AD-A238 943

MASTER COPY - FOR REPRODUCTION PURPOSES (2)

REPORT DOCUMENTATION PAGE

Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): ARO 23453.23-MA
6a. NAME OF PERFORMING ORGANIZATION: Stanford University	6b. OFFICE SYMBOL (If applicable)
7a. NAME OF MONITORING ORGANIZATION: U. S. Army Research Office	
6c. ADDRESS (City, State, and ZIP Code): Department of Electrical Engineering, Information Systems Laboratory, Stanford, CA 94305-4055	7b. ADDRESS (City, State, and ZIP Code): P. O. Box 12211, Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: U. S. Army Research Office	8b. OFFICE SYMBOL (If applicable)
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: DAAL03-86-K-0045	
8c. ADDRESS (City, State, and ZIP Code): P. O. Box 12211, Research Triangle Park, NC 27709-2211	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. P-23453-MA / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): Divide-and-Conquer Solutions of Least-Squares Problems for Matrices with Displacement Structure

12. PERSONAL AUTHOR(S): J. Chun and T. Kailath

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
reprint	FROM 1990 TO 1991	1991, January	18

16. SUPPLEMENTARY NOTATION: The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): divide-and-conquer, least squares, displacement structure, fast convolution, Toeplitz, Schur complements, generalized Schur algorithm

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Abstract. A divide-and-conquer implementation of a generalized Schur algorithm enables (exact and) least-squares solutions of various block-Toeplitz or Toeplitz-block systems of equations with $O(\alpha^3 n \log^2 n)$ operations to be obtained, where the displacement rank α is a small constant (typically between two to four for scalar near-Toeplitz matrices) independent of the size of the matrices.

91-05677

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Jagdish Chandra	22b. TELEPHONE (Include Area Code): (919) 549-0641 / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

91 7 19 088

SIAM J. MATRIX ANAL. APPL.
Vol. 12, No. 1, pp. 128–145, January 1991

DIVIDE-AND-CONQUER SOLUTIONS OF LEAST-SQUARES PROBLEMS FOR MATRICES WITH DISPLACEMENT STRUCTURE*

J. CHUN† AND T. KAILATH‡

Abstract. A divide-and-conquer implementation of a generalized Schur algorithm enables (exact and) least-squares solutions of various block-Toeplitz or Toeplitz-block systems of equations with $O(\alpha^3 n \log^2 n)$ operations to be obtained, where the displacement rank α is a small constant (typically between two to four for scalar near-Toeplitz matrices) independent of the size of the matrices.

Key words. divide-and-conquer, least squares, displacement structure, fast convolution, Toeplitz, Schur complements, generalized Schur algorithm

AMS(MOS) subject classifications. primary 65F05, 65F30; secondary 15A06

1. Introduction.

In recent years, there has been considerable research on fast algorithms for the solution of linear systems of equations with Toeplitz matrices. The Levinson and Schur algorithms allow solutions with $O(n^2)$ floating point operations (flops) for systems with $n \times n$ Toeplitz matrices.

In 1980 Brent, Gustavson, and Yun [5] described a scheme for obtaining a solution with $O(n \log^2 n)$ flops. This was based on two ideas—the use of the *Gohberg–Semencul formula* [11], [13], [17], [26] for the inverse of a Toeplitz matrix, and the use of divide-and-conquer (or doubling) techniques for computing (generators of) the Gohberg–Semencul formula.

Let $\mathbf{x}$ and $\mathbf{y}$ denote the first and last columns of $T^{-1} \in \mathbf{R}^{n \times n}$. Then if the first component of $\mathbf{x}$, say x_1, is nonzero, Gohberg and Semencul [13] showed that we could write

$$T^{-1} = \frac{1}{x_1}[L(\mathbf{x})L^T(\tilde{I}_n\mathbf{y}) - L(Z_n\mathbf{y})L^T(Z_n\tilde{I}_n\mathbf{x})], \qquad x_1 \neq 0,$$

where $\tilde{I}_n$ is the *reverse-identity matrix*, Z_n is the *shift matrix*,

$$\tilde{I}_n \equiv \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}, \qquad Z_n \equiv \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix},$$

and $L(\mathbf{v})$ is a lower-triangular Toeplitz matrix with first column $\mathbf{v}$. The significance of the Gohberg–Semencul formula in the present application is that the product of a vector and a lower- or upper-triangular Toeplitz matrix is equivalent to the convolution of two vectors, which can be done using $O(n \log n)$ flops (see, e.g., [4]).

Brent, Gustavson, and Yun used a divide-and-conquer scheme for a certain Euclidean algorithm to factorize row-permuted Toeplitz matrices (i.e., Hankel matrices), and to

* Received by the editors December 5, 1988; accepted for publication (in revised form) November 6, 1989. This work was supported in part by the U.S. Army Research Office under contract DAAL03-86-K-0045, the Strategic Defense Initiative Organization/Innovative Science and Technology, managed by the Army Research Office under contract DAAL03-87-K-0033, and the National Science Foundation under grant MIP-21315-A2.

† Information Systems Laboratory, Stanford University, Stanford, California 94305. Present address, General Electric Research and Development, Room KWD 218, P.O. Box 8, Schenectady, New York 12301 (chun@rascals.stanford.edu).

‡ Information Systems Laboratory, Stanford University, Stanford, California 94305 (tk@isl.stanford.edu).

obtain the vectors $\{\mathbf{x}, \mathbf{y}\}$ of the Gohberg–Semencul formula with $O(n \log^2 n)$ flops. Later Bitmead and Anderson [3] and Morf [21] used another approach based on the displacement-rank properties of matrix Schur complements, to obtain similar results; while this approach allows for generalization to non-Toeplitz matrices, the hidden coefficient in their proposed $O(n \log^2 n)$ constructions turned out to be extremely large (see Sexton, Shensa, and Speiser [25]). Later Musicus [22], de Hoog [11], Ammar and Gragg [2] used a more direct approach based on a combination of the Schur and Levinson algorithms to obtain better coefficients; in particular, Ammar and Gragg made a detailed study and claimed an operation count of $8n \log^2 n$ flops. With this count, the new (called superfast in [2]) method for solving (exactly determined) Toeplitz systems is faster than the one based on the Levinson algorithm whenever $n > 256$. We should mention here that Schur-algorithm-based methods are natural in the context of transmission-line and layered-earth models, so it is not a surprise that similar techniques were also conceived in those fields (see Choate [7], McClary [20], Bruckstein and Kailath [6]). A good source for background on the Levinson and Schur algorithms, transmission line models, displacement representations as mentioned and used in the present paper may be [14].

The method we have taken in this paper is in the spirit of the generalized Schur algorithm (see, e.g., [8], [9]). Our algorithm can be applied to non-Toeplitz matrices, and is simpler than the methods of Bitmead and Anderson [3] or Morf [21]. Furthermore, we can readily handle matrices such as $(T^TT)^{-1}$ and $(T^TT)^{-1}T^T$, where T may be a near-Toeplitz matrix or a rectangular block-Toeplitz matrix, or a Toeplitz-block matrix; in particular, therefore, we can also obtain the *least-squares* solutions of overdetermined Toeplitz and near-Toeplitz systems with $O(n \log^2 n)$ flops. Our algorithm is closely related to the algorithm of Musicus [22]. However, our presentation is conceptually much simpler (especially for the non-Toeplitz cases treated in [22]) than previous approaches; in particular, we do not use the relationship between the Schur algorithm and Levinson algorithms needed in [2], [11], and [22].

An outline of our approach is the following. For a matrix E,

$$E = \begin{bmatrix} E_{1,1} & E_{1,2} \\ E_{2,1} & E_{2,2} \end{bmatrix}, \qquad E_{1,1}, \text{ nonsingular,} \tag{1}$$

the Schur complement of $E_{1,1}$ in E is

$$S \equiv E_{2,2} - E_{2,1}E_{1,1}^{-1}E_{1,2}.$$

Note that matrices such as

$$S_1 \equiv T^{-1}, \qquad S_2 \equiv (T^TT)^{-1}, \qquad S_3 \equiv (T^TT)^{-1}T^T \tag{2}$$

can be identified as the Schur complements of the northwest blocks in the following *extended matrices*:

$$E_1 = \begin{bmatrix} T & I \\ -I & O \end{bmatrix}, \qquad E_2 = \begin{bmatrix} T^TT & I \\ -I & O \end{bmatrix}, \qquad E_3 = \begin{bmatrix} T^TT & T^T \\ -I & O \end{bmatrix}. \tag{3}$$

