TRANSFORMATION TECHNIQUES FOR TOEPLITZ AND TOEPLITZ-PLUS-HANKEL MATRICES PART II. ALGORITHMS

A. W. BOJANCZYK AND GEORG HEINIG

CTC96TR251, 8/96

Abstract. In the first part [13] of the paper transformations mapping Toeplitz and Toeplitz-plus-Hankel matrices into generalized Cauchy matrices were studied. In this second part fast algorithms for LU-factorization and inversion of generalized Cauchy matrices are discussed. It is shown that the combination of transformation pivoting techniques leads to algorithms for indefinite Toeplitz and Toeplitz-plus-Hankel matrices that are more stable that the classical ones. Special attention is paid to the symmetric and hermitian cases.

1. INTRODUCTION

To begin with let us recall the definition of a (generalized) Cauchy matrix. Let $c = (c_i)_1^n$ and $d = (d_j)_1^n$ be fixed n-tuples of complex numbers and $A = [a_{ij}]_1^n$ a given matrix. Then the *Cauchy rank* of A with respect to c and d is, by definition, the rank r of the matrix

$$\nabla_{c,d}(A) = [(c_i - d_j)a_{ij}]_1^n.$$

If r is small compared with the order of the matrix A, then A will be called (generalized) *Cauchy matrix*. Cauchy matrices in the classical sense are matrices with $(c_i - d_j)a_{ij} = 1$ for all i and j and have, therefore, Cauchy rank one.

Cauchy matrices appear directly in many applications, for example in rational interpolation (see [10], [2]). The motivation for the present paper is, however, the observation that they appear indirectly as the result of transformations of the form $A \to \mathcal{C}_1 A \mathcal{C}_2$ of Toeplitz and Toeplitz-plus-Hankel matrices and their generalizations where $\mathcal{C}_1$, $\mathcal{C}_2$ are related to discrete Fourier or real trigonometric transformations. This was first observed in [10], [11] for complex Fourier transformations and in [8] and [18] for some real trigonometric transformations. In the first part of our paper [13] we presented an account of different transformations transforming Toeplitz and Toeplitz-plus-Hankel matrices into Cauchy matrices. All these transformations can be carried out in an efficient and stable way [1].

One of the properties of Cauchy matrices is that permutations of columns and rows preserve the Cauchy structure. This is not the case with Toeplitz and Toeplitz-plus-Hankel matrices where the Toeplitz and Toeplitz-plus-Hankel structures are

[1] Actually the efficiency of the FFT depends on their length. Complexity $O(n \log n)$ can be reached only if n is a product of small primes. In a forthcoming paper it will be discussed how to overcome this difficulty (see [12]) using another approach.

destroyed by such permutations. This property of Cauchy matrices provides additional freedom in how matrix decomposition algorithms based on Gaussian elimination can be applied to Cauchy matrices. It turns out that instead of dealing with Toeplitz and Toeplitz-plus-Hankel matrices, it is worthwhile, from the numerical point of view, to deal with the corresponding Cauchy matrices. This is the subject of Part II of this paper.

Fast inversion algorithms for Cauchy matrices were proposed for the first time in [14]. The algorithms there are of Levinson type and are based on the "bordering method". In [6] and [7] algorithms of Schur type for matrices with "recursive structure" were presented that also include fast algorithms for Cauchy matrices. The Schur type algorithms are in particular convenient for parallel processing. Concerning more references in this development we refer to the survey paper [17]. In [10] both Levinson and Schur type (called type-I and type-II, respectively) algorithms were derived from the interpolation interpretation of Cauchy systems of equations. Based on the observation that the Cauchy rank is inherited by Schur complementation it was observed in [9] and [16] independently that Gaussian elimination can be carried out in an efficient way considering only the generators, which are the vectors with the help of which the matrix $\nabla_{c,d}(A)$ is "generated". This result is implicitly also contained in [10] (compare Theorem 5.1 and Proposition 5.1). The corresponding Gauss-Schur algorithms was presented in [10] and [8]. The difference between the *Algorithm GKO* in [8] and the type-II algorithm in [10] is that the type-II algorithm in [10] computes the LU-factorization of the Cauchy matrix and its inverse in parallel, while the *Algorithm GKO* in [8] computes only the LU-factorization of the original matrix (by the same formulas as the type-II algorithm) and uses backward substitution for the solution of the corresponding systems of equations.

Gaussian elimination can be combined with partial pivoting which leads, as numerical experiments show, in practice to fairly stable algorithms. However, as it is shown in [20], if one applies the fast Cauchy version of Gaussian elimination then it can happen that there is a growth in the generators during the computation. According to [19] in all examples in [20] the growth can be avoided by a proper choice of the generators. Moreover, in [18] an orthogonalization technique for the generators combined with some approximately complete, fast pivoting is proposed, and it is shown that by applying this technique the growth becomes moderate.

In the first part of this paper main attention was paid to transformations preserving properties like realness and symmetry. Once the Cauchy matrix is real and symmetric, the question is how to carry out pivoting while preserving the structure of the matrix. This question is one of the main topics of this paper [2].

Let us sketch the further contents of the paper. In Section 2 we consider algorithms for general Cauchy matrices. First we present the Gauss-Schur type algorithm *LU-Cauchy* for the LU-Factorization of a Cauchy matrix in the form how it was designed in [8] and in other form derived in [10]. We also present a type-I (Levinson-type) algorithm *UL-INVCAUCHY* for the UDL-factorization of the inverse matrix with a pure matrix derivation (and correcting a mistake in [10]).

[2] After the the first draft of the paper was completed we learned that T. Kailath and V. Olshevsky developed independently similar ideas in connection with symmetric pivoting for Cauchy matrices (see [15].

In Section 3 we discuss first the specifics of symmetric and hermitian Cauchy matrices and after that we adopt the Bunch-Kaufman-Parlett method for symmetric pivoting to the case of symmetric and hermitian Cauchy matrices. In Section 4 we discuss the application of this to matrices appearing as the result of transformations of Toeplitz and Toeplitz-plus-Hankel presented in the first part of the paper [13]. This concerns all types of complex transformations discussed in [13] and the sine-I transformation as an example of a real trigonometric transformation. The implementation for the other real transformations would be similar, but it is slightly more complicated.

A reason to use another transformation than sine-I could be that the efficiency of these transformations depend on their length, so sine-I is efficient if $n+1$ is a product of small primes, cosine-I if $n-1$ and cosine-II if n is a product of small primes.

The results of [18] suggest that the mixed sine-I/cosine-I transformation should not be used for transformation since a substantial growth of the elements during the computation could be expected. A better mixed transformation is the combination of cosine-II and sine-IV which is the transformation proposed in [18].

In Section 5 we present the results of numerical experiments. Experiments were carried out for complex hermitian Toeplitz-plus-Henkel and for real symmetric Toeplitz systems of linear equations.

2. FAST FACTORIZATION ALGORITHMS FOR GENERAL CAUCHY MATRICES

In this section we consider general Cauchy matrices $C = [a_{ij}]_1^n$. If the Cauchy rank of C with respect to $c = (c_i)_1^n$ and $d = (d_j)_1^n$ equals r then there exist vectors z_i and $y_j \in \mathbf{C}^r$ satisfying

$$(c_i - d_j)a_{ij} = z_i^T y_j \tag{2.1}$$

The relation (2.1) can be written as a displacement (Sylvester) equation

$$D(c)C - CD(d) = ZY^T, \tag{2.2}$$

where $Z = \mathrm{col}\,(z_i^T)_1^n$ and $Y = \mathrm{col}\,(y_i^T)_1^n$, $D(c) = \mathrm{diag}\,(c_i)_1^n$, $D(d) = \mathrm{diag}\,(d_j)_1^n$. Following [17] we call the matrices Y and Z generators of C.

