given in ΣLU representation. If A is singular, but of generic rank profile, we seek the ΣUL representation for the largest non-singular leading principal submatrix. The algorithm follows a divide-and-conquer matrix partitioning à la Strassen: let

$$A = \left[\begin{array}{c|c} A_{1,1} & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right], \quad \text{where} \quad \begin{cases} A_{1,1} \in \mathbb{K}^{M \times M}, \ A_{1,2}, A_{2,1}^{\mathrm{tr}} \in \mathbb{K}^{M \times (N-M)}, \\ A_{2,2} \in \mathbb{K}^{(N-M) \times (N-M)}. \end{cases} \tag{21}$$

If $A_{1,1}$ is non-singular, we consider the *Schur complement* $\Delta = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$. If both $A_{1,1}$ and A are non-singular, the inverse can be computed as

$$A^{-1} = \left[\begin{array}{c|c} A_{1,1}^{-1} + A_{1,1}^{-1}A_{1,2}\Delta^{-1}A_{2,1}A_{1,1}^{-1} & -A_{1,1}^{-1}A_{1,2}\Delta^{-1} \\ \hline -\Delta^{-1}A_{2,1}A_{1,1}^{-1} & \Delta^{-1} \end{array}\right].$$

The key property is:

Proposition 7. *If A, $A_{1,1}$, and Δ are the matrices defined above, $A_{1,1}$ is non-singular and if the top-left entry of A, $A[1,1] \neq 0$, then $\alpha_+(\Delta) \le \alpha_+(A)$.*

Proof. In case that A is a non-singular matrix, the stated displacement rank inequality for the Schur complement is proven (without the condition on $A[1,1]$) in Bitmead and Anderson (1980, Lemma 8) and it is also stated in Morf (1980). We will reduce the singular case to the non-singular case. Consider a minimum length ΣLU representation of A; namely, $A = \sum_{j=1}^{\alpha} L^{(j)}U^{(j)}$, and suppose without loss of generality that $(L^{(1)}U^{(1)})[1,1] \neq 0$. The latter condition is necessitated by our assumption on the non-vanishing of the top-left entry of A. Therefore, the parameterized matrix

$$A(\lambda) = (L^{(1)} + \lambda I)U^{(1)} + \sum_{j=2}^{\alpha} L^{(j)}U^{(j)} = A + \lambda U^{(1)} \in \mathbb{K}(\lambda)^{N\times N}$$

is non-singular of displacement rank α w.r.t. ϕ_+. Partitioning $A(\lambda)$ corresponding to (21), we obtain a parameterized Schur complement

$$\Delta(\lambda) = A_{2,2} + \lambda U_{2,2}^{(1)} - A_{2,1}(A_{1,1} + \lambda U_{1,1}^{(1)})^{-1}(A_{1,2} + \lambda U_{1,2}^{(1)}).$$

It follows from the non-singular version of this proposition that $\alpha_+(\Delta(\lambda)) \leq \alpha$. We may write $\Delta(\lambda)$ as power series in λ with matrix coefficients,

$$\Delta(\lambda) = \Delta + \lambda(U_{2,2}^{(1)} + A_{2,1}A_{1,1}^{-1}U_{1,1}^{(1)}A_{1,1}^{-1}A_{1,2} - A_{2,1}A_{1,1}^{-1}U_{1,2}^{(1)}) \\ + \text{higher order terms},$$

using the series expansion

$$\begin{aligned}(G + \lambda H)^{-1} &= (I + \lambda G^{-1}H)^{-1}G^{-1} \\ &= G^{-1} - \lambda G^{-1}HG^{-1} + \lambda^2(G^{-1}H)^2G^{-1} + \cdots.\end{aligned}$$

Since no negative powers of λ occur, the constant term in this power series for $\Delta(\lambda)$, which is Δ, cannot have a higher displacement rank than $\Delta(\lambda)$, which proves the proposition. ⊠

The last property of Toeplitz-like matrices that we need for our algorithm is the fact that their products remain Toeplitz-like. Because we encounter rectangular matrices in our algorithm, we first have to extend the definitions of the displacement operators to such matrices. By subscripting Z_N we shall indicate that the shift matrix Z is of dimensions $N \times N$; we define a rectangular displacement operator

$$\phi_+(X) = X - Z_M X Z_N^{\mathrm{tr}} \quad \text{for } X \in \mathbb{K}^{M\times N}.$$

Again, $\phi_+(X)$ is generated by $\alpha = \alpha_+(X) = \operatorname{rank} \phi_+(X)$ vectors $y_1, \ldots, y_\alpha \in \mathbb{K}^M$ and $z_1, \ldots, z_\alpha \in \mathbb{K}^N$: $\phi_+(X) = \sum_{j=1}^{\alpha} y_j z_j^{\mathrm{tr}}$. We now have the following product rule (cf. Pan 1992, Proposition A.3).

