MATHEMATICS OF COMPUTATION
VOLUME 64, NUMBER 212
OCTOBER 1995, PAGES 1557–1576

FAST GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING FOR MATRICES WITH DISPLACEMENT STRUCTURE

I. GOHBERG, T. KAILATH, AND V. OLSHEVSKY

ABSTRACT. Fast $O(n^2)$ implementation of *Gaussian elimination with partial pivoting* is designed for matrices possessing *Cauchy-like displacement structure*. We show how *Toeplitz-like*, *Toeplitz-plus-Hankel-like* and *Vandermonde-like* matrices can be transformed into Cauchy-like matrices by using Discrete Fourier, Cosine or Sine Transform matrices.

In particular this allows us to propose a new fast $O(n^2)$ Toeplitz solver GKO, which incorporates partial pivoting. A large set of numerical examples showed that GKO demonstrated stable numerical behavior and can be recommended for solving linear systems, especially with nonsymmetric, indefinite and ill-conditioned positive definite Toeplitz matrices. It is also useful for block Toeplitz and mosaic Toeplitz (Toeplitz–block) matrices.

The algorithms proposed in this paper suggest an attractive alternative to look-ahead approaches, where one has to jump over ill-conditioned leading submatrices, which in the worst case requires $O(n^3)$ operations.

0. INTRODUCTION

0.1. Displacement structure. Let matrices F, $A \in \mathbf{C}^{n\times n}$ be given. Let $R \in \mathbf{C}^{n\times n}$ be a matrix satisfying a Sylvester-type equation

$$\nabla_{\{F,A\}}(R) = F \cdot R - R \cdot A = G \cdot B, \tag{0.1}$$

with some rectangular matrices $G \in \mathbf{C}^{n\times\alpha}$, $B \in \mathbf{C}^{\alpha\times n}$, where the number α is small in comparison with n. The pair of matrices G, B in (0.1) is referred to as a *$\{F, A\}$-generator* of R and the smallest possible inner size α among all $\{F, A\}$-generators is called the *$\{F, A\}$-displacement rank* of R. The concept of *displacement structure* was first introduced in [21] using the Stein-type displacement operator $\nabla_{\{F,A\}}(\cdot) : \mathbf{C}^{n\times n} \to \mathbf{C}^{n\times n}$ given by

$$\nabla_{\{F,A\}}(R) = R - F \cdot R \cdot A. \tag{0.2}$$

Received by the editor August 4, 1994 and, in revised form, November 17, 1994.

1991 *Mathematics Subject Classification.* Primary 15A06, 15A09, 15A23, 15A57, 65F05, 65H10.

Key words and phrases. Gaussian elimination, partial pivoting, displacement structure, Toeplitz-like, Cauchy-like, Vandermonde-like, Toeplitz-plus-Hankel matrix, Schur algorithm, Levinson algorithm.

The research of the third author was supported in part by the Army Research Office under Grant DAAH04-93-G-0029. This manuscript is submitted for publication with the understanding that the U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

The variant (0.1) of displacement equation appeared later in [19]. The left-hand sides of (0.2) and (0.1) are often referred to as Toeplitz–like and Hankel–like displacement operators, respectively. The most general form of displacement structure, which clearly includes (0.1) and (0.2), was introduced in [24] using the operator

$$\nabla_{\{\Omega,\Delta,F,A\}}(R) = \Omega \cdot R \cdot \Delta^* - F \cdot R \cdot A^*. \tag{0.3}$$

A standard Gaussian elimination scheme applied for triangular factorization of R would require $O(n^3)$ operations. At the same time, displacement structure allows us to speed-up the triangular factorization of a matrix, or equivalently, Gaussian elimination. This speed-up is not unexpected, since all n^2 entries of a structured matrix are completely determined by a smaller number, $2\alpha n$, of the entries of its generator $\{G, B\}$ in the right-hand side of (0.1). Moreover, translating the Gaussian elimination procedure into appropriate operations on the generator gives fast $O(n^2)$ algorithms. This was first (implicitly) done by Schur in [31] for Hermitian Toeplitz matrices and for certain generalizations thereof, which were called quasi-Toeplitz matrices in [27]. Then it was progressively extended to more general displacement structures by Kailath and his colleagues. The most general results were presented in [24] (see also [25]). The name *generalized Schur algorithm* was coined there for any fast implementation of Gaussian elimination exploiting displacement structure as in (0.3).

0.2. **Basic classes of structured matrices.**

The now well-known classes of Toeplitz-like matrices, Vandermonde–like, or Cauchy–like matrices, etc., can be introduced by using any form (0.1) or (0.2) of displacement equation (see, e.g., [21, 19, 6, 1, 13]). For our purposes in this paper it will be more convenient to exploit the Sylvester-type displacement operator as in (0.1). Particular choices of matrices F and A in (0.1) lead to the definitions of basic classes of structured matrices, e.g.,

• *Toeplitz-like*	[21] :	$F = Z_1$,	$A = Z_{-1}$;
• *Toeplitz-plus-Hankel-like*	[18, 9, 30] :	$F = Y_{00}$,	$A = Y_{11}$;
• *Cauchy-like*	[19] :	$F = \operatorname{diag}(c_1, \ldots, c_n)$, $A = \operatorname{diag}(d_1, \ldots, d_n)$;	
• *Vandermonde-like*	[19, 13] :	$F = \operatorname{diag}(\frac{1}{x_1}, \ldots, \frac{1}{x_n})$, $A = Z_1$;	
• *Chebyshev-Vandermonde-like*	[22] :	$F = \operatorname{diag}(x_1, \ldots, x_n)$, $A = Y_{\gamma,\delta}$.	

Here,

$$Z_\phi = \begin{bmatrix} 0 & 0 & \cdots & 0 & \phi \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}, \qquad Y_{\gamma,\delta} = \begin{bmatrix} \gamma & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & \delta \end{bmatrix}, \tag{0.4}$$

i.e., Z_ϕ is the lower shift ϕ–circulant matrix and $Y_{\gamma,\delta} = Z_0 + Z_0^T + \gamma e_1 e_1^T + \delta e_n e_n^T$. The reason for such designations may be seen from the easily verified fact that the shift-invariance property of a Toeplitz matrix $T = [t_{i-j}]_{1 \le i,j \le n}$ implies that the $\{Z_1, Z_{-1}\}$-displacement rank of any Toeplitz matrix does not exceed 2. Therefore, a matrix with low $\{Z_1, Z_{-1}\}$-displacement rank is referred to as a Toeplitz–like matrix. Similar justification for the other types of displacement structure will be given in the main text below.

0.3. Transformation of structured matrices and pivoting. Let $R \in \mathbf{C}^{n \times n}$ satisfy the displacement equation (0.1), and let $T_1, T_2 \in \mathbf{C}^{n \times n}$ be two invertible matrices. It is straightforward to see that the matrix $\hat{R} = T_1^{-1} \cdot R \cdot T_2$ satisfies

$$\nabla_{\{\hat{F},\hat{A}\}}(\hat{R}) = \hat{F} \cdot \hat{R} - \hat{R} \cdot \hat{A} = \hat{G} \cdot \hat{B},$$

with $\hat{F} = T_1^{-1} \cdot F \cdot T_1$, $\hat{A} = T_2^{-1} \cdot A \cdot T_2$, $\hat{G} = T_1^{-1} \cdot G$, $\hat{B} = B \cdot T_2$. This enables one to change the form of the matrices F and A in the displacement equation and therefore to transform a structured matrix from one class to another. Ideas of this kind were utilized earlier by various authors. In [28], the translation of a matrix from one structured class to another was discussed in the context of the extension of known structured algorithms to the other basic structured classes. In [13] this technique was utilized for transformation of Vandermonde-like matrices and Cauchy–like matrices into Toeplitz–like matrices; this allowed us to exploit the FFT for reducing the complexity of computing matrix–vector products for matrices from all basic structured classes.

The judicious use of the transformation of Toeplitz–like matrices to generalized Cauchy matrices, i.e., matrices of the form

$$C = \left[\frac{z_i^T \cdot y_j}{c_i - d_j}\right]_{1 \le i,j \le n} \qquad (z_i, y_i \in \mathbf{C}^\alpha), \tag{0.5}$$

was suggested in [17]. The point is that a generalized Cauchy matrix clearly retains the same form (0.5) after any permutation of columns and rows; this allows incorporating pivoting techniques into fast algorithms for generalized Cauchy matrices.

Clearly, the matrix C possesses a Cauchy–like displacement structure, i.e., it satisfies (0.1) with F and A specified in §0.2, and with $G = [z_1 \ \ z_2 \ \ \cdots \ \ z_n]^T$, $B = [y_1 \ \ y_2 \ \ \cdots \ \ y_n]$. However, one must impose the restriction $c_i \neq d_j$ $(1 \le i, j \le n)$. It is a well-known fact that the inverse of a structured matrix also possesses a similar displacement structure (see, e.g., [21, 19, 6, 13, 25]). In particular, the inverse of the generalized Cauchy matrix in (0.5) is a matrix of the same form, i.e.,

$$C^{-1} = -\left[\frac{x_i^T \cdot w_j}{d_i - c_j}\right]_{1 \le i,j \le n} \tag{0.6}$$

with some $x_i, w_i \in \mathbf{C}^\alpha$. A fast algorithm with partial pivoting was suggested by Heinig [17] for inversion and solving linear systems with generalized Cauchy matrices. This algorithm computes in $O(\alpha n^2)$ operations the vectors $x_i, w_i \in \mathbf{C}^\alpha$ in (0.6); then the linear system $C \cdot x = b$ is solved by computing the matrix–

vector product $C^{-1} \cdot b$. Heinig's algorithm is of the Levinson type and it involves inner product calculations; it was shown in [17] that this disadvantage can be avoided by precomputing certain residuals. The algorithm for this purpose is of the Schur type; however the implicitly computed triangular factorization of C was not exploited in [17].