Now the matrices E in (3) have the following (generalized) *displacement representation*, for suitably chosen matrixes $\{F^f, F^b\}$:

$$E = \sum_{i=1}^{\alpha} K(\mathbf{x}_i, F^f)K^T(\mathbf{y}_i, F^b),$$

where $K(\mathbf{x}_i, F^f)$ and $K(\mathbf{y}_i, F^b)$ are lower triangular matrices whose j columns are $(F^f)^{(j-1)}\mathbf{x}_i$ and $(F^b)^{(j-1)}\mathbf{y}_i$, respectively. The smallest possible number α is called the

displacement rank of E with respect to $\{F^f, F^b\}$. For an example, let T be an $m \times n$ scalar Toeplitz matrix, with $m \geq n$. Then the matrix E_2 has displacement rank four with respect to $\{F, F\}$, where $F = \left[\begin{smallmatrix} Z_n & O \\ O & Z_n \end{smallmatrix}\right]$, and has a displacement representation [15],

$$\text{(4a)} \quad E_2 = \sum_{i=1}^{2} K(\mathbf{y}_i, F)K^T(\mathbf{x}_i, F) - \sum_{i=3}^{4} K(\mathbf{y}_i, F)K^T(\mathbf{x}_i, F), \qquad \mathbf{y}_i \equiv \begin{bmatrix} I_n & O \\ O & -I_n \end{bmatrix}\mathbf{x}_i.$$

If we define $\mathbf{x}_i^T \equiv [\mathbf{w}_i^T, \mathbf{v}_i^T]$, note that the matrix $K(\mathbf{x}_i, F)$ in (4a) has the form

$$\text{(4b)} \qquad \begin{bmatrix} L(\mathbf{w}_i) & O \\ L(\mathbf{v}_i) & O \end{bmatrix} \in \mathbf{R}^{2n \times 2n}, \qquad O \in \mathbf{R}^{n \times n},$$

where $L(\mathbf{w}_i)$ and $L(\mathbf{v}_i)$ are lower triangular Toeplitz matrices with first columns $\mathbf{w}_i$ and $\mathbf{v}_i$.

Given a displacement representation of E, we use a certain *generalized Schur algorithm* (see § 2) to successively compute displacement representations of the Schur complements of all the leading principal submatrices in E. For the above example, n steps of the generalized Schur algorithm will yield

$$\begin{bmatrix} O & O \\ O & (T^TT)^{-1} \end{bmatrix} = \sum_{i=1}^{2} K(\mathbf{u}_i, F)K^T(\mathbf{u}_i, F) - \sum_{i=3}^{4} K(\mathbf{u}_i, F)K^T(\mathbf{u}_i, F),$$

where the top n elements of $\mathbf{u}_i$ are zero. Therefore, if we denote the bottom n elements of $\mathbf{u}_i$ as $\mathbf{u}_{2,i}$, we can have the displacement representation

$$(T^TT)^{-1} = \sum_{i=1}^{2} L(\mathbf{u}_{2,i})L^T(\mathbf{u}_{2,i}) - \sum_{i=3}^{4} L(\mathbf{u}_{2,i})L^T(\mathbf{u}_{2,i}).$$

Now, the generalized Schur algorithm, which is a two-term polynomial recursion, can be implemented in a divide-and-conquer fashion with $O(\alpha^3 f(n) \log n)$ flops, where $f(n)$ denotes the number of operations for the multiplication of two polynomials. Therefore, if the multiplication of two polynomials is done again by divide-and-conquer, i.e., by using fast convolution algorithms, then the overall computation requires $O(\alpha^3 n \log^2 n)$ flops. Once we have a displacement representation of the desired Schur complement S, the matrix-vector multiplication, $S\mathbf{b}$, can be done with $O(\alpha n \log n)$ flops using fast convolutions. As an example, we can obtain the least squares solution for the Toeplitz system

$$T\mathbf{x} = \mathbf{b}, \qquad T \in \mathbf{R}^{m \times n}, \qquad m \geq n$$

as follows:

(i) Form $T^T\mathbf{b}$ using two fast convolutions,

(ii) Obtain a displacement representation of $(T^TT)^{-1}$ using the divide-and-conquer version of the generalized Schur algorithm,

(iii) Form $(T^TT)^{-1}(T^T\mathbf{b})$ using eight fast convolutions.

If we had obtained the displacement representation of $(T^TT)^{-1}T^T$ directly (using E_3), then step (i) above would not be needed.

2. Generalized Schur algorithm. After a brief review of basic concepts and definitions, we shall describe the generalized Schur algorithm of references [8], [9], and [15], but in a polynomial form important for the divide-and-conquer implementations. We shall need to recall some definitions and basic properties.

Generators of matrices. Let F^f and F^b be nilpotent matrices. The matrix

$$\nabla_{(F^f,F^b)}A \equiv A - F^f A F^{bT}$$

is called the *displacement* of A with respect to the *displacement operators* $\{F^f, F^b\}$. Define the (F^f, F^b)-displacement rank of A as rank $[\nabla_{(F^f,F^b)}A]$. Any matrix pair $\{X, Y\}$ such that

$$(5) \qquad \nabla_{(F^f,F^b)}A = XY^T, \qquad X \equiv [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\alpha], \qquad Y \equiv [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_\alpha]$$

is called a (*vector form*) *generator* of A with respect to $\{F^f, F^b\}$. The generator will be said to have *length* α. If the length α is equal to the displacement rank of A, we say that the generator is *minimal*. A generator such as $Y = X\Sigma$, where Σ is a diagonal matrix with 1 or -1 along the diagonal, is called a *symmetric generator*.

The following lemma [15], [16] establishes the connection between generators and displacement representations.

LEMMA. *Let E be an $m \times n$ matrix. If F^f and F^b are nilpotent, then the equation $\nabla_{(F^f,F^b)}E = \sum_1^\alpha \mathbf{x}_i\mathbf{y}_i^T$ has the unique solution $E = \sum_1^\alpha K(\mathbf{x}_i, F^f)K^T(\mathbf{y}_i, F^b)$, where $K(\mathbf{x}_i, F^f) \equiv [\mathbf{x}_i, F^f\mathbf{x}_i, \cdots, F^{f(n-1)}\mathbf{x}_i]$ and $K(\mathbf{y}_i, F^b) \equiv [\mathbf{y}_i, F^b\mathbf{y}_i, \cdots, F^{b(n-1)}\mathbf{y}_i]$.*

Choice of displacement operators. The generalized Schur algorithm operates with generators, and needs $O(\alpha mn)$ flops for sequential implementation and $O(\alpha^3 n \log^2 n)$ for divide-and-conquer implementation. Therefore, for a given matrix A, we should try to choose the displacement operators that give the smallest α. If the matrix A is an $n \times n$ Toeplitz matrix, the appropriate displacement operator F is Z_n, an $n \times n$ shift matrix. If A has some near-Toeplitz structure, then F would have forms such as

$$F = Z_n \oplus Z_m, \qquad F = \bigoplus_{i=1}^{n} Z_{n_i}, \qquad F = Z_n^\beta,$$

where $\oplus$ denotes the *direct sum*, $Z_n \oplus Z_m \equiv \left[\begin{smallmatrix} Z_n & O \\ O & Z_m \end{smallmatrix}\right]$, and $\oplus_{i=1}^n$ denotes the concatenated direct sum.

Example 1. Let $T = (t_{i-j})$ be an $m \times n$ pre- and post-windowed scalar Toeplitz matrix, i.e., $t_{i,j} = 0$ if $j > i$ or $i > m - n + j$ with $m \geq n$. Then it is easy to check that the matrix $C = (c_{i-j}) \equiv T^T T$ is also an (unwindowed) Toeplitz matrix, and with respect to $\{Z_n \oplus Z_n, Z_n \oplus Z_m\}$, E_3 in (3) has a generator $\{X, Y\}$ of length two, where

$$\mathbf{x}_1 = [c_0, c_1, \cdots, c_{n-1}, -1, 0, \cdots, 0]^T / c_0^{1/2},$$

$$\mathbf{x}_2 = [0, c_1, \cdots, c_{n-1}, -1, 0, \cdots, 0]^T / c_0^{1/2},$$

$$\mathbf{y}_1 = [c_0, c_1, \cdots, c_{n-1}, t_0, t_1, \cdots t_{m-n}, 0, \cdots, 0]^T / c_0^{1/2},$$

$$\mathbf{y}_2 = -[0, c_1, \cdots, c_{n-1}, t_0, t_1, \cdots t_{m-n}, 0, \cdots, 0]^T / c_0^{1/2}. \qquad \square$$

Example 2. If T is a Toeplitz-block matrix, i.e.,

$$(6) \qquad T = \begin{bmatrix} T_{1,1} & T_{1,2} \cdots T_{1,N} \\ T_{2,1} & T_{2,2} \cdots T_{2,N} \\ \vdots & \vdots \\ T_{M,1} & T_{M,2} \cdots T_{M,N} \end{bmatrix} \in \mathbf{R}^{m \times n}, \qquad T_{i,j} = \text{scalar } m_i \times n_j \text{ Toeplitz matrix},$$

then for the matrices E in (3), we choose [9], [15] the following displacement operators:

$$(7a) \qquad E_1: F^f = \left[\bigoplus_{i=1}^{M} Z_{m_i}\right] \oplus F_1, \qquad F^b = \left[\bigoplus_{i=1}^{N} Z_{n_i}\right] \oplus F_1, \qquad m = n,$$

$$(7b) \qquad E_2: F^f = \left[\bigoplus_{i=1}^{N} Z_{n_i}\right] \oplus F_1, \qquad F^b = \left[\bigoplus_{i=1}^{N} Z_{n_i}\right] \oplus F_1,$$

$$(7c) \qquad E_3: F^f = \left[\bigoplus_{i=1}^{N} Z_{n_i}\right] \oplus F_1, \qquad F^b = \left[\bigoplus_{i=1}^{N} Z_{n_i}\right] \oplus \left[\bigoplus_{i=1}^{M} Z_{m_i}\right],$$

where F_1 can be either Z_n or $\oplus_{i=1}^{N} Z_{n_i}$. However, for the divide-and-conquer implementation, we prefer to choose $\oplus_{i=1}^{N} Z_{n_i}$; see the remark in § 4.