Proposition 8. *Let $G \in \mathbb{K}^{L\times M}$ and $H \in \mathbb{K}^{M\times N}$ be rectangular matrices with displacement ranks $\gamma = \alpha_+(G)$ and $\delta = \alpha_+(H)$. Then $\phi_+(GH)$ can be generated by $\gamma + \delta + 1$ vectors.*

Proof. First, we observe that $I_M = Z_M^{\mathrm{tr}} Z_M + e_M e_M^{\mathrm{tr}}$, where I_M is the $M \times M$ identity matrix and e_M is the M^{th} unit vector. Therefore

$$\begin{aligned}
\phi_+(GH) &= GH - Z_L G I_M H Z_N^{\mathrm{tr}} \\
&= GH - (Z_L G Z_M^{\mathrm{tr}})(Z_M H Z_N^{\mathrm{tr}}) - Z_L G e_M e_M^{\mathrm{tr}} H Z_N^{\mathrm{tr}} \\
&= (G - Z_L G Z_M^{\mathrm{tr}}) H + Z_L G Z_M^{\mathrm{tr}} (H - Z_M H Z_N^{\mathrm{tr}}) - g h^{\mathrm{tr}} \\
&= \phi_+(G) H + Z_L G Z_M^{\mathrm{tr}} \phi_+(H) - g h^{\mathrm{tr}},
\end{aligned} \tag{22}$$

where $g = Z_L G e_M \in \mathbb{K}^L$ and $h = Z_N H^{\mathrm{tr}} e_M \in \mathbb{K}^N$. $\boxtimes$

We can now sketch the main algorithm (cf. Bitmead and Anderson 1980, p. 110).

Algorithm *Leading Principal Inverse*

Input: Vectors $y_1, \dots, y_\alpha, z_1, \dots, z_\alpha \in \mathbb{K}^N$ such that $A = \sum_{j=1}^{\alpha} L[\![y_j]\!] U[\![z_j^{\mathrm{tr}}]\!] \in \mathbb{K}^{N \times N}$ has generic rank profile.

Output: An integer $r \le N$ and vectors $\bar{y}_1, \dots, \bar{y}_{\bar{\alpha}}, \bar{z}_1, \dots, \bar{z}_{\bar{\alpha}} \in \mathbb{K}^r$ with $\bar{\alpha} \le \alpha$ such that with high probability

$$r = \operatorname{rank}(A) \quad \text{and} \quad A_r^{-1} = \sum_{k=1}^{\bar{\alpha}} U[\![\bar{y}_k^{\mathrm{tr}}]\!] L[\![\bar{z}_k]\!],$$

where A_r is the largest non-singular leading principal submatrix of A.

If $N \le \alpha$ then expand the ΣLU representation of A and compute A_r^{-1} explicitly; finally, from $\phi_-(A_r^{-1})$ explicitly determine the ΣUL representation and return.

Now, let the matrix A be partitioned as (21) with $M = \lceil N/2 \rceil$.

Step 1: *Call the algorithm recursively to process $A_{1,1}$.* Note that the ΣLU representation of $A_{1,1}$ is given by the first M entries of y_j and z_j. If the returned rank of $A_{1,1}$ is less than M, we are done. Otherwise, the algorithm has produced a ΣUL representation of $A_{1,1}^{-1}$.

Step 2: *Compute a ΣLU representation of length no more than α for the Schur complement $\Delta = A_{2,2} - A_{2,1} A_{1,1}^{-1} A_{1,2}$.* We further explain this task in the analysis of the algorithm. *If $\Delta[1,1] = 0$, then $M = \operatorname{rank}(A)$; else perform the next steps.*

Step 3: *Call the algorithm recursively to process Δ.* Note that, with high probability, $\operatorname{rank}(A) = M + \operatorname{rank}(\Delta) = r$.