We restrict ourselves to the following two cases:

Case A: $c_i \neq d_j$ for all i and j.

Case B: $c_i = d_i$ for all i and the c_i are pairwise different.

This covers all cases appearing in Part I. Clearly, in the case A the generators define C uniquely; in the case B the generators define C only for the off-diagonal entries.

It is obvious that the inverse of a Cauchy matrix is a Cauchy matrix again and has the same Cauchy rank. In fact, (2.2) implies

$$D(d)C^{-1} - C^{-1}D(c) = -XW^T,$$

where X and W are the solutions of the "fundamental" equations $CX = Z$ and $W^TC = Y^T$. In case A the solutions X and W define the matrix C^{-1} completely. In case B one has still to find the diagonal of C^{-1}. This can be done by solving also the equation $Cu = \mathbf{1}$ with $\mathbf{1} = (1\ 1\ \ldots 1)^T$. Then the diagonal of C^{-1} is given by

$$\operatorname{diag} C^{-1} = u - \left(\sum_{j \neq i} \frac{x_i^T w_j}{c_i - c_j} \right)_{i=1}^{n} .$$

A basic observation which leads to the construction of fast inversion and factorization algorithms is that the Cauchy structure is inherited under Schur complementation. To show this one has to apply the "magic wand" Schur complement formula for a partioned matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

with nonsingular A_{11} which reads as follows

$$A = \begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & A_{11}^{\#} \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix}, \tag{2.3}$$

where $A_{11}^{\#}$ denotes the Schur complement $A_{11}^{\#} = A_{22} - A_{21}A_{11}^{-1}A_{12}$. If A_{11} is 1×1 then (2.3) just illustrates one step of Gaussian elimination. Repeating these steps one gets an LU-factorization of the matrix A. A block LU-factorization of A is obtained in an analogous way by choosing pivots A_{11} as square submatrices of size $k \geq 2$.

Lemma 1. (cf.[9], [16]) *Let*

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

be a partitioned Cauchy matrix corresponding to (c, d), $c = (c_1, c_2)$, $d = (d_1, d_2)$ *is the partition of the tuples* c *and* d *according to that of* C, *and let the generators of* C *be given by*

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}.$$

Then the Schur complement $C_{11}^{\#}$ *is Cauchy matrix corresponding to* (c_2, d_2) *with generators*

$$Z^{(1)} = Z_2 - C_{21}C_{11}^{-1}Z_1, \qquad {Y^{(1)}}^T = Y_2^T - Y_1^T C_{11}^{-1} C_{12}.$$

This lemma leads to the following algorithm of LU-factorization of strongly nonsingular Cauchy matrices which, for the case A, is a part of the type-II algorithm described in [10] and coincides, also for case A, with the GKO-algorithm in [8].

Algorithm LU-CAUCHY

1. Initialization: $Z^{(0)} = Z = \begin{bmatrix} z_1^{(0)} \\ Z_2^{(0)} \end{bmatrix}$, $Y^{(0)} = Y = \begin{bmatrix} y_1^{(0)} \\ Y_2^{(0)} \end{bmatrix}$.

In case B: $d^{(0)} = \operatorname{diag} C$.

2. For $k = 0, \ldots, n-1$,

(a) Compute $a^{(k)}$, $l^{(k)} = (l_i^{(k)})_1^{n-k}$, $u^{(k)} = (u_j^{(k)})_1^{n-k}$ by

$$a^{(k)} = \frac{z_1^{(k)T} y_1^{(k)}}{c_{1+k} - d_{1+k}} \quad \text{(case A)}, \quad a^{(k)} = d_{11}^{(k)} \quad \text{(case B)}$$

$$l_i^{(k)} = \frac{1}{a^{(k)}} \frac{z_i^{(k)T} y_1^{(k)}}{c_{i+k} - d_{1+k}}, \quad u_j^{(k)} = \frac{1}{a^{(k)}} \frac{z_1^{(k)T} y_j^{(k)}}{c_{1+k} - d_{j+k}}.$$

(b) Compute

$$Z^{(k+1)} = \begin{bmatrix} z_1^{(k+1)T} \\ Z_2^{(k+1)} \end{bmatrix} = Z_2^{(k)} - l^{(k)} z_1^{(k)T},$$

$$Y^{(k+1)} = \begin{bmatrix} y_1^{(k+1)T} \\ Y_2^{(k+1)} \end{bmatrix} = Y_2^{(k)} - u^{(k)} y_1^{(k)T}.$$

(c) In case B compute $d^{(k+1)} = (d_i^{(k+1)})_1^{n-k-1}$ by

$$d_i^{(k+1)} = d_{i+1}^{(k)} - a^{(k)} l_i^{(k)} u_i^{(k)}.$$

The vectors $l^{(k)}$ and $u^{(k)}$ form the nonzero sections of rows and columns of the lower and the upper triangular factors of C, respectively.

If one wants to solve a Cauchy linear system then one can use this factorization algorithm followed by backward substitution. Another possibility consists in the recursive computation of the solution of the equations $CX = Z$ and $W^T C = Y^T$ and the application of the formula for the inverse matrix. This leads to the type-I algorithm described in [10] which computes C^{-1}. This way can be preferable if one has to solve several equation with one and the same matrix and different right-hand sides because multiplication by C^{-1} can be carried out in $O(n \log n)$ operations, see [10].

The algorithm *LU-CAUCHY* can be combined with usual partial pivoting. Thus the condition concerning the strong nonsingularity of C is not essential.

Let us show how to get a factorization for the inverse matrix A^{-1}. This leads to a Levinson-type algorithm for computing the generators of C^{-1} which is similar to that presented in [14] and coincides with that in [10] for the case A [3].

The Levinson-type recursions are based on the "bordering" formula for the inverse matrix

$$\begin{aligned} A^{-1} &= \begin{bmatrix} I & -A_{11}^{-1} A_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & (A_{11}^{\#})^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{21} A_{11}^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} A_{11}^{-1} - URL & -XR \\ -RY & R \end{bmatrix}, \end{aligned} \tag{2.4}$$

where

$$U = A_{11}^{-1} A_{12}, \quad L = A_{21} A_{11}^{-1}, \quad R = (A_{11}^{\#})^{-1}.$$

We apply this formula to Cauchy matrices $C = [a_{ij}]_1^n$ satisfying (2.1). Let C_k $(k = 1, \ldots, n)$ be the principal submatrices which are assumed to be nonsingular. Furthermore represent C_{k+1} as

$$C_{k+1} = \begin{bmatrix} C_k & q_k \\ p_k^T & s \end{bmatrix}.$$

The upper and lower factors of the UDL-factorization of C^{-1} are obtained from the solutions of the equations

$$C_k u_k = q_k \qquad l_k^T C_k = p_k^T. \tag{2.5}$$

[3] Let us note that the corresponding recursion formula in [10] is not correct in case some of the d_j's coincide.

In order to find a fast algorithm for the factorization of C^{-1} we have to describe the recursion for the vectors q_k and p_k and for the generators X_k and W_k of C_k^{-1}, i.e. for the solutions of the equations $C_k X_k = Z_k$ and $W_k^T C_k = Y_k^T$, and in the case B also for the diagonals of C_k^{-1}.

We introduce the notations

$$Z_k = \mathrm{col}\,(z_i^T)_1^k,\ Y_k = \mathrm{col}\,(y_j)_1^k,\ D_k(c) = \mathrm{diag}\,(c_i)_1^k,\ D_k(d) = \mathrm{diag}\,(d_j)_1^k.$$

Furthermore, let $\{e_i\}$ stand for the standard basis in $\mathbf{C}^k$.