In [14, 15] a fast implementation of Gaussian elimination with partial pivoting (fast GEPP) was designed for Cauchy–like matrices in the context of the factorization problem for rational matrix functions. This algorithm computes in $O(\alpha n^2)$ operations the triangular factorization of C; then the linear system is solved in $O(n^2)$ operations via forward and back substitution [16]. Moreover, this algorithm allows us to process not only matrices of the form (0.5) but also more general Cauchy-like matrices in which the two sets $\{c_1, \ldots, c_n\}$ and $\{d_1, \ldots, d_n\}$ are not disjoint. In the context of rational interpolation, the condition $c_i = d_j$ means that the corresponding rational matrix function has a pole c_i and zero d_j at the same point. Such Cauchy–like matrices are encountered in many important applications [2]; see for example [15] for fast algorithms (with partial, symmetric or 2×2 block pivoting) for matrix Nehari and Nehari–Takagi interpolation problems.

0.4. **Main results.** In this paper we observe that partial pivoting can be incorporated into fast algorithms not only for Cauchy–like matrices, but also for any class of matrices with displacement structure, which is defined by equation (0.1) with diagonal matrix F. Therefore, by incorporating partial pivoting into the generalized Schur algorithms we can design a fast $O(n^2)$ implementation of GEPP for Cauchy–like, Vandermonde–like and Chebyshev–Vandermonde–like matrices. Our algorithm is not restricted to strongly regular matrices, and is valid for arbitrary invertible matrices from any of the above three structured classes.

Furthermore, we propose a variety of formulas for transformation of Toeplitz–like, Toeplitz-plus-Hankel–like and Vandermonde–like matrices into Cauchy–like matrices; these formulas only require computing Fast Fourier, Cosine or Sine transforms of the 2α columns of the generator matrices G and B^T in (0.1), which is a fast and accurate operation and preserves the condition number of a matrix. This allows us to solve linear systems with matrices from any structured class, applying the fast GEPP algorithm derived in §2 for Cauchy–like matrices.

In particular, it leads to two new $O(n^2)$ Toeplitz solvers that incorporate partial pivoting. The first *Toeplitz solver GKO*, derived in §3 below, transforms a Toeplitz matrix into a Cauchy–like matrix using FFTs. The second solver, developed in §4 for the more general class of Toeplitz-plus-Hankel–like matrices, utilizes FCTs and FSTs.

In §5 we present a large set of numerical examples, and compare the numerical behavior of the new Toeplitz solver GKO with the Levinson and classical Schur algorithms and with standard Gaussian elimination with complete pivoting. The data in §5 suggest that GKO demonstrates stable behavior and moreover it remains reliable with "difficult" Toeplitz matrices, for which other structured algorithms propagate large errors. More comments on the conclusions from the numerical experiments are offered in the concluding §6.

1. Fast GEPP and Sylvester-type displacement equation

1.1. Generator recursion. Applying Gaussian elimination to an arbitrary matrix R_1 is equivalent to recursive Schur complementation, as shown by the factorization

$$R_1 = \begin{bmatrix} d_1 & u_1 \\ l_1 & R_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{d_1} l_1 & I \end{bmatrix} \cdot \begin{bmatrix} d_1 & u_1 \\ 0 & R_2 \end{bmatrix}, \tag{1.1}$$

where $R_2 = R_{22}^{(1)} - \frac{1}{d_1} l_1 u_1$ is a Schur complement of the (1,1) entry d_1 in the matrix R_1.

It is a well-known fact in displacement structure theory that the Schur complement of a structured matrix remains in the same structured class. In the most general form, this statement can be found in [24], where it appeared for the case of a generalized displacement structure as in (0.3) with lower triangular matrices Ω, Δ, F, A. Moreover, a generalized Schur algorithm presented there provides a constructive proof of this result. Below we shall give a variant from [14, 15] of the generalized Schur algorithm for a matrix R_1 satisfying the Sylvester-type displacement equation.

Lemma 1.1. *Let the matrix* $R_1 = [\begin{smallmatrix} d_1 & u_1 \\ l_1 & R_{22}^{(1)} \end{smallmatrix}]$ *satisfy the Sylvester-type displacement equation*

$$\nabla_{\{F_1, A_1\}}(R_1) = \begin{bmatrix} f_1 & 0 \\ * & F_2 \end{bmatrix} \cdot R_1 - R_1 \cdot \begin{bmatrix} a_1 & * \\ 0 & A_2 \end{bmatrix} = G_1 \cdot B_1 \qquad (G_1 \in \mathbf{C}^{n \times \alpha},\ B_1 \in \mathbf{C}^{\alpha \times n}). \tag{1.2}$$

If its $(1, 1)$ *entry* d_1 *is nonzero, then the Schur complement* $R_2 = R_{22}^{(1)} - \frac{1}{d_1} l_1 u_1$ *satisfies the Sylvester-type displacement equation*

$$F_2 \cdot R_2 - R_2 \cdot A_2 = G_2 \cdot B_2, \tag{1.3}$$

with

$$\begin{bmatrix} 0 \\ G_2 \end{bmatrix} = G_1 - \begin{bmatrix} 1 \\ \frac{1}{d_1} l_1 \end{bmatrix} \cdot g_1, \qquad [0 \ \ B_2] = B_1 - b_1 \cdot \begin{bmatrix} 1 & \frac{1}{d_1} u_1 \end{bmatrix}, \tag{1.4}$$

where g_1 *and* b_1 *are the first row of* G_1 *and the first column of* B_1, *respectively.*

Proof. Indeed, from (1.2) and the standard Schur complementation formula

$$R_1 = \begin{bmatrix} 1 & 0 \\ \frac{1}{d_1} \cdot l_1 & I \end{bmatrix} \cdot \begin{bmatrix} d_1 & 0 \\ 0 & R_2 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{1}{d_1} \cdot u_1 \\ 0 & I \end{bmatrix},$$

it follows that

$$\begin{bmatrix} f_1 & 0 \\ * & F_2 \end{bmatrix} \cdot \begin{bmatrix} d_1 & 0 \\ 0 & R_2 \end{bmatrix} - \begin{bmatrix} d_1 & 0 \\ 0 & R_2 \end{bmatrix} \cdot \begin{bmatrix} a_1 & * \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{d_1} \cdot l_1 & I \end{bmatrix} \cdot B_1 \cdot G_1 \cdot \begin{bmatrix} 1 & -\frac{1}{d_1} \cdot u_1 \\ 0 & I \end{bmatrix}.$$

Equating the (2,2) block entries, one obtains (1.4). □

1.2. **Partial pivoting.** Applying partial pivoting requires a move of the maximum-magnitude entry in the first column of R_1 to the (1,1) position using row interchange, or equivalently, a multiplication with the corresponding permutation matrix P_1, and then performing elimination :

$$P_1 \cdot R_1 = \begin{bmatrix} 1 & 0 \\ \frac{1}{d_1} l_1 & I \end{bmatrix} \cdot \begin{bmatrix} d_1 & u_1 \\ 0 & R_2 \end{bmatrix}. \tag{1.5}$$

Now assume that R_1 satisfies (1.2) in which F_1 is a diagonal matrix. In this case, after the row interchange, the matrix $\hat{R}_1 = P_1 \cdot R_1$ satisfies in fact the same displacement equation (1.2), with the diagonal matrix F_1 replaced by another diagonal matrix, $\hat{F} = P_1 \cdot F_1 \cdot P_1^T$, and with G_1 replaced by $\hat{G}_1 = P_1 \cdot G_1$. In particular, this means that a row interchange does not destroy the Cauchy–like, Vandermonde–like and Chebyshev–Vandermonde–like displacement structures. Moreover, in this case, Gaussian elimination with partial pivoting can be translated into the language of operations on the generator of a matrix as follows :

<u>Fast GEPP for a structured matrix</u>

- First recover from the generator the first column of $R_1 = [\begin{smallmatrix} d_1 & u_1 \\ l_1 & R_{22}^{(1)} \end{smallmatrix}]$.[1]
- Next determine the position, say $(k, 1)$, of the entry with maximum magnitude in the first column. Let P_1 be a permutation of the 1st and the kth entries. Interchange the 1st and the kth diagonal entries of F_1 in (1.2); interchange the 1st and kth rows in the matrix G_1 in (1.2).
- Then recover from the generator the first row of $P_1 \cdot R_1$.[1] Now one has the first column $[\begin{smallmatrix} 1 \\ \frac{1}{d_1} l_1 \end{smallmatrix}]$ of L and the first row $[d_1 \ u_1]$ of U in the LU factorization of the permuted matrix, $P_1 \cdot R_1$ in (1.5).
- Next compute by (1.4) a generator of the Schur complement $R_2 = R_{22}^{(1)} - \frac{1}{d_1} l_1 u_1$ of $P_1 \cdot R_1$ in (1.5).

Proceeding recursively, one finally obtains the factorization $R_1 = P \cdot L \cdot U$, where $P = P_1 \cdot P_2 \cdots P_{n-1}$ and P_k is the permutation used at the kth step of the recursion.

2. Fast GEPP for Cauchy–like matrices

Let $t_1, t_2, \ldots, t_n$ and $s_1, s_2, \ldots, s_n$ be $2n$ numbers, which for simplicity we assume to be distinct. In this section we consider the displacement operator of the form (0.1) with diagonal F and A :

$$F = D_t = \operatorname{diag}(t_1, t_2, \ldots, t_n); \qquad A = D_s = \operatorname{diag}(s_1, s_2, \ldots, s_n).$$

It can be easily checked that the $\{D_t, D_s\}$-displacement rank of an ordinary Cauchy matrix $C(t, s) = [\frac{1}{t_i - s_j}]_{1 \le i,j \le n}$ is equal to 1. Therefore, a matrix R_1 with low $\{D_t, D_s\}$-displacement rank will be referred to as a Cauchy–like matrix

[1]The computations for doing this depend on the form of the matrices F_1 and A_1 in displacement equation (1.2). In the next section these computations will be specified for Cauchy-like matrices.