Step 4: Consider the leading principal submatrix A_r partitioned as

$$A_r = \left[\begin{array}{c|c} A_{1,1} & A'_{1,2} \\ \hline A'_{2,1} & A'_{2,2} \end{array}\right], \quad \text{where} \quad \begin{cases} A_{1,1} \in \mathbb{K}^{M \times M},\ A'_{1,2}, A'^{\,\mathrm{tr}}_{2,1} \in \mathbb{K}^{M \times (r-M)}, \\ A'_{2,2} \in \mathbb{K}^{(r-M) \times (r-M)}. \end{cases}$$

At this point we have the ΣUL representations for $A_{1,1}^{-1}$ and for Δ'^{-1}, where $\Delta' = A'_{2,2} - A'_{2,1} A_{1,1}^{-1} A'_{1,2}$. *Compute (possibly non-minimum length) generators for $\phi_-(B'_{1,1})$, $\phi_-(B'_{1,2})$, and $\phi_-(B'_{2,1})$, where $B'_{1,2} = -A_{1,1}^{-1} A'_{1,2} \Delta'^{-1}$, $B'_{2,1} = -\Delta'^{-1} A'_{2,1} A_{1,1}^{-1}$, and $B'_{1,1} = A_{1,1}^{-1} - B'_{1,2} A'_{2,1} A_{1,1}^{-1}$. Finally, compute a minimum length ΣUL representation for*

$$A_r^{-1} = \left[\begin{array}{c|c} B'_{1,1} & B'_{1,2} \\ \hline B'_{2,1} & \Delta'^{-1} \end{array}\right]. \qquad \square$$

We can now state and prove the running time of the above algorithm.

Theorem 9. *Algorithm Leading Principal Inverse finishes after*

$$O(\alpha^2 N(\log N)^2 \operatorname{loglog} N)$$

arithmetic operations in $\mathbb{K}$. It requires $O(N \log N)$ random field elements that are uniformly sampled from a subset $S \subset \mathbb{K}$, and it returns with probability no less than $1 - 4N\alpha/\mathrm{card}(S)$ a correct rank and ΣUL representation of the largest leading principal submatrix.

Proof. Let $T(\alpha, N)$ denote the maximum number of arithmetic operations required for any input of dimension N and of at most α displacement rank. Step 1 requires at most $T(\alpha, \lceil N/2 \rceil)$ arithmetic operations. By Proposition 7, Step 3 requires at most $T(\alpha, \lfloor N/2 \rfloor)$ arithmetic operations. We shall show that both Step 2 and Step 4 have arithmetic complexity $O(\alpha^2 N \log N \operatorname{loglog} N)$. Hence, for a constant C, we must have

$$T(\alpha, N) \leq T(\alpha, \lceil N/2 \rceil) + T(\alpha, \lfloor N/2 \rfloor) + C\alpha^2 N \log N \operatorname{loglog} N,$$

which yields the arithmetic complexity $T(\alpha, N) = O(\alpha^2 N(\log N)^2 \operatorname{loglog} N)$.

In Step 2, we first compute generators for $\phi_+(\Delta)$ of length $\beta \leq 4\alpha + 8$, which we then reduce by Proposition 5 and 7 to a length of no more than α. The former is accomplished as follows: generators of length no more than $\alpha + 2$ can be derived for $\phi_+(A_{2,2})$ from generators for $\phi_+(A)$ by correcting for the shift into $A_{2,2}$ of parts of row M and parts of column M of A. Similarly, generators of length no more than $\alpha + 1$ can be derived for $\phi_+(A_{2,1})$ and $\phi_+(A_{1,2})$ (cf. Bitmead and Anderson 1980, Lemma 8). The ΣUL representation for $A_{1,1}^{-1}$ can be converted to a ΣLU representation of length no more than $\alpha + 2$ by using formula (20). We do not have ΣLU representations for the rectangular matrices $A_{2,1}$ and $A_{1,2}$. However, we have the ΣLU representation of A restricted to these submatrices. Thus, we may effectively use the product rule (22) of Proposition 8. For instance, the generators