Lemma 2. *1) The solutions $u_k = (u_{kj})_1^k$ of the equations* (2.5) *can be computed from the generators X_k, W_k via*

$$u_{kj} = \begin{cases} (d_j - d_{k+1})^{-1} e_j^T X_k (y_{k+1} - W_k^T q_k) & : \quad d_j \neq d_{k+1} \\ e_j^T X_k W_k^T (D_k(c) - d_{k+1} I_k)^{-1} q_k & : \quad d_j = d_{k+1} \end{cases}, \tag{2.6}$$

where

$$q_k = (D_k(c) - d_{k+1} I_k)^{-1} Z_k y_{k+1}. \tag{2.7}$$

Analogous equalities hold for the solution l_k.

2) The generators X_{k+1} for C_{k+1} can be computed from the generators of C_k via

$$X_{k+1} = \begin{bmatrix} X_k \\ 0 \end{bmatrix} - \frac{1}{\alpha_k} \begin{bmatrix} u_k \\ -1 \end{bmatrix} (z_{k+1}^T - p_k^T X_k), \tag{2.8}$$

where

$$p_k^T = z_{k+1}^T Y_k (c_{k+1} I_k - D_k(d))^{-1}, \quad \alpha_k = a_{k+1,k+1} - p_k^T u_k. \tag{2.9}$$

An analogous equality holds for the generator W_{k+1}.

3) The diagonal of C_{k+1}^{-1} can be computed from the diagonal of C_k^{-1} via

$$\mathrm{diag}\, C_{k+1}^{-1} = \left(\mathrm{diag}\, C_k^{-1} - \alpha_k^{-1} \mathrm{diag}\,(u_{ki} l_{ki})_{i=1}^k, \quad \alpha_k^{-1}\right). \tag{2.10}$$

From the displacement equality

$$D_k(d) C_k^{-1} - C_k^{-1} D_k(c) = -X_k W_k$$

we conclude that

$$\begin{aligned} & (D_k(d) - d_{k+1} I_k) C_k^{-1} (D_k(c) - d_{k+1} I_k)^{-1} Z_k y_{k+1} \\ = \;& (C_k^{-1} - X_k W_k^T (D_k(c) - d_{k+1} I_k)^{-1}) Z_k y_{k+1} \\ = \;& X_k (y_{k+1} - W_k^T q_k). \end{aligned}$$

Taking (2.7) into account we obtain

$$(D_k(d) - d_{k+1}) u_k = X_k (y_{k+1} - W_k^T q_k). \tag{2.11}$$

In case when $d_j \neq d_{k+1}$ this leads to the first equality in Lemma 2. In case when $d_j = d_{k+1}$ we obtain

$$0 = e_j^T C_k^{-1} q_k - e_j^T X_k W_k^T (D_k(c) - d_{k+1} I_k)^{-1} q_k.$$

This leads to the second equality in Lemma 2.

The recursion in 2) can be checked immediately and the recursion in 3) follows from (2.4).

□

This lemma leads to the following algorithm for the factorization of C^{-1} for a strongly nonsingular Cauchy matrix $C = [a_{ij}]_1^n$.

Algorithm UL-INVCAUCHY

1. Initialization: $X_1 = \frac{1}{a_{11}} Z_1$, $W_1 = \frac{1}{a_{11}} Y_1$, in the case B: $\operatorname{diag} C_1^{-1} = \frac{1}{a_{11}}$.
2. For $k = 1, \ldots, n-1$,
(a) Compute q_k by (2.7) and p_k by (2.9).
(b) Compute u_k by (2.6) and analogously l_k.
(c) Compute α_k by (2.9) and X_{k+1} by (2.8), and analogously W_{k+1}.
(d) In case B compute $\operatorname{diag} C_{k+1}^{-1}$ by (2.10).

Now the vectors $[-u_k^T \quad 1]^T$ form the nonzero sections of columns of the upper triangular factor and the vectors $[-l_k \quad 1]$ the lower triangular factor of the UDL-factorization of C^{-1}. The numbers α_k correspond to the diagonal factor. In order to get the inverse matrix one has only to store the generators $X = X_n$ and $W = W_n$ and, in case B, also the diagonal $\operatorname{diag} C_n^{-1}$. This leads also to a Levinson-type algorithm for solving Cauchy systems. From the numerical view point it may be less accurate when the nodes are close to each other (part of the reason for this is the the presence of the division by $(d_j - d_{k+1})$ in the formula (2.6). Furthermore, pivoting is more expensive and the algorithm is not very suitable for parallelization due to inner product calculations. But the Levinson-type algorithm is important because it provides a simple updating formula for the inverse matrix or for the solution of systems of equations.

3. SYMMETRIC PIVOTING

In this section we describe certain features of fast Cauchy solvers that can be exploited to lower the number of operations when the matrices under consideration are symmetric or hermitian. This will lead us to symmetric (or hermitian) fast Cauchy solvers. We next describe how these symmetric solvers can be applied to the special Cauchy matrices obtained from Toeplitz and Toeplitz-plus-Hankel matrices. We start with some general observations.

3.1. **Dependent Generators.** If the generators Z and Y of the Cauchy matrix depend via a relation

$$Y = ZK \quad \text{or} \quad Y^T = Z^* K, \tag{3.1}$$

where K is an $r \times r$ matrix, then it is sufficient to compute in the algorithm *LU-CAUCHY* only one of the generators. This follows immediately from the fact that the relation (3.1) is inherited by Schur complementation. This also applies to the algorithm *UL-INVCAUCHY*.

The following is easily checked.

Proposition 3.1. 1) *If $c_i = d_i$ $(i = 1, \ldots, r)$ and C is symmetric then r is even and the first relation of* (3.1) *holds for certain Z and*

$$K = \begin{bmatrix} 0 & I_{r/2} \\ -I_{r/2} & 0 \end{bmatrix}.$$

2) *If $c_i = -\bar{d}_i$ $(i = 1, \ldots, r)$ and C is hermitian then the second relation of* (3.1) *holds for certain Z and*

$$K = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}.$$

The first assertion follows from the fact that in this case $\nabla_{c,d}(C)$ is skew symmetric. Using symmetric Gaussian elimination, it can be easily shown that a skew-symmetric matrix S has even rank and admits the representation $S = ZKZ^T$, where K has the from as in Proposition 3, case (1). The second assertion follows from the fact that in this case $\nabla_{c,d}(C)$ is hermitian.

For some Cauchy matrices obtained as the result of transformations in part I the generators of the Stein displacement are connected rather them those of the Sylvester displacement [4]. That means we have a relation

$$C - D(c)CD(d) = ZKZ^T \quad \text{or} \quad C - D(c)CD(d) = ZKZ^* \tag{3.2}$$

for a certain $r \times r$ matrix K and a $n \times r$ matrix Z. Matrices satisfying a relation (3.2) are sometimes called *(generalized) Pick matrices.* Clearly, they are also (generalized) Cauchy matrices.

If C satisfies (3.2) then it is not immediately clear how the relation between the generators are inherited in the Schur complement. Our suggestion is to transform the Stein displacement equation (3.2) into a Sylvester displacement equation using linear fractional transformations of the nodes c_i and d_j. The following is easily checked.