[19]. More specifically, R is a Cauchy–like matrix if

(2.1)
$$\nabla_{\{D_t, D_s\}}(R_1) = D_t \cdot R_1 - R_1 \cdot D_s = \begin{bmatrix} \varphi_1^T & \varphi_2^T & \cdots & \varphi_n^T \end{bmatrix}^T \cdot \begin{bmatrix} \psi_1 & \psi_2 & \cdots & \psi_n \end{bmatrix},$$

with $\varphi_i \in \mathbf{C}^{1\times\alpha}$ and $\psi_i \in \mathbf{C}^{\alpha\times 1}$ ($i = 1, 2, \ldots, n$), and α a small number. It is easy to see that

$$R_1 = \left[\frac{\varphi_i \cdot \psi_j}{t_i - s_j}\right]_{1\le i,j\le n}. \tag{2.2}$$

Recall that for Cauchy–like matrices to implement the fast GEPP, one has only to show how to recover the first row and column of R_1 from its generator (see §1). The formula (2.2) makes this easy to do, as is described in

Algorithm 2.1. Fast GEPP for a Cauchy–like matrix

Complexity	$4\alpha n^2$ operations.
Input	A Cauchy–like matrix R_1 given by its generator (see (2.1)).
Output	The factorization $R_1 = P \cdot L \cdot U$ of a permuted version of R_1, where P is a permutation, the matrix L is a unit lower triangular matrix, and U is an upper triangular matrix.
Initialization	Set $L = [l_{ij}]_{i,j=1}^n$, $U = [u_{ij}]_{i,j=1}^n$ to be zero matrices, and P to be the identity matrix.

```
for $k = 1:n$
    for $j = k:n$
        $l_{jk} = \frac{\varphi_j \cdot \psi_k}{t_j - s_k}$
    end
    find $k \le q \le n$    so that    $|l_{qk}| = \max_{k\le j\le n} |l_{jk}|$
    $u_{kk} = l_{qk}$
    swap $t_k$ and $t_q$
    swap $\varphi_k$ and $\varphi_q$
    swap $k$th and $q$th rows in $L$
    swap $k$th and $q$th rows in $P$
    for $j = k+1:n$
        $u_{kj} = \frac{\varphi_k \cdot \psi_j}{t_k - s_j}$
    end
    $l_{kk} = 1$
    for $j = k+1:m$
        $l_{jk} = \frac{l_{jk}}{u_{kk}}$
        $\psi_j = \psi_j - \psi_k \cdot \frac{u_{kj}}{u_{kk}}$
        $\varphi_j = \varphi_j - \varphi_k l_{jk}$
    end
end
```

3. Transformation of Toeplitz-like matrices to Cauchy-like matrices

In this section we consider a displacement operator of the form (0.1) with

$$F = Z_1, \qquad A = Z_{-1},$$

where Z_ϕ is the lower shift circulant matrix defined in (0.4). As was mentioned in the introduction, it is easy to check that any Toeplitz matrix $T = [t_{i-j}]_{1\le i,j\le n}$ satisfies the equation

(3.1)
$$\nabla_{\{Z_1, Z_{-1}\}}(T) = Z_1 \cdot T - T \cdot Z_{-1} = \begin{bmatrix}1\\0\\ \vdots\\0\end{bmatrix} \cdot \begin{bmatrix} t_{n-1} - t_{-1} \\ \vdots \\ t_1 - t_{-n+1} \\ t_0 \end{bmatrix}^T + \begin{bmatrix} t_0 \\ t_{-n+1} + t_1 \\ \vdots \\ t_{-1} + t_{n-1}\end{bmatrix} \cdot \begin{bmatrix}0\\ \vdots \\0\\1\end{bmatrix}^T,$$

i.e., the $\{Z_1, Z_{-1}\}$-displacement rank of a Toeplitz matrix does not exceed 2. By analogy with (3.1), a matrix with low $\{Z_1, Z_{-1}\}$-displacement rank is referred to as a *Toeplitz-like* matrix. Observe that this definition slightly deviates from the one in [21], where the displacement operator of the form $\nabla_{\{Z_0, Z_0^T\}}(T) = T - Z_0 \cdot T \cdot Z_0^T$ was exploited. However, both definitions above describe in fact the *same* class of matrices, viz., those having a low displacement rank; the actual displacement rank will depend on the actual choice of displacement operator.

Proposition 3.1. *Let $R \in \mathbf{C}^{n\times n}$ be a Toeplitz-like matrix satisfying*

(3.2) $$\nabla_{\{Z_1, Z_{-1}\}}(R) = Z_1 \cdot R - R \cdot Z_{-1} = G \cdot B,$$

where $G \in \mathbf{C}^{n\times\alpha}$ and $B \in \mathbf{C}^{\alpha\times n}$. Then $\mathcal{F} \cdot R \cdot D_0^{-1} \cdot \mathcal{F}^*$ is a Cauchy-like matrix:

$$\nabla_{\{D_1, D_{-1}\}}(\mathcal{F} \cdot R \cdot D_0^{-1} \cdot \mathcal{F}^*)$$
$$= D_1 \cdot (\mathcal{F} \cdot R \cdot D_0^{-1} \cdot \mathcal{F}^*) - (\mathcal{F} \cdot R \cdot D_0^{-1} \cdot \mathcal{F}^*) \cdot D_{-1} = \hat{G} \cdot \hat{B}.$$

Here $\mathcal{F} = \frac{1}{\sqrt{n}}[e^{\frac{2\pi i}{n}(k-1)(j-1)}]_{1\le k,j\le n}$ stands for the (normalized) Discrete Fourier Transform matrix,

$$D_1 = \operatorname{diag}(1, e^{\frac{2\pi i}{n}}, \ldots e^{\frac{2\pi i}{n}(n-1)}), \qquad D_{-1} = \operatorname{diag}(e^{\frac{\pi i}{n}}, e^{\frac{3\pi i}{n}}, \ldots, e^{\frac{(2n-1)\pi i}{n}}),$$
$$D_0 = \operatorname{diag}(1, e^{\frac{\pi i}{n}}, \ldots, e^{\frac{(n-1)\pi i}{n}}),$$

and

(3.3) $$\hat{G} = \mathcal{F} \cdot G, \qquad \hat{B}^* = \mathcal{F} \cdot D_0 \cdot B^*.$$

Proof. The assertions of the proposition follow immediately from the well-known factorizations

(3.4) $$Z_1 = \mathcal{F}^* \cdot D_1 \cdot \mathcal{F} \quad \text{and} \quad Z_{-1} = D_0^{-1} \cdot \mathcal{F}^* \cdot D_{-1} \cdot \mathcal{F} \cdot D_0.$$

Substituting (3.4) into (3.2), and multiplying by $\mathcal{F}$ from the left and by $D_0^{-1} \cdot \mathcal{F}^*$ from the right, one obtains the assertions of the proposition. □

Proposition 3.1 offers a new $O(n^2)$ algorithm for solving linear systems with Toeplitz–like matrices. More precisely, let R be a Toeplitz–like matrix, given by a $\{Z_1, Z_{-1}\}$-generator $\{G, B\}$. Just transform R into the Cauchy–like matrix, $\mathscr{F} \cdot R \cdot D_0^{-1} \cdot \mathscr{F}^*$. This transformation is reduced by (3.3) to computing the $\{D_1, D_{-1}\}$-generator for the Cauchy–like matrix $\mathscr{F} \cdot R \cdot D_0^{-1} \cdot \mathscr{F}^*$ by applying 2α FFT's on 2α columns of the matrices G and B^*. Then applying Algorithm 2.1 to this Cauchy–like matrix, one obtains the factorization

$$R = \mathscr{F}^* \cdot P \cdot L \cdot U \cdot \mathscr{F} \cdot D_0,$$

which allows an $O(n^2)$ solution of linear systems with coefficient matrix R via two FFTs and forward and back substitution.

In the next section another method for transforming a Toeplitz–like matrix into a Cauchy–like matrix is given. This method is valid for the more general class of Toeplitz–plus–Hankel–like matrices.

4. Transformation of Toeplitz–plus–Hankel–like matrices to Cauchy–like matrices

In this section we consider a displacement operator of the form (0.1) with

$$F = Y_{00}, \qquad A = Y_{11},$$

where $Y_{\gamma,\delta}$ is defined as in (0.4). It is easy to see that the sum of any Toeplitz and Hankel matrices $R = T + H = [t_{i-j}]_{1\le i,j\le n} + [h_{i+j-2}]_{1\le i,j\le n}$ satisfies

$$\text{(4.1)} \qquad \operatorname{rank}\nabla_{\{Y_{00},Y_{11}\}}(T+H) = Y_{00} \cdot (T+H) - (T+H) \cdot Y_{11} \le 4$$

(cf. [18, 9, 30]). Indeed, let $R = T+H = [r_{ij}]_{1\le i,j\le n}$ and $A = \nabla_{\{Y_{00},Y_{11}\}}(R) = [a_{ij}]_{1\le i,j\le n}$. Note that the matrix $Y_{\gamma,\delta}$ is essentially a combination of lower and upper shifts. Therefore, for $2 \le i, j \le n-1$ we have $a_{i,j} = r_{i-1,j} + r_{i+1,j} - r_{i,j-1} - r_{i,j+1}$. It is easy to see that the latter expression is zero for both the Toeplitz and Hankel components of the matrix R. From this it follows that only nonzero entries of $A = \nabla_{\{Y_{00},Y_{11}\}}(T+H)$ can appear in its first and last rows and its first and last columns, and (4.1) follows. Thus the $\{Y_{00}, Y_{11}\}$-displacement rank of $T+H$ does not exceed 4. By analogy with (4.1), we shall refer to any matrix R with low $\{Y_{00}, Y_{11}\}$-displacement as a *Toeplitz-plus-Hankel-like* matrix.

