NORTH-HOLLAND

High Performance Algorithms for Toeplitz and Block Toeplitz Matrices

K. A. Gallivan and S. Thirumalai

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois

and

P. Van Dooren and V. Vermaut

Université Catholique de Louvain
Louvain, Belgium

Submitted by André Ran

ABSTRACT

In this paper, we present several high performance variants of the classical Schur algorithm to factor various Toeplitz matrices. For positive definite block Toeplitz matrices, we show how hyperbolic Householder transformations may be blocked to yield a block Schur algorithm. This algorithm uses BLAS3 primitives and makes efficient use of a memory hierarchy. We present three algorithms for indefinite Toeplitz matrices. Two of these are based on look-ahead strategies and produce an exact factorization of the Toeplitz matrix. The third produces an inexact factorization via perturbations of singular principal minors. We also present an analysis of the numerical behavior of the third algorithm and derive a bound for the number of iterations to improve the accuracy of the solution. For rank-deficient Toeplitz least-squares problems, we present a variant of the generalized Schur algorithm that avoids breakdown due to an exact rank-deficiency. In the presence of a near rank-deficiency, an approximate rank factorization of the Toeplitz matrix is produced. Finally, we suggest an algorithm to solve the normal equations resulting from a real Toeplitz least-squares problem based on transforming to Cauchy-like matrices. This algorithm exploits both realness and symmetry in the normal equations.

1. INTRODUCTION

Algorithms to solve Toeplitz matrices can be broadly classified into two categories, namely, the *Levinson* type and the *Schur* type. The Levinson type algorithms produce factorizations of the inverse of the Toeplitz matrix

LINEAR ALGEBRA AND ITS APPLICATIONS 241–243:343–388 (1996)

655 Avenue of the Americas, New York, NY 10010
0024-3795/96/$15.00
SSDI 0024-3795(95)00649-4

such as $T^{-1} = LDL^T$ and $T^{-1} = QR$, while the Schur type algorithms produce factorizations of the Toeplitz matrix itself such as $T = LDL^T$ and $T = QR$. In addition, the two approaches differ in the kinds of computational primitives used during the factorization.

In [30] Schur derived a fast recursive algorithm to check if a power series is analytic and bounded in the unit disc. Interestingly, the recursions proposed in this algorithm provide a fast factorization of matrices with *displacement rank* 2. It is well known that Toeplitz matrices have a displacement rank of 2 [23]. More generally block Toeplitz matrices with a block size of m have a displacement rank of $2m$. In this paper we discuss several high performance variants of the classical Schur algorithms to factor symmetric block Toeplitz matrices. Specifically we discuss routines to factor symmetric positive definite, positive semidefinite, and indefinite matrices. Algorithms to obtain the QR factorization of exactly and nearly rank deficient Toeplitz matrices are also discussed.

In this paper the classical Schur algorithm for obtaining the Cholesky factorization of symmetric positive definite block Toeplitz matrices [8, 9] is generalized to the block Toeplitz matrix case using a block generalization of the hyperbolic Householder reflectors. The block generalization of the Schur algorithm and various blocking schemes differing in the amount of storage and computational primitives used are described in Section 2. Blocking the hyperbolic Householder transformations allows us to apply these transformations using BLAS 3 primitives rather than the BLAS 2 primitives that are required for plain hyperbolic Householder transformations. On machines with a memory hierarchy this provides us with a faster algorithm.

For symmetric indefinite block Toeplitz matrices the Schur algorithm breaks down if the matrix has singular principal minors. A scheme to modify the block Schur algorithm by perturbing the generators and obtaining an approximate factorization of the matrix is described in Section 3. The approximate solution is then improved through iterative refinement. The numerical behavior of this method to circumvent the singularities is studied. If an exact factorization of the indefinite block Toeplitz matrix is desired, then one would have to look ahead over the singular or near singular principal minors. Look-ahead algorithms based on the Levinson algorithm have appeared in the literature [4, 12] but suffer from the same reduced parallelism relative to the Schur algorithm mentioned above and are limited to point Toeplitz matrices. Look-ahead Schur algorithms based on orthogonal polynomials exist [18] but are limited to point Toeplitz matrices. In Section 3 we present two look-ahead Schur algorithms for point and block Toeplitz matrices and compare the two from a computational viewpoint.

The classical Schur algorithm can be generalized to obtain the QR factorization of block Toeplitz matrices [5]. If the Toeplitz matrix is rank

deficient, then we present a modification of the generalized Schur algorithm in Section 4 to obtain the QR factorization by pruning the generators of the Toeplitz matrix. If the matrix is nearly rank deficient, then this method produces a low-rank approximation of the Toeplitz matrix.

Finally we discuss algorithms to factor Toeplitz matrices by converting them to Cauchy type matrices. Toeplitz matrices can be converted using the discrete Fourier transform into Cauchy type matrices that allow pivoting during the factorization [15, 21]. These algorithms also have the same complexity, $O(n^2)$, as the Schur algorithm. The problem with this method is that any real-valued Toeplitz matrix is converted to a complex Cauchy type matrix and the entire factorization algorithm proceeds in complex arithmetic. This is computationally expensive. Similarly, any symmetry in the Toeplitz matrix is ignored in this algorithm. In Section 5 we present a modification to this algorithm that allows us to work in real arithmetic and also exploit the symmetric structure of the matrix. This yields a rank revealing algorithm for the factorization of a semidefinite block Toeplitz matrix that is computationally less expensive than the algorithm presented in [15, 21].

2. SYMMETRIC POSITIVE DEFINITE BLOCK TOEPLITZ MATRICES

In this section we present a block generalization of the classical Schur algorithm [8, 9] using block hyperbolic Householder reflectors. Block hyperbolic Householder transformations can be applied at the BLAS 3 rate rather than plain householder transformations, which are applied at the BLAS 2 rate. On machines with a memory hierarchy this provides us with a significant improvement in performance. Various blocking strategies that differ in the computational primitives required during the construction are presented. The cost of applying these transformations is also discussed.

2.1. The Classical Schur Algorithm

Let T be an $mp \times mp$ symmetric positive definite block Toeplitz matrix with a block size of $m \times m$ whose first block row is given by $[\widehat{T}_1\ \widehat{T}_2\ \cdots\ \widehat{T}_{p-1}\ \widehat{T}_p]$. Let Z be a block right shift matrix. The Schur algorithm is based on the fact that the displacement of a block Toeplitz matrix T, defined as $T - Z^T T Z$, has a rank of at most $2m$ [23]. The derivation of the Schur algorithm to compute the Cholesky factorization of a symmetric positive definite block Toeplitz matrix is outlined below.

Since $\widehat{T}_1$ is a symmetric positive definite matrix, we can find its Cholesky factorization $\widehat{T}_1 = L_1 L_1^T$, where L_1 is an $m \times m$ lower triangular matrix.

Let $T_j = L_1^{-1}\widehat{T}_j$. It is easy to see that $T_1 = L_1^T$. We now define two matrices $G_1(T)$ and $G_2(T)$ as follows [6, 23]:

$$G_1(T) = \begin{pmatrix} T_1 & T_2 & T_3 & \dots & T_p \\ 0 & T_1 & T_2 & \dots & T_{p-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & T_2 \\ 0 & 0 & \cdots & 0 & T_1 \end{pmatrix}$$

$$G_2(T) = \begin{pmatrix} 0 & T_2 & T_3 & \dots & T_p \\ 0 & 0 & T_2 & \dots & T_{p-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & & \ddots & \ddots & T_2 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{1}$$

from which it follows that

$$T = \begin{bmatrix} G_1^T(T) & G_2^T(T) \end{bmatrix} \begin{bmatrix} I_{mp} & 0 \\ 0 & -I_{mp} \end{bmatrix} \begin{bmatrix} G_1(T) \\ G_2(T) \end{bmatrix} = G^T W_{mp} G, \tag{2}$$

where

$$G = \begin{bmatrix} G_1(T) \\ G_2(T) \end{bmatrix} \quad \text{and} \quad W_{mp} = \begin{bmatrix} I_{mp} & 0 \\ 0 & -I_{mp} \end{bmatrix}. \tag{3}$$

If we can obtain a transformation matrix U that satisfies the property $U^T W_{mp} U = W_{mp}$ such that $UG = R$, where R is upper triangular, then we have

$$\begin{aligned} T &= G^T W_{mp} G = G^T U^T W_{mp} U G \\ &= \begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} I_{mp} & 0 \\ 0 & -I_{mp} \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \\ &= R^T R, \end{aligned} \tag{4}$$

which gives us the Cholesky factorization of T [8]. The transformation matrix U, which satisfies the property $U^T W_{mp} U = W_{mp}$, is called a hyperbolic Householder transformation [26]. The basic properties of hyperbolic Householder reflectors are discussed in Section 2.2. Since the matrix G comprises two upper triangular block Toeplitz matrix, we show in Section 2.4 that considerable computational savings can be obtained by working with

a generator matrix defined using the first block rows of G_1 and G_2 as

$$\text{Gen} = \begin{bmatrix} T_1 & T_2 & \cdots & T_{p-1} & T_p \\ 0 & T_2 & \cdots & T_{p-1} & T_p \end{bmatrix}. \tag{5}$$

It can also be seen that the above generator matrix Gen is obtained by a factorization of the displacement of the block Toeplitz matrix into

$$T - Z^T T Z = \text{Gen}^T \begin{bmatrix} I_m & 0 \\ 0 & -I_m \end{bmatrix} \text{Gen} \tag{6}$$

Note that when $\widehat{T}_1$ is not positive definite we can consider the more general decomposition $T_1 = L_1 \, \Sigma \, L_1^T$, where Σ is some signature matrix with ± 1 on diagonal. This will exist provided $\widehat{T}_1$ has nonsingular leading principal submatrices. The blocks T_j are obtained by $T_j = (L_1 \Sigma)^{-1} \widehat{T}_j$ and the W_{mp} matrix becomes

$$W_{mp} = \begin{bmatrix} I_p \otimes \Sigma & 0 \\ 0 & -I_p \otimes \Sigma \end{bmatrix}. \tag{7}$$

We then again use hyperbolic Householder transformations (now with respect to the new signature matrix W_{mp}) to reduce G to an upper triangular matrix. A detailed discussion of the Schur algorithm for indefinite Toeplitz matrices is presented in Section 3.

2.2. Hyperbolic Householder Transformations

In [8], Cybenko and Berry use hyperbolic Householder transformations [26] to reduce the generator matrix G of a scalar Toeplitz matrix to an upper triangular matrix. We extend their idea to block hyperbolic Householder transformations (required in the block Schur algorithm), using representations very similar to those proposed in [2] and [29].

Let W be a diagonal matrix whose entries are either $+1$ or -1. It is easy to verify that the matrix W satisfies the equalities

$$W^2 = I \qquad \text{and} \qquad W^T = W. \tag{8}$$

Any matrix U that satisfies the equation $U^T W U = W$ is called a W-unitary matrix. Let x be a column vector such that $x^T W x \neq 0$. A hyperbolic Householder matrix is defined as

$$U_x = W - \frac{2 x x^T}{x^T W x}. \tag{9}$$

One easily checks [8, 27] that U_x is W-unitary, i.e., $U_x^T W U_x = W$. These transformations can be used to map one vector to another as long as they have the same hyperbolic norm, i.e., if $a^T W a = b^T W b$. In our algorithm, we

reduce the generator matrix to an upper triangular matrix by successively zeroing elements below the diagonal of columns of the G matrix in (3). Given a column vector u, we would like to find a hyperbolic Householder matrix U_x such that

$$U_x u = -\sigma e_j, \tag{10}$$

where e_j is a column vector whose jth element is 1 and other elements are 0 and σ is a constant. We assume here that $e_j^T W e_j = 1$; i.e., the jth component corresponds to a $+1$ in W. Also the vectors u we consider will have positive hyperbolic norm when the matrix T we decompose is positive definite. Choosing

$$\sigma = \frac{u_j}{|u_j|}\sqrt{u^T W u} \tag{11}$$

then u and σe_j have the same hyperbolic norm. If we take $x = Wu + \sigma e_j$, it can be shown that U_x is a hyperbolic Householder transformation that maps u to $-\sigma e_j$.

2.3. Block Hyperbolic Householder Representations

If we have to perform a sequence of hyperbolic Householder transformations we could block these transformations together and then apply this block to the appropriate matrices. This allows us to use level 3 BLAS primitives rather than level 2 BLAS operations if we applied the transformations sequentially. Storage efficient ways to block regular Householder transformations are derived in [2] and [29]. We extend these methods to hyperbolic Householder transforms.

Suppose $U^{(r)} = U_r U_{r-1} \cdots U_2 U_1$ is a product of r $n \times n$ hyperbolic Householder matrices. The matrix U can be written in two forms corresponding to the VY form and the YTY^T form derived in [2] and [29]. The two forms of the VY representation differ in the types of primitives they use.

LEMMA 1. *Suppose $U^{(k)} = W^k + V_k Y_k^T$ is a product of k $n \times n$ hyperbolic Householder matrices, where V_k and Y_k are $n \times k$ matrices. If*

$$U_{k+1} = W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^T W x_{k+1}} \qquad \text{and} \qquad z_{k+1} = \frac{-2x_{k+1}^T U^{(k)}}{x_{k+1}^T W x_{k+1}},$$

then

$$U^{(k+1)} = U_{k+1}U^{(k)} = W^{k+1} + V_{k+1}Y_{k+1}^T,$$

where $V_{k+1} = [WV_k \quad x_{k+1}]$ and $Y_{k+1} = [Y_k \quad z_{k+1}^T]$. We call this the first VY form.

Proof. If $r = 1$ then, $U^{(1)} = U_1 = W - 2x_1x_1^T/(x_1^TWx_1)$ and we assign $V_1 = x_1$ and $Y_1 = -2x_1/x_1^TWx_1$ in order to have the desired form

$$\begin{aligned} U_{k+1}U^{(k)} &= \left(W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^TWx_{k+1}}\right)(W^k + V_kY_k^T) \\ &= W^{k+1} + WV_kY_k^T - \frac{2x_{k+1}x_{k+1}^TU^{(k)}}{x_{k+1}^TWx_{k+1}} \\ &= W^{k+1} + WV_kY_k^T + x_{k+1}z_{k+1} \\ &= W^{k+1} + [WV_k \quad x_{k+1}]\begin{bmatrix} Y_k^T \\ z_{k+1} \end{bmatrix} \\ &= W^{k+1} + V_{k+1}Y_{k+1}^T. \end{aligned}$$

■

LEMMA 2. *Suppose $U^{(k)} = W^k + V_kY_k^T$ is a product of k $n \times n$ hyperbolic Householder matrices, where V_k and Y_k are $n \times k$ matrices. If*

$$U_{k+1} = W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^TWx_{k+1}} \qquad \text{and} \qquad z_{k+1} = \frac{-2x_{k+1}^TW^k}{x_{k+1}^TWx_{k+1}},$$

then

$$U^{(k+1)} = U_{k+1}U^{(k)} = W^{k+1} + V_{k+1}Y_{k+1}^T,$$

where $V_{k+1} = [U_{k+1}V_k \qquad x_{k+1}]$ and $Y_{k+1} = [Y_k \quad z_{k+1}^T]$. We call this the second VY form.