Proposition 3.2. *Let ϕ be a linear fractional transformation, $\phi(\lambda) = (\alpha\lambda - \beta)/(\alpha\lambda + \beta)$. and let $c_i = \phi(x_i)$, $d_j = \phi(y_j)$. Then the Stein displacement equation $C - D(c)CD(d) = R$ is equivalent to the Sylvester displacement equation*

$$D(x)C + CD(y) = \frac{1}{2\alpha\beta} D(\alpha x + \beta \mathbf{1}) R D(\alpha y + \beta \mathbf{1}) \ .$$

3.2. **Bunch-Kaufman-Parlett Pivoting Algorithm.** Now we discuss the problem of pivoting for the symmetric and hermitian cases. In the symmetric (or hermitian) case by working with one generator only we hope to decrease the number of arithmetic operation required in the unsymmetric case. However, if proper care is not taken, pivoting will destroy the symmetry. In order to keep the symmetry throughout the computation partial pivoting can be replaced by diagonal pivoting. However, simple diagonal pivoting, in general, does not bound the rate of growth in the data. The rate of growth can be bounded if some sort of block diagonal pivoting is used. Several block diagonal pivoting techniques were discussed in the literature for computing factor of (indefinite) symmetric matrices (see [3], [4], [5]). The most widely used is the technique presented in [5]. We will refer to this technique as *Bunch-Kaufman-Parlett* pivoting algorithm, or the *BKP* algorithm for short. This algorithm uses block diagonal pivoting with blocks of order 1 or 2. In what follows we will outline the *BKP* algorithm for factorization of a general indefinite symmetric matrix, and next we will show how it can be incorporated into fast symmetric Cauchy solvers.

For a given symmetric matrix A, the *BKP*-algorithm computes a permutation P, a unit lower triangular matrix M and a block diagonal matrix D with with block of order 1 or 2 such that

$$P^T AP = MDM^T \ . \tag{3.3}$$

The permutation matrix P depends on a parameter $0 < \alpha < 1$ which controls the rate of growth of pivots. The rate is minimized for $\alpha = (1 + \sqrt{17})/8$.

[4]The Stein displacement is often called *discrete time* and the Sylvester displacement *continuous time* Lyapunov displacement (cf.[15])

The factorization proceeds in stages. In stage $k+1$ ($k = 0, \ldots, n-1$) one operates on the matrix $A^{(k)}$, with $A^{(0)} = A$. Given $A^{(k)}$, a symmetric permutation of rows and columns is applied to $A^{(k)}$ so a satisfactory diagonal pivot block of order 1 or 2 can be determined. If $P^{(k)}$ is such a permutation, and

$$(P^{(k)})A^{(k)}(P^{(k)})^T = \begin{bmatrix} D^{(k)} & E^T \\ E & G \end{bmatrix} , \tag{3.4}$$

with $D^{(k)}$ and G symmetric, then the pivot block is $D^{(k)}$, and $A^{(k+1)}$ is computed as the Schur complement of $D^{(k)}$, i.e.

$$A^{(k+1)} = G - E(D^{(k)})^{-1}E^T .$$

The significant elements of the new columns of M are computed as $E(D^{(i-1)})^{-1}$. The whole process is iterated until all columns of M are determined.

The most important aspect of the method is a procedure for finding the permutation matrix $P^{(k)}$ and determining the order of the pivot block $D^{(k)}$. In [5] this is done in two steps. First, one checks whether a suitable diagonal permutation that would bound pivot element growth exists. If such a pivot cannot be found a satisfactory 2×2 pivot is calculated instead.

A single iteration, that is the transitions from $A^{(k)}$ to $A^{(k+1)}$ is summarized in *Algorithm BKP*. There, in order to simplify notation, superscripts were dropped. Also, following [5], I_{ij} is used to denote a permutation which interchanges row i with row j.

Algorithm BKP

Let $A = A^{(k)} = [a_{ij}]$ be the current $(n-k) \times (n-k)$ matrix. One iteration of the algorithms computes a permutation P, pivot block D of order s ($s = 1$ or 2), s columns of the lower triangular factor M, and the Schur complement $A^{(k+1)}$ corresponding to D.

1. Initialization: Set $\alpha = (1 + \sqrt{17})/8$.
2. (a) Search for a 1×1 diagonal pivot, i.e. set $s = 1$,
   - (a1) compute $\lambda = |a_{m1}| = \max_{2 \le i \le n-k} |a_{i1}|$,
   - (a2) if $|a_{11}| \ge \alpha\lambda$ the element c_{11} is a good pivot element, set $P = I$ and go to (c),
   - (a3) compute $\sigma = \max_{i \ne m} |a_{i1}|$,
   - (a4) if $|a_{11}|\sigma \ge \alpha\lambda^2$ the element a_{11} is still a good pivot element, set $P = I$ and go to (c),
   - (a5) if $|a_{mm}| \ge \alpha\sigma$ the element a_{mm} is a good pivot element, set $P = I_{1m}$ and go to (c).

   (b) There is not any satisfactory 1×1 diagonal pivot, set $s = 2$ and $P = I_{2m}$.

   (c) Compute
   - (c1) set $PAP^T = \begin{bmatrix} D & E^T \\ E & G \end{bmatrix}$, where D is $s \times s$,
   - (c2) compute the Schur complement $A^{(k+1)}$ of D,
   - (c3) compute the significant elements of the new columns in M as ED^{-1}.

The algorithm easily extends to the case of a hermitian A. A repeated application of *Algorithm BKP* will produce the desired decomposition (3.3).

The decomposition can be used to solve a linear system of equations $Ax = b$. Given the symmetric decomposition $PCP^T = MDM^T$ this can be accomplished in $MS_r(n) = n^2$ multiplications and $AS_r(n) = n^2$ additions for a real A. For a complex matrix most operations has to be performed in complex arithmetic. A complex multiplication is equivalent to four real multiplications and two real additions. A complex addition is equivalent to two real additions. Thus in the complex case, given the symmetric (or hermitian) decomposition of A, the cost of solving the linear system $Ax = b$ is $MS_c(n) = 4n^2$ multiplications and $AS_c(n) = 4n^2$ additions.

3.3. **Symmetric pivoting on generators.** Recall that the possibility to speed up the factorization procedure for generalized Cauchy matrices from $O(n^3)$ to $O(n^2)$ complexity is based on the fact that it is sufficient to carry out all operations on the generators. Concerning pivoting the following obvious observation is in order.

If C is a Cauchy matrix with generators Z and Y and P is a permutation matrix then $\tilde{C} = PCP^T$ has the generators PZ and PY and the nodes of $\tilde{C}$ are the corresponding permutations of the original nodes.

The above observation when combined with *Algorithm LU-Cauchy* and *Algorithm BKP* leads to *Algorithm BKP-Cauchy* which is described below.

Let $A_{i:j}$ denote a matrix composed of rows i through j of the matrix A. Similarly, if $J \subset \{1, 2, ..., n\}$ then A_J will denote a matrix composed of all rows of A which indices are taken from J (in the increasing order).

Algorithm BKP-Cauchy

Let C be an $n \times n$ Cauchy matrix defined by the generators Y and $Z = YK$, the nodes (c, d) where $c_i \neq c_j$ for $i \neq j$, and the diagonal diag (C) if $c = d$. The algorithms computes the permutation P, the pivot block D of order s ($s = 1$ or 2) s columns of the lower triangular factor M, and the generators $\hat{Y}$ and $\hat{Z}$, such that $\hat{Z} = \hat{Y}K$.