In what follows we shall use the fact that the matrices $Y_{\gamma,\delta}$ with $\gamma, \delta \in \{1, -1\}$ or $\gamma = \delta = 0$, can be diagonalized by Fast Trigonometric Transform matrices. In particular, the following statement holds (see for example [5]).

Lemma 4.1. *Let* $Y_{\gamma,\delta}$ *be defined as in* (0.4). *Then*

$$Y_{00} = \mathscr{S} \cdot D_S \cdot \mathscr{S}, \qquad Y_{11} = \mathscr{C} \cdot D_C \cdot \mathscr{C}^T,$$

where

$$\mathscr{C} = \left[\sqrt{\tfrac{2}{n}}\left(q_j \cos \tfrac{(2k-1)(j-1)\pi}{2n}\right)\right]_{1\le k,j\le n}, \qquad \mathscr{S} = \left[\sqrt{\tfrac{2}{n+1}} \cdot \sin \tfrac{kj\pi}{n+1}\right]_{1\le k,j\le n}$$

are (normalized) Discrete Cosine Transform-II *and Discrete Sine Transform*-I *matrices, respectively* $(q_1 = \frac{1}{\sqrt{2}}, q_2 = \cdots = q_n = 1)$, *and*

$$D_C = 2 \cdot \operatorname{diag}\left(1, \cos\frac{\pi}{n}, \ldots, \cos\frac{(n-1)\pi}{n}\right),$$

$$D_S = 2 \cdot \operatorname{diag}\left(\cos\frac{\pi}{n+1}, \ldots, \cos\frac{n\pi}{n+1}\right).$$

Lemma 4.1 immediately yields the following result, which allows us to transform a Toeplitz–plus–Hankel–like matrix to a Cauchy–like matrix.

Proposition 4.2. *Let R be a Toeplitz–plus–Hankel–like matrix satisfying*

$$\nabla_{\{Y_{00}, Y_{11}\}}(R) = Y_{00} \cdot R - R \cdot Y_{11} = G \cdot B, \tag{4.2}$$

where $G \in \mathbf{C}^{n\times\alpha}$ and $B \in \mathbf{C}^{\alpha\times n}$. Then $\mathcal{S} \cdot R \cdot \mathcal{C}$ is a Cauchy–like matrix :

$$\nabla_{\{D_S, D_C\}}(\mathcal{S} \cdot R \cdot \mathcal{C}) = D_S \cdot (\mathcal{S} \cdot R \cdot \mathcal{C}) - (\mathcal{S} \cdot R \cdot \mathcal{C}) \cdot D_C = \hat{G} \cdot \hat{B}.$$

Here, as in Lemma 4.1, *$\mathcal{S}$ and $\mathcal{C}$ stand for the Discrete Cosine*-II *and Sine*-I *Transform matrices, respectively, and*

$$\hat{G} = \mathcal{S} \cdot G, \qquad \hat{B}^T = \mathcal{C}^T \cdot B^T. \tag{4.3}$$

Proposition 4.2 suggests the following $O(n^2)$ algorithm for solving linear systems with a Toeplitz–plus–Hankel–like matrix R, given by its $\{Y_{00}, Y_{11}\}$-generator $\{G, B\}$. One has to transform R into the Cauchy–like matrix $\mathcal{S} \cdot R \cdot \mathcal{C}$. According to (4.3), this transformation can be obtained by computing the $\{D_S, D_C\}$-generator for the Cauchy–like matrix $\mathcal{S} \cdot R \cdot \mathcal{C}$, applying α FSTs to the columns of G and applying α inverse FCTs to the columns of B^T. Then applying Algorithm 2.1 to the Cauchy–like matrix $\mathcal{S} \cdot R \cdot \mathcal{C}$, one obtains the factorization

$$R = \mathcal{S} \cdot P \cdot L \cdot U \cdot \mathcal{C}^T,$$

which allows $O(n^2)$ solution of a linear system with matrix R via FCT, FST, forward and back substitution.

Finally, Proposition 4.2 also enables fast solution of linear systems with Toeplitz–like matrices. To show this, we remark that for an arbitrary matrix R,

$$\operatorname{rank}\nabla_{\{Y_{00}, Y_{11}\}}(R) \le 2 \cdot \operatorname{rank}\nabla_{\{Z_1, Z_{-1}\}}(R) + 4. \tag{4.4}$$

Indeed, assume that

$$\operatorname{rank}(Z_1 \cdot R - R \cdot Z_{-1}) = \alpha;$$

then

$$\operatorname{rank}(Z_1^T \cdot R - R \cdot Z_{-1}^T) = \alpha.$$

By adding the two latter equalities, one obtains $\operatorname{rank}\nabla_{\{Z_1+Z_1^T, Z_{-1}+Z_{-1}^T\}}(R) \le 2 \cdot \alpha$. Finally, the matrix $Z_\phi + Z_\phi^T$ is a rank-two perturbation of the matrix $Y_{\gamma,\delta}$, and (4.4) follows.

The equality (4.4) shows that Toeplitz–like matrices indeed belong to the more general class of Toeplitz–plus–Hankel–like matrices. Therefore, they can be processed by the algorithm based on Proposition 4.2. However, the $\{Y_{00}, Y_{11}\}$-displacement rank of a Toeplitz–like matrix can be twice as big as its $\{Z_1, Z_{-1}\}$-displacement rank (for example, for an ordinary Toeplitz matrix the above two displacement ranks are 4 and 2, respectively). Hence, the $\{D_1, D_{-1}\}$-displacement rank of the Cauchy–like matrix $\mathscr{F} \cdot R \cdot D_0^{-1} \cdot \mathscr{F}^*$ (see §3) will also be about twice as big as the $\{D_S, D_C\}$-displacement rank of the Cauchy–like matrix $\mathscr{S} \cdot R \cdot \mathscr{C}$. Hence, processing a Toeplitz–like matrix by the method of §4 can be more expensive in comparison with the method suggested in §3. However, this is not the case when R is a real matrix. In this situation, the new $\{D_S, D_C\}$-generator of $\mathscr{S} \cdot R \cdot \mathscr{C}$ will also be real, and hence all further computations in Algorithm 2.1 will remain in the real domain. However, the new $\{D_1, D_{-1}\}$-generator of $\mathscr{F} \cdot R \cdot D_0^{-1} \cdot \mathscr{F}^*$ will generally be complex, which will reduce the advantage of the lower displacement rank. We shall study the numerical behavior of the methods of §§3 and 4 in §6. For completeness, however, let us first describe how to transform Vandermonde-like matrices to Cauchy-like matrices.

5. Transformation of Vandermonde-like matrices to Cauchy-like matrices

In this section we consider a displacement operator of the form (0.1) with

$$F = D_{\frac{1}{x}} = \operatorname{diag}(\frac{1}{x_1}, \frac{1}{x_2}, \dots, \frac{1}{x_n}), \qquad A = Z_1^T,$$

where x_i are nonzero. It can be easily checked that the Vandermonde matrix $V(x) = [x_i^{j-1}]_{1 \le i, j \le n}$ satisfies the equation

(5.1)
$$\begin{aligned}\nabla_{\{D_{\frac{1}{x}}, Z_1^T\}}(V(x)) &= D_{\frac{1}{x}} \cdot V(x) - V(x) \cdot Z_1^T \\ &= [\tfrac{1}{x_1} - x_1^{n-1}, \dots, \tfrac{1}{x_n} - x_n^{n-1}]^T \cdot [1 \quad 0 \quad \cdots \quad 0],\end{aligned}$$

(cf. [19, 13]). By analogy with (5.1) we shall refer to any matrix R with low $\{D_{\frac{1}{x}}, Z_1^T\}$-displacement rank as a *Vandermonde-like* matrix. Observe that although this definition slightly deviates from the one in [19], it describes in fact the *same* class of matrices.

The next statement is the counterpart of Proposition 3.1 for Vandermonde-like matrices, and it is deduced with exactly the same arguments.

Proposition 5.1. *Let R be a Vandermonde-like matrix satisfying*

$$\nabla_{\{D_{\frac{1}{x}}, Z_1^T\}}(R) = D_{\frac{1}{x}} \cdot R - R \cdot Z_1^T = G \cdot B,$$

where $G \in \mathbf{C}^{n \times \alpha}$, $B \in \mathbf{C}^{\alpha \times n}$. As in Proposition 3.1, let $\mathscr{F}$ stand for the normalized DFT matrix and $D_1 = \operatorname{diag}(1, e^{\frac{2\pi i}{n}}, \dots, e^{\frac{2\pi i}{n}(2n-1)})$. Then $R \cdot \mathscr{F}^*$ is a Cauchy-like matrix:

$$\nabla_{\{D_{\frac{1}{x}}, D_1^*\}}(R \cdot \mathscr{F}^*) = D_{\frac{1}{x}} \cdot (R \cdot \mathscr{F}^*) - (R \cdot \mathscr{F}^*) \cdot D_1^* = G \cdot (B \cdot \mathscr{F}^*).$$

This proposition suggests an efficient method for transforming a Vandermonde-like matrix into a Cauchy-like matrix. This shows that Algorithm 2.1 is also useful for solving linear systems with Vandermonde-like coefficient matrices.