Proof. If $r = 1$ then, $U^{(1)} = U_1 = W - 2x_1x_1^T/(x_1^TWx_1)$ and we assign $V_1 = x_1$ and $Y_1 = -2x_1/x_1^TWx_1$ in order to have the desired form

$$\begin{aligned} U_{k+1}U^{(k)} &= \left(W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^TWx_{k+1}}\right)(W^k + V_kY_k^T) \\ &= W^{k+1} + U_{k+1}V_kY_k^T - \frac{2x_{k+1}x_{k+1}^TW^k}{x_{k+1}^TWx_{k+1}} \\ &= W^{k+1} + U_{k+1}V_kY_k^T + x_{k+1}z_{k+1} \\ &= W^{k+1} + [U_{k+1}V_k \quad x_{k+1}]\begin{bmatrix} Y_k^T \\ z_{k+1} \end{bmatrix} \\ &= W^{k+1} + V_{k+1}Y_{k+1}^T. \end{aligned}$$

■

LEMMA 3. *Suppose* $U^{(k)} = W^k + Y_k T_k Y_k^T W^{k-1}$ *is a product of* k $n \times n$ *hyperbolic Householder matrices, where* Y_k *is a* $n \times k$ *matrix and* T_k *is a* $k \times k$ *matrix. If*

$$U_{k+1} = W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^T W x_{k+1}}, \qquad a_{k+1} = -\frac{2}{x_{k+1}^T W x_{k+1}}(x_{k+1}^T Y_k T_k)$$

and

$$b_{k+1} = -\frac{2}{x_{k+1}^T W x_{k+1}},$$

then

$$U^{(k+1)} = U_{k+1}U^{(k)} = W^{k+1} + Y_{k+1}T_{k+1}Y_{k+1}^T W^k,$$

where

$$Y_{k+1} = [WY_k \quad x_{k+1}] \qquad \textit{and} \qquad T_{k+1} = \begin{bmatrix} T_k & 0 \\ a_{k+1} & b_{k+1} \end{bmatrix}.$$

Proof. For $k = 1$ it can be seen that $U_1 = W + Y_1 T_1 Y_1^T$, where $Y_1 = x_1$ and $T_1 = -2/x_1^T W x_1$.

$$\begin{aligned}
U^{(k+1)} &= \left(W - \frac{2x_{k+1}x_{k+1}^T}{x_{k+1}^T W x_{k+1}}\right)(W^k + Y_k T_k Y_k^T W^{k-1}) \\
&= W^{k+1} + x_{k+1}\left(-\frac{2}{x_{k+1}^T W x_{k+1}}\right)(x_{k+1}^T W^k) + (WY_k)T_k(Y_k^T W^{k-1}) \\
&\quad + x_{k+1}\left(-\frac{2}{x_{k+1}^T W x_{k+1}} x_{k+1}^T Y_k T_k\right)(Y_k^T W^{k-1}) \\
&= W^{k+1} + x_{k+1}b_{k+1}(x_{k+1}^T W^k) + (WY_k)T_k(Y_k^T W^{k-1}) \\
&\quad + x_{k+1}a_{k+1}(Y_k^T W^{k-1}) \\
&= W^{k+1} + [WY_k \quad x_{k+1}]\begin{bmatrix} T_k & 0 \\ a_{k+1} & b_{k+1} \end{bmatrix}\begin{bmatrix} Y_k^T W^{k-1} \\ x_{k+1}^T W_k \end{bmatrix} \\
&= W^{k+1} + Y_{k+1}T_{k+1}Y_{k+1}^T W^k.
\end{aligned}$$

■

The three blocking schemes discussed above differ in the computational primitives employed (*dotproducts or saxpys*) and the amount of storage. A detailed performance analysis of the three blocking schemes is presented in [13].

2.4. The Factorization Algorithm

The following algorithm is used to reduce matrix G (3) described in Section 2.1 to an upper triangular matrix. This algorithm is essentially the same as the one described in [8] except that we are dealing with blocks instead of elements. We describe the algorithm using an example as follows. Let $T = G^T W_{mp} G$, where G and W_{mp} are

$$G = \left(\begin{array}{c|cccccc} T_1 & T_2 & T_3 & T_4 & \cdots & T_p \\ 0 & T_1 & T_2 & T_3 & \ddots & T_{p-1} \\ 0 & 0 & T_1 & T_2 & \ddots & T_{p-2} \\ 0 & 0 & 0 & T_1 & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \\ \hline 0 & \boxed{T_2} & T_3 & T_4 & \cdots & T_p \\ 0 & 0 & \boxed{T_2} & T_3 & \ddots & T_{p-1} \\ 0 & 0 & 0 & \boxed{T_2} & \ddots & T_{p-2} \\ 0 & 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \end{array}\right)$$

$$W_{mp} = \begin{pmatrix} I_{mp} & 0 \\ 0 & -I_{mp} \end{pmatrix}. \tag{12}$$

The goal of this algorithm is to reduce G into an upper triangular matrix using block hyperbolic Householder matrices. Since the first column of the generator is already in the right form we only use the generator matrix from the second row down. The first row of the upper submatrix of the generator is the first block row of the triangular factor of the Toeplitz matrix. The first step in this algorithm therefore involves eliminating the first diagonal in the lower half of the generator matrix (the boxed T_2 blocks in (12)). If this is done while maintaining the Toeplitz structure of the remaining portion of the matrix (the submatrix from the third row downward), we can repeat the process on the smaller generator till we triangularize G.

Consider the matrix formed by stacking the second block row of the upper submatrix and the first block row of the lower submatrix as

$$G' = \begin{pmatrix} 0 & T_1 & T_2 & T_3 & \cdots & T_{p-1} \\ 0 & T_2 & T_3 & T_4 & \cdots & T_p \end{pmatrix}. \tag{13}$$

Let U_1 be a block hyperbolic Householder transformation that eliminates T_2 using T_1. Applying this to G' we get

$$U_1 G' = \begin{pmatrix} 0 & \widetilde{T}_1 & \widetilde{T}_2 & \widetilde{T}_3 & \cdots & \widetilde{T}_{p-1} \\ 0 & 0 & \widehat{T}_3 & \widehat{T}_4 & \cdots & \widehat{T}_p \end{pmatrix}. \tag{14}$$

The matrix formed by stacking the third row of the upper submatrix and the second row of the lower submatrix is just a shifted version of G'. Similarly all matrices constructed by stacking the corresponding rows in the two halves of the generator matrix are shifted versions of the G' matrix in (13). Hence, all the work that was needed to zero out the diagonal row of T_2 in the lower submatrix was done in the first step. At this stage, the generator matrix G has a Toeplitz submatrix in its upper half (from the third row onward) and another Toeplitz submatrix in its lower half as

$$G = \left(\begin{array}{cc|ccccc} T_1 & T_2 & T_3 & T_4 & \cdots & T_p \\ 0 & \widetilde{T}_1 & \widetilde{T}_2 & \widetilde{T}_3 & \ddots & \widetilde{T}_{p-1} \\ \hline 0 & 0 & \widetilde{T}_1 & \widetilde{T}_2 & \ddots & \widetilde{T}_{p-2} \\ 0 & 0 & 0 & \widetilde{T}_1 & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \\ \hline 0 & 0 & \boxed{\widehat{T}_3} & \widehat{T}_4 & \cdots & \widehat{T}_p \\ 0 & 0 & 0 & \boxed{\widehat{T}_3} & \ddots & \widehat{T}_{p-1} \\ 0 & 0 & 0 & 0 & \ddots & \widehat{T}_{p-2} \\ 0 & 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & \vdots \end{array}\right) \tag{15}$$

The second row of the upper submatrix of G is the second block row of the triangular factor of the Toeplitz matrix. The process is then repeated on the two lower right submatrices of the generator in (15). After $p-2$ steps the generator is completely triangularized.

Note that in addition to being able to work with only two block rows, we can work with the same two block rows because the reduced generator in the next step has the same lower block row but the upper block row is shifted by one block to the right. Before this shift is made the upper block row must be stored in the right place in the triangular factor of the original Toeplitz matrix. At the first step of the algorithm, this reduced

matrix, which we refer to as the generator matrix, is

$$\text{Gen} = \begin{pmatrix} T_1 & T_2 & T_3 & \cdots & T_p \\ 0 & T_2 & T_3 & \cdots & T_p \end{pmatrix}. \tag{16}$$

Also, we see that in the first step T_1 is upper triangular because by construction $T_1 = L_1^T$. The diagonal elements of T_1 are sequentially used to zero out all the elements in the corresponding column of the lower block (T_2). This implies that at each step of the algorithm the block hyperbolic Householder matrices are computed using vectors that have one nonzero element in their upper half and a non zero lower half. This means that the V, Y matrices in the first two forms and the Y matrix in the third form have more sparsity than usual. The sparsity patterns of the matrices V, Y and Y, T and their performance implications can be found in [13]. In this paper we provide a summary of the computational costs involved in blocking the hyperbolic Householder reflector.

The blocking scheme described in Lemma 1 requires two reduction primitives (matrix vector products) at each step. For a block Toeplitz matrix with block size m, if the m hyperbolic Householder reflectors at each step of the Schur algorithm are blocked, then the total flop count is $2.33m^3 + 3.75m^2 + 8m$. Also, applying the blocked reflector to a generator of size $2m \times mp$ requires $5m^3p + 3m^2p$ operations performed at the BLAS 3 (matrix multiplication) rate.

If the blocking scheme described in Lemma 2 is used, one matrix vector product and one rank 1 update are used at each step of the blocking process. The total flop count to block the reflectors is $2m^3 + 3m^2 + 8m$ and the cost of applying the blocked reflector to the rest of the generator is $5m^3p + 2m^2p$.

The blocking scheme described in Lemma 3 requires two reduction primitives like in Lemma 1 but the cost of blocking m reflectors is $1.33m^3 + 3.75m^2 + 8m$, which is less than the two schemes mentioned above. On the other hand, applying the blocked reflector in this form to the rest of the generator is the most expensive, requiring $5m^3p + 5m^2p$ flops.

From this discussion it can be seen that there are definite tradeoffs in implementing the three blocking schemes and implementation choices must be made following a detailed performance analysis taking into consideration the architecture of the machine at hand.

2.5. LDL^T Factorization of a s.p.d. Block Toeplitz Matrix

In this section we derive another form of the block hyperbolic Householder reflector that is used to obtain an LDL^T factorization of a symmetric positive definite block Toeplitz matrix as opposed to a Cholesky factorization. This blocking scheme can be used if the matrix is symmetric indefinite unless there is a breakdown. Modifications to the Schur algorithm in the

presence of breakdowns are discussed in Section 3.

Consider a symmetric positive definite block Toeplitz matrix T having blocks $\widehat{T}_i, i = 1, \ldots, p$ of dimension $m \times m$. The generator for such a Toeplitz matrix can be written as

$$G = \begin{pmatrix} I & T_2 & T_3 & \cdots & T_p \\ 0 & T_2 & T_3 & \cdots & T_p \end{pmatrix}, \tag{17}$$

where $T_i = \widehat{T}_1^{-1}\widehat{T}_i, i = 1, \ldots, p$. The generator matrix shown above gives us a factorization of the displacement of the Toeplitz matrix T

$$\begin{aligned} T - Z^T T Z &= G^T \begin{pmatrix} \widehat{T}_1 & 0 \\ 0 & -\widehat{T}_1 \end{pmatrix} G \\ &= G^T W G, \end{aligned} \tag{18}$$

where Z is the block right shift matrix of size $mp \times mp$. The first step of the Schur algorithm for such a generator is trivial. After the shift at the end of the first step, the generator for the second step is

$$G^{(2)} = \begin{pmatrix} I & T_2 & T_3 & \cdots & T_{p-1} \\ T_2 & T_3 & T_4 & \cdots & T_p \end{pmatrix}. \tag{19}$$

If we choose a block hyperbolic Householder reflector U such that $U^T \widehat{W} U = W$, where $\widehat{W}$ is also block diagonal, then the factorization obtained is of the form LDL^T, where D is block diagonal. If T_{sc} is the Schur complement of T w.r.t. the first leading $m \times m$ block and $\widehat{Z}$ is a block right shift matrix of size $m(p-1) \times m(p-1)$, then

$$\begin{aligned} T_{\mathrm{sc}} - \widehat{Z}^T T_{\mathrm{sc}} \widehat{Z} &= {G^{(2)}}^T \begin{pmatrix} \widehat{T}_1 & 0 \\ 0 & -\widehat{T}_1 \end{pmatrix} G^{(2)} \\ &= {G^{(2)}}^T U^T \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} U G^{(2)} \\ &= \left(\widehat{G}^{(2)}\right)^T \widehat{W} \widehat{G}^{(2)}, \end{aligned}$$

where

$$\widehat{G}^{(2)} = \begin{pmatrix} I & \widetilde{T}_2 & \widetilde{T}_3 & \cdots & \widetilde{T}_{p-1} \\ 0 & \widetilde{T}_3 & \widetilde{T}_4 & \cdots & \widetilde{T}_p \end{pmatrix} \quad \text{and} \quad \widehat{W} = \begin{pmatrix} \widehat{\Sigma}_1 & 0 \\ 0 & \widehat{\Sigma_2} \end{pmatrix}.$$

From the above equations we see that if $T = LDL^T$, then

$$\begin{aligned} L(m+1 : 2m, m+1 : mp) &= \begin{pmatrix} I & \widetilde{T}_2 & \widetilde{T}_3 & \cdots & \widetilde{T}_{p-1} \end{pmatrix} \\ D(m+1 : 2m, m+1 : 2m) &= \Sigma_1. \end{aligned} \tag{20}$$

From this discussion it is obvious that we need to construct a block hyperbolic Householder reflector U such that

$$U^T \begin{pmatrix} \widehat{\Sigma}_1 & 0 \\ 0 & \widehat{\Sigma}_2 \end{pmatrix} U = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \tag{21}$$

$$U \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{22}$$

The steps to construct the block reflector U are shown below. From (21) and (22) it can be seen that

$$\widehat{\Sigma}_1 = \Sigma_1 + X^T \Sigma_2 X \tag{23}$$

and

$$U^{-1} \begin{bmatrix} I \\ 0 \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} \Rightarrow U^{-1} = \begin{bmatrix} I & Y \\ X & Z \end{bmatrix} \tag{24}$$

U^{-1} can be factored as

$$U^{-1} = \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} \begin{bmatrix} I & Y \\ 0 & W \end{bmatrix}, \tag{25}$$

where $Z = XY + W$ and

$$U = \begin{bmatrix} I & -YW^{-1} \\ 0 & W^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -X & I \end{bmatrix}. \tag{26}$$

Substituting for U, Σ, and $\widehat{\Sigma}$ in (21) we get

$$\widehat{\Sigma}_1 = \Sigma_1 + X^T \Sigma_2 X \tag{27}$$

$$-\widehat{\Sigma}_1 Y W^{-1} = X^T \Sigma_2 \tag{28}$$

$$\Sigma_2 = W^{-T}(Y^T \widehat{\Sigma}_1 Y + \widehat{\Sigma}_2) W^{-1}. \tag{29}$$

If we choose $W = I$, then we have

$$\widehat{\Sigma}_1 = \Sigma_1 + (X^T \Sigma_2) X \tag{30}$$

$$Y = -\widehat{\Sigma}_1^{-1}(X^T \Sigma_2) \tag{31}$$

$$\widehat{\Sigma}_2 = \Sigma_2 - Y^T \widehat{\Sigma}_1 Y = \Sigma_2 + (X^T \Sigma_2)^T Y. \tag{32}$$

It can be seen from the above description that the primitives used in this blocking scheme are of the BLAS 3 type (matrix multiplication). The cost of obtaining the block reflector in this form is $6.83m^3 + m^2$ flops. This is substantially higher than the cost of the previous blocking schemes but the operations are performed at a higher rate (BLAS 3 rate versus BLAS 2 for the other schemes). The advantage of this scheme over the others is that applying the block reflector to the rest of the generator of size $2m \times mp$ requires $4m^3 p$ flops, which is significantly less than that of the other blocking schemes.