1. Initialization: Set $\alpha = (1 + \sqrt{17})/8$, and $s = 1$,
2. (a) Look for a 1×1 diagonal pivot,
   - (a1) compute the off diagonal entries of the first column of C, $C_1 = (a_{i1})_1^n = (\mathrm{diag}\,(c_{2:n}) - d_1 I)^{-1} Y_{2:n} K y_1$, and set $\lambda = |a_{m1}| = \max_{2 \leq k \leq n} |a_{k1}|$,
   - (a2) if $|a_{11}| \geq \alpha\lambda$ set $P = I$, $D = a_{11}$, $E = C_1$, and go to (c),
   - (a3) compute the off diagonal entries of the mth column of C, $C_m = (a_{im})_1^m = (\mathrm{diag}\,(c_J) - d_m I)^{-1} Y_J K y_m$, where $J = \{1, ..., m-1, m+1, ..., n\}$, and set $\sigma = \max_{i \neq m} |a_{im}|$,
   - (a4) if $|a_{11}|\sigma \geq \alpha\lambda^2$ set $P = I$, $D = a_{11}$, $E = C_1$, and go to (c),
   - (a5) if $|a_{mm}| \geq \alpha\sigma$ set $P = I_{1m}$, $D = a_{mm}$, $E = C_m$, and go to (c)

   (b) Set $s = 2$,
   - (b1) $P = I_{2m}$,
   - (b2) $D = \begin{bmatrix} a_{11} & a_{1m} \\ a_{m1} & a_{mm} \end{bmatrix}$, and $E = [C_1, C_m]$.

   (c) Compute other relevant quantities
   - (c2) permute Y, E, b, c, d and diag (C) according to P,
   - (c3) compute the generator $\hat{Y} = Y_{s+1:n} - \hat{E}D^{-1}Y_{1:s}$,

where $\hat{E}$ equals E with its first s rows dropped,

(c4) compute entries in rows $s+1$ through n in the first s columns of the lower triangular factor M as $\hat{E}D^{-1}$,

(c5) compute diag $(\hat{C})$ = diag (C) − diag $(ED^{-1}E^T)$.

A repeated application of *Algorithm BKP-Cauchy* will produce the desired decomposition (3.3). The cost of the algorithm will depend on the order $n \times r$ of generators, and on how often the step (a3) followed by (c) are executed. If (a3) is always performed and is always followed by (c), then, for a real matrix, it is maximal and is $\frac{3}{2}n^2r$ multiplications and $\frac{3}{2}n^2r$ additions. On the other hand if (a2) is always followed by (b) or (c) then the cost is only n^2r multiplications and n^2r additions. The cost of *Algorithm BKP* for a complex matrix is roughly four times that for a real matrix.

Estimates of the cost (measured by the number of real multiplications) of computing the solution of a symmetric $n \times n$ linear system $Cx = b$ given the generators of C are given in Table 1.

matrix C	rhs b	cost (real multiplications) max	min
complex	complex	$4(\frac{3}{2}n^2r + n^2)$	$4(n^2r + n^2)$
real	complex	$\frac{3}{2}n^2r + 2n^2$	$n^2r + 2n^2$
real	real	$\frac{3}{2}n^2r + n^2$	$n^2r + n^2$

Table 1. Cost of solving $Cx = b$.

3.4. **Computation of generators based on DFT.** In this subsection we provide some more detailed description of generators for Cauchy matrices appearing in Section 2, Part I as the result of DFT transformed Toeplitz matrices.

For a given real vector $a = (a_j)$, the matrix-vector multiplication $\mathcal{F}(\xi)a$ can be accomplished via FFT in $5n \log n$ real arithmetic operations (assuming that $n = 2^t$ for some positive integer t). These kind of matrix-vector multiplications compute generators of the Cauchy matrices considered in Sections 2, Part I. As only a small number of operations of the type $\mathcal{F}(\xi)a$ are performed in transforming Toeplitz into Cauchy matrices, we will not include the cost of these matrix-vector operations in the overall estimation of the amount of solving the resulting Cauchy linear systems.

Let $T = [a_{i-j}]$ and $H = [s_{i+j}]$ be $n \times n$ Toeplitz and Hankel matrices, respectively. We will now list generators corresponding to the transformations considered in Sections 2.1-2.4, Part I.

(a) `HTfftHC` - hermitian Toeplitz to complex hermitian Cauchy.

Suppose that $a_{-k} = \bar{a}_k$ and $c_i = d_i = \omega_j = \exp(2\pi \mathrm{i} j/n)$ ($\mathrm{i} = \sqrt{-1}$) According to Theorem 4, Part I,

$$\tilde{C} = \mathcal{F}(1)T\mathcal{F}(1)^*$$

is a hermitian Cauchy matrix. More important for our purposes is the fact that the matrix $C = \frac{n\mathrm{i}}{2}\mathrm{diag}\,(\omega)\tilde{C}$ is a symmetric Cauchy matrix with $c = d$. The quantities defining C can be derived from the vectors

$$a' = [\frac{a_0}{2}, a_1, \ldots, a_{n-1}]^T, \quad a'' = [na_0, (n-1)a_1, \ldots, a_{n-1}]^T, \tag{3.5}$$

The generator Y and the diagonal of C are given by

$$\operatorname{diag}(C) = \frac{n\mathrm{i}}{2}\operatorname{diag}(\omega) f \ , \tag{3.6}$$

$$Y = \begin{bmatrix} e_1 & 1 \\ \vdots & \vdots \\ e_n & 1 \end{bmatrix} , \tag{3.7}$$

where

$$e = \operatorname{Im}\mathcal{F}(\xi)a' \quad \text{and} \quad f = \operatorname{Re}\mathcal{F}(\xi)a'' \ . \tag{3.8}$$

Finally, $Z = YK$ with $K = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$.

Since C is symmetric and $c = d$ *Algorithm BKP-Cauchy* can be readily applied to C.

(b) `HTfftSC` - hermitian Toeplitz to real symmetric Cauchy.

Let ω, e, f, $\operatorname{diag}(C)$, Y, K and $\tilde{C}$ be like in (a). The fractional transformation

$$c_j = \frac{\zeta + \omega_j}{\zeta - \omega_j} \cdot \mathrm{i} \tag{3.9}$$

maps the complex node vector ω into a real node vector $c = (c_j)$. The matrix

$$C = \operatorname{diag}(c_j + \mathrm{i})\, \tilde{C} \operatorname{diag}(c_j - \mathrm{i})$$

is a real symmetric Cauchy matrix. The relevant quantities defining C are the same as in the case (a) except for the nodes and the diagonal elements, which are now c and $d = c$, and $\operatorname{diag}(C) = \operatorname{diag}(c_j^2 + 1) f$.

(c) `STfftSC` - symmetric Toeplitz to symmetric Cauchy .

Let $\xi = \mathrm{i}$ and $\omega = (\omega_j)_1^n$ with $\omega_j = \exp((4j+1)\pi\mathrm{i}/2n)$. Then, according to Theorem 6, Part I,

$$\tilde{C} = \mathcal{F}(\mathrm{i}) T \mathcal{F}(\mathrm{i})^T$$

is a symmetric Cauchy matrix. The fractional transformation (3.9) transforms ω into a real vector c. Now, the matrix

$$C = \frac{1}{2\mathrm{i}} \operatorname{diag}(1 - c_j)\, \tilde{C} \operatorname{diag}(1 - c_j)$$

is a symmetric Cauchy matrix with the entries

$$c_{ij} = \frac{e_i + e_j}{c_i + c_j} \ ,$$

where

$$e = \mathcal{F}(\mathrm{i})a' + \mathcal{F}(-\mathrm{i})a' \tag{3.10}$$

and a' is defined by (3.5). Thus the matrix C is defined by the nodes c and $d = -c$, and the generators Y and $Z = YK$ where

$$Y = \begin{bmatrix} e_1 & 1 \\ \vdots & \vdots \\ e_n & 1 \end{bmatrix} \text{ and } K = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ .$$

Algorithm BKP-Cauchy can be now applied to C. The cost of solving $Tx = b$ is analogous to that in (a) for a complex T, and that in (b) for a real T.

Remark 3.1. *Note from (3.10) that when T is real then e is real also. Thus we have an alternative transformation (to the transformation in (b)) that takes a real symmetric Toeplitz matrix into a real symmetric Cauchy matrix. We will refer to this transformation in the Section on numerical experiments as* `RTfftSC`.