6. Numerical experiments with fast Toeplitz solvers

We performed a large number of computer experiments with the algorithms developed in the present paper to investigate their behavior in floating-point arithmetic and to compare them with other available algorithms. In particular, we solved a linear system $T \cdot x = b$ for various choices of the Toeplitz matrix T and right-hand side b, using the following algorithms :

(1)	GECP	$O(n^3)$	Gaussian elimination with complete pivoting.
(2)	Levinson	$O(n^2)$	Levinson algorithm.
(3)	Schur	$O(n^2)$	Classical Schur algorithm for triangular factorization of T, and then back and forward substitution.
(4)	GKO	$O(n^2)$	Transformation of T to a Cauchy–like matrix on the basis of Proposition 4.1 and then use of Algorithm 2.1.
(5)	TpH	$O(n^2)$	Transformation of T to a Cauchy–like matrix on the basis of Proposition 5.1 and then use of Algorithm 2.1.

All the algorithms (1) – (5) were implemented using the C language on a SUN workstation. To estimate the condition number and Frobenius norm $\|T\|_F$ of a matrix we used routines from LAPACK, and for computing Fast Transforms we used routines from FFTPACK.

The data on the time required by each of the above algorithms are given in Table 1. The authors have to make a proviso that the test programs were not completely optimized for time performance. Nevertheless, these data provide an approximation for the real complexities of the five compared algorithms.

Table 1. Time (seconds)

n	GECP	Levinson	Schur	GKO	TpH
15	0.01	0.00	0.00	0.02	0.00
20	0.02	0.00	0.00	0.02	0.01
25	0.05	0.00	0.00	0.04	0.02
30	0.08	0.00	0.00	0.05	0.02
40	0.21	0.01	0.00	0.09	0.05
50	0.57	0.01	0.01	0.14	0.06
60	1.48	0.02	0.01	0.20	0.09
70	2.36	0.02	0.02	0.27	0.16
80	3.99	0.02	0.02	0.35	0.17
90	5.88	0.04	0.03	0.43	0.25
100	8.23	0.04	0.03	0.56	0.26
110	11.15	0.05	0.04	0.66	0.30
120	14.48	0.06	0.04	0.77	0.36
130	18.96	0.07	0.05	0.94	0.45
140	23.58	0.08	0.07	1.23	0.66
150	28.61	0.09	0.06	1.25	0.61
160	34.83	0.10	0.07	1.46	0.72
170	43.76	0.11	0.08	1.64	0.84
180	49.02	0.14	0.09	1.83	0.97
190	56.82	0.14	0.10	2.04	1.00
200	64.81	0.16	0.11	2.21	1.01
210	74.64	0.18	0.12	2.44	1.29
220	84.99	0.19	0.14	2.67	1.25
230	96.94	0.21	0.15	2.90	1.36

The standard mode of computation was in single precision with unit round-off error $u_s = 2^{-23} \approx 1.19 \times 10^{-7}$. The algorithms GECP and GKO were also implemented in double precision, for which unit round-off $u_d = 2^{-56} = 1.4 \times 10^{-17}$. For the solutions x_{dGECP} and x_{dGKO} computed by the double precision versions of GECP and GKO, we computed the relative error

$$de\_s = \frac{\|x_{\mathrm{dGECP}} - x_{\mathrm{dGKO}}\|}{\|x_{\mathrm{dGECP}}\|}.$$

If de_s was of the order of 10^{-k}, we regarded the first k digits in the mantissa of x_{dGECP} to be exact and relied on them when computing the relative error

$$e\_s = \frac{\|x_{\mathrm{dGECP}} - \tilde{x}\|}{\|x_{\mathrm{dGECP}}\|}$$

for the solution $\tilde{x}$ computed in single precision for each algorithm (1) – (5). For each example we also computed the relative residual error as

$$r\_e = \frac{\|T \cdot \tilde{x} - b\|}{\|b\|}.$$

As is well known, the error in the computed solution $\tilde{x}$ and the residual error are related by

$$\frac{1}{k_2(T)} \cdot \frac{\|T \cdot \tilde{x} - b\|}{\|b\|} \le \frac{\|x - \tilde{x}\|}{\|x\|} \le k_2(T) \cdot \frac{\|T \cdot \tilde{x} - b\|}{\|b\|},$$

i.e., a small residual error implies an accurate solution for well-conditioned matrices, but not necessarily for ill-conditioned matrices.

We solved linear systems using the above algorithms for different Toeplitz matrices and different right-hand sides b. In particular, for each Toeplitz matrix specified below we constructed b by accumulating the product $b = T_n \cdot [1 \quad 1 \quad \cdots \quad 1]^T$ in double precision. This choice allows us to avoid the situation in which ill-conditioning of the matrix is reflected in the growth of the norm of a solution vector. We describe the results of several numerical experiments.

6.1. **Positive definite Toeplitz matrices.**

For positive definite Toeplitz matrices with positive reflection coefficients, Cybenko showed in [8] that the Levinson algorithm guarantees the same size of residual error as the stable Cholesky factorization. He pointed out that his proof does not extend to the case where there are negative reflection coefficients.

Example 1. In [32] (see also [4]) it was observed that the Levinson algorithm can produce residual errors that are larger by a factor of the order of $10^3 - 10^5$ than those of a general stable method (Cholesky factorization), when applied to the prolate matrix with $w = \frac{1}{4}$ for which reflection coefficients alternate in sign; the prolate matrix is defined by

$$T_n = [t_{i-j}]_{1 \le n}, \quad \text{where } t_k = \begin{cases} 2\omega & \text{if } k = 0, \\ \frac{\sin(2\pi\omega k)}{\pi k} & \text{otherwise} \end{cases} \qquad (0 \le \omega \le \tfrac{1}{2}).$$

Background on prolate matrices can be found in [32]. We mention here only that they possess remarkable spectral and conditioning properties. For small ω, their eigenvalues are clustered around 0 and 1, which makes them extremely ill-conditioned. In fact, $k_2(T_n) \approx \frac{1}{p(n,\omega)} e^{\gamma n}$, for some γ and $p(n, \omega)$, where $k_2(T)$ stands for the spectral condition number, $k_2(T) = \|T\| \cdot \|T^{-1}\|$. The data in Table 2 confirm the conclusion of [32] that the Levinson algorithm can give poor results. However, it turns out, very interestingly, that the classical Schur algorithm and new Toeplitz solver GKO applied to the prolate matrix remain as accurate as the stable GECP algorithm.

TABLE 2. Prolate matrix with $\omega = \frac{1}{4}$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	4e+2	1e+0	2e+0	2e+0	7e-6	5e-8	3e-7	7e-8	3e-6	2e-8	1e-5	1e-7	3e-5	2e-7
10	2e+6	2e+0	3e+0	3e+0	9e-3	7e-8	3e-1	2e-6	2e-2	1e-7	5e-2	1e-7	8e-4	3e-7
20	6e+13	3e+0	4e+0	4e+0	4e-1	6e-8	9e+1	1e-4	1e+0	2e-7	2e+0	4e-7	9e-1	2e-6
40	5e+16	4e+0	9e+1	6e+0	1e+0	7e-7	8e+0	3e-4	1e+0	5e-7	2e+0	2e-6	1e+0	6e-6
60	1e+17	5e+0	8e+1	8e+0	1e+0	1e-6	1e+1	4e-4	3e+0	8e-7	1e+0	7e-7	1e+0	4e-6
80	5e+17	6e+0	3e+2	9e+0	1e+0	7e-7	2e+1	5e-4	2e+0	2e-6	1e+0	9e-7	1e+0	2e-5
100	2e+17	7e+0	1e+2	1e+1	1e+0	5e-7	2e+1	2e-3	9e+0	3e-6	1e+0	1e-6	1e+0	5e-5
120	2e+18	8e+0	8e+1	1e+1	3e+0	3e-6	3e+2	7e-3	2e+1	9e-6	1e+0	2e-6	2e+0	8e-5

Example 2. It turned out that a prolate matrix is not an isolated example where the Levinson algorithm is less accurate than the other algorithms compared. In fact, it was typical for the cases where the reflection coefficients did not keep the same sign and the matrix was not well conditioned. Thus, another example is the *Gaussian Toeplitz matrix* $T_n = [t_{i-j}]_{1 \le n}$, where $t_k = a^{k^2}$, $0 < a < 1$, with a close to 1. Background on Gaussian Toeplitz matrices can be found in [29]. We mention here only that this positive definite matrix arises as a discretization of Gaussian convolution, and that

$$k_2(T_n) \ge \frac{1+a}{(1-a^2)(1-a^4)\cdots(1-a^{2(n-1)})}.$$

One can check that the reflection coefficients of T_n also alternate in sign. The data for $a = 0.9$ are given in Table 3 below.

TABLE 3. Gaussian Toeplitz matrix with $a = 0.9$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	6e+3	4e+0	2e+0	8e+0	8e-5	5e-8	7e-6	7e-8	5e-5	6e-8	7e-5	1e-7	3e-4	2e-7
10	1e+6	6e+0	3e+0	1e+1	1e-2	5e-8	3e-1	2e-6	5e-2	9e-8	2e-2	2e-7	1e-3	2e-7
20	2e+8	9e+0	4e+0	2e+1	4e-1	1e-7	2e+1	6e-5	4e-1	1e-7	1e-1	2e-7	2e-1	2e-6
40	3e+9	1e+1	6e+0	3e+1	9e-1	9e-8	6e+1	2e-4	1e+0	2e-7	4e+0	6e-7	5e+0	6e-6
60	5e+9	2e+1	8e+0	4e+1	9e-1	1e-7	7e+1	3e-4	1e+0	3e-7	5e+0	9e-7	4e+0	4e-6
80	6e+9	2e+1	9e+0	5e+1	1e+0	1e-7	8e+1	3e-4	1e+0	3e-7	1e+1	2e-6	3e+0	2e-5
100	6e+9	2e+1	1e+1	5e+1	9e-1	2e-7	7e+1	3e-4	1e+0	4e-7	2e+0	9e-7	1e+1	5e-5
120	7e+9	2e+1	1e+1	6e+1	9e-1	1e-7	7e+1	3e-4	1e+0	4e-7	2e+0	1e-6	5e+1	8e-5
140	7e+9	2e+1	1e+1	6e+1	8e-1	2e-7	7e+1	3e-4	2e+0	4e-7	2e+1	2e-6	7e+0	5e-5
160	7e+9	2e+1	1e+1	7e+1	8e-1	2e-7	7e+1	3e-4	2e+0	4e-7	1e+1	2e-6	2e+1	2e-5

It is interesting that for a Gaussian Toeplitz matrix with $a = 0.99$, the Levinson algorithm failed for $n \ge 24$, i.e., it indicated the occurrence of singular minors. At the same time, the data for the other algorithms are close to those given in Table 3.