Example 3. On the other hand, if the matrix T in (3) is a block-Toeplitz matrix with $\beta \times \beta$ blocks,

$$(8) \qquad T = \begin{bmatrix} B_0 & B_{-1} & \cdot & B_{-N+1} \\ B_1 & B_0 & \cdot & B_{-N+2} \\ \cdot & \cdot & \cdot & \cdot \\ B_{M-1} & B_{M-2} & \cdot & B_{-N+M} \end{bmatrix} \in \mathbf{R}^{m \times n}, \quad B_k \in \mathbf{R}^{\beta \times \beta}, \quad m \equiv M\beta, \quad n \equiv N\beta,$$

then for the extended matrices E, we should choose [8], [9] the displacement operators

$$(9) \qquad F^f = Z_n^{\beta} \oplus Z_n^{\beta}, \qquad F^b = Z_n^{\beta} \oplus Z_m^{\beta},$$

where for E_1 we assumed that T is a square $n \times n$ matrix.

Generators of the above and other extended block-Toeplitz or Toeplitz-block matrices can be found in [8].

Polynomial form of generators. In general, the displacement operators F^f and F^b for both extended block-Toeplitz matrices and extended Toeplitz-block matrices have the form

$$(10) \qquad F = \bigoplus_{i=1}^{N} Z_{n_i}^{\beta}, \qquad n \equiv \sum_{i=1}^{N} n_i.$$

We shall say that the displacement operator F in (10) has N *sections*. One of the key operations in generalized Schur algorithms is matrix-vector multiplication $F\mathbf{v}$, i.e., a *sectioned shift* operation. With the polynomial representation of vectors, the shift operations has a nice algebraic expression. For a given vector $\mathbf{v}$, let $v(z)$ denote the polynomial whose coefficient for the term z^i is the $(i+1)$st component of the vector, i.e.,

$$(11) \qquad \mathbf{v} = [v_0, v_1, v_2, \cdots, v_{n-1}]^T \leftrightarrow v(z) = v_0 + v_1 z + v_2 z^2 + \cdots v_{n-1} z^{n-1}.$$

Then,

$$Z_n \mathbf{v} \equiv \mathbf{v}' = [0, v_0, v_1, \cdots, v_{n-2}]^T \leftrightarrow v(z) z \bmod z^n.$$

In general, for the matrix whose displacement operator is the F in (10), let us define integers $\{\delta_i\}$ by

$$\delta_i = \sum_{k=1}^{i} n_k, \qquad \delta_1 < \delta_2 < \cdots < \delta_N.$$

Let $v(z)$ and $\theta(z)$ be polynomials of degree less than or equal to $n-1$, and define the degree at most $(n_i - 1)$ polynomial, $v_i(z)$, by

$$(12a) \qquad v(z) = v_1(z) + z^{\delta_1} v_2(z) + z^{\delta_2} v_3(z) + \cdots + z^{\delta_{N-1}} v_{n_N}(z).$$

Given two polynomials $v(z)$ and $\theta(z)$, and the displacement operator F in (10), the (*polynomial form*) *displacement operator* $\circledast_F$ is defined by the following operation:

$$(12b) \qquad v(z)\circledast_F\theta(z)\equiv r(z)\equiv r_1(z)+z^{\delta_1}r_2(z)+z^{\delta_2}r_3(z)+\cdots+z^{\delta_N-1}r_{n_N}(z),$$

where

$$(12c) \qquad r_i(z)\equiv v_i(z)\theta(z^{\beta}) \bmod z^{n_i},$$

i.e., $r_i(z)$ is the polynomial $v_i(z)\theta(z^{\beta})$ after chopping off the higher degree terms, so that $r_i(z)$ has the degree at most (n_i-1).

Let

$$X=[\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_\alpha], \qquad Y=[\mathbf{y}_1,\mathbf{y}_2,\cdots,\mathbf{y}_\alpha]$$

be a generator of a matrix A with respect to certain $\{F^f, F^b\}$, and let

$$\mathbf{x}_i\leftrightarrow x_i(z), \qquad \mathbf{y}_i\leftrightarrow y_i(w).$$

Then we call the pair of polynomial vectors $\{X(z), Y(w)\}$, where

$$X(z)\equiv[x_1(z),x_2(z),\cdots,x_\alpha(z)], \qquad Y(w)\equiv[y_1(w),y_2(w),\cdots,y_\alpha(w)],$$

a (*polynomial form*) *generator of* A, with respect to (*polynomial form*) *displacement operator* $\{\circledast_{F^f}, \circledast_{F^b}\}$.

Example 1 (*continued*). The matrix E_3 in (3) has a generator $\{X(z), Y(w)\}$ with respect to $\{\circledast_{F^f}, \circledast_{F^b}\}$, where $F^f = Z_n \oplus Z_n$, $F^b = Z_n \oplus Z_m$, and

$$x_1(z)=[c_0+c_1z+\cdots+c_{n-1}z^{n-1}-z^n]c_0^{-1/2},$$

$$x_2(z)=[c_1z+cz^2+\cdots+c_{n-1}z^{n-1}-z^n]c_0^{-1/2},$$

$$y_1(w)=[c_0+c_1w+\cdots+c_{n-1}w^{n-1}+t_0w^n+t_1w^{n+1}+\cdots+t_{m-n}w^m]c_0^{-1/2},$$

$$y_2(w)=-[c_1w+\cdots+c_{n-1}w^{n-1}+t_0w^n+t_1w^{n+1}+\cdots+t_{m-n}w^m]c_0^{-1/2}.$$

Also note that

$$x_1(z)\circledast_{F^f}z=[c_0z+c_1z^2+\cdots+c_{n-2}z^{n-1}-z^{n+1}]c_0^{-1/2},$$

$$y_1(w)\circledast_{F^b}w$$

$$=[c_0w+c_1w^2+\cdots+c_{n-2}w^{n-1}+t_0w^{n+1}+t_1w^{n+2}+\cdots+t_{m-n}w^{m+1}]c_0^{-1/2}. \quad \square$$

Next we note that for given vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\mathbf{a}^T\mathbf{b}\neq 0$, we can always find [8] matrices Θ and Ψ such that

$$(13) \qquad \mathbf{a}^T\Theta=[a_1',0,0,\cdots,0], \qquad \mathbf{b}^T\Psi=[b_1',0,0,\cdots,0], \qquad \Theta\cdot\Psi^T=I,$$

and therefore, $\mathbf{a}^T\mathbf{b}=a_1'b_1'$. We define polynomial matrices $\Theta(z)$ and $\Psi(w)$ by

$$(14) \qquad \Theta(z)\equiv\Theta\begin{bmatrix} z & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \qquad \Psi(w)\equiv\Psi\begin{bmatrix} w & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}.$$

We also remark that if $\mathbf{a}=\mathbf{b}$, then we could choose $\Psi(w)=\Theta(w)$, and if $\mathbf{b}=\Sigma\mathbf{a}$, where $\Sigma\equiv I_p\oplus -I_q$, then $\Psi(w)=\Theta(w)\Sigma$, so that we only need to find, and post-multiply by, $\Theta(z)$.

Generalized Schur algorithm. Let a matrix E have a generator $\{X_0(z), Y_0(w)\}$ with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, and define $E_{i,j}$ by

$$E = \begin{bmatrix} E_{1,1} & E_{1,2} \\ E_{2,1} & E_{2,2} \end{bmatrix} \in \mathbf{R}^{m \times n},$$

where $E_{1,1}$ is a $k \times k$ strongly nonsingular matrix, i.e., the one with all nonsingular leading submatrices. The k-step generalized Schur algorithm [8], [9], [15] presented below in polynomial form gives a generator of the matrix

$$\begin{bmatrix} O & O \\ O & S \end{bmatrix}, \qquad S \equiv E_{2,2} - E_{2,1} E_{1,1}^{-1} E_{1,2} \in \mathbf{R}^{(m-k)\times(n-k)},$$

with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, or equivalently, a generator of S with respect to $\{\otimes_{\bar{F}^f}, \otimes_{\bar{F}^b}\}$, where $\bar{F}^f$ and $\bar{F}^b$ denote the trailing square submatrices of size $(m-k)$ and $(n-k)$ of F^f and F^b, respectively.