(d) `HTpHfftHC` - hermitian Toeplitz-plus-Hankel to hermitian Cauchy

We now consider the case when the Toeplitz-plus-Hankel $A = T+H$ is hermitian. As remarked in Section 3, Part I, if $\xi = -\eta = \mathrm{i}$ then

$$C = \mathcal{F}(\xi)A\mathcal{F}(\eta)^T$$

is hermitian. The transformation from A to C is based on the vectors

$$\begin{aligned} t_{\mathrm{col}} &= \quad \left[\tfrac{t_0}{2}, t_1, ..., t_{n-1}\right]^T, \quad t'_{\mathrm{col}} = \left[\frac{t_0}{2}n, (n-1)t_1, ..., t_{n-1}\right]^T \\ t_{\mathrm{row}} &= \quad \left[\tfrac{t_0}{2}, t_{-1}, ..., t_{-n+1}\right]^T, \quad t'_{\mathrm{row}} = \left[n\frac{t_0}{2}n, (n-1)t_{-1}, ..., t_{-n+1}\right]^T \end{aligned}$$

to transform T, and on the vectors

$$h_{\mathrm{row}} = \left[\frac{s_{n-1}}{2}, s_n, ..., s_{2n-2}\right]^T, \; h_{\mathrm{col}} = \left[\frac{s_{n-1}}{2}, s_{n-2}, ..., s_0\right]^T ,$$

to transform H. These vectors are transformed via $\mathcal{F}(\xi)$ or $\mathcal{F}(\eta)$. The transformed vectors are combined according to Theorem 7 to give $n \times 4$ generators Z and Y of C. The generators satisfy $\bar{Z} = YK$ for

$$K = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} .$$

The hermitian version of *Algorithm BKP-Cauchy* can then be applied.

3.5. **Computation of generators based on trigonometric transformations.** As was pointed out in Section 3, the trigonometric transformations have the advantage over the DFT that they are real and hence allow to stay in reals when applied to real data. There are numerous fast algorithms for applying trigonometric transformations, see [22], [23], [21]. However, as they are all of order $O(n \log n)$ for vectors of length n, a simple implementation based on the complex FFT can be used (in all cases when n is large) as it will only effect lower terms in the cost of solving the resulting Cauchy linear system.

The sine-I transformation in Theorem 9 is straightforward to implement and was used in numerical tests. For a Toeplitz-plus-Hankel matrix A this transformation operates on vectors

$$\begin{aligned}
t_{\text{col}} &= \left[\frac{t_0}{2}, t_1, ..., t_{n-1}\right]^T \\
t_{\text{row}} &= \left[\frac{t_0}{2}, t_{-1}, \ldots, t_{-n+1}\right]^T \\
t_{\text{col}-\text{row}} &= [t_0, t_1 + t_{-1}, \ldots, t_{n-1} + t_{-n+1}]^T \\
t'_{\text{col}-\text{row}} &= [nt_0, (n-1)(t_1 + t_{-1}), \ldots, t_{n-1} + t_{-n+1})]^T
\end{aligned}$$

to transform T, and on the vectors

$$\begin{aligned}
h_{\text{col}} &= \left[\frac{s_{n-1}}{2}, s_n, \ldots, s_{2n-2}\right]^T \\
h_{\text{row}} &= \left[\frac{s_{n-1}}{2}, s_{n-2}, \ldots, s_0\right]^T \\
s_{\text{col}-\text{row}} &= [s_{n-1}, s_{n-2} + s_n, \ldots, s_0 + s_{2n-2}]^T \\
s'_{\text{col}-\text{row}} &= [ns_{n-1}, (n-1)(s_{n-2} + s_n), \ldots, s_0 + s_{2n-2}]^T
\end{aligned}$$

to transform H. The transformed vectors are combined according to Theorem 9 to give $n \times 4$ generators Z and Y of C. The generators satisfy $\bar{Z} = YK$ for

$$K = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The hermitian version of *Algorithm BKP-Cauchy* can then be applied. We will refer to this transformation as `HTpHsineHC`.

As stated in Theorem 9, for a real symmetric Toeplitz matrix, the transformed matrix is a 2×2 block diagonal with blocks that are symmetric and each having Cauchy rank 2. The symmetric version of *Algorithm BKP-Cauchy* can be applied to each of the two blocks. This leads to a potential saving of 50% in the cost of solving a real symmetric Toeplitz system of linear equations. We will refer to this transformation as `STsineSC`.

4. NUMERICAL EXPERIMENTS

Numerical experiments were conducted on a MacIntosh PowerBook 180 computer using MATLAB 4.1. The relative machine precision was eps $= 2.2e - 16$. Memory limitations did not allow us to solve systems larger than 150×150.

In the tests we generated indefinite Toeplitz or Toeplitz-plus-Hankel matrices as follows. A matrix $\hat{A}$ was first generated randomly. Next $\hat{A}$ was modified to obtain another matrix A according to the formula

$$A = \hat{A} - (\lambda + \text{eps}^{\alpha})I_n$$

where λ was an eigenvalue of $\hat{A}$, eps the machine relative precision, and $0 < \alpha < 1$. Such modification does not change the underlying structure of the original Toeplitz or Toeplitz-plus-Hankel matrix. By varying α we could influence the magnitude of the condition number of A. We also varied the dimension n of A.

The right hand side vector b in the linear system $Ax = b$ was chosen in such a way so the true solution x was a vector of ones. This was because such a vector does not reflect potential ill-conditioning of the matrix and hence may expose numerical deficiencies of the solvers.

In the tests we measured the residual error $\mathrm{res}\, x$ in the computed solution $\tilde{x}$ as follows

$$\mathrm{res}\, x = \frac{||A\tilde{x} - b||}{||A|| \cdot ||\tilde{x}|| + ||b||} \,.$$

We also measured the relative error $\mathrm{res}\, C$ in the symmetric (or hermitian) decomposition of C as

$$\mathrm{res}\, C = \frac{||C - LML^T||}{||C||} \,,$$

and the relative error $\mathrm{res}\, A$ in the decomposition of A as

$$\mathrm{res}\, A = \frac{||A - FLML^T G||}{||A||} \,,$$

where F and G were the transformations that related the matrix A with the matrix C. The ratio of $\mathrm{res}\, A$ and $\mathrm{res}\, C$ measures the degradation of accuracy caused by the transformation from A to C and back to A. For the algorithm *LU-CAUCHY*, $\mathrm{res}\, C$ and $\mathrm{res}\, A$ are defined in an analogous way. For the algorithm *UL-INVCAUCHY*, $\mathrm{res}\, C$ and $\mathrm{res}\, A$ would have to be computed for C^{-1}, hence only $\mathrm{res}\, x$ was recorded.

4.1. **Hermitian Toeplitz-plus-Hankel.** Solving systems of equations with a complex hermitian Toeplitz-plus-Hankel coefficient matrix we compared the following algorithms.

- **ALG 1.1:** Use `HTpHfftHC` to transform the matrix into a hermitian Cauchy matrix. Then apply algorithm *LU-CAUCHY* in combination with algorithm *BKP-Cauchy* for pivoting.
- **ALG 1.2:** Instead of `HTpHfftHC` use the real transformation `HTpHsineHC`. (The advantage is that the resulting Cauchy matrix is real if the original matrix is real). Then apply *LU-CAUCHY* in combination with *BKP-Cauchy* as in **ALG 1.1**.
- **ALG 1.3:** Apply the transformation `HTpHfftHC` and algorithm *LU-CAUCHY* but instead of *BKP-Cauchy* pivoting as in **ALG 1.1**, use the standard partial pivoting.
- **ALG 1.4:** Use `HTpHfftHC`. Instead of algorithm *LU-CAUCHY* apply algorithm *UL-INVCAUCHY* combined with *BKP-Cauchy* pivoting.