Example 3. For a given set of reflection coefficients, a Toeplitz matrix can be recovered easily by tracing the Levinson algorithm backwards. The results in Table 4, where the reflection coefficients are chosen to alternate in sign, show another example of the lack of numerical stability of the Levinson algorithm.

TABLE 4. A matrix with equal sign-alternating reflection coefficients $\rho_i = (-1)^i 0.3$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	7e+0	2e+0	2e+0	3e+0	3e-7	7e-8	8e-8	7e-8	3e-8	4e-8	5e-7	2e-7	3e-7	1e-7
10	9e+1	4e+0	3e+0	4e+0	3e-6	6e-8	3e-6	2e-7	4e-7	9e-8	6e-7	2e-7	5e-7	4e-7
20	3e+4	5e+0	4e+0	6e+0	7e-4	1e-7	4e-4	2e-6	2e-5	1e-7	2e-5	4e-7	1e-4	2e-6
40	5e+9	7e+0	6e+0	9e+0	8e-1	2e-7	2e+1	9e-4	1e+1	5e-7	1e+0	8e-7	7e-1	6e-6
60	1e+15	9e+0	8e+0	1e+1	2e+0	3e-7	7e+1	3e-3	1e+1	8e-7	8e-1	9e-7	4e+0	5e-6
80	3e+16	1e+1	2e+1	1e+1	2e+0	5e-7	4e+1	7e-3	1e+1	1e-6	1e+0	1e-6	9e-1	2e-5
100	5e+16	1e+1	2e+1	1e+1	4e+0	1e-6	8e+1	8e-3	2e+1	1e-5	9e-1	1e-6	9e-1	5e-5
120	3e+16	1e+1	4e+1	1e+1	2e+0	8e-7	1e+1	8e-3	7e+0	5e-6	1e+0	2e-6	1e+0	8e-5
140	3e+16	1e+1	2e+1	2e+1	1e+1	3e-6	6e+1	9e-3	2e+2	8e-5	3e+0	2e-6	2e+0	5e-5
160	5e+17	2e+1	3e+1	3e+1	5e+0	2e-6	8e+2	1e-2	4e+2	3e-4	2e+0	8e-6	9e-1	1e-5

Example 4. Finally, we tested some cases where the reflection coefficients of a positive definite Toeplitz matrix all had the same sign. One such example with reflection coefficients $\rho_i = 0.1$ is shown in Table 5.

TABLE 5. Toeplitz matrix with equal-sign reflection coefficients $\rho_i = +0.1$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	2e+0	2e+0	2e+0	3e+0	8e-8	6e-8	5e-8	3e-8	7e-8	5e-8	2e-7	1e-7	3e-7	3e-7
10	3e+0	3e+0	3e+0	7e+0	2e-7	7e-8	1e-7	7e-8	1e-7	7e-8	4e-7	1e-7	5e-7	3e-7
20	9e+0	5e+0	4e+0	2e+1	2e-7	5e-8	3e-7	7e-8	2e-7	5e-8	3e-7	7e-8	4e-6	2e-6
40	2e+2	9e+0	6e+0	5e+1	1e-6	5e-8	7e-7	6e-8	6e-6	9e-8	6e-7	7e-8	9e-5	6e-6
60	6e+3	1e+1	8e+0	8e+1	8e-5	5e-8	5e-5	1e-7	2e-4	8e-8	3e-5	4e-7	4e-4	4e-6
80	3e+5	2e+1	9e+0	1e+2	4e-4	5e-8	7e-4	1e-7	2e-3	6e-8	4e-4	2e-7	1e-1	2e-5
100	1e+7	2e+1	1e+1	2e+2	3e-2	8e-8	4e-1	4e-7	1e-1	8e-8	3e-2	6e-7	7e+0	5e-5
120	7e+8	2e+1	1e+1	2e+2	2e+1	4e-7	4e+0	2e-6	3e-1	1e-7	1e-1	1e-7	1e+3	7e-4
140	3e+10	3e+1	1e+1	3e+2	2e+0	5e-8	9e+1	8e-5	2e+0	7e-8	1e+0	1e-6	2e+2	5e-5
160	2e+12	3e+1	1e+1	4e+2	3e+0	1e-7	6e+2	5e-5	4e+0	1e-7	5e-1	3e-7	1e+1	6e-6
180	9e+13	3e+1	1e+1	4e+2	5e+0	1e-7	4e+4	5e-3	7e+0	8e-8	2e+0	2e-7	1e+2	3e-5
200	4e+15	4e+1	2e+1	5e+2	3e+0	5e-8	2e+5	6e-2	1e+1	2e-7	9e-1	3e-7	1e+1	5e-5
220	8e+16	4e+1	8e+1	6e+2	2e+0	2e-7	2e+4	1e-2	5e+0	2e-7	1e+0	4e-7	6e+0	1e-5
240	1e+17	5e+1	2e+2	7e+2	1e+0	1e-7	4e+4	6e-1	2e+0	4e-7	1e+0	2e-6	2e+3	3e-4

Observe that for this case the residual errors of the Levinson algorithm became large starting from $n = 125$, whereas all other algorithms demonstrated good backward stability. At first glance this occurrence contradicts the Cybenko result. But in fact, owing to error accumulation, the reflection coefficients, computed by the Levinson algorithm (and by the classical Schur algorithm) for $n > 125$ were no longer of the same sign, so that the Cybenko analysis is not applicable in this situation. Also note that the latter fact means that in numerical computations one cannot rely on some property of a Toeplitz matrix which is theoretically predicted by a particular application; because of round-off in the computer representation of a matrix, as well as a result of further error accumulation, this property can be lost at some stage of the computation. We shall give another example of a problem of this kind below.

The data in Tables 2 – 5 suggest that there is a subclass of positive definite Toeplitz matrices for which the numerical properties of the Levinson algorithm are worse than for the stable GECP method. At the same time observe that in all the above examples, the classical Schur algorithm behaves similarly to the stable GECP. This relates to recent results of [4], where backward stability of the classical Schur algorithm (which is referred to as FACTOR there) was claimed for the class of positive definite Toeplitz matrices. More precisely, it was asserted there that for a positive definite Toeplitz matrix T, the classical

Schur algorithm computes the factorization

$$T = [t_{i-j}]_{1\le i,j\le n} = \tilde{L}\cdot\tilde{L}^T + \Delta T \quad \text{with } \|\Delta T\| = O(ut_0n^3), \tag{6.1}$$

where u is the machine precision. In practice, we found the estimate in (6.1) to be pessimistic. Moreover, in all examples the size of the residual vector, computed via the classical Schur algorithm, did not depend on the size n of the positive definite Toeplitz matrix (experiments in [4] led to the same observation for the closely related Bareiss algorithm).

Example 5. We conclude this subsection with the following example, which reinforces a point made for the example in Table 5.

TABLE 6. Another Toeplitz matrix with equal-sign reflection coefficients $\rho_i = +0.5$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	2e+1	3e+0	2e+0	8e+0	0e+0	0e+0	0e+0	0e+0	0e+0	0e+0	3e-7	1e-7	1e-6	2e-7
10	2e+3	7e+0	3e+0	2e+1	1e-5	9e-8	2e-15	0e+0	1e-5	2e-8	2e-5	2e-7	1e-5	2e-7
20	7e+7	1e+1	4e+0	6e+1	6e-1	3e-8	2e+0	7e-7	7e-1	4e-8	2e-2	2e-7	1e+0	2e-6
40	2e+17	3e+1	2e+1	2e+2	4e+0	1e-7	5e+1	1e-4	1e+1	6e-7	5e+0	4e-7	1e+2	4e-6
60	5e+18	4e+1	3e+2	3e+2	1e+0	5e-7	2e+1	3e-4	9e-1	1e-7	1e+0	3e-7	5e+1	2e-5
80	2e+18	5e+1	2e+4	5e+2	1e+0	4e-7	2e+0	5e-4	1e+0	7e-7	1e+0	2e-6	1e+0	3e-6
100	2e+21	3e+4	3e+2	2e+4	3e+0	2e-6	4e+8	1e+0	2e+6	1e+0	1e+0	3e-6	1e+0	5e-6
120	5e+24	3e+8	1e+4	1e+8	1e+0	1e-4	2e+13	1e+0	4e+14	1e+0	1e+0	4e-6	1e+0	1e-5
140	4e+29	3e+13	3e+7	1e+13	4e+0	2e-3	4e+19	1e+0	1e+20	1e+0	1e+0	1e-6	1e+0	1e-6
150	1e+33	2e+17	1e+7	1e+17	1e+2	3e-3	Inf	Inf	3e+27	1e+0	1e+0	1e-6	1e+0	3e-6

Observe that in Table 6 the size of the residual error for the classical Schur algorithm started to grow for $n > 90$. At first glance, this contradicts the result in (6.1). But in fact, owing to error accumulation, the reflection coefficients actually computed by the classical Schur (and Levinson) algorithm deviate from the given number $+0.5$. Moreover, starting from $n > 90$, they became greater than 1 in magnitude, i.e., the Toeplitz matrix turned out to be indefinite. Observe that the analysis in [4] assumes that the actually computed reflection coefficients are positive. Therefore, that analysis is valid only for a special subclass of positive definite Toeplitz matrices and is not applicable for this situation. In [3] it was shown that to guarantee that the reflection coefficients of a positive definite Toeplitz matrix, computed via the classical Schur algorithm, will remain less than 1 in magnitude, one has to require that $u \cdot k_2(T)$ be less than some constant of the order of $O(n^2)$. The conclusion is that if a Toeplitz matrix is ill-conditioned, then its Schur complements can lose their theoretically expected positive definiteness at some stage of the computation. As we shall see in the next subsection, the classical Schur algorithm loses accuracy in examples with indefinite Toeplitz matrices. However, the results in the next subsection show that the algorithm GKO continues to yield satisfactory performance also in these examples.