ALGORITHM (k-step generalized Schur algorithm).
Input: Generator of E, $\{X_0(z), Y_0(w)\}$; displacement operator $\{\otimes_{F^f}, \otimes_{F^b}\}$;
Number of steps k.
Output: Generator of S $\{X_k(z), Y_k(w)\}$
Procedure GeneralizedSchur
begin
for $i := 0$ to $k-1$ **do begin**
$\mathbf{a}^T := [z^{-i} X_i(z)]_{z=0}$;
$\mathbf{b}^T := [z^{-i} Y_i(z)]_{z=0}$;
Find $\Theta_i(z)$ and $\Psi_i(w)$ to transform $\mathbf{a}^T$ and $\mathbf{b}^T$ such as (13);
$X_{i+1}(z) = X_i(z) \otimes_{F_f} \Theta_i(z)$; $\quad Y_{i+1}(w) = Y_i(w) \otimes_{F_b} \Psi_i(w)$
end
return $\{X_k(z), Y_k(w)\}$
end

Remark. The polynomial vectors, $X_i(z)$ and $Y_i(w)$, have degrees $m-1$ and $n-1$, respectively, for all i. Each step eliminates the nonzero lowest degree term, and therefore the terms of $X_i(z)$ and $Y_i(w)$ whose degrees are less than z^i and w^i are zeros.

By applying the generalized Schur algorithm we can obtain generators, or equivalently displacement representations, for various interesting Schur complements.

3. Divide-and-conquer implementation. The (sequential) k-step generalized Schur algorithm in § 2 can also be implemented efficiently using the divide-and-conquer approach. We shall only explain how to find $X_k(z)$; essentially the same argument applies for $Y_k(w)$.

Let us define $\Theta_{p:q}(z)$ and $X_{p:q}(z)$ by

$$\Theta_{p:q}(z) \equiv \Theta_p(z)\Theta_{p+1}(z) \cdots \Theta_q(z),$$

$$X_{p:q}(z) \equiv X_{0:q}(z) \otimes_{F^f} \Theta_{0:p-1}(z), \qquad X_{0:q}(z) \equiv X_0(z) \bmod z^{q+1},$$

where $0 \leqq p \leqq q$. The polynomial matrix $\Theta_{p:q}(z)$ has a degree $q-p+1$. The polynomial vector $X_{p:q}(z)$ has degree q, and is obtained by dropping from $X_p(z)$ all terms of degree higher than z^q. Also note the useful properties,

$$[x(z) \otimes_F \theta_1(z)] \otimes_F \theta_2(z) = x(z) \otimes_F [\theta_1(z)\theta_2(z)],$$

$$[x_1(z) + x_2(z)] \otimes_F \theta(z) = [x_1(z) \otimes_F \theta(z)] + [x_2(z) \otimes_F \theta(z)].$$

These properties and the fact that $\Theta_{p:q}(z)$ is completely determined by $X_{p:q}(z)$ allow a divide-and-conquer implementation of the generalized Schur algorithm.

Given $X_{p:q}(z)$, we can compute $\Theta_{p:q}(z)$ as follows. If $p = q$, then we are successful, and compute $\Theta_{p:p}(z) = \Theta_p(z)$. Otherwise, we choose an "appropriate" (see § 4) *division point* r such that $p < r < q$, and try to solve the smaller subproblem of finding $\Theta_{p:r-1}(z)$, given $X_{p:r-1}(z)$. Once we know $\Theta_{p:r-1}(z)$, we can compute $X_{r:q}(z)$ by

$$X_{r:q}(z) = X_{0:q}(z) \otimes_{F/} \Theta_{0:r-1}(z) = [X_{0:q}(z) \otimes_{F/} \Theta_{0:p-1}(z)] \otimes_{F/} \Theta_{p:r-1}(z) \tag{15a}$$

$$= X_{p:q}(z) \otimes_{F/} \Theta_{p:r-1}(z). \tag{15b}$$

Now we again try to find $\Theta_{r:q}(z)$ given $X_{r:q}(z)$. After we obtain $\Theta_{r:q}(z)$, we can combine the two results, $\Theta_{p:r-1}(z)$ and $\Theta_{r:q}(z)$, by multiplication,

$$\Theta_{p:q}(z) = \Theta_{p:r-1}(z)\Theta_{r:q}(z). \tag{16}$$

Programming details of the above *recursive generalized Schur algorithm* are shown in the Appendix.

The previous recursive description can be visualized nonrecursively using *trees* (See Figs. 1 and 2). Each node in the tree is annotated with the *rules*: "find," "apply," and "combine,"

$f_{p:p}$: Find $\Theta_{p:p}(z)$,

$a_{p:q}$: $Y_{r:q}(z) := X_{p:q}(z) \otimes_F \Theta_{p:r-1}(z)$,

$c_{p:q}$: $\Theta_{p:q}(z) := \Theta_{p:r-1}(z)\Theta_{r:q}(z)$.

We traverse the tree in *post-order* (i.e., follow the order labeled on each node of the tree), and evaluate the rules.

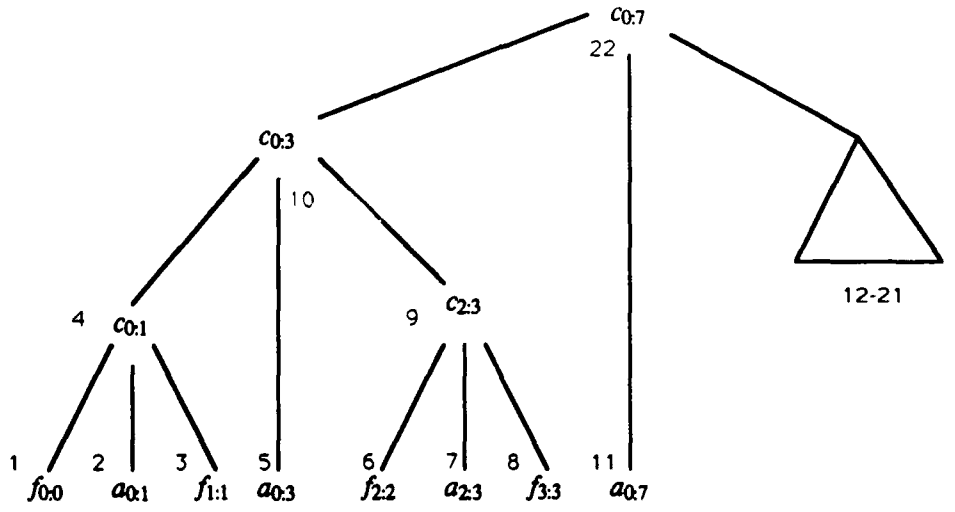

FIG. 1. *Sequence of computations for Example* 4.

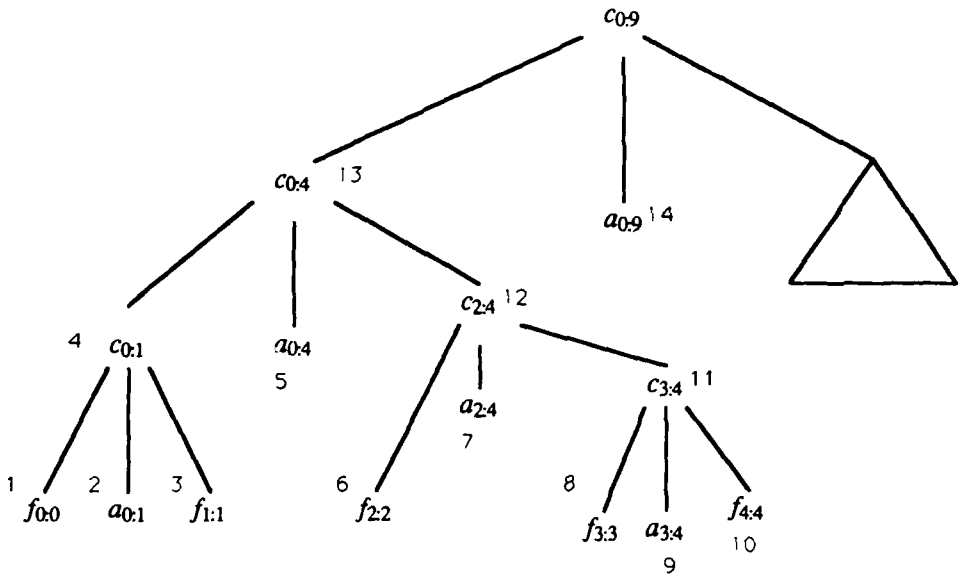

FIG. 2. *Sequence of computations for Example* 5.

Now, we shall consider two examples in detail.