ALG 1.3 was included in order to check whether symmetric pivoting may cause less accurate results than standard partial pivoting. **ALG 1.4** was included in order to check whether the algorithm *UL-INVCAUCHY* may produce less accurate results than *LU-CAUCHY*. Representative results are given in Table 2.

size	cond		ALG 1.1	ALG 1.2	ALG 1.3	ALG 1.4
80	1.8887e+03	res x	4.1286e-15	7.2777e-14	3.4174e-14	9.5423e-15
		res A	7.6986e-15	1.9160e-15	9.6248e-14	
		res C	3.9243e-16	1.9160e-15	1.3053e-15	
	3.8526e+04	res x	3.1655e-15	9.3151e-14	9.5612e-15	5.5622e-14
		res A	6.9822e-15	5.0378e-15	7.1205e-14	
		res C	5.3745e-16	5.0378e-15	6.4230e-15	
	1.6275e+07	res x	2.7303e-15	1.1814e-13	1.0432e-14	2.3952e-13
		res A	6.9938e-15	1.3900e-14	8.3806e-14	
		res C	4.7229e-16	1.3900e-14	1.4163e-14	
	2.6182e+15	res x	3.2986e-15	1.2818e-13	7.8452e-15	3.3623e-13
		res A	3.9141e-15	9.2986e-14	7.1720e-14	
		res C	7.8609e-16	1.6000e-14	7.0866e-15	
120	2.9335e+03	res x	7.3268e-15	2.8937e-13	5.1263e-14	3.1281e-13
		res A	1.8614e-14	5.5601e-13	1.6654e-13	
		res C	2.0574e-15	4.0819e-14	1.2077e-14	
	2.9691e+06	res x	6.9222e-15	2.9267e-13	5.1182e-14	3.8811e-13
		res A	1.8270e-14	5.4112e-13	1.7444e-13	
		res C	1.4279e-15	4.0414e-14	1.4117e-14	
	2.7449e+10	res x	7.1947e-15	2.4564e-13	5.1537e-14	2.1526e-13
		res A	1.7800e-14	5.4142e-13	1.6047e-13	
		res C	1.3676e-15	4.2591e-14	1.3015e-14	
	2.0951e+15	res x	5.5690e-15	3.2705e-13	5.4942e-14	4.8662e-13
		res A	1.7484e-14	5.1699e-13	1.7495e-13	
		res C	2.0453e-15	3.9838e-14	1.4298e-14	
150	3.6807e+03	res x	1.1590e-14	2.4397e-13	2.8876e-14	2.1523e-13
		res A	2.8423e-14	5.5647e-13	3.0573e-13	
		res C	1.0333e-15	3.8046e-15	2.8126e-14	
	3.5337e+06	res x	1.0409e-14	2.3590e-13	2.2918e-14	5.7461e-13
		res A	2.6824e-14	3.5981e-13	2.6781e-13	
		res C	8.5036e-16	4.7821e-15	1.4756e-14	
	3.3233e+10	res x	9.7645e-15	2.3049e-13	2.3672e-14	4.7922e-13
		res A	2.6093e-14	3.9866e-13	2.8928e-13	
		res C	1.2998e-15	1.4901e-14	9.4463e-15	
	2.8970e+15	res x	8.1843e-15	2.0285e-13	1.4991e-14	6.1183e-13
		res A	2.6317e-14	3.6961e-13	2.4262e-13	
		res T	2.9684e-15	3.9237e-15	3.0188e-14	

Table 2. Complex Hermitian Toeplitz-plus-Hankel A.

The table suggests that accuracy of the symmetric pivoting on the Cauchy matrix C is comparable to that of unsymmetric pivoting. The accuracy of the decomposition of A is slightly lower that that of C. This is caused by the transformation of A into C, and possibly could be improved by a more careful implementation of

FFT-like transformations. The residual error of the solution is comparable to that one of the decomposition of A.

The cost of the three methods (excluding the cost of the transformations) can be derived from Table 1 with $r = 4$. These costs (measured by the number of real multiplications) are summarized in Table 3.

	multiplications max	multiplications min
ALG 1.1	$28n^2$	$20n^2$
ALG 1.2	$28n^2$	$20n^2$
ALG 1.3	$36n^2$	$36n^2$
ALG 1.4	$52n^2$	$52n^2$

Table 3. Cost of solving $Ax = b$.

4.2. **Real symmetric Toeplitz.** Solving systems of equations with a real Toeplitz coefficient matrix we compared the following algorithms.

ALG 2.1: Use `RTfftSC` [5] to transform the matrix into a real symmetric Cauchy matrix. Then apply algorithm *LU-CAUCHY* in combination with algorithm *BKP-Cauchy* for pivoting.

ALG 2.2: Use `HTfftSC` for the transformation into a real symmetric Cauchy matrix. Then proceed as in **ALG 2.1**. *BKP-Cauchy* as in **ALG 1.1**.

ALG 2.3: Apply the transformation `HTfftHC` to transform the matrix into a complex hermitian matrix. Then proceed as in **ALG 2.1**.

ALG 2.4: Apply the transformation of the nonsymmetric standard choice in Section 2.1 of Part I.(This is the type-II algorithm proposed in [10] where a general Toeplitz matrix is transformed into a complex matrix with the Cauchy rank 2). Then proceed as in **ALG 2.1**.

ALG 2.5: Use `RTsineSC` to transform the matrix into a direct sum of two real symmetric Cauchy matrices of about half the size. Then proceed as in **ALG 2.1** for the two matrices.

Results of numerical tests are given in Table 4.

[5] See Subsection 3.4

size	cond		**ALG 2.1**	**ALG 2.2**	**ALG 2.3**	**ALG 2.4**	**ALG 2.5**
80	4.99e+02	res x	4.74e-15	2.61e-16	2.19e-16	8.96e-15	1.18e-14
		res T	1.03e-14	5.25e-16	5.87e-16	1.12e-15	3.84e-14
		res C	2.90e-15	1.21e-16	5.82e-16	2.90e-16	2.58e-17
	3.02e+05	res x	6.06e-15	1.73e-15	1.76e-15	1.05e-14	9.72e-16
		res T	7.57e-14	4.28e-15	4.34e-15	1.12e-14	1.41e-14
		res C	6.11e-14	5.26e-17	1.52e-15	5.14e-15	4.45e-17
	2.68e+09	res x	4.31e-15	1.68e-15	1.70e-15	1.05e-14	7.57e-16
		res T	8.34e-15	2.80e-15	2.88e-15	1.13e-14	1.42e-14
		res C	6.53e-16	5.64e-17	7.13e-16	3.99e-16	5.35e-18
	1.17e+16	res x	4.54e-15	1.64e-15	1.60e-15	5.05e-15	8.76e-16
		res T	8.19e-15	3.25e-15	3.54e-15	1.14e-14	1.42e-14
		res C	6.90e-16	3.86e-17	5.76e-16	6.16e-16	4.15e-17
120	3.58e+03	res x	2.16e-09	2.19e-15	2.27e-15	1.62e-15	3.68e-14
		res T	6.41e-09	1.04e-14	5.27e-15	5.85e-15	1.43e-13
		res C	5.82e-10	4.85e-17	3.09e-15	2.02e-16	9.54e-18
	4.24e+05	res x	8.21e-13	5.65e-16	6.09e-16	3.29e-15	1.78e-14
		res T	2.92e-12	9.51e-16	9.30e-16	5.81e-15	3.37e-14
		res C	3.64e-13	1.08e-16	1.40e-16	2.11e-15	1.68e-17
	3.33e+09	res x	4.06e-13	5.38e-16	7.74e-16	3.36e-15	1.79e-14
		res T	1.28e-12	9.96e-16	9.93e-16	5.83e-15	3.33e-14
		res C	4.00e-12	4.05e-17	1.42e-16	1.80e-15	9.76e-18
	1.73e+16	res x	8.88e-16	4.97e-16	5.75e-16	1.89e-15	1.77e-14
		res T	3.67e-15	1.04e-15	1.02e-15	5.81e-15	3.32e-14
		res C	4.76e-15	6.24e-17	2.35e-16	1.50e-15	7.68e-18
150	1.04e+03	res x	1.21e-09	2.34e-15	2.33e-15	3.61e-14	5.31e-15
		res T	1.36e-08	9.02e-15	7.74e-15	2.33e-14	3.49e-14
		res C	6.33e-09	7.89e-17	1.31e-15	3.15e-16	4.74e-18
	5.80e+05	res x	2.47e-14	7.21e-15	7.28e-15	1.96e-14	7.06e-14
		res T	1.24e-13	1.66e-14	1.83e-14	2.47e-14	1.41e-13
		res C	1.12e-12	6.34e-17	1.68e-15	8.44e-16	2.46e-17
	4.37e+09	res x	5.12e-13	7.29e-15	7.31e-15	2.07e-14	6.96e-14
		res T	1.13e-11	1.13e-14	1.09e-14	2.42e-14	1.45e-13
		res C	2.89e-11	7.91e-17	3.43e-15	1.23e-15	2.93e-17
	2.63e+16	res x	2.28e-15	5.17e-15	5.27e-15	6.23e-15	1.30e-14
		res T	3.37e-14	1.34e-14	1.36e-14	2.55e-14	1.40e-13
		res C	8.44e-15	1.08e-16	4.83e-15	1.35e-15	3.60e-18