3. SYMMETRIC INDEFINITE BLOCK TOEPLITZ MATRICES

In Section 2 we described the Schur algorithm to obtain a Cholesky (LL^T) factorization and an LDL^T factorization of a block Toeplitz matrix. In this section, we discuss modifications to the classical Schur to obtain an LDL^T factorization of a symmetric indefinite block Toeplitz matrix. We begin by discussing a possible degeneracy for indefinite matrices and then present a few techniques to overcome these degenerate steps in the Schur algorithm.

The following theorem states that if the block Toeplitz matrix T is positive definite, it can be shown that the block reflector U (22) always exists at every step of the Schur algorithm.

THEOREM 4. *Given a symmetric positive definite block Toeplitz T, at every step of the Schur algorithm, one can always construct a block reflector U, such that* (21) *and* (22) *are satisfied.*

Proof. See [13]. ■

3.1. Modifications to the Schur Algorithm for the Indefinite Case

If the block Toeplitz matrix T is symmetric indefinite, then the Schur algorithm could break down because of a singular $\widehat{\Sigma}_1$ (see (31)). Even if $\widehat{\Sigma}_1$ is badly conditioned the Schur algorithm would produce an inaccurate factorization. If at any step of the Schur algorithm $\widehat{\Sigma}_1$ is found to be well conditioned, then one can proceed with the Schur algorithm exactly as described in Section 2.5 to the next step.

There are two ways in which one can, in the event of degeneracy, avoid the problem of near or total breakdown of the Schur algorithm. The first method involves perturbing the pivot element of the generator such that the matrix $\widehat{\Sigma}_1$ in (31) is invertible. This method of "boosting" the pivot block provides an inexact factorization of the block Toeplitz matrix. Iterative refinement may be used to correct the solution of such a system. The other method of avoiding degeneracy is to look ahead a few steps of the Schur algorithm, till a well-conditioned principal minor can be obtained. These two techniques are discussed in Sections 3.2 and 3.3.

3.2. Approximate Factorization of Indefinite Toeplitz Matrices Using Perturbations

We outline a modification to the Schur algorithm to factor a symmetric indefinite block Toeplitz matrix with singular principal minors. As indicated in the previous subsection, if the matrix has a singular principal minor, then the hyperbolic Householder reflector cannot be constructed

and the Schur algorithm breaks down. If the pivot block is perturbed such that the matrix $\widehat{\Sigma}_1$ becomes nonsingular, then the Schur algorithm can be continued. This provides an approximate factorization of block Toeplitz matrix.

The blocking scheme used in this subsection is different from the one discussed in the previous subsection. The scheme used is a modification of the techniques discussed in Section 2. Consider a symmetric indefinite block Toeplitz matrix T with block size $m \times m$ whose first block row is given as $\widehat{T}_i, i = 1, \ldots, p$. If $\widehat{T}_1$ is nonsingular and $\widehat{T}_1 = PL_1\Sigma_1 L_1^T P^T$ (P is a permutation matrix), then the generator for the Toeplitz matrix is given as

$$\text{Gen} = \begin{pmatrix} T_1 & T_2 & \cdots & T_p \\ 0 & T_2 & \cdots & T_p \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & -\Sigma_1 \end{pmatrix}, \tag{33}$$

where $T_i = (L\Sigma_1)^{-1} P^T \widehat{T}_i, i = 1, \ldots, p$ and Σ_1 is a diagonal signature matrix. If the leading block $\widehat{T}_1$ is singular, then the generator is given as

$$\text{Gen} = \begin{pmatrix} 0.5(\widehat{T}_1 + I_m) & \widehat{T}_2 & \cdots & \widehat{T}_p \\ 0.5(\widehat{T}_1 - I_m) & \widehat{T}_2 & \cdots & \widehat{T}_p \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} I_m & 0 \\ 0 & -I_m \end{pmatrix},$$

where I_m is an identity matrix of size m.

At each step of the Schur algorithm, a block hyperbolic Householder matrix is constructed using the first block column of the generator at that step. Let us consider the blocking schemes discussed in Section 2. A sequence of hyperbolic Householder matrices is constructed such that the diagonal element of the upper block is used to zero out all the elements of the column below it. At the jth step of the process of zeroing out the lower block, the vector u has the form $[0, \ldots, 0, u_j, \ldots, u_{2m}]$. Let the hyperbolic norm of u be $u^T W u$. A hyperbolic Householder reflector can transform a vector u to another vector b such that $u^T W u = b^T W b$. If we choose b to be $-\sigma e_j$ (using u_j to zero out the column), then $b^T W b = W(j,j)\sigma^2$. If $\text{sign}(W(j,j)) \neq \text{sign}(u^T W u)$, then one cannot obtain a reflector U such that $Uu = -\sigma e_j$. We would have to look for an alternate nonzero pivot element in the column of u that has the same signature as the sign of $u^T W u$. Let this be u_k. The element u_k can be permuted to the jth position and can be used as a pivot element to zero out the column below it.

Let us first assume that the hyperbolic norms of all the u vectors during the block reflector generation process are nonzero. The case of a zero hyperbolic norm is discussed later. The blocking schemes discussed in Section 2 can be easily extended to the indefinite case in the presence of permutations of the kind described above. Let us consider the VY blocking scheme. A derivation of the YTY^T form can be obtained similarly.

Let us consider a particular step in the Schur algorithm. Let the generator and signature matrix Gen and W satisfy the following displacement equation

$$\widetilde{T} - \widetilde{Z}^T \widetilde{T} \widetilde{Z} = \mathrm{Gen}^T W \ \mathrm{Gen}. \tag{34}$$

Consider the first step of the blocking process. Let P_1 be the permutation matrix to get the correct pivot element in place. The hyperbolic reflector U_1 is given as

$$U_1 = \widetilde{W}_1 - \frac{2\widetilde{x}_1 \widetilde{x}_1^T}{\widetilde{x}_1^T \widetilde{W}_1 \widetilde{x}_1}, \tag{35}$$

where $\widetilde{W}_1 = P_1 W_1 P_1^T$ (where $W_1 = W$) and $\widetilde{x}_1 = P_1 x$. Let us denote the first block column of the generator Gen that is used to produce the block reflector as A. The reflector U_1 is applied to a permuted version of A,

$$\begin{aligned} U_1 P_1 A &= \widetilde{W}_1 - \frac{2\widetilde{x}_1 \widetilde{x}_1^T}{\widetilde{x}_1^T \widetilde{W}_1 \widetilde{x}_1} P_1 A \\ U^{(1)} A &= \left(P_1 W_1 + (P_1 x_1) \left(-\frac{2x_1^T}{x_1^T W_1 x_1} \right) \right) A \\ U^{(1)} A &= \left(P_1 W_1 + v_1 y_1^T \right) A. \end{aligned} \tag{36}$$

The reflector $U^{(1)}$ shown above is W-unitary in the following sense

$$U^{(1)^T} \widetilde{W}_1 U^{(1)} = W, \tag{37}$$

where $\widetilde{W}_1 = P_1 W P_1^T$. This result is derived as follows:

$$\begin{aligned} U^{(1)^T} \widetilde{W}_1 U^{(1)} &= \left(P_1 W_1 + v_1 y_1^T \right)^T \widetilde{W}_1 \left(P_1 W_1 + v_1 y_1^T \right) \\ &= W_1 P_1^T \widetilde{W}_1 P_1 W_1 + y_1 x_1^T P_1^T \widetilde{W}_1 P_1 x_1 y_1^T \\ &\quad + y_1 x_1^T P_1^T \widetilde{W}_1 P_1 W_1 + W_1 P_1^T \widetilde{W}_1 P_1 x_1 y_1^T \\ \text{If} \quad P_1^T \widetilde{W}_1 P_1 &= W_1 \\ U^{(1)^T} \widetilde{W}_1 U^{(1)} &= W_1 + y_1 x_1^T W_1 x_1 y_1^T + y_1 x_1^T + x_1 y_1^T \\ &= W_1. \end{aligned} \tag{38}$$

Let $C^{(1)} = P_1 W_1$, $V^{(1)} = v_1$ and $Y^{(1)} = y_1$, we show by induction that at the $(i+1)$th step the block relector has the form $U^{(i+1)} = C^{(i+1)} +$

$V^{(i+1)}Y^{(i+1)^T}$, where $C^{(i+1)} = P^{(i+1)}W^{(i+1)}$. At the first step $P^{(1)} = P_1$, $W^{(1)} = W_1$ and $U^{(1)} = U_1$. Assume that $U^{(i)}$ has been obtained in the correct form. We show that $U^{(i+1)}$ can be obtained in the correct form.

At the $(i+1)$th step $\widetilde{W}_{i+1}$ is given by

$$\begin{aligned}\widetilde{W}_{i+1} &= P_{i+1}\left(\begin{array}{c|c} I_i & 0 \\ \hline 0 & \widetilde{W}_i(i+1:2m, i+1:2m) \end{array}\right) P_{i+1}^T \\ &= P_{i+1}W_{i+1}P_{i+1}^T \end{aligned} \tag{39}$$

and

$$\begin{aligned} U^{(i+1)} &= U_{i+1}P_{i+1}U^{(i)} \\ &= \left(P_{i+1}W_{i+1} + P_{i+1}x_{i+1}y_{i+1}^T\right)\left(C^{(i)} + V^{(i)}Y^{(i)^T}\right) \\ &= P_{i+1}W_{i+1}C^{(i)} \\ &\quad + \left(P_{i+1}W_{i+1}V^{(i)} \mid P_{i+1}x_{i+1}\right)\left(\begin{array}{c} Y^{(i)^T} \\ \hline y_{i+1}^T\left(C^{(i)} + V^{(i)}Y^{(i)^T}\right) \end{array}\right) \\ &= C^{(i+1)} + V^{(i+1)}Y^{(i+1)^T}, \end{aligned}$$

where

$$\begin{aligned} C^{(i+1)} &= P_{i+1}W_{i+1}C^{(i)} \\ &= P_{i+1}W_{i+1}P^{(i)}W^{(i)} \\ &= \left(P_{i+1}P^{(i)}\right)\left(P^{(i)^T}W_{i+1}P^{(i)}W^{(i)}\right) \\ &= P^{(i+1)}W^{(i+1)}. \end{aligned} \tag{40}$$

The block hyperbolic Householder transformation at the end of m steps has the form $U^{(m)} = C^{(m)} + V^{(m)}Y^{(m)^T}$. From (37) we know that $U^{(1)^T}\widetilde{W}_1 U^{(1)} = W$, where $\widetilde{W}_1 = P_1 W P_1^T$. It can be shown by induction that

$$\begin{aligned} U^{(m)^T}W_{\text{next}}U^{(m)} &= W \\ W_{\text{next}} &= P_m \cdots P_1 W P_1^T \cdots P_m^T \\ &= P^{(m)}WP^{(m)}, \end{aligned} \tag{41}$$

where W_{next} is the signature matrix for the next Schur step.

If the hyperbolic norm of any column is zero, then the Schur algorithm breaks down. The column of the generator is perturbed such that the hyperbolic norm of the column is of the order of $|\delta|$. An algorithm for the

perturbation of such a column of the generator is:

if $(u^T\widetilde{W}_{j-1}u = 0)$ *then*
  $u_j \rightarrow$ pivot element
  $a = (0, \ldots, 0, u_{j+1}, \ldots, u_{2m})$
  if $(\widetilde{W}_{j-1}(j,j) = 1)$ *then*
    $u_j = \sqrt{\widetilde{W}_{j-1}(j,j)(|\delta| - a^T\widetilde{W}_{j-1}a)}$
  else
    $u_j = \sqrt{\widetilde{W}_{j-1}(j,j)(-|\delta| - a^T\widetilde{W}_{j-1}a)}$
  end
else if $(u^T\widetilde{W}_{j-1}u > 0)$ *then* $\quad (+|\epsilon| \text{ say})$
  if $(\widetilde{W}_{j-1}(j,j) = 1)$ *then*
    $u_j \rightarrow$ pivot element
    $a = (0, \ldots, 0, u_{j+1}, \ldots, u_{2m})$
    $u_j = \sqrt{\widetilde{W}_{j-1}(j,j)(|\delta| + |\epsilon| - a^T\widetilde{W}_{j-1}a)}$
  else
    $u_k \rightarrow$ pivot element $\quad (\widetilde{W}_{j-1}(k,k) = 1 \text{ say})$
    $a = (0, \ldots, 0, u_j, \ldots, u_{k-1}, 0, u_{k+1}, \ldots, u_{2m})$
    $u_k = \sqrt{\widetilde{W}_{j-1}(k,k)(|\delta| + |\epsilon| - a^T\widetilde{W}_{j-1}a)}$
  end
else $\quad (u^T\widetilde{W}_{j-1}u = -|\epsilon| \text{ say})$
  if $(\widetilde{W}_{j-1}(j,j) = 1)$ *then*
    $u_k \rightarrow$ pivot element $\quad (\widetilde{W}_{j-1}(k,k) = -1 \text{ say})$
    $a = (0, \ldots, 0, u_j, \ldots, u_{k-1}, 0, u_{k+1}, \ldots, u_{2m})$
    $u_k = \sqrt{\widetilde{W}_{j-1}(k,k)(-|\delta| - |\epsilon| - a^T\widetilde{W}_{j-1}a)}$
  else
    $u_j \rightarrow$ pivot element
    $a = (0, \ldots, 0, u_{j+1}, \ldots, u_{2m})$
    $u_j = \sqrt{\widetilde{W}_{j-1}(j,j)(-|\delta| - |\epsilon| - a^T\widetilde{W}_{j-1}a)}$
  end
end

The perturbation of a column of the pivot block column of the generator with zero hyperbolic norm allows us to continue the factorization process but introduces numerical instability into the algorithm. One way to circumvent the possible numerical instability of the Schur algorithm is to use iterative refinement on the system of equations. A similar perturbation technique has been used in [7] for the Levinson algorithm. They use the approximate factorization as a preconditioner in the conjugate-gradient al-

gorithm. The iterative refinement technique we propose requires less work than the preconditioned conjugate-gradient algorithm per iteration.