Example 4. Pseudoinverse of pre- and post-windowed Toeplitz matrices. Consider the matrix E_3 in Example 1, where

$$T^TT=\begin{bmatrix}16 & 8 & 4 & 1\\ 8 & 16 & 8 & 4\\ 4 & 8 & 16 & 8\\ 1 & 4 & 8 & 16\end{bmatrix},\qquad T^T=\begin{bmatrix}3 & 2 & 1 & 1 & -1 & 0 & 0 & 0\\ 0 & 3 & 2 & 1 & 1 & -1 & 0 & 0\\ 0 & 0 & 3 & 2 & 1 & 1 & -1 & 0\\ 0 & 0 & 0 & 3 & 2 & 1 & 1 & -1\end{bmatrix}.$$

We would like to find a displacement representation of $(T^TT)^{-1}T^T$. This can be done by the four-step recursive generalized Schur algorithm. The input to the algorithm is a generator $\{X_0(z), Y_0(w)\}$ of

$$E_3=\begin{bmatrix}T^TT & T^T\\ -I & O\end{bmatrix},$$

with respect to $\{\circledast_{f^f}, \circledast_{F^b}\}$, where $F^f=Z_n\oplus Z_n$, $F^b=Z_n\oplus Z_m$. The output $\{X_4(z), Y_4(w)\}$ is a generator of $(T^TT)^{-1}T^T$ with respect to $\{\circledast_{Z_n}, \circledast_{Z_m}\}$. The computational sequence is illustrated in Fig. 1, where it is assumed that the division points were chosen successively by two, one, and three.

[1] $f_{0:0}:\Theta_{0:0}(z)=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}z & \\ & 1\end{bmatrix}$ because $X_{0:0}(z)=[4,0]$.

[2] $a_{0:1}:X_{1:1}(z)=X_{0:1}(z)\circledast_{F^f}\Theta_{0:0}(z)=[4+2z,2z]\circledast_{F^f}\Theta_{0:0}(z)=[4z,-2z]$.

[3] $f_{1:1}:\Theta_{1:1}(z)=\frac{2}{\sqrt{3}}\cdot\begin{bmatrix}1 & +\frac{1}{2}\\ -\frac{1}{2} & -1\end{bmatrix}\begin{bmatrix}z & \\ & 1\end{bmatrix}$.

[4] $c_{0:1}:\Theta_{0:1}(z)=\Theta_{0:0}(z)\Theta_{1:1}(z)=\frac{2}{\sqrt{3}}\cdot\begin{bmatrix}z^2 & -z/2\\ -z/2 & 1\end{bmatrix}$.

[5] $a_{0:3}: X_{2:3}(z) = X_{0:3}(z) \circledast_{Ff} \Theta_{0:1}(z) = \frac{2}{\sqrt{3}} \cdot [3z^2 + 3z^3/2, -z^3/4].$

[6] $f_{2:2}: \Theta_{2:2}(z) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$ because $X_{2:2}(z) = \frac{2}{\sqrt{3}} \cdot [3z^2, 0].$

[7] $a_{2:3}: X_{3:3}(z) = X_{2:3}(z) \circledast_{Ff} \Theta_{2:2}(z) = \frac{2}{\sqrt{3}} \cdot [3z^3, z^3/4].$

[8] $f_{3:3}: \Theta_{3:3}(z) = \frac{12}{\sqrt{143}} \cdot \begin{bmatrix} 1 & \frac{1}{12} \\ -\frac{1}{12} & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}.$

[9] $c_{2:3}: \Theta_{2:3}(z) = \Theta_{2:2}(z)\Theta_{3:3}(z) = \frac{12}{\sqrt{143}} \cdot \begin{bmatrix} z & z/12 \\ \frac{1}{12} & 1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}.$

[10] $c_{0:3}: \Theta_{0:3}(z) = \Theta_{0:1}(z)\Theta_{2:3}(z) = \frac{24}{\sqrt{3}\sqrt{143}} \cdot \begin{bmatrix} z^4 - z^2/24 & z^3/12 - z/12 \\ -z^3/12 + z/12 & -z^2/24 + 1 \end{bmatrix}.$

[11] $a_{0:7}: X_{4:7}(z) = [4 + 2z + z^2 + z^3/4 - z^4/4, 2z + z^2 + z^3/4 - z^4/4] \circledast_{Ff} \Theta_{0:3}(z)$

$$= [(4 + 2z + z^2 + z^3/4, 2z + z^2 + z^3/4) - z^4(\tfrac{1}{4}, \tfrac{1}{4})] \circledast_{Ff} \Theta_{0:3}(z)$$

$$= -z^4[(\tfrac{1}{4}, \tfrac{1}{4})\Theta_{0:3}(z) \bmod z^4]$$

$$= -\frac{6z^4}{\sqrt{3}\sqrt{143}}[z/12 - z^2/24 - z^3/2, 1 - z/2 - z^2/24 + z^3/12].$$

Because T^TT is symmetric, $\Psi_{0:3}(w) = \Theta_{0:3}(w)\Sigma$, where $\Sigma = 1 \oplus -1$, and therefore,

$$Y_{4:13}(w) = [(4 + 2z + z^2 + z^3/4) + z^4(3/4 + z/2 + z^2/4 - z^4/4),$$
$$(2z + z^2 + z^3/4) + z^4(3/4 + z/2 + z^2/4 + z^3/4 - z^4/4)] \circledast_{Fb} \Theta_{0:3}(w)\Sigma$$
$$= \frac{z^4 6}{\sqrt{3}\sqrt{143}}[1/4z + z^2/24 - 3z^3/2 + 49z^4/24 + 11z^5/8 + 13z^6/24 + 3z^7/2,$$
$$-3 - z/2 + z^2/8 - 2z^3/3 + 11z^4/8 - 13/24z^5 - z^6/8 - z^7/12].$$

Therefore,

$$(T^TT)^{-1}T^T = \gamma^2[L(\mathbf{x}_1)L^T(\mathbf{y}_1) + L(\mathbf{x}_2)L^T(\mathbf{y}_2)], \qquad \gamma = \frac{6}{\sqrt{3}\sqrt{143}},$$

where $L(\mathbf{x}_i)$ and $L(\mathbf{y}_i)$ are the lower triangular Toeplitz matrices whose first columns are $\mathbf{x}_i$ and $\mathbf{y}_i$, respectively, and

$$\mathbf{x}_1 = [0, -\tfrac{1}{12}, \tfrac{1}{24}, \tfrac{1}{2}]^T,$$
$$\mathbf{x}_2 = [-1, \tfrac{1}{2}, \tfrac{1}{24}, -\tfrac{1}{12}]^T,$$
$$\mathbf{y}_1 = [0, \tfrac{1}{4}, \tfrac{1}{24}, -\tfrac{3}{2}, \tfrac{49}{24}, \tfrac{11}{8}, \tfrac{13}{24}, \tfrac{3}{2}]^T,$$
$$\mathbf{y}_2 = [-3, -\tfrac{1}{2}, \tfrac{1}{8}, -\tfrac{2}{3}, \tfrac{11}{8}, -\tfrac{13}{24}, -\tfrac{1}{8}, \tfrac{1}{12}]^T.$$

Remark 1. For a symmetric generator of length two with $\beta = 1$, the 2×2 polynomial matrix $\Theta(z)$ in (14) can have the form (hyperbolic reflection)

$$\Theta_i(z) = \begin{bmatrix} ch_i z & sh_i \\ -sh_i z & -ch_i \end{bmatrix}, \qquad ch_i^2 - sh_i^2 = 1.$$

Let

$$\Theta_{p:q}(z) \equiv \Theta_p(z)\Theta_{p+1}(z) \cdots \Theta_q(z) \equiv \begin{bmatrix} \Theta_{1,1}(z) & \Theta_{1,2}(z) \\ \Theta_{2,1}(z) & \Theta_{2,2}(z) \end{bmatrix}.$$

Then, by induction, we can easily prove that

$$z^{q-p+1}\Theta_{1,1}(z^{-1}) = (-1)^{q-p+1}\Theta_{2,2}(z), \qquad z^{q-p+1}\Theta_{1,2}(z^{-1}) = (-1)^{q-p+1}\Theta_{2,1}(z).$$

Therefore, we need to compute and store only two entries of $\Theta_{p:q}(z)$.

Remark 2. For an unwindowed scalar Toeplitz matrix, the matrix E_2 in (3) has displacement rank four, whereas the matrix E_3 has displacement rank five. Therefore, when we solve Toeplitz least squares problems, it is more efficient to find a displacement representation of $(T^T T)^{-1}$ rather than of $(T^T T)^{-1} T^T$. With the notation in (4), the matrix E_2 for an unwindowed scalar Toeplitz matrix $T = (t_{i-j}) \in \mathbf{R}^{m \times n}$ $(m \geqq n)$ has a generator [15],

$$\mathbf{w}_1 = T^T \mathbf{t}_1 / \|\mathbf{t}_1\|, \qquad \mathbf{w}_2 = \mathbf{t}_2, \qquad \mathbf{w}_3 = Z_n Z_n^T \mathbf{w}_1, \qquad \mathbf{w}_4 = Z_n \mathbf{l},$$

$$\mathbf{t}_1 \equiv [t_0, t_1, \cdots, t_{m-1}]^T, \qquad \mathbf{t}_2 \equiv [0, t_{-1}, \cdots, t_{1-n}]^T, \qquad \mathbf{l} \equiv [t_{m-1}, \cdots, t_{m-n}]^T,$$

$$\mathbf{v}_1 = \mathbf{v}_3 = \mathbf{e}_1 / \|\mathbf{t}_1\|, \qquad \mathbf{v}_2 = \mathbf{v}_4 = 0,$$

where $\|\cdot\|$ denotes the Euclidean norm, and $\mathbf{e}_1$ is the vector with one in the first position, and zeros elsewhere.