6.2. Symmetric indefinite Toeplitz matrices.

Example 6. In the next example we generated a Toeplitz matrix with random entries in the interval $(-1, 1)$. This matrix is not positive definite and may have several ill-conditioned principal submatrices. The common approach to avoid error accumulation is to apply a look-ahead strategy, e.g., to skip over these singular or ill-conditioned submatrices using some in general unstructured algorithm. However, the complexity of such a look-ahead algorithm depends on the number of "difficult" submatrices, and may become $O(n^3)$ in the worst cases. The data in Table 7 show that the $O(n^2)$ Algorithm GKO is a good alternative to the look-ahead strategy, and that it gives numerical results similar to those with the stable GECP algorithm.

TABLE 7. Symmetric Toeplitz matrix with random entries in $(-1, 1)$

					GECP		Levinson		Schur		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	5e+1	2e+0	2e+0	4e+0	9e-7	5e-8	1e-5	5e-7	2e-5	3e-6	7e-7	4e-8	2e-6	2e-7
10	3e+1	5e+0	3e+0	9e+0	3e-7	9e-8	1e-4	2e-5	6e-4	8e-5	6e-7	9e-8	5e-7	2e-7
20	4e+1	1e+1	4e+0	1e+1	4e-7	2e-7	4e-6	3e-6	5e-6	3e-6	5e-7	4e-7	5e-6	6e-7
40	1e+2	2e+1	6e+0	1e+1	1e-6	5e-7	2e-4	5e-5	3e-4	1e-4	5e-6	1e-6	1e-5	6e-6
60	3e+2	4e+1	8e+0	5e+1	4e-6	3e-7	7e-4	2e-4	4e-4	1e-4	4e-6	7e-7	3e-5	4e-6
80	4e+2	5e+1	9e+0	3e+1	7e-6	8e-7	5e-4	4e-5	7e-4	2e-4	1e-4	7e-6	3e-6	1e-6
100	2e+3	5e+1	1e+1	4e+1	8e-6	7e-7	1e+0	1e-2	4e-3	3e-4	5e-5	2e-6	1e-4	5e-5
120	8e+1	7e+1	1e+1	5e+1	2e-6	1e-6	3e-4	2e-4	3e-4	1e-4	3e-6	3e-6	8e-5	8e-5
140	1e+3	8e+1	1e+1	1e+2	5e-6	7e-7	1e-3	1e-4	5e-3	2e-4	2e-5	4e-7	4e-4	5e-5
160	1e+2	9e+1	1e+1	9e+1	4e-6	1e-6	1e-3	3e-4	4e-4	2e-4	6e-6	2e-6	3e-5	3e-5
180	2e+2	1e+2	1e+1	1e+2	1e-5	1e-6	3e-2	1e-2	1e-2	5e-3	4e-6	1e-6	1e-4	4e-5
200	3e+2	1e+2	1e+1	1e+2	7e-6	8e-7	1e-3	3e-4	6e-3	2e-3	4e-6	7e-7	2e-4	1e-4
220	1e+2	1e+2	1e+1	3e+1	3e-6	4e-6	5e-4	6e-4	8e-4	1e-3	4e-6	4e-6	9e-5	4e-5
240	2e+2	1e+2	2e+1	2e+2	5e-6	1e-6	3e-4	1e-4	2e-3	3e-4	2e-5	3e-6	4e-4	3e-4

6.3. **Nonsymmetric Toeplitz matrices.**

Example 7. For the next example we generated a random nonsymmetric Toeplitz matrix with entries chosen randomly in the interval $(-1, 1)$. The data in Table 8 shows also for this case that the accuracy of Algorithm GKO is close to that for GECP.

TABLE 8. Nonsymmetric Toeplitz matrix with random entries in $(-1, 1)$

					GECP		Levinson		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r	e-s	e-r
5	9e+0	3e+0	2e+0	2e+0	1e-7	5e-8	1e-6	4e-7	3e-7	2e-7	1e-7	1e-7
10	5e+0	6e+0	3e+0	6e+0	7e-8	8e-8	7e-5	4e-5	3e-7	3e-7	5e-7	3e-7
20	2e+1	1e+1	4e+0	2e+1	5e-7	1e-7	2e-5	4e-6	7e-7	2e-7	3e-6	2e-6
40	2e+2	2e+1	6e+0	3e+1	6e-6	3e-7	2e-3	4e-4	1e-5	7e-7	2e-5	4e-6
60	5e+1	3e+1	8e+0	5e+1	1e-6	4e-7	9e-5	2e-5	3e-6	2e-7	1e-5	4e-6
80	3e+2	5e+1	9e+0	4e+1	2e-6	6e-7	1e-3	9e-5	3e-6	5e-7	6e-5	4e-6
100	1e+2	6e+1	1e+1	3e+1	1e-6	8e-7	8e-5	5e-5	8e-6	4e-6	5e-5	9e-6
120	9e+1	7e+1	1e+1	5e+1	2e-6	8e-7	4e-4	3e-4	4e-5	6e-6	2e-4	4e-5
140	3e+2	8e+1	1e+1	1e+2	5e-6	8e-7	5e-4	2e-4	4e-5	2e-6	1e-4	5e-5
160	5e+1	1e+2	1e+1	1e+2	2e-6	9e-7	6e-3	3e-3	4e-6	2e-6	3e-5	3e-5
180	1e+2	1e+2	1e+1	7e+1	3e-6	1e-6	5e-4	7e-5	3e-5	2e-6	7e-4	4e-5
200	1e+2	1e+2	1e+1	6e+1	2e-6	2e-6	9e-3	2e-3	6e-5	8e-6	8e-4	8e-5
220	4e+2	1e+2	1e+1	7e+1	1e-5	2e-6	8e-2	1e-2	1e-5	3e-6	2e-4	2e-5
240	9e+2	1e+2	2e+1	1e+2	3e-5	1e-6	2e-1	4e-2	2e-4	8e-6	3e-3	2e-4

6.4. **Mosaic Toeplitz (or Toeplitz–block) matrices.** We also applied the algorithms GECP, GKO and TpH for solving linear systems with mosaic Toeplitz [19] (the designation Toeplitz–block was also used, see [7]) matrices to test problems where the displacement rank of a matrix is greater than for ordinary Toeplitz matrices. In particular, we checked matrices of the form

$$T_{2n} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \tag{6.2}$$

where A, B, C, D are themselves Toeplitz matrices.

Example 8. In the following table one can find the data for the case where A and D are prolate matrices with $\omega = \frac{1}{4}$, and the matrices B and C are equal to the identity matrix I.

TABLE 9. Mosaic-I

					GECP		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r
10	9e+2	4e+0	3e+0	6e+0	2e-5	1e-7	6e-6	2e-7	7e-6	4e-7
20	4e+6	5e+0	4e+0	9e+0	4e-2	2e-7	1e-1	2e-7	1e-1	3e-6
40	1e+14	8e+0	6e+0	1e+1	5e+0	5e-7	2e+0	2e-6	4e+1	2e-5
60	1e+16	9e+0	1e+1	2e+1	4e+2	1e-6	3e+3	3e-5	7e+2	1e-5
80	3e+16	1e+1	1e+1	2e+1	1e+3	3e-6	2e+2	4e-6	2e+4	5e-4
100	6e+16	1e+1	2e+1	2e+1	1e+5	5e-5	9e+2	4e-6	1e+5	1e-4
120	2e+17	1e+1	2e+1	2e+1	1e+3	2e-5	1e+2	2e-5	4e+2	9e-5
140	2e+18	1e+1	6e+1	2e+1	7e+1	1e-6	1e+1	9e-6	4e+1	1e-4
160	2e+17	2e+1	8e+1	3e+1	2e+2	3e-6	1e+2	2e-5	1e+2	3e-5
180	2e+17	2e+1	7e+1	3e+1	1e+3	1e-5	7e+1	1e-5	5e+3	2e-3
200	2e+17	2e+1	3e+1	3e+1	6e+2	7e-6	4e+3	2e-5	5e+3	4e-5
220	6e+17	2e+1	4e+1	3e+1	4e+2	6e-6	2e+2	8e-6	3e+3	3e-4
240	3e+17	2e+1	4e+1	3e+1	3e+2	8e-6	4e+2	1e-5	5e+4	9e-4

Example 9. The next example is a well-conditioned matrix of the form (6.2), where A and D were prolate matrices with $\omega = \frac{1}{4}$, and the matrices B and C are $2 \cdot I$.