Let us consider the system of equations $Tx = b$, where T is an indefinite symmetric block Toeplitz with singular principal submatrices. Using the perturbation technique described above we obtain an approximate factorization

$$T + \delta T = LDL^T. \tag{42}$$

We solve the system of equations to get x_1

$$LDL^T x_1 = b \tag{43}$$

and then compute the residual r_1

$$r_1 = -Tx_1 + b. \tag{44}$$

Using the correction term Δx_1 obtained from

$$LDL^T \Delta x_1 = r_1 \tag{45}$$

we improve the estimated solution by

$$x_2 = x_1 + \Delta x_1. \tag{46}$$

The algorithm then becomes

Construct $LDL^T = T + \delta T$ using the Schur algorithm.
Solve $LDL^T x_1 = b$, and set $r_1 = -Tx_1 + b$.
for $i = 1, \ldots$
    Solve $LDL^T \Delta x_i = r_i$
    if $\|\Delta x_i\| < \text{tol}\ \|x_i\|$ *then* stop
    else
        $x_{i+1} = x_i + \Delta x_i$
        $r_{i+1} = -Tx_{i+1} + b$
    endif
endfor

From the error analysis of [32] we know that the computed quantities $\overline{x}_i$, $\Delta\overline{x}_i$, and $\overline{r}_i$ satisfy the following identities

$$\overline{r}_i = -T\overline{x}_i + b + \delta\overline{r}_i \quad = \quad r_i + \delta\overline{r}_i \qquad \text{with} \qquad \|\delta\overline{r}_i\| \le \epsilon_i \|T\|\ \|\overline{x}_i\| \tag{47}$$

$$(LDL^T + \delta T_i)\Delta\overline{x}_i = \overline{r}_i \qquad \text{with} \qquad \|\delta T_i\| \le \eta_i \|L\|^2\ \|D\|, \tag{48}$$

where ϵ_i, η_i are of the order of the machine precision of the computer. From these equations we obtain

$$(T + \delta T + \delta T_i)\Delta\overline{x}_i = b - T\overline{x}_i + \delta\overline{r}_i \tag{49}$$

and after some rewriting

$$r_{i+1} = b - T(\overline{x}_i + \Delta\overline{x}_i) \quad = \quad (\delta T + \delta T_i)\Delta\overline{x}_i - \delta\overline{r}_i$$

or also

$$\begin{aligned} r_{i+1} &= (\delta T + \delta T_i)(T + \delta T + \delta T_i)^{-1}(r_i + \delta \overline{r}_i) - \delta \overline{r}_i \\ &= \Delta T_i (T + \Delta T_i)^{-1} r_i - T(T + \Delta T_i)^{-1} \delta \overline{r}_i, \end{aligned}$$

where the terms δT and δT_i, which are typically of the same order, have been grouped together in ΔT_i. Defining $M_i = \Delta T_i \, T^{-1}$ we have

$$r_{i+1} = M_i (I + M_i)^{-1} r_i - (I + M_i)^{-1} \delta \overline{r}_i. \tag{50}$$

If we can now obtain that $\max_i \|\Delta T_i \; T^{-1}\| = \gamma \ll 1$ then the above equation is a difference equation that will converge linearly, with a factor $\beta = \gamma(1-\gamma)$, to a steady-state value of

$$\|r_\infty\| \approx \frac{1}{1-\beta}\frac{1}{1-\gamma}\|\delta r_{\max}\| = \frac{1}{1-2\gamma}\|\delta r_{\max}\| \leq \frac{\epsilon_{\max}}{1-2\gamma}\|T\|\|x\|. \tag{51}$$

Since our assumption is that γ is small, this final residual is about what one can expect from a stable algorithm. If we obtain that $\gamma = \sqrt[k]{\epsilon}$ then the number of iteration steps to get "convergence" to this result would be k.

As shown above it is important to bound $\|\delta T \; T^{-1}\|$ in the construction of the factorization. Since LDL^T is only an approximate decomposition of T (but an exact decomposition of $T + \delta T$), we have the freedom to perturb T so as to obtain a better bound for $\delta T \; T^{-1}$. In this subsection we show how to obtain this by selective perturbations introduced in the Schur algorithm. Similar ideas have been developed independently for the Levinson algorithm by Concus and Saylor [7].

At the ith step of the Schur algorithm we apply a block hyperbolic Householder transformation U_i to the generator $G'(i)$ to get $G'(i+1)$, i.e., $U_i G'(i) = G'(i+1)$. The corresponding decomposition for the Toeplitz matrix is

$$\begin{aligned} T &= \begin{bmatrix} G_1^T(i) \; G_2^T(i) \end{bmatrix} \widehat{U}_i W \widehat{U}_i \begin{bmatrix} G_1(i) \\ G_2(i) \end{bmatrix} \\ &= \begin{bmatrix} G_1^T(i+1) G_2^T(i+1) \end{bmatrix} W \begin{bmatrix} G_1(i+1) \\ G_2(i+1) \end{bmatrix}, \end{aligned}$$

where $\widehat{U}_i$ is essentially a block arrangement of identity matrices and U_i blocks. Hence,

$$\|\widehat{U}_i\|_2 = \|U_i\|_2 \qquad \text{and} \qquad \|\widehat{U}_i^{-1}\|_2 = \|U_i^{-1}\|_2. \tag{52}$$

If we now perturb the generator matrix $G'(i)$ by a perturbation of norm $\delta\|G(1)\|_2$ then the equivalent perturbation $\|\Delta G(1)\|$ of $G(1)$ is bounded by

$$\|\Delta G(1)\| \leq \delta \|U_1^{-1}\|_2 \cdots \|U_{i-1}^{-1}\|_2 \; \|G(1)\|$$

and that of T is proportional to $\delta\|U_1^{-1}\|_2 \cdots \|U_{i-1}^{-1}\|_2 \|T\|$. In other words, the norms of the inverses of the block transformations performed thus far

act as a growth factor in the back transformations of the perturbation to the original matrix. Another factor that we have to be concerned about is that the transformation U_i for which the δ perturbation was done will have a norm of approximately $1/\delta$ and the norm of the next generator $G(i+1)$ will be increased by that amount. Numerical errors in subsequent steps will thus be proportional to this value and when transforming these back to the original matrix T we find again that we have to keep

$$\epsilon \|U_1\| \cdots \|U_{n-1}\|$$

bounded. Experience has shown that for each perturbation δ performed at a certain step i, there will be two block transformations of norm approximately $1/\delta$. For hyperbolic Householder transformations, $\|U\| = \|U^{-1}\|$. Hence, the total error due to one perturbation is

$$\frac{\|\Delta T\|}{\|T\|} = \delta + \frac{\epsilon}{\delta^2}. \tag{53}$$

We choose δ so as to minimize the above expression. The value of δ that minimizes the above expression is $\sqrt[3]{2\epsilon}$ or $\delta \approx \sqrt[3]{\epsilon}$. This gives us

$$\begin{aligned}
\gamma &= \|\Delta T\, T^{-1}\| \\
&\leq \|\Delta T\| \;\; \|T^{-1}\| \\
&\leq \frac{\|\Delta T\|}{\|T\|} \mathrm{cond}(T) \\
&\approx \delta + \frac{\epsilon}{\delta^2} \qquad \text{(if } T \text{ is well conditioned)} \\
&\approx \sqrt[3]{\epsilon} \qquad \text{(if we set } \delta = \sqrt[3]{\epsilon}\text{)}.
\end{aligned} \tag{54}$$

The subsequent number of steps of iterative refinement would be three. The above analysis holds true if we perturb the generator matrix just once.

Let us consider the case when we need to perturb twice. Let δ_1 and δ_2 be the two perturbations at steps i and j respectively. The total perturbation to the original Toeplitz matrix can be expected to be of the following order

$$\begin{aligned}
\|\delta T\| &= \left(\delta_1 \left\|U_1^{-1}\right\| \cdots \left\|U_{i-1}^{-1}\right\| + \delta_2 \left\|U_1^{-1}\right\| \cdots \left\|U_{j-1}^{-1}\right\|\right) \|T\| \\
&\approx \left(\delta_1 + \frac{\delta_2}{\delta_1^2}\right) \|T\|.
\end{aligned} \tag{55}$$

The numerical error due to the block transformations of norms approximately equal to $1/\delta_1^2$ and $1/\delta_2^2$ is

$$\begin{aligned}
\text{Numerical errors} &= \epsilon \|U_1\| \cdots \|U_{n-1}\| \;\; \|T\| \\
&= \frac{\epsilon}{\delta_1^2 \delta_2^2}.
\end{aligned} \tag{56}$$

The total error due to the two factors is

$$\frac{\|\Delta T\|}{\|T\|} = \delta_1 + \frac{\delta_2}{\delta_1^2} + \frac{\epsilon}{\delta_1^2 \delta_2^2}. \tag{57}$$

The above expression is minimized by choosing $\delta_1 = \sqrt[9]{\epsilon}$ and $\delta_2 = \sqrt[3]{\epsilon}$. This means that we would require nine iterations to get to machine precision. It is impossible to know ahead of time how many perturbations one requires to carry on with the Schur algorithm. If, upon performing one perturbation of $\sqrt[3]{\epsilon}$, we see during the Schur algorithm that another perturbation is needed, we would have to backtrack to the first perturbation and change the value of δ_1 from $\sqrt[3]{\epsilon}$ to $\sqrt[9]{\epsilon}$. This is usually very wasteful of computation. Also, if the number of times the generator needs to be perturbed increases, the accuracy is lost very quickly and we might have to look for other ways to handle such cases. From our experiments with Toeplitz matrices, we have observed that even for Toeplitz matrices with several singular minors one perturbation is sufficient. So, in practice, it might be safe to assume that a large number of systems can be solved by perturbing the generator only once and the above analysis holds. For systems where this is not the case the algorithms discussed below are applicable.

We now present an example of a symmetric Toeplitz matrix with a singular principal minor. Consider the following block Toeplitz matrix T with a block size of 2.

$$T(1:2,1:2) = \begin{pmatrix} 0.04324379151529 & 0.29158091418984 \\ 0.29158091418984 & 0.67982106506507 \end{pmatrix}$$

$$T(1:2,3:4) = \begin{pmatrix} 0.00769818621115 & 0.06684223751856 \\ 0.38341565075489 & 0.41748597445781 \end{pmatrix}$$

$$T(1:2,5:6) = \begin{pmatrix} 0.68677271236050 & 0.93043649472782 \\ 0.58897664285683 & 0.84616689050857 \end{pmatrix}$$

$$T(1:2,7:8) = \begin{pmatrix} 0.52692877758617 & 0.65391896229885 \\ 0.09196489075756 & 0.41599935685098 \end{pmatrix}$$

This matrix has a singular principal minor ($T(1:4,1:4)$ is singular). At the second step of the Schur algorithm, while blocking the two hyperbolic Householder transformations, the second column of the pivot block column of the generator has zero hyperbolic norm. We introduce a perturbation of $\sqrt{10^{-16}} \approx 10^{-5}$. The norm of the block hyperbolic Householder after perturbation is $2.2172e{+}07$ and the norm of U_4 is $2.821e{+}07$. This indicates that a single perturbation of δ produces two block hyperbolic Householder transformations of norm approximately equal to $1/\delta$. The norm of $\delta T.T^{-1}$ is $5.5761e-04$. If we consider $x = (1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1)^T$, then $b = (3.2074\ \ 3.7154\ \ 2.4177\ \ 3.6918\ \ 2.0762\ \ 4.0332\ \ 2.6206\ \ 4.3022)$. We find $\|x - x_1\| = 3.1699e-04$. Using iterative refinement, we find that after one step $\|x - x_2\| = 9.7515e-08$, after the second step $\|x - x_3\| = 3.2389e-11$, and after the third step $\|x - x_4\| = 3.5231e-15$, which is approximately equal to the machine precision. Note that this is consistent with the analysis above.

3.3. Look-Ahead Schur Algorithms

Perturbing the generators in the event of singularities during the Schur algorithm produces an approximate factorization of the block Toeplitz matrix. Iterative refinement is needed to improve the accuracy of the solution. If an exact factorization of a symmetric indefinite block Toeplitz matrix is desired, then we would have to deal with the singular principal minors of the Toeplitz matrix in a different way.

One important way of avoiding the singular principal minors during the Schur algorithm is to look ahead over the singularities. This technique may also be used when the principal minors are badly conditioned. Look-ahead techniques were originally proposed to improve the numerical robustness of the Lanczos algorithm applied to an indefinite matrix T in the presence of singular and nearly singular leading principal minors in T [25]. Most of the techniques related to these developments are based on the theory of orthogonal polynomials [17] or equivalently on that of T conjugate directions. This theory is in turn closely connected to that of Hankel matrices and the Padé algorithm [3] and of Toeplitz matrices and the Levinson algorithm [12]. In both cases one constructs the decomposition $L^{-1}TL^{-T} = D$ where T is the given Toeplitz matrix. The rows of L^{-1} are the conjugate directions or also contain the coefficients of the orthogonal polynomials. Look-ahead techniques have been proposed and yielded algorithms with satisfactory numerical behavior [3, 4, 12, 18, 25].

The look-ahead Schur algorithm proposed in [18] is based on orthogonal polynomials and does not extend to block Toeplitz matrices. Look-ahead Schur algorithms for Toeplitz systems with exactly singular principal minors have been proposed in [10, 24].

In Sections 3.3.1 and 3.3.2 we discuss two look-ahead Schur algorithms that are based entirely on matrix operations and hence extend easily to block Toeplitz matrices.