Example 5. Displacement representation for the inverse of a Sylvester matrix. Let T denote the following Sylvester matrix,

$$T \equiv \begin{bmatrix} 2 & 0 & 0 & 1 & 0 \\ 1 & 2 & 0 & 2 & 1 \\ 3 & 1 & 2 & 1 & 2 \\ 0 & 3 & 1 & 1 & 1 \\ 0 & 0 & 3 & 0 & 1 \end{bmatrix} \tag{17}$$

and suppose that it is desired to obtain a displacement representation of T^{-1}. Then the appropriate extended matrix is

$$E_1 = \begin{bmatrix} T & I \\ -I & O \end{bmatrix}, \tag{18}$$

and it is easy to see that the following $\{X_0(z), Y_0(w)\}$ is a generator of E_1 with respect to $\{\circledast_{F^f}, \circledast_{F^b}\}$, where $F^f = Z_5 \oplus Z_5$, $F^b = Z_3 \oplus Z_2 \oplus Z_5$;

$$X_0(z) \equiv [x_1(z), x_2(z), x_3(z)], \qquad Y_0(w) \equiv [y_1(w), y_2(w), y_3(w)],$$

$$x_1(z) = 2 + z + 3z^2 - z^5, \qquad x_2(z) = 1 + 2z + z^2 + z^3 - z^8, \qquad x_3(z) = 1, \tag{19a}$$

$$y_1(w) = 1, \qquad y_2(w) = w^3, \qquad y_3(w) = w^5. \tag{19b}$$

Now the five-step recursive generalized Schur algorithm gives a desired generator of T^{-1}, with respect to $\{Z_5, Z_5\}$, and a possible computational sequence is shown in Fig. 2, where the division points are chosen successively as two, one, three, and four.

[1] $$f_{0:0}:\Theta_{0:0}(z)=\begin{bmatrix} z & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \Psi_{0:0}(w)=\begin{bmatrix} w & 0 & 0 \\ w/2 & 1 & 0 \\ w/2 & 0 & 1 \end{bmatrix}.$$

[2] $$a_{0:1}:X_{1:1}(z)=[2z, 3z/2, -z/2], \qquad Y_{1:1}(w)=[w,0,0].$$

[3] $$f_{1:1}:\Theta_{1:1}(z)=\begin{bmatrix} z & -\frac{3}{4} & -\frac{1}{4} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \Psi_{1:1}(w)=\begin{bmatrix} w & 0 & 0 \\ 3w/4 & 1 & 0 \\ -w/4 & 0 & 1 \end{bmatrix}.$$

[4] $$c_{0:1}:\Theta_{0:1}(z)=\begin{bmatrix} z^2 & -3z/4-1/2 & z/4-1/2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\Psi_{0:1}(w)=\begin{bmatrix} w^2 & 0 & 0 \\ w^2/2+3w/4 & 1 & 0 \\ w^2/2-w/4 & 0 & 1 \end{bmatrix}.$$

[5] $$a_{0:4}:X_{2:4}(z)=[2z^2+z^3+3z^4, -5z^2/4-5z^3/4, -5z^2/4+3z^3/4],$$

$$\begin{aligned} Y_{2:4}(w) &= Y_{0:4}(w)\circledast_{F^b}\Psi_{0:1}(w) \\ &= [(1,0,0)\Psi_{0:1}(w) \bmod w^3]+w^3[(0,w,0)\Psi_{0:1}(w) \bmod w^2] \\ &= [w^2+3w^4/4, w^3, 0]. \end{aligned}$$

[6] $$f_{2:2}:\Theta_{2:2}(z)=\begin{bmatrix} z & \frac{5}{8} & \frac{5}{8} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \Psi_{2:2}(w)=\begin{bmatrix} w & 0 & 0 \\ -5w/8 & 1 & 0 \\ -5w/8 & 0 & 1 \end{bmatrix}.$$

[7] $$a_{2:4}:X_{3:4}(z)=[2z^3+z^4, -5z^3/8+15z^4/8, 11z^3/8+15z^4/8],$$

$$\begin{aligned} Y_{3:4}(w) &= Y_{2:4}(w)\circledast_{F^b}\Psi_{2:2}(w) \\ &= [(w^2,0,0)\Psi_{2:2}(w) \bmod w^3]+w^3[(3w/4,1,0)\Psi_{2:2}(w) \bmod w^2] \\ &= [-5w^4/8, w^3, 0]. \end{aligned}$$

[8] $$f_{3:3}:\Theta_{3:3}(z)=\begin{bmatrix} 0 & 1 & 0 \\ z & \frac{16}{5} & \frac{11}{5} \\ 0 & 0 & 1 \end{bmatrix}, \qquad \Psi_{3:3}(w)=\begin{bmatrix} -16w/5 & 1 & 0 \\ w & 0 & 0 \\ -11w/5 & 0 & 1 \end{bmatrix}.$$

[9] $$a_{3:4}:X_{4:4}(z)=[-5z^4/8, 7z^4, 6z^4], \qquad Y_{4:4}(w)=[w^4, -5w^4/8, 0].$$

[10] $$c_{4:4}:\Theta_{4:4}(z)=\begin{bmatrix} z/(2\sqrt{2}) & 28/(5\sqrt{2}) & \frac{6}{5} \\ -5z/(16\sqrt{2}) & 1/(2\sqrt{2}) & -\frac{3}{4} \\ 0 & 0 & 1 \end{bmatrix},$$

$$\Psi_{4:4}(w)=\begin{bmatrix} w(2\sqrt{2}) & 5/(16\sqrt{2}) & 0 \\ -28w/(5\sqrt{2}) & 1/(2\sqrt{2}) & 0 \\ -12\sqrt{2}w/5 & 0 & 1 \end{bmatrix}.$$

Operations [11]–[13] are obvious. After evaluating, $c_{3:4}$, $c_{2:4}$, and $c_{0:4}$, we obtain $\Theta_{0:4}(z)$ and $\Psi_{0:4}(w)$, and finally,

[14] $$a_{0:9}: X_{0:9}(z) = [x_1(z), x_2(z), x_3(z)] \otimes_{F^f} \Theta_{0:4}(z)$$
$$= z^5[(-1, -z^3, 0) \otimes_{F^f} \Theta_{0:4}(z)]$$
$$= z^5[(-1, -z^3, 0)\Theta_{0:4}(z) \bmod z^5] = z^5[u_1(z), u_2(z), u_3(z)],$$

where

$$u_1(z) = -z/(2\sqrt{2}) - z^2/(2\sqrt{2}) + z^3/\sqrt{2} + z^4/\sqrt{2},$$
$$u_2(z) = 4/(5\sqrt{2}) + 4z/\sqrt{2} + 16z^2/(5\sqrt{2}) - 28z^3/(5\sqrt{2}) - 28z^4/(5\sqrt{2}),$$
$$u_3(z) = 2/5 + z/5 + 2z^2/5 + z^3/5 - 6z^4/5.$$

$$Y_{0:9}(w) = [y_1(w), y_2(w), y_3(w)] \otimes_{F^b} \Psi_{0:4}(w)$$
$$= w^5[(0, 0, 1) \otimes_{F^b} \Psi_{0:4}(w)] = w^5[v_1(w), v_2(w), v_3(w)],$$

where

$$v_1(w) = -12\sqrt{2}w/5 + 12w^2/(5\sqrt{2}) + 12w^3/(5\sqrt{2}) - 12w^4/(5\sqrt{2}),$$
$$v_2(w) = -w/\sqrt{2} + w^2/(2\sqrt{2}) + w^3/(2\sqrt{2}) - w^4/(2\sqrt{2}),$$
$$v_3(w) = 1.$$

Therefore,

$$T^{-1} = L(\mathbf{u}_1)L^T(\mathbf{v}_1) + L(\mathbf{u}_2)L^T(\mathbf{v}_2) + L(\mathbf{u}_3)L^T(\mathbf{v}_3),$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the vectors whose jth component is the coefficient of z^{j-1} and w^{j-1} of $u_i(z)$ and $v_i(w)$, respectively. □

Remark 3. If we had chosen the displacement operator $F^f = Z_5 \oplus Z_3 \oplus Z_2$, $F^b = Z_3 \oplus Z_2 \oplus Z_5$ for the matrix T in (17) we would have the same generator (19) for E_1, but the obtained generator of T^{-1} would be the one with respect to $\{Z_3 \oplus Z_2, Z_5\}$ rather than with respect to $\{Z_5, Z_5\}$. The displacement ranks of T^{-1} with respect to both displacement operators are two, but the above procedure gives nonminimal generators of length three.