$$Z_{N-M} A_{2,1} Z_M^{\mathrm{tr}} \phi_+(A_{1,1}^{-1})$$

arising in the computation of generators for $\phi_+(A_{2,1}A_{1,1}^{-1})$ are found by multiplying each y vector of the generators of $\phi_+(A_{1,1}^{-1})$; first by Z_M^{tr}, then by $A_{2,1}$, and finally by Z_{N-M}. Clearly, from the ΣLU representation of A restricted to $A_{2,1}$, such multiplication can be carried out for a single vector in $O(\alpha N \log N \times \operatorname{loglog} N)$ arithmetic operations. Therefore, the computation of the generators for $A_{2,1}A_{1,1}^{-1}A_{1,2}$ dominates this step at a cost of $O(\alpha^2 N \log N \operatorname{loglog} N)$ arithmetic operations.

The tasks of Step 4 are carried out similarly. After converting the ΣLU representation of A to a ΣUL representation using formula (19), we can obtain generators of length no more than $\alpha + 3$ for $\phi_-(A'_{2,1})$ and $\phi_-(A'_{1,2})$. Note that here we need a generalized ϕ_- operator on rectangular matrices. Then, as in Step 2 with a product formula for $\phi_-(GH)$ dual to (22), we find generators for

$$\begin{aligned}
&\phi_-(B'_{1,2}) \quad \text{with} \quad \alpha_-(B'_{1,2}) \leq 3\alpha + 5,\\
&\phi_-(B'_{2,1}) \quad \text{with} \quad \alpha_-(B'_{2,1}) \leq 3\alpha + 5,\\
&\phi_-(B'_{1,1}) \quad \text{with} \quad \alpha_-(B'_{1,1}) \leq 6\alpha + 10.
\end{aligned}$$

Finally, the generators for the blocks can be "puzzled" together to a generator of $\phi_-(A_r^{-1})$ of length no more than $13\alpha + 22$. Note that the length is the sum of the individual lengths corrected by two extra generators, which make up for the "cross" of a row and a column missing in the shift of the individual blocks. Finally, we reduce the ΣUL representation of A_r^{-1} to minimum length, again appealing to a dual of Proposition 5. The overall cost in this step is again dominated by the implementation of the product formula, which is $O(\alpha^2 N \log N \operatorname{loglog} N)$.

Finally, we argue that the algorithm produces, with the stated probability, the correct result. By Proposition 7, the displacement rank of the Schur complement Δ is no more than α. Furthermore, Δ has generic rank profile, as can be deduced from the factorization

$$A = \left[\begin{array}{c|c} I_M & 0 \\ \hline A_{2,1}A_{1,1}^{-1} & I_{N-M} \end{array}\right] \cdot \left[\begin{array}{c|c} A_{1,1} & A_{1,2} \\ \hline 0 & \Delta \end{array}\right].$$

Thus the algorithm produces a correct result if the randomizations of Proposition 5 needed in Steps 2 and 4 result in correct ΣLU representations and the recursive calls return correct ΣUL representations. Let $Pf(\alpha, N)$ be the probability that the algorithm fails to compute a correct result. We have the recursive estimate

$$Pf(\alpha, N) \leq Pf(\alpha, \lceil N/2 \rceil) + Pf(\alpha, \lfloor N/2 \rfloor) + \frac{2\alpha(\alpha+1)}{\operatorname{card} S},$$

and $Pf(\alpha, L) = 0$ for all $L \leq \alpha$, which by induction yields

$$Pf(\alpha, N) \leq \left(\frac{2N}{\alpha+1} - 1\right) \frac{2\alpha(\alpha+1)}{\operatorname{card} S} \quad \text{for } N > \alpha.$$

The number of random elements required, $El(\alpha, N)$, satisfies the recursive inequality

$$El(\alpha, N) \leq El(\alpha, \lceil N/2 \rceil) + El(\alpha, \lfloor N/2 \rfloor) + 4N - 4,$$

and $El(\alpha, L) = 0$ for all $L \leq \alpha$, which yields $El(\alpha, N) = O(N \log N)$. ⊠

Appendix B

In Theorem 8 we have supposed that Step C3 of the algorithm Block Wiedemann of §4 is carried out by a single processor, i.e., sequentially. One of the referees has suggested the following parallelization that with n processors requires only about $N(1 + \log_2 n)/n^2$ matrix-vector multiplications for Step C3, thus leading to a total of about

$$\frac{N}{m} + \frac{N}{n} + \frac{(1+\log_2 n)N}{n^2} + O(1)$$

parallel matrix-times-vector products in the entire Block Wiedemann algorithm. This improvement has practical potential in applications where the matrix-vector multiplication is costly, like in the polynomial factoring algorithm of Kaltofen and Lobo (1994). Note that in such applications one also chooses $m \gg n$.