3.3.1. Algorithm 1. Consider a $mp \times mp$ block Toeplitz or quasi-block Toeplitz matrix T with a block size of $m \times m$. Let the displacement equation of this matrix be

$$T - Z^T T Z = G_0^T \Sigma_0 G_0$$

where

$$G = \begin{pmatrix} H_{00} & H_{01} & H_{02} & \cdots & H_{0p-1} \\ G_{00} & G_{01} & G_{02} & \cdots & G_{0p-1} \end{pmatrix}$$

and

$$\Sigma_0 = \begin{pmatrix} \Sigma_{01} & 0 \\ 0 & \Sigma_{02} \end{pmatrix}. \tag{58}$$

The Schur algorithm proceeds by applying a Σ-unitary transformation U_0 ($U_0^T \Sigma_1 U_0 = \Sigma_0$) to G_0 such that

$$\widetilde{G}_0 = U_0 G_0 = \begin{pmatrix} \widetilde{H}_{00} & \widetilde{H}_{01} & \widetilde{H}_{02} & \cdots & \widetilde{H}_{0p-1} \\ 0 & \widetilde{G}_{01} & \widetilde{G}_{02} & \cdots & \widetilde{G}_{0p-1} \end{pmatrix},$$

$$\Sigma_1 = \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{12} \end{pmatrix}.$$

It was shown earlier that if $H_{00}^T \Sigma_{01} H_{00} + G_{00}^T \Sigma_{02} G_{00}$ is singular, then the Schur algorithm breaks down. If it is badly conditioned, the factorization would have significant numerical errors. It can be seen that $H_{00}^T \Sigma_{01} H_{00} + G_{00}^T \Sigma_{02} G_{00}$ is the $(1,1)$ block of the Toeplitz or quasi-Toeplitz matrix. More generally, if the $2m, 3m, \ldots, (k-1)m$ principal minors are singular or badly conditioned and the km principal minor is well conditioned, then to preserve numerical accuracy we would have to look ahead over the $(k-1)m$ steps of the Schur algorithm. Let the matrix T be partitioned as

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{12}^T & T_{22} \end{bmatrix}, \tag{59}$$

where T_{11} is the $km \times km$ principal minor of T that is well conditioned. If we are to "jump" over $(k-1)m$ steps of the Schur algorithm, we also require that the off-diagonal entries of $T_{11}^{-1} T_{12}$ not be too large. A detailed discussion on the determination of the look-ahead step size (denoted here by k) can be found in [4] and [12]. We restrict our discussion to the look-ahead scheme after the determination of the step size k.

The first step in this look-ahead scheme is the computation of the first km rows of the Toeplitz or quasi-Toeplitz matrix given by $[T_{11} \mid T_{12}]$. From this we obtain the diagonal block and the upper triangular factor of the Toeplitz or quasi-Toeplitz matrix by an $O(n^3)$ "slow" algorithm such as the Bunch–Kaufman for symmetric indefinite matrices. The first km rows of the block Toeplitz matrix can be obtained from the generator matrix and the signature in $O(m^2 p)$ flops.

Let the matrix $[T_{11} \mid T_{12}]$ be factored into

$$[T_{11} \mid T_{12}] = D_k L_k^T, \tag{60}$$

where $D_k = T_{11}$ and L_k^T is a $km \times mp$ matrix with a leading identity matrix of size km

$$\begin{aligned} L_k^T &= (I_{km} \mid T_{11}^{-1} T_{12}) \\ &= (I_{km} \mid \widehat{L}_k^T). \end{aligned} \tag{61}$$

The Schur complement of the Toeplitz matrix T w.r.t. the kmth principal minor is

$$T_{\mathrm{sc}}^{(k)} = T(km+1:mp, km+1:mp) - \widehat{L}_k D_k \widehat{L}_k^T. \tag{62}$$

The Schur algorithm can be continued if we obtain a factorization of $T_{\mathrm{sc}}^{(k)}$ that is of the form shown in (58). Since the displacement rank of the Schur complement of a block Toeplitz matrix is $2m$, such a factorization exists. We now proceed to show how such a factorization can be obtained.

Let us denote the matrix $T(km+1:mp, km+1:mp)$ as $\widetilde{T}$. Let $\widehat{Z}$ be a block right shift matrix of size $(p-k)m$. The displacement of the Schur complement $T_{\mathrm{sc}}^{(k)}$ is given as

$$T_{\mathrm{sc}}^{(k)} - \widehat{Z}^T T_{\mathrm{sc}}^{(k)} \widehat{Z} = \widetilde{T} - \widehat{Z}^T \widetilde{T} \widehat{Z} - \widehat{L}_k D_k \widehat{L}_k^T + \widehat{Z}^T \widehat{L}_k D_k \widehat{L}_k^T \widehat{Z}. \tag{63}$$

If the generator G_0 is partitioned as

$$\begin{aligned} G_0 &= \left(\begin{array}{ccc|ccc} H_{00} & \cdots & H_{0k-1} & H_{0k} & \cdots & H_{0p-1} \\ G_{00} & \cdots & G_{0k-1} & G_{0k} & \cdots & G_{0p-1} \end{array}\right) \\ &= (\widetilde{G}_0 \mid \widehat{G}_0), \end{aligned} \tag{64}$$

then

$$\widetilde{T} - \widehat{Z}^T \widetilde{T} \widehat{Z} = \widehat{G}_0^T \Sigma_0 \widehat{G}_0 \tag{65}$$

and

$$T_{\mathrm{sc}}^{(k)} - \widehat{Z}^T T_{\mathrm{sc}}^{(k)} \widehat{Z} = \widehat{G}_0^T \Sigma_0 \widehat{G}_0 - \widehat{L}_k D_k \widehat{L}_k^T + \widehat{Z}^T \widehat{L}_k D_k \widehat{L}_k^T \widehat{Z}. \tag{66}$$

Factoring $D_k = L_{D_k} \Sigma_{D_k} L_{D_k}^T$, where Σ_{D_k} is a diagonal matrix with ± 1 entries, the right-hand side of the above equation can be rewritten as

$$\begin{pmatrix} \widehat{G}_0^T & \widehat{L}_k L_{D_k} & \widehat{Z}^T \widehat{L}_k L_{D_k} \end{pmatrix} \begin{pmatrix} \Sigma_0 & 0 & 0 \\ 0 & -\Sigma_{D_k} & 0 \\ 0 & 0 & \Sigma_{D_k} \end{pmatrix} \begin{pmatrix} \widehat{G}_0 \\ L_{D_k} \widehat{L}_k^T \\ L_{D_k} \widehat{L}_k^T \widehat{Z} \end{pmatrix}$$

Hence, we have

$$\Delta T_{\mathrm{sc}}^{(k)} = T_{\mathrm{sc}}^{(k)} - \widehat{Z}^T T_{\mathrm{sc}}^{(k)} \widehat{Z} = \widehat{G}^T W \widehat{G}. \tag{67}$$

This indicates that we can readily obtain a generator for the Schur complement. The problem with (67) is that the generator $\widehat{G}$ has a rank of at most $2km + 2m$. We know that the minimal generator of a block Toeplitz matrix has rank $2m$. We, therefore, have to reduce the generator shown

above so that a minimal generator is obtained. The displacement of the Schur complement, $\Delta T_{sc}^{(k)}$, is a symmetric indefinite matrix of rank $2m$. To obtain a rank $2m$ factorization of this matrix one would have to use the Bunch–Kaufman algorithm. A brief description of a delayed update version of the algorithm follows. Consider the ith step of the Bunch–Kaufman algorithm, and let the partial factorization of the matrix $\Delta T_{sc}^{(k)}$ be

$$\Delta T_{sc}^{(k)} = \left[\begin{array}{c|c} d & u \\ \hline u^T & X \end{array}\right], \tag{68}$$

where d is a block diagonal matrix with 1×1 or 2×2 blocks. The next step is the computation of the first row of the matrix X. This can be obtained by computing the corresponding row of $\Delta T_{sc}^{(k)}$ and updating it with $u^T d u$. It must be noted that the matrix $\Delta T_{sc}^{(k)}$ can be stored in its factored form and when a certain row is needed it can be computed using the factorization in (67). For example, the jth row of $\Delta T_{sc}^{(k)}$ is given by $\widehat{g}_j^T W \widehat{G}$ and requires $O(mkn)$ flops where m is the block size, k is the look-ahead step size, and n is the number of columns of $\widehat{G}$. After obtaining the first row of X, the maximum element of this row is computed. If the $(1,1)$ element of X can be used as the pivot (for a detailed description of the Bunch–Kaufman algorithm see [16]), then this row can be used to compute the next row of the factorization. If the $(1,1)$ element cannot be used as a pivot, another row of the matrix X needs to be computed in the same way as described above. In some cases this new row becomes the pivot row. In others the first row and the new row are used to define a 2×2 pivot block, which is used in the elimination. After $\widehat{m}$ steps with $s_i \times s_i$ pivot blocks, where $\sum_{i=1}^{\widehat{m}} s_i = 2m$, the generator of the $\Delta T_{sc}^{(k)}$ is obtained.

This look-ahead algorithm requires $2km + 2m$ of storage for the generator $\widehat{G}$. In addition, during the reduction of $\widehat{G}$ to G_k a Bunch–Kaufman like pivoting strategy is applied to obtain 1×1 or 2×2 pivot blocks that are used to compute the hyperbolic Householder transforms. The pivot search strategy requires reduction primitives to find the column with maximum hyperbolic norm. In Section 3.3.2 we present an alternate look-ahead Schur algorithm that requires less storage and in some cases less computation than this method and avoids the Bunch–Kaufman pivoting strategy all together. Hence, reduction primitives that perform poorly on distributed memory machines are avoided.

3.3.2. Algorithm 2. In this section we discuss another look-ahead Schur algorithm that requires less storage than the previous scheme and avoids the reduction primitives used in the Bunch–Kaufman pivoting strategy. A similar algorithm has been developed independently by Sayed and Kailath [28].

Let T be a general symmetric, block Toeplitz matrix of dimension $N \times N$ and block size $m \times m$, i.e., row of T be given as

$$T = \begin{bmatrix} T_0 & T_1 & \cdots & T_{p-1} \\ T_1^T & T_0 & \ddots & T_{p-2} \\ \vdots & \ddots & \ddots & \vdots \\ T_{p-1}^T & T_{p-2}^T & \cdots & T_0 \end{bmatrix}, \qquad T_0 = T_0^T, \qquad N = m \times p. \tag{69}$$

Let Z be a block right shift matrix of size mp; then the displacement equation of the matrix T can be written as

$$T - Z^T T Z = G^T \Sigma G. \tag{70}$$

Let us assume that T_0 is ill conditioned. A look-ahead Schur step would be needed to preserve numerical accuracy of the factorization. In addition, let us assume that the $m, 2m, \ldots, (k-1)m$ principal minors are ill conditioned and that the km principal minor is well conditioned. Partition T and Z conformally as

$$T = \left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline T_{21} & T_{22} \end{array}\right], \qquad Z = \left[\begin{array}{c|c} Z_{11} & Z_{12} \\ \hline 0 & Z_{22} \end{array}\right], \tag{71}$$

where T_{11} and Z_{11} are of dimension $mk \times mk$ (a multiple of the block size) and T_{11} is assumed to be invertible (this is always possible by choosing k large enough). Let us also assume that all the conditions for determining the look-ahead step size of k as discussed in [12] are satisfied. We now derive updating formulas for the Schur complement of a matrix T with low displacement rank and show that it also has low displacement rank. The rank $2m$ factorization of the displacement of the Schur complement provides the generator for the subsequent steps of the Schur algorithm. This part is related to the work of [22], but is not contained in it.

Define

$$X = T_{11}^{-1} T_{12}, \qquad X^T = T_{12}^T T_{11}^{-1}, \qquad U = \left[\begin{array}{c|c} I & -X \\ \hline & I \end{array}\right]; \tag{72}$$

then it follows that

$$U^T T U = \left[\begin{array}{c|c} T_{11} & \\ \hline & T_{\mathrm{sc}} \end{array}\right], \qquad T_{\mathrm{sc}} = T_{22} - T_{12}^T T_{11}^{-1} T_{12}, \tag{73}$$

where T_{sc} is the Schur complement of T with respect to T_{11}. Applying $U^T(\cdot)U$ to (70) yields

$$U^T T U - (U^T Z^T U^{-T})\, U^T T U\, (U^{-1} Z U) = U^T G^T \Sigma G U. \tag{74}$$

Note that

$$U^{-1}ZU = \left[\begin{array}{c|c} Z_{11} & \widehat{Z}_{12} \\ \hline & Z_{22} \end{array}\right], \qquad \widehat{Z}_{12} = [I \mid X]Z\left[\frac{-X}{I}\right]. \tag{75}$$

Using (73) and (75) we can reduce (74) to

$$\left[\begin{array}{c|c} T_{11} & \\ \hline & T_{\mathrm{sc}} \end{array}\right] - \left[\begin{array}{c|c} Z_{11}^T & \\ \hline \widehat{Z}_{12}^T & Z_{22}^T \end{array}\right] \left[\begin{array}{c|c} T_{11} & \\ \hline & T_{\mathrm{sc}} \end{array}\right] \left[\begin{array}{c|c} Z_{11} & \widehat{Z}_{12} \\ \hline & Z_{22} \end{array}\right] = U^T G^T \Sigma G U. \tag{76}$$

Equating the $(1,2)$ and $(2,2)$ positions in the above equation we have

$$\begin{aligned} M &= Z_{11}^T T_{11} \widehat{Z}_{12} + [I \mid 0] G^T \Sigma G \left[\frac{-X}{I}\right] = 0, \\ \Delta T_{\mathrm{sc}} &= T_{\mathrm{sc}} - Z_{22}^T T_{\mathrm{sc}} Z_{22} \\ &= \widehat{Z}_{12}^T T_{11} \widehat{Z}_{12} + \left[-X^T \middle| I\right] G^T \Sigma G \left[\frac{-X}{I}\right]. \end{aligned} \tag{77}$$

Substituting for $\widehat{Z}_{12}$ from (75) we can further simplify M and ΔT_{sc} to

$$M = [I \mid 0] \left\{ Z^T \left[\frac{I}{X^T}\right] T_{11} [I \mid X] Z + G^T \Sigma G \right\} \left[\frac{-X}{I}\right] = 0 \tag{78}$$

$$\Delta T_{\mathrm{sc}} = [-X^T \mid I] \left\{ Z^T \left[\frac{I}{X^T}\right] T_{11} [I \middle| X] Z + G^T \Sigma G \right\} \left[\frac{-X}{I}\right]. \tag{79}$$

Substituting for X in the matrix in the middle of the above equations we get

$$\begin{aligned} W &= Z^T \left[\frac{T_{11}^T}{T_{12}^T}\right] T_{11}^{-1} [T_{11} \mid T_{12}] Z + G^T \Sigma G \\ &= \left[Z^T \left[\frac{T_{11}^T}{T_{12}^T}\right] \middle| G^T \right] \left[\begin{array}{c|c} T_{11}^{-1} & 0 \\ \hline 0 & \Sigma \end{array}\right] \left[\frac{[T_{11} \mid T_{12}] Z}{G}\right]. \end{aligned} \tag{80}$$

This expression can now be further simplified to prove that the rank of ΔT_{sc} is at most α. To prove this we first need the following lemma.