Table 4. *Real Symmetric Toeplitz T*

The table suggests that accuracy of the symmetric pivoting on the Cauchy matrix C in all but one case is comparable to that of unsymmetric pivoting. The single exception is the method **ALG 2.1** which for large n and well-conditioned matrices looses about half of the number of accurate digits. This loss of accuracy is most probably connected with the linear fractional function which maps a unit circle onto a unit interval. However, the related method **ALG 2.2** produces an accurate decomposition of C. Reasons for loss of accuracy in **ALG 2.1** will have to be investigated further.

The accuracy of the decomposition of T as measured by $resT$ is slightly lower that that of the decomposition of C. This is caused by the transformation of T into C, and possibly could be improved by a more careful implementation of FFT-like transformations. The residual error of the solution is comparable to that of the decomposition of T.

The cost of the five methods (excluding the cost of the transformations) can be derived from Table 1 where $r = 2$ and are summarized in Table 5. Note that except for **ALG 2.5** all other transformation transform a real right hand side vector to a complex vector. Thus **ALG 2.5** is the least costly among all five algorithms discussed in this paper.

	real multiplications	
	max	min
ALG 2.1	$5n^2$	$4n^2$
ALG 2.1	$5n^2$	$4n^2$
ALG 2.3	$16n^2$	$12n^2$
ALG 2.4	$20n^2$	$20n^2$
ALG 2.5	$2n^2$	$\frac{3}{2}n^2$

Table 5. Cost of solving $Tx = b$.

References

[1] A.W. Bojanczyk, R.P. Brent, F.R. de Hoog, D.R. Sweet, On the stability of the Bareiss and related Toeplitz factorization algorithms, *SIAM J.Matrix Anal.Appl.* 1: 40-57 (1995).

[2] T. Boros, A. Sayed, T. Kailath, Structured matrices and unconstrained rational interpolation problems. *Linear Algebra Appl.*, to appear.

[3] J. R. Bunch, Analysis of the diagonal pivoting method, *SIAM J. Numer.Anal.* 8: 656-680 (1971).

[4] J. K. Bunch, L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, *Math.Comp.* 31: 162-179 (1977).

[5] J. Bunch, L. Kaufman, B. Parlett, Decomposition of a symmetric matrix, Numerische Mathematik, vol 27, 95-109 (1976).

[6] I. Gohberg, I. Koltracht, P. Lancaster, Efficient solution of linear systems of equations with recursive structure. *Linear Algebra Appl.* 80: 81-113 (1986).

[7] I. Gohberg, T. Kailath, I. Koltracht, P. Lancaster, Linear complexity parallel algorithms for linear systems of equations with recursive structure, *Linear Algebra Appl.* 88/99: 271-316 (1987).

[8] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian elimination with partial pivoting for matrices with displacement structure, *Math.of Comp.* 64: 1557-1576 (1995).

[9] I. Gohberg and V. Olshevsky, Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems, *Integral Equ. and Operator Theory* 20,1: 44-83 (1994).

[10] G. Heinig, Inversion of generalized Cauchy matrices and other classes of structured matrices. In *The IMA Volumes in Mathematics and Its Applications*, Springer-Verlag, volume 69, Lectures presented at the IMA Workshop on Linear Algebra in Signal Processing, Minneapolis April 1992, A. Bojanczyk and G. Cybenko, Editors, pp. 63-82.

[11] G. Heinig, Inversion of Toeplitz-like matrices via generalized Cauchy matrices and rational interpolation. In: *Systems and Network: Mathematical Theory and Applications*, Akademie Verlag 1994, vol.*2*, 707-711.

[12] G. Heinig, Transformation approaches for fast and stable solution of Toeplitz systems and polynomial equations. *Proceedings of the workshop "Recent Advances in Applied Mathematics"*, Kuwait University, May 1996.

[13] G. Heinig, A. Bojanczyk, Transformation techniques for Toeplitz and Toeplitz-plus-Hankel matrices, I. Transformations, *Linear Algebra Appl.* (submitted)

[14] G. Heinig, K. Rost, *Algebraic methods for Toeplitz-like matrices and operators.* Birkhäuser Verlag , Basel-Boston-Stuttgart 1984.

[15] T. Kailath, V. Olshevsky, Symmetric and Bunch-Kaufman pivoting for partially structured Cauchy-like matrices with applications to Toeplitz-like linear equations, *Linear Algebra Appl.*. (submitted)

[16] T. Kailath, A. Sayed, Fast algorithms for generalized displacement structures, In: *Recent Advances in Mathematical Theory of Systems, Control, Network, and Signal Processing II*, Proceedings of the MTNS-91 (H. Kimura, S. Kodama, Eds), Mita Press, Japan, 27-32 (1992).

[17] T. Kailath, A. Sayed, Displacement structure: Theory and applications, *SIAM Review* 37,3:297-386 (1995).

[18] Ming Gu, Stable and efficient algorithms for structured systems of linear equations, *SIAM J.on Matrix Analysis* (submitted)

[19] V. Olshevsky, *personal communication.*

[20] D. Sweet, R. Brent, Error analysis of a fast partial pivoting method for structured matrices. In T. Luk,ed., *Advanced Signal Processing Algorithms, Proc. of SPIE*, vol.2363 (1995), 266-280.

[21] M. Tasche, Fast algorithms for discrete Chebyshev-Vandermonde transforms and applications, *Numerical Algorithms* 5: 453-464 (1993).

[22] C. Van Loan, *Computational Framework for the Fast Fourier Transform*, SIAM, Philadelphia 1992.

[23] M. Vetterli, H. J. Nussbaumer, Simple FFT and DCT algorithms with reduced number of operations, *Signal Processing* 6: 267–278 (1984).

Electrical Engineering Department, Cornell University, Ithaca, N.Y. 14853, USA
E-mail address: adamb@toeplitz.ee.cornell.edu

Department of Mathematics, Kuwait University, POB 5969,, Safat 13060, Kuwait
E-mail address: georg@math-1.sci.kuniv.edu