Remark 4. The following extended matrix:

(20) $$\begin{bmatrix} T & \mathbf{b} \\ -I & 0 \end{bmatrix}, \quad T = \text{Sylvester matrix}$$

also has a displacement rank of three. We could as well obtain the solution $T^{-1}\mathbf{b}$ directly by applying the recursive generalized Schur algorithm to (20): the last column of X, where $\{X, \mathbf{y}\}$ is the computed generator of $T^{-1}\mathbf{b}$ with respect to $\{Z_n, 1\}$, can be shown to be the solution $T^{-1}\mathbf{b}$.

4. Polynomial products with fast convolutions. The product of two polynomials of degree d_1 and d_2 can be performed efficiently using $d = d_1 + d_2 + 1$ point fast cyclic convolution algorithms [4]. A d-point fast cyclic convolution needs $O(d \log d)$ flops. Among others, Fast Fourier Transforms (FFTs) can be used for convolutions, and Ammar and Gragg [2] carefully examined the use of FFTs for a doubling algorithm for square Toeplitz systems of equations. We shall only consider the subtle complications that arise in the recursive generalized Schur algorithm in this paper.

The polynomial matrix-matrix product of (16) needs α^3 of $q-p$ point cyclic convolutions. The polynomial vector-matrix product of (15b) has α^2 of scalar polynomial products of the form, $x(z)\circledast_{F^f}\theta(z)$, where $x(z)$ is a polynomial with nonzero terms of $z^p, z^{p+1}, \cdots, z^q$. Let us assume that

$$0<\delta_1<\cdots<\delta_l\leqq p<\delta_{l+1}<\cdots<\delta_s\leqq r<\delta_{s+1}<\cdots<\delta_t\leqq q<\delta_{t+1}<\cdots<\delta_N.$$

Then

$$x'(z)\equiv x(z)\circledast_{F^f}\theta(z) \tag{21a}$$

$$=[z^{\delta_l}x_{l+1}(z)+z^{\delta_{l+1}}x_{l+2}(z)+\cdots+z^{\delta_s}x_{s+1}(z)$$

$$+\cdots+z^{\delta_t}x_{t+1}(z)]\circledast_{F^f}\theta(z) \tag{21b}$$

$$=[z^{\delta_l}x_{l+1}(z)+\cdots+z^{\delta_{s-1}}x_s(z)]\circledast_{F^f}\theta(z) \tag{22a}$$

$$+z^{\delta_s}[x_{s+1}(z)\theta(z^\beta) \bmod z^{n_{s+1}}] \tag{22b}$$

$$+z^{\delta_{s+1}}[x_{s+2}(z)\theta(z^\beta) \bmod z^{n_{s+2}}] \tag{22c}$$

$$\cdots$$

$$+z^{\delta_t}[x_{t+1}(z)\theta(z^\beta) \bmod z^{n_{t+1}}]. \tag{22d}$$

The terms in (22a) do not need to be computed because these terms will be summed to zeros after adding all the partial sums in the vector-matrix multiplication of (15b). Recall that $x_i(z)$ has degree n_i, and $\theta(z^\beta)$ has degree $\beta^{(q-p+1)}$. Therefore, the product $x_i(z)\theta(z^\beta)$ from (22b) to (22d) can be performed by

$2n_i+1$ point cyclic convolutions if degree $[\theta(z^\beta)]\geqq$ degree $[x_i(z)]$,

$n_i+\beta^{(q-p+1)}+1$ point cyclic convolutions if degree $[\theta(z^\beta)]<$ degree $[x_i(z)]$.

Remark 5. Note that two $d/2$ point convolutions take $cd\log(d/2)$ flops if one d point convolution takes $cd\log d$ flops. Therefore, the polynomial product (21) is more efficient for the displacement operator F^f with more sections, because such displacement operators break a long convolution into many smaller convolutions. Therefore, for a given matrix we prefer to choose a displacement operator with as many sections as possible, while keeping the displacement rank minimal. Also we remark that the first and last terms (22b) and (22d) need smaller point convolutions.

If the dimensions of the matrix are powers of two, then we can always choose the center division point $r=\lceil(p+q)/2\rceil$. This *balanced division* (or doubling) gives the least number of computations, in general. For this case, let $\eta\equiv p-q$, and $T(\eta)$ denote the number of computations for one recursion. Then

$$T(\eta)\leqq 2T(\eta/2)+W(\eta),\qquad W(\eta)\equiv O(\alpha^3\eta\log\eta),$$

and therefore, we can show [1] that the k-step recursion takes

$$T(k)\leqq O(\alpha^3 k\log^2 k).$$

However, in most cases the doubling is not possible, and for such circumstances, the desirable choice of r is such that $r-p$ and $q-r+1$ are highly composite numbers (so that fast convolution algorithms can be applied efficiently), as well as r is close to $(q-p)/2$ (so as to achieve balancing).

Matrix-vector products using displacement representation. The final step of finding solutions for linear equations is the matrix-vector multiplication $S\mathbf{b}$, given a displacement representation of $S \in \mathbf{R}^{m\times n}$,

$$S = \sum_{i=1}^{\alpha} K(\mathbf{x}_i, F^f)K^T(\mathbf{y}_i, F^b), \tag{23}$$

where the length α is a multiple of the block size β, $\alpha = \beta\delta$, say, and

$$F^f = \bigoplus_{i=1}^{M} Z^{\beta}_{m_i}, \quad F^b = \bigoplus_{i=1}^{N} Z^{\beta}_{n_i}, \quad m = \sum_{i=1}^{M} m_i, \quad n = \sum_{i=1}^{N} n_i.$$

The expression in (23) can be rewritten in the *block displacement form*

$$S = \sum_{i=1}^{\delta} K_\beta(X_i, F^f)K_\beta^T(Y_i, F^b), \qquad X_i \in \mathbf{R}^{m\times\beta}, \quad Y_i \in \mathbf{R}^{n\times\beta}, \tag{24}$$

where

$$K_\beta(X_i, F^f) = [X_i, F^f X_i, F^{f2}X_i, \cdots, F^{f[(m/\beta)-1]}X_i] \in \mathbf{R}^{m\times n}, \tag{25a}$$

$$K_\beta(Y_i, F^b) = [Y_i, F^b Y_i, F^{b2}Y_i, \cdots, F^{b[(n/\beta)-1]}Y_i] \in \mathbf{R}^{n\times n}. \tag{25b}$$

Furthermore, because F^f and F^b have M and N sections, respectively, (25a) and (25b) have the forms

$$K_\beta(X_i, F^f) = \begin{bmatrix} K_\beta(X_{1,i}, Z^{\beta}_{m_1}) & O \\ K_\beta(X_{2,i}, Z^{\beta}_{m_2}) & O \\ \cdot & \cdot \\ K_\beta(X_{M,i}, Z^{\beta}_{m_M}) & O \end{bmatrix}, \qquad K_\beta(Y_i, F^b) = \begin{bmatrix} K_\beta(Y_{1,i}, Z^{\beta}_{n_1}) & O \\ K_\beta(Y_{2,i}, Z^{\beta}_{n_2}) & O \\ \cdot & \cdot \\ K_\beta(Y_{N,i}, Z^{\beta}_{n_N}) & O \end{bmatrix},$$

where $K_\beta(X, Z^\beta)$ is the block lower triangular Toeplitz matrix with the first column block X. The matrix O denotes a null matrix of appropriate size such that $K_\beta(X_i, F^f)$ and $K_\beta(Y_i, F^b)$ are $m \times n$ and $n \times n$ matrices, respectively.

To see how to use convolutions for the product

$$K_\beta(X_i, F^f)K_\beta^T(Y_i, F^b)\mathbf{b}$$

it is enough to consider matrix-vector multiplications of the form $K_\beta(X, Z^\beta)\mathbf{b}$. Note that $K_\beta(X, Z^\beta)\mathbf{b}$ can be expressed as sum of β products of scalar lower triangular Toeplitz matrix and vectors. As an example,

$$\begin{bmatrix} a_0 & c_0 & & \\ a_1 & c_1 & & \\ a_2 & c_2 & a_0 & c_0 \\ a_3 & c_3 & a_1 & c_1 \end{bmatrix}\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ a_2 & a_1 & a_0 & \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix}\begin{bmatrix} b_0 \\ 0 \\ b_2 \\ 0 \end{bmatrix} + \begin{bmatrix} c_0 & & & \\ c_1 & c_0 & & \\ c_2 & c_1 & c_0 & \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix}\begin{bmatrix} b_1 \\ 0 \\ b_3 \\ 0 \end{bmatrix}. \tag{26}$$

The multiplications in the right sides of (26) can be done by fast convolutions, and therefore, so can the multiplication $S\mathbf{b}$.