LEMMA 5. *Let*

$$W = \begin{bmatrix} F_{11}^T & F_{21}^T \\ F_{12}^T & F_{22}^T \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}, \tag{81}$$

where Σ_1 *and* $W_{11} = F_{11}^T \Sigma_1 F_{11} + F_{21}^T \Sigma_2 F_{21}$ *are invertible. Then there always exists a transformation* H *such that*

$$H^T \begin{bmatrix} \widetilde{\Sigma}_1 & 0 \\ 0 & \widetilde{\Sigma}_2 \end{bmatrix} H = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \tag{82}$$

$$H \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} = \begin{bmatrix} \widehat{F}_{11} & \widehat{F}_{12} \\ 0 & \widehat{F}_{22} \end{bmatrix}. \tag{83}$$

Proof. See [14]. ∎

To simplify (80) we now must apply this lemma to construct a transformation H such that

$$H^T \left[\begin{array}{c|c} \widetilde{T}_{11}^{-1} & \\ \hline & \widetilde{\Sigma} \end{array}\right] H = \left[\begin{array}{c|c} T_{11}^{-1} & \\ \hline & \Sigma \end{array}\right], \tag{84}$$

$$H \left[\begin{array}{c} [T_{11} \quad T_{12}]Z \\ \hline G \end{array}\right] = \left[\begin{array}{c|c} \widehat{T}_{11} & \widehat{T}_{12} \\ \hline 0 & \widehat{G}_2 \end{array}\right], \tag{85}$$

where $\widetilde{T}_{11}$ and $\widehat{T}_{11}$ are matrices of size $mk \times mk$, G has dimensions $\alpha \times N$, and $\widehat{G}_2$ has dimensions $\alpha \times (N - mk)$. To apply the above lemma we only need to show that W_{11} is invertible since T_{11} is invertible by assumption. From (76) it follows that

$$T_{11} = Z_{11}^T T_{11} Z_{11} + G_1^T \Sigma G_1, \qquad \text{where } G_1 = G \left[\frac{I}{0}\right]. \tag{86}$$

From (80), W_{11} equals

$$W_{11} = [I \mid 0] \left\{ Z^T \left[\frac{T_{11}^T}{T_{12}^T}\right] T_{11}^{-1} [T_{11} \mid T_{12}] Z + G^T \Sigma G \right\} \left[\frac{I}{0}\right] \tag{87}$$

and since

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ 0 & Z_{22} \end{bmatrix}, \qquad G = [G_1 \quad G_2], \tag{88}$$

we have

$$W_{11} = Z_{11}^T T_{11} T_{11}^{-1} T_{11} Z_{11} + G_1^T \Sigma G_1 = T_{11}, \tag{89}$$

which thus shows that W_{11} is invertible as well.

Applying (84) and (85) to (80) we obtain

$$W = \left[\begin{array}{c|c} \widehat{T}_{11}^T & 0 \\ \hline \widehat{T}_{12}^T & \widehat{G}_2^T \end{array}\right] \left[\begin{array}{c|c} \widetilde{T}_{11}^{-1} & 0 \\ \hline 0 & \widetilde{\Sigma} \end{array}\right] \left[\begin{array}{c|c} \widehat{T}_{11} & \widehat{T}_{12} \\ \hline 0 & \widehat{G}_2 \end{array}\right]. \tag{90}$$

Inserting this in (78) and (79) yields

$$M = [I \mid 0]W \left[\frac{-X}{I}\right] = \widehat{T}_{11}^T \widetilde{T}_{11}^{-1} [\widehat{T}_{11} \mid \widehat{T}_{12}] \left[\frac{-X}{I}\right] = 0$$

$$\Delta T_{\text{sc}} = \widehat{G}_2^T \widetilde{\Sigma} \widehat{G}_2 + [-X^T \mid I] \left[\frac{\widehat{T}_{11}^T}{\widehat{T}_{12}^T}\right] \widetilde{T}_{11}^{-1} [\widehat{T}_{11} \mid \widehat{T}_{12}] \left[\frac{-X}{I}\right]. \tag{91}$$

Since $M = 0$ and $\widehat{T}_{11}$ and $\widetilde{T}_{11}$ are invertible, we have

$$[\widehat{T}_{11} \mid \widehat{T}_{12}] \left[\frac{-X}{I}\right] = 0,$$

which yields,

$$\Delta T_{\text{sc}} = \widehat{G}_2^T \widetilde{\Sigma} \widehat{G}_2. \tag{92}$$

This establishes a new displacement identity where $\widetilde{\Sigma}$ and $\widehat{G}_2$ are obtained from (84)–(85).

The above description of the algorithm did not provide a method to construct the transformation H. We now outline one method to construct the matrix H. Assuming that T_{11} is invertible, we know that the matrix H satisfies the following

$$H \left[\frac{[T_{11} \mid T_{12}]Z}{G}\right] = \left[\begin{array}{c|c} T_{11} & T_{12} \\ \hline 0 & \widehat{G}_2 \end{array}\right] \tag{93}$$

$$H^* \begin{bmatrix} T_{11}^{-1} & \\ & \Sigma \end{bmatrix} H = \begin{bmatrix} T_{11}^{-1} & \\ & \Sigma \end{bmatrix}. \tag{94}$$

Let $H = RQ$, where R is upper block triangular and Q is unitary. Let G be partitioned as $[G_1 \,|\, G_2]$, where G_1 has dimensions $\alpha \times mk$. Let R be partitioned as

$$R = \left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline 0 & R_{22} \end{array}\right]. \tag{95}$$

From (93), it can be seen that H must satisfy

$$H \left[\frac{T_{11}Z_{11}}{G_1}\right] = \begin{bmatrix} T_{11} \\ 0 \end{bmatrix} \Rightarrow RQ \left[\frac{T_{11}Z_{11}}{G_1}\right] = \begin{bmatrix} T_{11} \\ 0 \end{bmatrix}. \tag{96}$$

The first step involves a QR factorization:

$$\left[\begin{array}{c} T_{11}Z_{11} \\ \hline G_1 \end{array}\right] = Q^* \left[\begin{array}{c} B \\ \hline 0 \end{array}\right] = \left[Q_1^* \mid Q_2^*\right] \left[\begin{array}{c} B \\ \hline 0 \end{array}\right] = Q_1^* B. \tag{97}$$

From (96) and (97), we obtain

$$\left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline 0 & R_{22} \end{array}\right] \left[\begin{array}{c} B \\ \hline 0 \end{array}\right] = \left[\begin{array}{c} T_{11} \\ \hline 0 \end{array}\right] \Rightarrow R_{11}B = T_{11} \Rightarrow R_{11} = T_{11}B^{-1}. \tag{98}$$

Substituting for H in (94) we obtain,

$$R^* \left[\begin{array}{cc} T_{11}^{-1} & \\ & \Sigma \end{array}\right] R = Q \left[\begin{array}{cc} T_{11}^{-1} & \\ & \Sigma \end{array}\right] Q^*$$

$$\left[\begin{array}{c|c} R_{11}^* T_{11}^{-1} R_{11} & R_{11}^* T_{11}^{-1} R_{12} \\ \hline R_{12}^* T_{11}^{-1} R_{11} & R_{12}^* T_{11}^{-1} R_{12} + R_{22}^* \Sigma R_{22} \end{array}\right] = \left[\begin{array}{c} Q_1 \\ \hline Q_2 \end{array}\right] \left[\begin{array}{cc} T_{11}^{-1} & \\ & \Sigma \end{array}\right] \left[Q_1^* \mid Q_2^*\right]. \tag{99}$$

Equating the (1, 2) position in the above matrix equation after some simplification we obtain

$$R_{12}^* = Q_2 \left[\begin{array}{c} Z_{11} \\ \Sigma G_1 \end{array}\right]. \tag{100}$$

Equating the (2, 2) position in (99) and rearranging the terms, we have

$$R_{22}^* \Sigma R_{22} = Q_2 \left[\begin{array}{cc} T_{11}^{-1} & \\ & \Sigma \end{array}\right] Q_2^* - R_{12}^* T_{11}^{-1} R_{12}. \tag{101}$$

The matrix H is then computed as a product of R and Q.

This algorithm is of course only conceptual. It does not describe how to track the condition number of T_{11}. For this we refer to techniques as those described in [4, 12, 18]. If no look ahead is necessary, then the blocking scheme discussed in Section 2.5 can be used to compute H. If a look ahead of size km is required, then H can be computed as shown in Lemma 5. It should be pointed out that when T_{11} is well conditioned then the transformation H and its construction should give no numerical problems.

3.3.3. Comparison of the two algorithms. In this section we compare the two look-ahead algorithms from a computational and numerical stand point. Consider a block Toeplitz matrix with a block size of m. Further, let us consider a look-ahead step size of km at some stage of the Schur

algorithm. Let the size of the Schur complement following the look-ahead step be $lm \times lm$.

In Algorithm 1, the Bunch–Kaufman pivoting strategy would have to be applied to obtain the generator for the Schur complement. In the worst case, we would have $2m$ steps with each step requiring two rows of ΔT_{sc} be computed and contributing a 1×1 pivot. This would mean that a total of $4m$ reduction operations, each of length lm are done throughout the algorithm. Computing one row of ΔT_{sc} for example, say the ith row, is done as $\widehat{g}_i^T W \widehat{G}$. It can be seen that computing one row of ΔT_{sc} costs

$$\text{total flops } = 8m^2 + 4m^2k^2 + 4m^2(k+1)l. \tag{102}$$

As mentioned earlier, in the worst case there are $2m$ steps requiring two rows at each step. Also, at each step the rows computed need to be updated with the factorization computed till the previous step. At the jth step this requires $2(j-1)lm$ operations. Hence, the total cost of the entire algorithm is

$$\begin{aligned} &= 2m\ 2\ (8m^2 + 4m^2k^2 + 4m^2(k+1)l) + 2m \sum_{j=1}^{2m} 2\ 2(j-1)lm \\ &= 16m^4l + 8m^3l + 16m^3kl + 16m^3k^2 + 32m^3. \end{aligned} \tag{103}$$

In comparison, if we use Algorithm 2, the computation of the matrix H (described in (93 through 101)) requires a QR factorization of the matrix

$$\begin{bmatrix} T_{11}Z_{11} \\ G_1 \end{bmatrix}, \tag{104}$$

which has a dimension of $m(k+2) \times km$. The cost of QR factorization of an $M \times N$ matrix is $4M^2N - 2MN^2 + 2N^3/3$. For the matrix in (104) the computational cost would be

$$\begin{aligned} &= 4m^2(k+2)^2mk - 2(mk)^2(k+2)m + (2/3)(mk)^3 \\ &= 2.67m^3k^3 + 12m^3k^2 + 16m^3k. \end{aligned} \tag{105}$$

We then have to compute R_{12} from (100). The total number of operations to compute R_{12} is

$$8m^3k + 16m^3 + 2m^2k. \tag{106}$$

If we assume $R_{22} = I$, then the number of operations required to compute Σ from (101) is

$$= 2m^3k^2 + 16m^3k + 16m^3 + 8m^2. \tag{107}$$

The cost of applying H to the generator of size $m(k+2) \times lm$ is just the cost of applying Q_2 to the generator. This cost is

$$= 4m^3kl + 8m^3l. \tag{108}$$

The total cost of this method is found by adding (105, 106, 107, 108) together. This gives us

$$= 4m^3kl + 8m^3l + 2.67m^3k^3 + 14m^3k^2 + 40m^3k + 2m^2k + 32m^3 + 8m^2. \tag{109}$$

Comparing (109) and (103) and factoring the common multipliers we have

$$\begin{aligned} 2m^3k^2 + 12m^3kl + 16m^4l \quad &\text{vs.} \quad 2.67m^3k^3 + 40m^3k + 2m^2k + 8m^2 \\ k^2 + 6kl + 8ml \quad &\text{vs.} \quad 1.33k^3 + 20k + \frac{k}{m} + \frac{4}{m}. \end{aligned} \tag{110}$$

Consider an example where $m = 4$ and $l = 100$. It can be seen from (110) that for look-ahead step sizes greater than 24 Algorithm 1 is less expensive than Algorithm 2.

Hence for small block sizes, if the look-ahead step size is large, the Bunch–Kaufman-based look-ahead algorithm is faster than the one without pivoting. Note that in this calculation the cost of the reduction operation has not figured in. The results are not very different for serial machines. For parallel machines, the reduction operations give rise to several synchronization points but the reduction is done in parallel. For Algorithm 2 the computation of H is a serial bottleneck. It is possible on some parallel machines that Algorithm 1 will have a wider range of applicability than on a sequential machine. From this it is clear that the two algorithms have distinct ranges of applicability. The performance implications of these algorithms on serial and parallel machines is currently being investigated.

It has been shown that in certain pathological cases, the Bunch–Kaufman algorithm may not be able to detect, accurately, the rank of a low rank matrix [31]. Algorithm 1 relies on obtaining a rank 2 factorization of the displacement of the Schur complement after each look-ahead step. The process is stopped after one or two steps of the Bunch–Kaufman algorithm. In such cases, Algorithm 1 would produce only an approximate factorization and the exact solution would have to be obtained using iterative refinement. Iterative refinement may also be used to improve the accuracy of the solution in the second look-ahead algorithm, if the solution is inaccurate.

4. *QR* FACTORIZATION OF BLOCK TOEPLITZ MATRICES

The Schur algorithm can be generalized to obtain the QR factorization of block Toeplitz matrices due to the low displacement rank of the matrix $T^T T$. This generalized Schur algorithm has been outlined in [5] for scalar Toeplitz matrices and can be trivially extended to block Toeplitz matrices. In this section we present a modification of the generalized Schur algorithm for rank deficient Toeplitz matrices. It is shown that for exactly rank deficient block Toeplitz matrices, in the event of a degeneracy, the rank of the generator matrix can be dropped by 2. This reduces the complexity of the generalized Schur algorithm. For numerically rank deficient block Toeplitz matrices this algorithm yields a low-rank approximation.

The generalized Schur algorithm described in [5] was applied to block Toeplitz systems with full column rank. In several applications in signal and image processing the Toeplitz systems are related to rank deficient least-squares problems and hence regularization has to be applied to yield an acceptable solution.

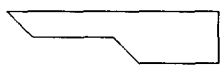
A standard approach would be to apply Tikhonov regularization which still yields a matrix in the same class of matrices. If T is a $N \times N$ Hermitian (and semidefinite) Toeplitz matrix, then both T and $T + \alpha I$ are Toeplitz Hermitian and hence of displacement rank 2. Similarly, if T is a general $M \times N$ Toeplitz matrix, then both T^*T and $T^*T + \alpha I$ have a displacement rank of at most 4. The complexity of this approach would thus be that of a Toeplitz solver, i.e., $O(N^2)$.

For some applications, T is a large matrix, and its rank r is small compared to the dimensions of T ($r \ll \min\{M, N\}$). This fact is not exploited in the standard approach because the regularized problems yield full-rank matrices. One would expect that the Toeplitz algorithms should only require $O(Nr)$ operations instead since the Cholesky decomposition of a low-rank semidefinite matrix A is

$$A = U_r^* U_r,$$

where U_r in a $r \times N$ "upper-triangular" matrix or rank r. Depending of the given matrix, the rank profile of U_r will be of the type

or



Matrices of displacement rank 2 are always of the first type, whereas matrices of displacement rank 4 can be of both types.