During Step C1, store the vector blocks

$$\boldsymbol{u}_j = B^{j\lceil N/n^2 \rceil} \boldsymbol{z} = B^{j\lceil N/n^2 \rceil - 1} \boldsymbol{y}$$

for all $j = 1, \ldots, n-1$. Note that these blocks are available, since the $B^i \boldsymbol{y}$ are computed for $0 \le i < N/m + N/n + 2n/m + 1$. Define $\boldsymbol{u}_0 = \boldsymbol{z}$. By binary search, we find the exponent k, where $0 \le k \le L/\lceil N/n^2 \rceil \le n$, such that

$$B^{k\lceil N/n^2\rceil}\widehat{w} \neq \mathbf{0} \quad \text{but} \quad B^{(k+1)\lceil N/n^2\rceil}\widehat{w} = \mathbf{0}. \tag{23}$$

Temporarily, let $l = k\lceil N/n^2 \rceil$. The value of

$$B^l\widehat{w} = B^l \boldsymbol{z} c^{(L)} + B^{l+1}\boldsymbol{z} c^{(L+1)} + \cdots + B^{D-L+l}\boldsymbol{z} c^{(D)}$$

is found by having each processor compute the vector

$$\sum_{i=j\lceil N/n^2\rceil}^{(j+1)\lceil N/n^2\rceil-1} B^i \boldsymbol{z} c^{(L+i-l)} = \sum_{i=0}^{\lceil N/n^2\rceil-1} B^i \boldsymbol{u}_j c^{(L+i+j\lceil N/n^2\rceil-l)}.$$

The latter requires no more than N/n^2 parallel matrix-times-vector products using Horner evaluation. Once an exponent k satisfying (23) is discovered, one sequentially computes $B^{k\lceil N/n^2\rceil+i}\widehat{w}$ for $i = 1, 2, \ldots$ until the zero vector is produced.

Acknowledgements

Thanks to Austin Lobo for discussions on the theory and implementation of the block Wiedemann method and generalized Levinson-Durbin method, and to Martin Morf for his advice on the asymptotically fast Toeplitz-like solver. Thanks also to the two referees for their valuable suggestions.

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York 12181
E-mail: `kaltofen@cs.rpi.edu`

Bitmead, R. R. and Anderson, B. D. O., "Asymtotically fast solution of Toeplitz and related systems of linear equations," *Linear Algebra Applic.* **34**, pp. 103–116 (1980).

Brent, R. P., Gustavson, F. G., and Yun, D. Y. Y., "Fast solution of Toeplitz systems of equations and computation of Padé approximants," *J. Algorithms* **1**, pp. 259–295 (1980).

Cantor, D. G. and Kaltofen, E., "On fast multiplication of polynomials over arbitrary algebras," *Acta Inform.* **28**/7, pp. 693–701 (1991). Available from `anonymous@ftp.cs.rpi.edu` in directory `kaltofen`.

Coppersmith, D., "Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm," *Math. Comput.* **62**/205, pp. 333–350 (1994).

Delsarte, P., Genin, Y. V., and Kamp, Y. G., "A generalization of the Levinson algorithm for Hermitian Toeplitz matrices with any rank profile," *IEEE Trans. Acoustics, Speech, and Signal Process.* ASSP-**33**/4, pp. 964–971 (1985).

DeMillo, R. A. and Lipton, R. J., "A probabilistic remark on algebraic program testing," *Information Process. Letters* **7**/4, pp. 193–195 (1978).

Díaz, A., Hitz, M., Kaltofen, E., Lobo, A., and Valente, T., "Process scheduling in DSC and the large sparse linear systems challenge," in *Proc. DISCO '93*, Springer

Lect. Notes Comput. Sci. **722**, edited by A. Miola; pp. 66–80, 1993. Available from `anonymous@ftp.cs.rpi.edu` in directory `kaltofen`.