5. Concluding remarks. We have presented $O(\alpha^3 n \log^2 n)$ algorithms for the determination of exact and least squares solutions of linear systems with matrices having

(generalized) displacement rank α. Such algorithms for exact solutions have been studied by several authors, most recently by Ammar and Gragg [2] for Toeplitz systems. They also made a very close study of the implementation of the convolution operation in an attempt to obtain the smallest coefficient. Although we have not attempted so close an operation count for the more general algorithm in our paper, the hidden constant in the operation counts for solving Toeplitz least squares problems is quite high because $\alpha = 4$ for the matrices E_2 or E_3 (see (3)) with a full rectangular T. Also we conjecture that our algorithm suffers numerical stability problem when $E_{1,1}$ in (1) has a leading principal submatrix that is close to singular; nevertheless we might hope that numerical refinements devised for the Schur algorithm (see, e.g., Koltracht and Lancaster [18]) may be carried over to the divide-and-conquer framework as well.

We also mention that the fast algorithms for Hankel and close-to-Hankel matrices in [10] can be implemented with divide-and-conquer fashion using the spirit in this paper.

Appendix. We shall summarize the explanation in § 3 using a Pascal-like recursive procedure. First, note that the polynomial $\Theta_{p:q}(z)$ (and $\Psi_{p:q}(z)$) has $q - p + 2$ terms. The first column of $\Theta_{p:q}(z)$ has terms ranging from degree z to z^{q-p+1}, and the other columns have terms from 1 to z^{q-p}. Hence, by shifting the first column by one position, we can store $\Theta_{p:q}(z)$ and $\Psi_{p:q}(z)$ in the array "Poly" from p to q slots inclusive:

```
Poly: array [1..α, 1..α, 0..MAX-1] of record
    θ: coefficients;
    ψ: coefficients
end;
```

The computation of $\Theta_{p:q}(z)$ is sequential, i.e., once we compute $\Theta_{p:q}(z)$, we do not need to keep $\Theta_{p:r-1}(z)$, and therefore, the array "Poly" can be kept as a single global variable.

The polynomial vector $X_{p:q}(z)$ has $q - p + 1$ terms, and therefore, can be stored in an array type GENERATORS:

```
type
    GENERATORS = array [1..α, 0..MAX-1] of record
        x: coefficient;
        y: coefficient
    end
```

However, $X_{p:q}(z)$ cannot be kept as a global variable, and local copies should be maintained until we compute $X_{r:q}(z)$.

Now we can describe the recursive generalized Schur algorithm as follows.

```
ALGORITHM (recursive k-step generalized Schur algorithm).
Input: Generator of E, {X0(z), Y0(w)}; displacement operator {⊛F^f, ⊛F^b};
    Number of steps, k.
Output: Generator of S, {Xk(z), Yk(w)};
procedure RecursiveSchur
    var
        G, LowerG: GENERATORS;
    begin
        Find(0, k-1, G);
        Apply(0, k, n, G, LowerG);
        return(LowerG)
    end
```

The procedure Find (p, q, G) computes $\Theta_{p:q}(z)$, and $\Psi_{p:q}(w)$ given $\{X_{p:q}(z), Y_{p:q}(w)\}$, and the procedure Apply $(p, r, q, G, \text{Lower}G)$ returns Lower$G$ = $\{X_{r:q}(z), Y_{r:q}(w)\}$ given $G = \{X_{p:q}(z), Y_{p:q}(w)\}$

```
procedure Find(p, q: index; G: GENERATORS);
    var
        r: index;
        G, LowerG: GENERATORS;
    begin
        if p = q then begin
            Compute Θ_{p:q}(z) and Ψ_{p:q}(w);
            return
        end
    r := appropriate integer close to ⌈(p+q)/2⌉;
    Find(p. r-1, G);
    Apply(p, r, q, G, LowerG);
    Find(r, q, LowerG);
    (* Use fast convolution for polynomial products *)
    Θ_{p:q}(z) := Θ_{p:r-1}(z)Θ_{r,q}(z);
    Ψ_{p:q}(w) := Ψ_{p:r-1}(w)Ψ_{r,q}(w)
end

procedure Apply (p, r, q: index; G: GENERATORS; var LowerG: GENERATORS);
    begin
        (* Use fast convolution for polynomial products *)
        X_{r:q}(z) := X_{p:q}(z) ⊛_{F^l} Θ_{p:r-1}(z);
        Y_{r:q}(w) := Y_{p:q}(w) ⊛_{F^b} Ψ_{p:r-1}(w);
        LowerG := {X_{r:q}(z), Y_{r:q}(w)}
        Free the storage of {X_{p:q}(z), Y_{p:q}(w)};
        return (LowerG);
    end
```

REFERENCES

[1] A. AHO, J. HOPCROFT, AND J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974, p. 305.

[2] G. AMMAR AND W. GRAGG, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 61–76.

[3] G. BITMEAD AND B. D. O. ANDERSON, *Asymptotically fast solution of Toeplitz and related systems of linear equations*, Linear Algebra Appl., 34 (1980), pp. 103–116.

[4] R. BLAHUT, *Fast Algorithms for Digital Signal Processing*, Addison-Wesley, Reading, MA, 1985.

[5] R. BRENT, F. GUSTAVSON, AND D. YUN, *Fast solution of Toeplitz systems of equations and computation of Padé approximants*, J. Algorithms, 1 (1980), pp. 259–295.

[6] A. BRUCKSTEIN AND T. KAILATH, *Doing inverse scattering the fast(est) way*, Tech. Report, Department of Electrical Engineering, Stanford University, Stanford, CA, July 1985.

[7] W. CHOATE, *A fast algorithm for normal incidence seismograms*, Geophysics, 47 (1982), pp. 196–202.

[8] J. CHUN, *Fast array algorithms for structured matrices*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, Stanford, CA, June 1989.

[9] J. CHUN AND T. KAILATH, *Generalized displacement structure for block-Toeplitz, Toeplitz-block and Toeplitz-derived matrices*, NATO Conference on Signal Processing, Leuven, Belgium, August, 1988.

[10] ———, *Displacement structure for Hankel, Vandermonde and related matrices*, IMA Conference, Minneapolis, MN, June, 1988.

[11] F. DE HOOG, *A new algorithm for solving Toeplitz systems of equations*, Linear Algebra Appl., 88/89 (1987), pp. 122–138.

[12] I. GOHBERG AND I. FEL'DMAN, *Convolution Equations and Projection Methods for Their Solutions*, Trans. Math. Monographs, Vol. 41, American Mathematical Society, Providence, RI, 1974.

[13] I. GOHBERG AND A. SEMENCUL, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201–233.

[14] T. KAILATH, *Signal processing applications of some moment problems*, Proc. Sympos. Appl. Math., 37 (1987), pp. 71–109.

[15] T. KAILATH AND J. CHUN, *Generalized Gohberg–Semencul formulas for matrix inversion*, in the Gohberg Anniversary Collection, Vols. I and II, H. Dym, S. Goldberg, M. Kaashoek, P. Lancaster, eds., Birkhäuser-Verlag, Basel, Switzerland, 1989.

[16] T. KAILATH, S. KUNG, AND M. MORF, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407; see also Bull. Amer. Math. Soc., 1 (1979), pp. 769–773.

[17] T. KAILATH, A. VIEIRA, AND M. MORF, *Inverses of Toeplitz operators, innovations, and orthogonal polynomial*, SIAM Rev., 20 (1978), pp. 106–119.

[18] I. KOLTRACHT AND P. LANCASTER, *Threshold algorithms for the prediction of reflection coefficients in a layered medium*, Geophysics, 53 (1988), pp. 908–919.

[19] H. LEV-ARI AND T. KAILATH, *Triangular Factorization of Structured Hermitian Matrices*, Operator Theory, Advances and Applications, Vol. 18, Birkhäuser, Boston, 1986, pp. 301–324.

[20] W. MCCLARY, *Fast seismic inversion*, Geophysics, 48 (1983), pp. 1371–1372.

[21] M. MORF, *Doubling algorithms for Toeplitz and related equations*, in Proc. IEEE Internat. Conference on Acoustics, Speech and Signal Processing, Denver, CO, 1980, pp. 954–959.

[22] B. MUSICUS, *Levinson and fast Cholesky algorithms for Toeplitz and almost Toeplitz matrices*, Res. Report, Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, 1981.

[23] I. SCHUR, *Über Potenzreihen, die im Innern des Einheitskreises beschrankt sind*, J. Reine Angew. Math., 147 (1917), pp. 205–232.

[24] ———, *On Power Series Which Are Bounded in the Interior of the Unit Circle*. I, Operator Theory, Advances and Applications, Vol. 18, Birkhäuser, Boston, (1986), pp. 31–60. (English translation of [23].)

[25] H. SEXTON, M. SHENSA, AND J. SPEISER, *Remarks on a displacement-rank inversion method for Toeplitz systems*, Linear Algebra Appl., 45 (1982), pp. 127–130.

[26] W. TRENCH, *An algorithm for inversion of finite Toeplitz matrices*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 515–522.