Consider a column rank deficient point Toeplitz matrix T of size $M \times N$. Let the matrix have l consecutive linearly dependent columns $T(:,k),\ldots,$ $T(:,k+l-1)$. We show that in this case a very particular property holds in the generator obtained at the start of the kth step of the generalized Schur algorithm. As seen in [5], the generator for the matrix T^TT is of the form

$$\Delta T^TT = \left[G_1^T \mid G_2^T \mid G_3^T \mid G_4^T\right] \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & -I & 0 \\ 0 & 0 & 0 & -I \end{bmatrix} \left[\begin{array}{c} G_1 \\ \hline G_2 \\ \hline G_3 \\ \hline G_4 \end{array}\right], \tag{111}$$

where $G_i, i = 1,\ldots,4$ is of size $1 \times N$. Let us denote the generator of T^TT, at the ith step of the generalized Schur algorithm by

$$G^{(i)} = \begin{bmatrix} G_1^{(i)} \\ G_2^{(i)} \\ G_3^{(i)} \\ G_4^{(i)} \end{bmatrix}, \tag{112}$$

where $G^{(i)}$ is of size $4 \times (N-i+1)$. Since the matrix T^TT is positive semidefinite and since the matrix T has l linearly dependent columns $k,\ldots,$ $k+l-1$, the Schur complement of T^TT w.r.t. the $(k-1)$th principal minor has the form

$$T_{\text{sc}}^{(k-1)} = \left[\begin{array}{c|c} 0_{l,l} & 0_{l,(N-k+1-l)} \\ \hline 0_{(N-k+1-l),l} & X \end{array}\right], \tag{113}$$

where X is a $(N-k-l) \times (N-k-l)$ matrix with nonzero entries. The displacement of the Schur complement, $T_{\text{sc}}^{(k-1)}$ also has the same sparsity pattern. The generator at the start of kth step of the generalized Schur algorithm is (the superscript indicating the kth step has been dropped for convenience)

$$G = \left[\begin{array}{c|ccc} g_{11} & g_{12} & \cdots & g_{1(N-k+1)} \\ g_{21} & g_{22} & \cdots & g_{2(N-k+1)} \\ g_{31} & g_{32} & \cdots & g_{3(N-k+1)} \\ g_{41} & g_{42} & \cdots & g_{4(N-k+1)} \end{array}\right]. \tag{114}$$

Instead of applying a hyperbolic Householder transform to zero out the first column using g_{11}, we first apply two orthogonal transforms to zero

out g_{21} and g_{41} using g_{11} and g_{31} respectively. Let Q_1 and Q_3 be those transforms. The sparsity pattern of the generator will then be as shown:

$$\begin{bmatrix} Q_1^T & \\ & Q_3^T \end{bmatrix} G = \left[\begin{array}{c|cccc} \widehat{g}_{11} & \cdots & \widehat{g}_{1l} & \widehat{g}_{1(l+1)} & \cdots & \widehat{g}_{1(N-k+1)} \\ 0 & \cdots & 0 & \widehat{g}_{2(l+1)} & \cdots & \widehat{g}_{2(N-k+1)} \\ \widehat{g}_{31} & \cdots & \widehat{g}_{3l} & \widehat{g}_{3(l+1)} & \cdots & \widehat{g}_{3(N-k+1)} \\ 0 & \cdots & 0 & \widehat{g}_{4(l+1)} & \cdots & \widehat{g}_{4(N-k+1)} \end{array}\right]. \tag{115}$$

Since the $(1,1)$ element of the Schur complement is 0, we have

$$\widehat{g}_{11}^2 - \widehat{g}_{31}^2 = 0 \tag{116}$$

and since the first row of the displacement of the Schur complement is zero we have

$$\widehat{g}_{11}[\widehat{g}_{11} \quad \widehat{g}_{12} \quad \cdots \quad \widehat{g}_{1(N-k+1)}] - \widehat{g}_{31}[\widehat{g}_{31} \quad \widehat{g}_{32} \quad \cdots \quad \widehat{g}_{3(N-k+1)}] = 0. \tag{117}$$

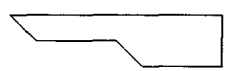
The above equations yield

$$[\widehat{g}_{11} \quad \widehat{g}_{12} \quad \cdots \quad \widehat{g}_{1(N-k+1)}] = [\widehat{g}_{31} \quad \widehat{g}_{32} \quad \cdots \quad \widehat{g}_{3(N-k+1)}]. \tag{118}$$

Hence, the first and third rows of the generator shown in (115) can be dropped. Also, since the first l columns of the reduced generator are "zero" we can skip the next l steps of the generalized Schur algorithm. The generator at the start of the $(k+l)$th step of the generalized Schur algorithm is

$$\begin{bmatrix} \widehat{g}_{2(l+1)} & \cdots & \widehat{g}_{2(N-k+1)} \\ \widehat{g}_{4(l+1)} & \cdots & \widehat{g}_{4(N-k+1)} \end{bmatrix}. \tag{119}$$

Since the matrix T^TT is a positive semidefinite matrix, if the pivot column of the generator has a zero hyperbolic norm, then the $(1,1)$ element of the displacement of the Schur complement will be zero and the entire row will also be zero. A detection of a zero hyperbolic norm of the pivot column of the generator is therefore sufficient to drop the rank of the generator. The next time the pivot column of the generator has zero hyperbolic norm, the rank of the generator again drops by two causing the Schur algorithm to terminate with an upper triangular factor of the form



This reduction in the generator size and rank avoids breakdowns. The algorithm has as many steps as the number of linearly independent columns in T. The complexity of the algorithm therefore is $O(Nr)$ (where r is the column rank of T) as opposed to $O(N^2)$ for matrices with full column rank.

If the matrix T has columns that are nearly linearly dependent on the other columns, i.e., it is nearly rank deficient, then the hyperbolic norm of the pivot column of the generator at those steps will be nonzero. In this case a simple thresholding mechanism applied to the above algorithm can be used to obtain an approximate low-rank decomposition of the matrix $T^T T$

$$A + \delta A = U_r^* U_r. \tag{120}$$

In case one is solving least-squares problems, it is easy also to use the obtained decomposition to perform a few steps of iterative refinement on the seminormal equations.

If the matrix T is a block Toeplitz matrix of block size m, then the generator at the start of the generalized Schur algorithm has rank $4m$. Again, if the hyperbolic norm of the generator is zero, then the Schur complement will have a leading "zero." Also since the matrix $T^T T$ is semidefinite, the entire row of the displacement of the Schur complement will be zero and the rank of the generator can be dropped by two by dropping the two identical rows with opposite signatures.

The algorithm proposed in this section is a significant simplification over a similar approach proposed in [20], which uses the Levinson algorithm with look ahead. We include an example to illustrate the above algorithm. Consider a Toeplitz matrix T

$$T = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 & 2 & 2 & 3 \\ 6 & 5 & 4 & 3 & 2 & 1 & 2 & 2 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 2 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 \\ 11 & 10 & 9 & 8 & 7 & 6 & 5 & 4 \\ 12 & 11 & 10 & 9 & 8 & 7 & 6 & 5 \\ 13 & 12 & 11 & 10 & 9 & 8 & 7 & 6 \\ 14 & 13 & 12 & 11 & 10 & 9 & 8 & 7 \\ 15 & 14 & 13 & 12 & 11 & 10 & 9 & 8 \end{bmatrix}.$$

Columns 3, 4, and 5 are linearly dependent of the first two columns, while 6, 7, and 8 are again linearly independent. The (generalized) Schur algorithm uses the following generator for the matrix $A = T^*T$:

$$G_{(0)}(:, 1:4) = \begin{bmatrix} 34.7851 & 31.6228 & 28.4605 & 25.2982 \\ 0 & 4.0000 & 3.0000 & 2.0000 \\ 0 & 31.6228 & 28.4605 & 25.2982 \\ 0 & 15.000 & 14.000 & 13.000 \end{bmatrix}$$

$$G_{(0)}(:, 5:8) = \begin{bmatrix} 22.1359 & 19.2611 & 16.5876 & 14.2877 \\ 1.0000 & 2.0000 & 2.0000 & 3.0000 \\ 22.1359 & 19.2611 & 16.5876 & 14.2877 \\ 12.000 & 11.000 & 10.000 & 9.0000 \end{bmatrix}.$$

Two steps of the (generalized) Schur algorithm generate the first two rows of the upper triangular factor of T^*T. At the beginning of the third step the first column of the generator has a Σ-norm equal to zero

$$G_{(2)} = \begin{bmatrix} 1.0000 & 2.0000 & 3.0000 & 4.0000 & 3.9091 & 3.4545 \\ -0.7583 & -1.5166 & -2.2749 & -0.9113 & -0.5391 & 0.9020 \\ -1.2515 & -2.5030 & -3.7545 & -3.7545 & -3.4860 & -2.2985 \\ -0.0936 & -0.1873 & -0.2809 & 0.0827 & 0.4783 & 1.1324 \end{bmatrix}.$$

We then use Householder transformations to eliminate $G_{(2)}(2,1)$ using $G_{(2)}(1,1)$ and $G_{(2)}(4,1)$ using $G_{(2)}(3,1)$. This gives us the generator

$$\widetilde{G}_{(2)} = \begin{bmatrix} -1.2550 & -2.5100 & -3.7650 & -3.7379 & -3.4406 & -2.2076 \\ 0 & 0 & 0 & 1.6908 & 1.9324 & 2.8060 \\ -1.2550 & -2.5100 & -3.7650 & -3.7379 & -3.4406 & -2.2076 \\ 0 & 0 & 0 & -0.3626 & -0.7370 & -1.3008 \end{bmatrix}.$$

Since the first and third rows of the generator are equal and have signatures of opposite signs, they can be removed and the generator for the next step will have only two columns. Also, it can be seen that the first three columns of this generator are zeros and this means that we can skip the corresponding rows in the upper triangular factor U_r. The next step would use the generator

$$G_{(5)} = \begin{bmatrix} 1.6908 & 1.9324 & 2.8060 \\ -0.3626 & -0.7370 & -1.3008 \end{bmatrix},$$

and the factorization process continues. This finally yields the triangular factor U_r as

$$U_r(:, 1:4) = \begin{bmatrix} -34.785 & -31.623 & -28.461 & -25.298 \\ & 1.000 & 2.000 & 3.000 \\ & & & \\ & & & \end{bmatrix}$$

$$U_r(:, 5:8) = \begin{bmatrix} -22.136 & -19.261 & -16.588 & -14.288 \\ 4.000 & 3.909 & 3.455 & 2.182 \\ & -1.651 & -1.817 & -2.587 \\ & & 1.618 & 1.708 \\ & & & -1.578 \end{bmatrix}.$$

The backward error δA of the matrix $A = T^*T$ defined as

$$\|\delta A\| = \frac{\|A - U_r^* U_r\|}{\|A\|}$$

is 3.57×10^{-15}, which is of the order of the machine precision ($\epsilon \approx 2.22 \times 10^{-16}$). This shows the good numerical behavior of the regularization algorithm.

The numerical behavior of this algorithm was good because the above example was exactly rank deficient. If there is a sharp drop in the singular values of the matrix, this algorithm will yield accurate results. However, if there is no sharp drop, then this algorithm may produce an inaccurate factorization due to the sensitivity of Schur complements [31]. The algorithm discussed in the next section addresses this issue.

5. CONVERSION TO CAUCHY TYPE MATRICES

In Section 4, we discussed modifications to the QR factorization algorithm for block Toeplitz matrices proposed by Chun et al. [5]. The modified algorithm could be used to obtain the QR factorization of an exactly rank deficient block Toeplitz matrix. If the Toeplitz matrix happened to be numerically rank deficient, then only a low approximation of the block Toeplitz matrix could be obtained. This was because any form of pivoting applied to the generalized Schur algorithm would destroy the displacement structure of the block Toeplitz matrix.

In [15, 21] it was shown that if Toeplitz matrices were converted to Cauchy type matrices, then the factorization of such matrices could be carried out with pivoting. The drawbacks of the algorithms proposed in [15, 21] were that complex-valued FFTs were used to convert a real valued

Toeplitz matrix into a complex-valued Cauchy type matrix. The algorithms were not able to exploit any symmetry in the Toeplitz or quasi-Toeplitz matrix to reduce the computational complexity.

In this section we present a modification to the algorithms in [15, 21] to factor a symmetric semidefinite quasi-Toeplitz matrix using only real arithmetic and exploiting the symmetric property of the matrix. This algorithm can be used to obtain a rank revealing factorization of the matrix T^TT, where T is a rank deficient Toeplitz matrix. Rank deficient Toeplitz matrices arise in image reconstruction and system identification problems.

In [20] Hansen and Gesmar present a look-ahead-like algorithm for fast orthogonalization of rank deficient Toeplitz matrices and in [11] Eldén and Park present a modification to the algorithm proposed in [5], where they delay the application of the ill-conditioned skew hyperbolic transforms to obtain an approximate factorization. Both algorithms do not involve any pivoting since they deal with Toeplitz matrices only. The algorithm presented in this section does not have this limitation due to the conversion to Cauchy type matrices.

5.1. Rank Factorization of Positive Semidefinite Quasi-Toeplitz Matrices

Consider a symmetric positive semidefinite quasi-Toeplitz matrix T of size $N \times N$. Let the displacement equation of this matrix be given as

$$T - ZTZ^T = \widehat{G}\Sigma\widehat{G}^T, \tag{121}$$

where Z is a circulant matrix of size $N \times N$

$$Z = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \ddots & \ddots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}. \tag{122}$$

Note that the matrix Z in (121) is a circulant matrix and not a lower shift as used in Section 4.

A Cauchy type matrix can be defined as any matrix that has the following displacement structure

$$\begin{aligned} & D_fC - CD_b = G_1G_2^T \\ \text{or} \quad & C - D_fCD_b = G_1G_2^T, \end{aligned} \tag{123}$$

where D_f and D_b are diagonal matrices. It was shown in [15, 21] that if (121) is converted to (123) using the discrete Fourier transform, then Gaussian elimination with partial pivoting can be applied to obtain a factorization. The problem with this method is that the Fourier transform converts the real-valued Toeplitz matrix into a complex-valued Cauchy type matrix and increases the complexity of the algorithm.

If we can obtain a real-valued transform that block diagonalizes the circulant matrix, then applying that transform to (121) would convert it to (123), where D_f and D_b are real-valued block diagonal matrices. A *Hartley* transform H of size N converts the circulant matrix Z shown above into a matrix with an X-shaped non zero structure. A permutation P can then be applied to obtain a block diagonal matrix with 1×1 or 2×2 blocks. Applying the transformation $\widehat{H} = PH$ of the appropriate size to (121) yields

$$\begin{aligned} \widehat{H}T\widehat{H}^T - (\widehat{H}Z\widehat{H}^T)(\widehat{H}T\widehat{H}^T)(\widehat{H}Z^T\widehat{H}^T) &= \widehat{H}\widehat{G}\Sigma\widehat{G}^T\widehat{H}^T \\ C - ACA^T &= G\Sigma G^T, \end{aligned} \tag{124}$$

where $C = \widehat{H}T\widehat{H}^T$ is a Cauchy type matrix and $A = \widehat{H}Z\widehat{H}^T$ is a block diagonal matrix with 1×1 and 2×2 blocks. If we obtain a factorization $C = LDL^T$ of the Cauchy type matrix, then the corresponding factorization of the quasi-Toeplitz matrix T will be $T = \widehat{H}^T LDL^T \widehat{H}$. The next step in this algorithm is obtaining a factorization of the form LDL^T of the Cauchy type matrix C in (124).