TABLE 10. Mosaic-II

					GECP		GKO		TpH	
n	$k_2(T)$	$\|T\|_F$	$\|x\|$	$\|b\|$	e-s	e-r	e-s	e-r	e-s	e-r
10	3e+0	7e+0	3e+0	9e+0	1e-7	8e-8	9e-7	5e-7	7e-7	5e-7
20	3e+0	9e+0	4e+0	1e+1	4e-7	2e-7	1e-6	8e-7	4e-6	3e-6
40	3e+0	1e+1	6e+0	2e+1	6e-7	3e-7	2e-6	1e-6	2e-5	1e-5
60	3e+0	2e+1	8e+0	2e+1	5e-7	3e-7	3e-6	1e-6	2e-5	1e-5
80	3e+0	2e+1	9e+0	3e+1	6e-7	3e-7	4e-6	2e-6	5e-5	3e-5
100	3e+0	2e+1	1e+1	3e+1	7e-7	5e-7	8e-6	3e-6	1e-4	7e-5
120	3e+0	2e+1	1e+1	3e+1	8e-7	6e-7	9e-6	5e-6	2e-4	1e-4
140	3e+0	3e+1	1e+1	4e+1	8e-7	5e-7	6e-6	3e-6	2e-4	8e-5
160	3e+0	3e+1	1e+1	4e+1	1e-6	7e-7	1e-5	6e-6	6e-5	4e-5
180	3e+0	3e+1	1e+1	4e+1	1e-6	7e-7	1e-5	6e-6	1e-4	5e-5
200	3e+0	3e+1	1e+1	4e+1	9e-7	6e-7	1e-5	6e-6	3e-4	1e-4
220	3e+0	3e+1	1e+1	4e+1	1e-6	5e-7	1e-5	8e-6	2e-4	8e-5
240	3e+0	3e+1	2e+1	5e+1	1e-6	8e-7	2e-5	9e-6	8e-4	4e-4

The data in Tables 9, 10 indicate that Algorithm GKO is reliable also for Toeplitz-like matrices. Its numerical behavior does not depend on the size of the matrix and is close to that of GECP. For well-conditioned Toeplitz-like matrices, GKO computes an accurate solution, and for ill-conditioned Toeplitz-like matrices, it produces a small residual error.

7. CONCLUSIONS

The following conclusions are made entirely on the basis of our experiments, only a small part of which was described in the previous section.

GECP. There is no numerical superiority of fast $O(n^2)$ Toeplitz solvers over the general-purpose algorithm $O(n^3)$ GECP. This relates to the results of [10, 11], where it was found that there is little difference between the structured condition number and condition number of a positive definite Toeplitz matrix.

Levinson algorithm. For positive definite Toeplitz matrices with positive reflection coefficients, the Levinson algorithm is as stable, in practice, as GECP and GKO, the results are consistent with those of Cybenko [8]. For positive definite Toeplitz matrices, whose reflection coefficients do not have the same sign, the Levinson algorithm may be less accurate and can produce residual errors larger than those of stable GECP. With symmetric indefinite and nonsymmetric Toeplitz matrices, the Levinson algorithm is less accurate than GECP and GKO.

Classical Schur algorithm. With positive definite Toeplitz matrices the classical Schur algorithm has stable numerical behavior, close to that of GECP. With nonsymmetric and indefinite Toeplitz matrices, it is less accurate than GECP. Also with extremely ill-conditioned positive definite Toeplitz matrices, for which the actually computed reflection coefficients are not bounded by unity, the classical Schur algorithm is also worse than GECP and GKO.

Algorithm TpH. The numerical behavior of Algorithm TpH applied to Toeplitz matrices is slightly worse in comparison with GECP and GKO.

Algorithm GKO. Algorithm GKO showed stable numerical behavior, close to that of GECP. For well-conditioned Toeplitz matrices it computes an accurate solution, and for ill-conditioned Toeplitz matrices it produces small residual errors. It is the only one of the compared algorithms that was reliable for indefinite and nonsymmetric Toeplitz matrices. We are currently investigating the numerical properties of the transformations of Toeplitz-like matrices that preserve the symmetry of the original matrix, thus admitting the possibility of symmetric and Bunch-Kaufman pivoting, rather than just partial pivoting.

ACKNOWLEDGMENT

The authors wish to thank the referee for useful suggestions.

BIBLIOGRAPHY

1. G. Ammar and P. Gader, *New decompositions of the inverse of a Toeplitz matrix*, Signal Processing, Scattering and Operator Theory, and Numerical Methods, Proc. Internat. Sympos. MTNS-89, vol. III, Birkhäuser, Boston, 1990, pp. 421–428.
2. J. Ball, I. Gohberg, and L. Rodman, *Interpolation of rational matrix functions*, OT45, Birkhäuser Verlag, Basel, 1990.
3. A. W. Bojanczyk, private communication.
4. A. W. Bojanczyk, R. P. Brent, F. R. de Hoog, and D. R. Sweet, *On the stability of the Bareiss and related Toeplitz factorization algorithms*, SIAM J. Matrix Anal. Appl. **16** (1995), 40–57.
5. E. Bozzo and C. Di Fiore, *On the use of certain matrix algebras associated with discrete trigonometric transforms in matrix displacement decomposition*, SIAM J. Matrix Anal. Appl. **16** (1994), 312–326.
6. J. Chun and T. Kailath, *Fast triangularization and orthogonalization of Hankel and Vandermonde matrices*, Linear Algebra Appl. **151** (1991), 199–228.
7. ——, *Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices*, SIAM J. Matrix Anal. Appl. **15** (1994), 114–128.
8. G. Cybenko, *The numerical stability of Levinson-Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. Statist. Comput. **1** (1980), 303–319.
9. I. Gohberg and I. Koltracht, *Efficient algorithm for Toeplitz plus Hankel matrices*, Integral Equations Operator Theory **12** (1989), 136–142.
10. I. Gohberg, I. Koltracht, and D. Xiao, *On the solution of Yule-Walker equations*, Proc. SPIE Conference of Advanced Algorithms and Architectures for Signal Processing IV, vol. 1566, July 1991, pp. 14–22.
11. ——, *Condition and accuracy of algorithms for computing Schur coefficients of Toeplitz matrices*, SIAM J. Matrix Anal. Appl. **15** (1994), 1290–1309.
12. I. Gohberg and V. Olshevsky, *Circulants, displacements and decompositions of matrices*, Integral Equations Operator Theory **15** (1992), 730–743.
13. ——, *Complexity of multiplication with vectors for structured matrices*, Linear Algebra Appl. **202** (1994), 163–192.

14. ——— *Fast algorithm for matrix Nehari problem*, Proc. MTNS-93, Systems and Networks: Mathematical Theory and Applications, vol. 2, Invited and Contributed Papers (U. Helmke, R. Mennicken and J. Sauers, eds.), Akademie Verlag, 1994, pp. 687–690.
15. ———, *Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems*, Integral Equations Operator Theory **20** (1994), 44–83.
16. G. Golub and C. Van Loan, *Matrix computations*, 2nd ed., John Hopkins Univ. Press, Baltimore, MD, 1989.
17. G. Heinig, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, Linear Algebra in Signal Processing, IMA volumes in Mathematics and its Applications, vol. 69, 1994, pp. 95–114.
18. G. Heinig, P. Jankowski, and K. Rost, *Fast inversion of Toeplitz-plus-Hankel matrices*, Numer. Math. **52** (1988), 665–682.
19. G. Heinig and K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Operator Theory, vol. 13, Birkhäuser, Basel, 1984.
20. T. Kailath, *Signal processing applications of some moment problems*, Moments in Mathematics (H. Landau, ed.), vol. 37, Amer. Math. Soc., Providence, RI, 1987, pp. 71–109.
21. T. Kailath, S. Kung and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl. **68** (1979), 395–407.
22. T. Kailath and V. Olshevsky, *Displacement structure approach to Chebyshev-Vandermonde and related matrices*, Integral Equations Operator Theory, 1995 (to appear).
23. ———, *Displacement structure approach to polynomial Vandermonde and related matrices*, preprint, 1994.
24. T. Kailath and A. H. Sayed, *Fast algorithms for generalized displacement structures*, Recent Advances in Mathematical Theory of Systems, Control, Networks and Signal Processing II, Proc. MTNS-91 (H. Kimura, S. Kodama, eds.), Mita Press, Japan, 1992, pp. 27–32.
25. ———, *Displacement structure: theory and applications*, SIAM Rev. (to appear).
26. P. Lancaster and M. Tismenetsky, *Theory of matrices with applications*, 2nd ed., Academic Press, Orlando, 1985.
27. H. Lev-Ari and T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory: Advances and Applications, (I. Gohberg, ed.), vol. 18, Birkhäuser, Boston, 1986, pp. 301–324.
28. V. Pan, *On computations with dense structured matrices*, Math. Comp. **55** (1990), 179–190.
29. J. Pasupathy and R. A. Damodar, *The Gaussian Toeplitz matrix*, Linear Algebra Appl. **171** (1992), 133–147.
30. A. Sayed, H. Lev-Ari, and T. Kailath, *Fast triangular factorization of the sum of quasi-Toeplitz and quasi-Hankel matrices*, Linear Algebra Appl. **191** (1993), 77–106.
31. I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind*, J. Reine Angew. Math. **147** (1917), 205–232; English transl., *Operator Theory: Advances and Applications* (I. Gohberg, ed.), vol. 18, Birkhäuser, Boston, 1986, pp. 31–88.
32. J. M. Varah, *The prolate matrix*, Linear Algebra Appl. **187** (1993), 269–278.

School of Mathematical Sciences, Tel Aviv University, Ramat Aviv 69978, Israel
E-mail address: gohberg@math.tau.ac.il

Information Systems Laboratory, Stanford University, Stanford California 94305-4055
E-mail address, T. Kailath: tk@rascals.stanford.edu
E-mail address, V. Olshevsky: olshevsk@rascals.stanford.edu