GOHBERG, I., KAILATH, T., and KOLTRACHT, I., "Efficient solution of linear systems of equations with recursive structure," *Linear Algebra Applic.* **80**, pp. 81–113 (1986).

KAILATH, T., KUNG, S.-Y., and MORF, M., "Displacement ranks of matrices and linear equations," *J. Math. Analysis Applications* **68**, pp. 395–407 (1979).

KALTOFEN, E., "Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems," in *Proc. AAECC-10*, Springer Lect. Notes Comput. Sci. **673**, edited by G. Cohen, T. Mora, and O. Moreno; pp. 195–212, 1993. Available from `anonymous@ftp.cs.rpi.edu` in directory `kaltofen`.

KALTOFEN, E. and LOBO, A., "Factoring high-degree polynomials by the black box Berlekamp algorithm," in *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, edited by J. von zur Gathen and M. Giesbrecht; ACM Press, New York, N. Y., pp. 90–98, 1994. Available from `anonymous@ftp.cs.rpi.edu` in directory `kaltofen`.

KALTOFEN, E. and PAN, V., "Processor-efficient parallel solution of linear systems II: the positive characteristic and singular cases," *Proc. 33rd Annual Symp. Foundations of Comp. Sci.*, pp. 714–723 (1992).

KALTOFEN, E. and SAUNDERS, B. D., "On Wiedemann's method of solving sparse linear systems," in *Proc. AAECC-9*, Springer Lect. Notes Comput. Sci. **539**; pp. 29–38, 1991. Available from `anonymous@ftp.cs.rpi.edu` in directory `kaltofen`.

LAMACCHIA, B. A. and ODLYZKO, A. M., "Solving large sparse linear systems over finite fields," in *Advances in Cryptology: CRYPTO '90*, Lect. Notes Comput. Sci. **537**, edited by A. J. Menezes and S. Vanstone; Springer Verlag, pp. 109–133, 1991.

LENSTRA, A. K., LENSTRA, JR., H. W., MANASSE, M. S., and POLLARD, J. M., "The factorization of the ninth Fermat number," *Math. Comput.* **61**/203, pp. 319–349 (1993).

MASSEY, J. L., "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory* IT-**15**, pp. 122–127 (1969).

MCCLELLAN, M. T., "The exact solution of systems of linear equations with polynomial coefficients," *J. ACM* **20**, pp. 563–588 (1973).

MOENCK, R. T. and CARTER, J. H., "Approximate algorithms to derive exact solutions to systems of linear equations," *Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci.* **72**, pp. 65–73 (1979).

MORF, M., "Doubling algorithms for Toeplitz and related equations," in *Proc. 1980 IEEE Internat. Conf. Acoust. Speech Signal Process.*; IEEE, pp. 954–959, 1980.

PAN, V., "Parameterization of Newton's iteration for computations with structured matrices and applications," *Computers Math. Applic.* **24**/3, pp. 61–75 (1992).

POMERANCE, C. and SMITH, J. W., "Reduction of huge, sparse matrices of finite fields via created catastrophes," *Experiment. Math.* **1**, pp. 89–94 (1992).

SCHWABE, E. J., BLELLOCH, G. E., FELDMANN, A., GHATTAS, O., GILBERT, J. R., MILLER, G. L., O'HALLARON, D. R., SHEWCHUK, J. R., and TENG, S.-H., "A separator-based framework for automated partitioning and mapping of parallel algorithms for numerical solution of PDEs," *Manuscript*, Carnegie Mellon Univ., Pittsburgh, Pennsylvania, July 1993.

SCHWARTZ, J. T., "Fast probabilistic algorithms for verification of polynomial identities," *J. ACM* **27**, pp. 701–717 (1980).

WIEDEMANN, D., "Solving sparse linear equations over finite fields," *IEEE Trans. Inf. Theory* IT-**32**, pp. 54–62 (1986).

ZIPPEL, R., "Probabilistic algorithms for sparse polynomials," *Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci.* **72**, pp. 216–226 (1979).

ZIPPEL, R., *Effective Polynomial Computations*; Kluwer Academic Publ., Boston, Massachusetts, 1993; 384pp.