It must be noted that the Cauchy type matrix C is not explicitly computed but is implicitly available from the matrices A, G, and Σ. Reconstructing any column of the Cauchy matrix from (124) would require solving the Lyapunov equations. Let the columns of the matrix C and G^T be partitioned conformally with the block structure of A. The ith column block of the matrix C, denoted by c_i, satisfies the equation

$$c_i - Ac_i a_i^T = G\Sigma g_i^T, \tag{125}$$

where a_i is the ith diagonal block of the matrix A and g_i is the corresponding block row of the generator G. The above equation can then be written as

$$A^T c_i - c_i a_i^T = A^T G\Sigma g_i^T. \tag{126}$$

Any block row of c_i given by c_{ji} then satisfies the Lyapunov equation

$$a_j^T c_{ji} - c_{ji} a_i^T = a_j^T g_j \Sigma g_i^T. \tag{127}$$

If $j \neq i$, then the matrices a_j and a_i have different eigenvalues and the Lyapunov equation can be solved for c_{ij}. For $j = i$, the Lyapunov equation cannot be solved. The diagonal blocks of C, therefore, cannot be computed from A, G and Σ. We, therefore, need to precompute the diagonal blocks of C from the original quasi-Toeplitz matrix.

Having outlined the method to compute any column and the diagonal block of the Cauchy type matrix, we now proceed to describe the algorithm to obtain the factorization of C.

Let the block diagonal of the Cauchy matrix C be denoted by D. Since the matrix T is positive semidefinite, it can be argued that searching for the diagonal block d_i with the highest determinant is sufficient to locate a pivot block. Let P_1 be the permutation matrix to get the diagonal block d_i to the pivot position d_1. Also, let $P_1 C P_1^T$ be partitioned as

$$P_1 C P_1^T = \left[\begin{array}{c|c} d_1 & l_1^T \\ \hline l_1 & C_1 \end{array}\right]. \tag{128}$$

Let us define the matrix X,

$$X = \left[\begin{array}{c|c} I & 0 \\ \hline -l_1 d_1^{-1} & I \end{array}\right]; \tag{129}$$

then applying $X\,(\cdot)\,X^T$ to

$$\begin{aligned} P_1 C P_1^T - (P_1 A P_1^T)(P_1 C P_1^T)(P_1 A^T P_1^T) &= P_1 G \Sigma G^T P_1^T \\ \widehat{C} - \widehat{A}\widehat{C}\widehat{A}^T &= \widehat{G}\Sigma\widehat{G}^T \end{aligned} \tag{130}$$

yields

$$\left[\begin{array}{c|c} d_1 & 0 \\ \hline 0 & C_{\mathrm{sc}} \end{array}\right] - \left[\begin{array}{c|c} A_{11} & 0 \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c|c} d_1 & 0 \\ \hline 0 & C_{\mathrm{sc}} \end{array}\right] - \left[\begin{array}{c|c} A_{11}^T & A_{21}^T \\ \hline 0 & A_{22}^T \end{array}\right] = X\widehat{G}\Sigma\widehat{G}^T X^T \tag{131}$$

If we obtain a generator for C_{sc} that satisfies the displacement equation of the form

$$C_{\mathrm{sc}} - A_{22} C_{\mathrm{sc}} A_{22}^T = G_{\mathrm{sc}} \Sigma_{\mathrm{sc}} G_{\mathrm{sc}}^T, \tag{132}$$

then we would have finished the first step of the factorization algorithm. It can be seen that the above equation is identical in form to (76) and hence the procedure developed in Section 3.3.2 can be used to obtain the generator of C_{sc}.

Alternately, another technique to update the generators can be used. Partitioning $\widehat{G}^T$ conformally as $\widehat{G}^T = [\widehat{G}_1^T \;\; \widehat{G}_2^T]$ and equating the $(2,2)$

position in (131) we have

$$C_{\rm sc} - A_{22}C_{\rm sc}A_{22}^T = \big(\widehat{G}_2 - l_1 d_1^{-1}\widehat{G}_1\big)\Sigma\big(\widehat{G}_2 - l_1 d_1^{-1}\widehat{G}_1\big)^T + A_{21}d_1A_{21}^T, \quad (133)$$

where $A_{21} = A_{22}l_1d_1^{-1} - l_1d_1^{-1}A_{11}$. The last term of (133), $A_{21}d_1A_{21}^T$, can be expanded as

$$\begin{aligned} &= \big(A_{22}l_1d_1^{-1} - l_1d_1^{-1}A_{11}\big)d_1\big(d_1^{-1}l_1^TA_{22}^T - A_{11}^Td_1^{-1}l_1^T\big) \\ &= \big(A_{22}l_1A_{11}^T - l_1d_1^{-1}A_{11}d_1A_{11}^T\big)A_{11}^{-T}d_1^{-1}A_{11}^{-1} \\ &\quad \times \big(A_{11}l_1^TA_{22}^T - A_{11}d_1A_{11}^Td_1^{-1}l_1^T\big). \end{aligned} \quad (134)$$

Equating the $(1,1)$, $(1,2)$, and $(2,1)$ positions of (130) we have

$$d_1 - A_{11}d_1A_{11}^T = \widehat{G}_1\Sigma\widehat{G}_1^T \quad (135)$$

$$l_1 - A_{22}l_1A_{11}^T = \widehat{G}_2\Sigma\widehat{G}_1^T. \quad (136)$$

Inserting the above equations in (134) yields

$$A_{21}d_1A_{21}^T = \big(\widehat{G}_2 - l_1d_1^{-1}\widehat{G}_1\big)\Sigma\widehat{G}_1^TA_{11}^{-T}d_1^{-1}A_{11}^{-1}\widehat{G}_1\Sigma\big(\widehat{G}_2 - l_1d_1^{-1}\widehat{G}_1\big)^T. \quad (137)$$

Substituting (137) in (133), $\Delta C_{\rm sc} = C_{\rm sc} - A_{22}C_{\rm sc}A_{22}^T$ has the form

$$C_{\rm sc} = \big(\widehat{G}_2 - l_1d_1^{-1}\widehat{G}_1\big)\big(\Sigma + \Sigma\widehat{G}_1^TA_{11}^{-T}d_1^{-1}A_{11}^{-1}\widehat{G}_1\Sigma\big)\big(\widehat{G}_2 - l_1d_1^{-1}\widehat{G}_1\big)^T. \quad (138)$$

Using the Sherman–Morrison–Woodbury formula and (135) it can be shown that

$$\big(\Sigma + \Sigma\widehat{G}_1^TA_{11}^{-T}d_1^{-1}A_{11}^{-1}\widehat{G}_1\Sigma\big) = \big(\Sigma^{-1} - \widehat{G}_1^Td_1^{-1}\widehat{G}_1\big)^{-1}. \quad (139)$$

Hence, the update equations for the generator and the signature matrices are

$$G_{\rm sc} = \widehat{G}_2 - l_1d_1^{-1}\widehat{G}_1 \quad (140)$$

$$\Sigma_{\rm sc}^{-1} = \Sigma^{-1} - \widehat{G}_1^Td_1^{-1}\widehat{G}_1. \quad (141)$$

Having obtained the generator for the next step, we update the diagonal matrix D as

$$D_{\rm next} = D - l_1D_1^{-1}l_1^T. \quad (142)$$

This defines all the information to proceed with the next step of the factorization. Carrying the factorization to completion in a similar manner, one obtains a factorization of C of the form LDL^T and a factorization of T for the form $\widehat{H}^TLDL^T\widehat{H}^T$.

5.2. *Extensions of the Algorithm*

The above algorithm extends very easily to block Toeplitz matrices because the block circulant matrix Z can also be diagonalized by a permuted version of the Hartley transform.

The pivoting strategy used in the above algorithm was block diagonal pivoting. In some cases this may not be sufficient. If at any stage of the algorithm all the diagonal pivots of the Schur complement are ill conditioned, it is possible to revert to the complex arithmetic version without too much overhead and continue the factorization.

6. CONCLUSIONS

In this paper we have presented several high performance variants of the Schur algorithm to solve block Toeplitz matrices. Based on the existing Schur type algorithms and the algorithms discussed in this paper a high performance library is currently being developed. In the past there have been efforts to develop libraries for point Toeplitz matrices [1, 19] on serial machines using the Levinson algorithm. The proposed library can be used to solve point and block Toeplitz matrices on parallel machines. On parallel machines, the Levinson algorithm suffers from reduced parallelism. The Schur algorithm-based library will be developed for distributed memory machines such as the Cray T3D and shared memory/vector-pipeline machines such as the Cray C90. A detailed performance analysis of the algorithms on the various high performance architectures will impact the implementation choices.

The authors thank Michael Stewart for some useful suggestions on Section 3.2 of the paper. The authors also thank Cray Research Inc. for summer support provided to Srikanth Thirumalai. The work of the first three authors is supported by the National Science Foundation under Grants NSF CCR-9120105 and CCR-9209349, and by ARPA under a subcontract of Grant No. ARPA/NIST 60NANB2D1272. V. Vermaut is supported by an FDS-94 grant of the Université Catholique de Louvain.

REFERENCES

1 O. B. Arushanian, M. K. Samarin, V. V. Voevoedin, E. E. Tyrtyshnikov, B. S. Garbow, J. M. Boyle, W. R. Cowell, and K. W. Dritz, The Toeplitz Package Users' Guide, Technical report, Argonne National Laboratory, 1983.

2 C. Bischof and C. Van Loan, The WY representation for products of Householder matrices, *SIAM J. Sci. Stat. Comput.* 8:s2–s13 (1987).

3 S. Cabay and R. Meleshko, A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices, *SIAM J. Matrix Anal. Appl.* 14:735–765 (1993).
4 T. F. Chan and P. C. Hansen, A look-ahead Levinson algorithm for indefinite Toeplitz systems, *SIAM J. Matrix Anal. Appl.* 13:490–506 (1992).
5 J. Chun, T. Kailath, and H. Lev-Ari, Fast parallel algorithms for QR and triangular factorization, *SIAM J. Sci. Stat. Comput.* 8:899–913 (1987).
6 J. Chun and T. Kailath, Generalized Displacement Structure for Block Toeplitz, Toeplitz Block and Toeplitz Derived Matrices, Informations Systems Lab., Stanford University, CA, 1988.
7 P. Concus and P. Saylor, A modified direct preconditioner for indefinite symmetric Toeplitz systems, *Linear Algebra Appl.*, to appear.
8 G. Cybenko and M. Berry, Hyperbolic Householder algorithms for factoring structured matrices, *SIAM J. Matrix Anal. Appl.* 11:499–520 (1990).
9 J.-M. Delosme, I. C. F. Ipsen, and C. C. Paige, The Cholesky Factorization, Schur Complements, Correlation Coefficients, Angles between Vectors, and the QR Factorization, Technical report, Yale University, 1988.
10 Ph. Delsarte, Y. Genin, and Y. Kamp, Pseudo-Carathéodory functions and Hermitian Toeplitz matrices, *Philips J. Res.* 41:1–54 (1986).
11 L. Eldén and H. Park, Accurate Least Squares Solutions for Toeplitz Matrices, Technical report, Linkoping University, Sweden, 1994.
12 R. W. Freund and H. Zha, Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems, *Linear Algebra Appl.* 188:255 (1993).
13 K. Gallivan, S. Thirumalai, and P. Van Dooren, On Solving Block Toeplitz Matrices Using a Block Schur Algorithm, Technical report, CSRD, University of Illinois at Urbana-Champaign, 1994 [a shorter version appears in *Proceedings, 1994 International Conference on Parallel Processing*, pp. 274–281].
14 K. Gallivan, S. Thirumalai, and P. Van Dooren, A block Toeplitz look-ahead Schur algorithm, in *SVD in Signal Processing III, Algorithms, Architectures and Applications* (M. Moonen and B. De Moor, Eds.), pp. 199–206, Elsevier, Amsterdam, 1995.
15 I. Gohberg, T. Kailath, and V. Olshevsky, Gaussian Elimination with Partial Pivoting for Structured Matrices, Technical report, Information Systems Lab., Stanford University, 1994.
16 Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, John Hopkins Univ. Press, 1989.
17 M. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, ii, *SIAM J. Matrix Anal. Appl.* 15:15–58 (1994).
18 M. Gutknecht and M. Hochbruck, Look-Ahead Levinson and Schur Algorithms for Non-Hermitian Toeplitz Systems, Technical report, IPS, ETH Zurich, 1994.
19 P. C. Hansen and T. F. Chan, FORTRAN subroutines for general Toeplitz systems, *ACM Trans. Math. Software* 18(3):256–273 (1992).
20 P. C. Hansen and H. Gesmar, Fast orthogonal decomposition of rank deficient Toeplitz matrices, *Num. Algorithms* 4:151–166 (1993).

21 G. Heinig, Inversion of generalized Cauchy matrices and other classes of structured matrices, in *Linear Algebra for Signal Processing*, The IMA Volumes in Mathematics and its Applications, Vol. 69, (A. Bojanczyk and G. Cybenko, Eds.), Springer-Verlag, 1995.
22 T. Kailath and A. Sayed, Fast algorithms for generalized displacement structures, in *Recent Advances in Mathematical Theory of Systems*, (H. Kimura and S. Kodoma, Eds.), pp. 27–32, Proc. MTNS-91, 1992.
23 T. Kailath, S.-Y. Kung, and M. Morf, Displacement ranks of matrices and linear equations, *J. Math. Appl.* 68:395–407 (1979).
24 D. Pal and T. Kailath, Fast triangular factorization and inversion of Hermitian, Toeplitz related matrices with arbitrary rank profile, *SIAM J. Matrix Anal. Appl.* 14(4):1016–1042 (1993).
25 B. Parlett, Reduction to tridiagonal form and minimal realizations, *SIAM J. Matrix Anal. Appl.* 13:567–593 (1992).
26 C. M. Rader and A. O. Steinhardt, Hyperbolic Householder transformations, *IEEE Trans. Acoust. Speech Signal Process.* 34:1589–1602 (1986).
27 C. M. Rader and A. O. Steinhardt, Hyperbolic Householder transforms, *SIAM J. Matrix Anal. Appl.* 9:269–290 (1988).
28 A. Sayed and T. Kailath, A look-ahead block Schur algorithm for Toeplitz-like matrices, *SIAM J. Matrix Anal. Appl.* 16(2) (1995).
29 R. Schreiber and C. Van Loan, A storage-efficient WY representation for products of Householder transformations, *SIAM J. Sci. Stat. Comput.* 10(1):53–57 (1989).
30 I. Schur, Uber potenzreihen die im Inneren des Einheitskreises beschrankt sind, *J. Reine Angew. Math.* 147:205–232 (1917).
31 M. Stewart and P. Van Dooren, Stability Issues in the Factorization of Structured Matrices, Technical report, CSRD, University of Illinois at Urbana-Champaign, 1994, Submitted for publication.
32 J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Oxford, England, 1965.

Received 9 December 1994; revised 27 June 1995