Factorizations and linear system solvers for matrices with Toeplitz structure *

Daniel Kressner† and Paul Van Dooren‡

June 2000

*This document presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET (contract number BRRT-CT97-5040) and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from `wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/SLWN2000-2.ps.Z`

†University of Chemnitz-Zwickau, Department of Mathematics, Germany

‡Université Catholique de Louvain, CESAME, B-1348 Louvain-la-Neuve, Belgium

Abstract

In this report we describe new routines for several factorizations of matrices with Toeplitz or block Toeplitz structure and show how this can be used to solve the corresponding systems of equations or least squares systems of equations. We also describe certain implementation details and show how to handle matrices of low rank or of low bandwidth.

Contents

1 Introduction

Matrices with special structure play a fundamental role in systems and control. In signal processing and statistics one typically encounters matrices with Toeplitz structure :

$$T = \begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{-(n-1)} & \dots & t_{-1} & t_0 \end{bmatrix}$$

where the t_i are scalars, or their block equivalent, where the t_i are then $k \times k$ matrices. In control, one often has to deal with Hankel matrices :

$$H = \begin{bmatrix} h_0 & h_1 & \dots & h_{n-1} \\ h_1 & .\cdot^{\cdot} & .\cdot^{\cdot} & \vdots \\ \vdots & .\cdot^{\cdot} & .\cdot^{\cdot} & \vdots \\ h_{n-1} & \dots & \dots & h_{2n-2} \end{bmatrix}$$

where the h_i are scalars, or their block equivalent, where the h_i are $k \times k$ matrices. In this report we describe numerical methods to handle Toeplitz matrices and block Toeplitz matrices that are real symmetric and positive definite or semi-definite. We also show how these methods extend to matrices that have a Toeplitz-like structure, which includes a certain class of Hankel matrices.

2 The Schur algorithm

One of the basic algorithms for symmetric Toeplitz matrices is the Schur algorithm, which has been shown to be backward stable in the case that the matrix is positive definite. We describe briefly this special case since it is the cornerstone for all variants described in this report. A more elaborate description of the algorithm can be found in [3]. We start with the matrix

$$\nabla T \doteq T - Z^T T Z, \quad Z \doteq \begin{bmatrix} 0 & 1 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 0 \end{bmatrix}.$$

The matrix ∇T is called the "displacement" of T and clearly has low rank since

$$\nabla T = \begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_1 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ t_{n-1} & 0 & \dots & 0 \end{bmatrix}. \tag{1}$$

From the form of this matrix it follows that it is indefinite and of rank 2. Therefore it can be factorized as follows :

$$\nabla T = G^T \Sigma G, \quad \Sigma \doteq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G \doteq \begin{bmatrix} x_0 & x_1 & \ldots & x_{n-1} \\ 0 & y_1 & \ldots & y_{n-1} \end{bmatrix}, \tag{2}$$

where $y_i = x_i = t_i/\sqrt{t_0}$. The matrix G is also called the *generator* of T and is related to T via the equation

$$T - Z^T T Z = G^T \Sigma G. \tag{3}$$

One shows that a sequence of $n-1$ transformations of this generator G yields the Cholesky factorization of T :

$$T = U^T \cdot U, \quad U \doteq \begin{bmatrix} u_{1,1} & u_{1,2} & \ldots & u_{n,n} \\ & u_{2,2} & \ldots & u_{2,n} \\ & & \ddots & \vdots \\ & & & u_{n,n} \end{bmatrix}. \tag{4}$$

The fact that element y_0 of G equals 0 and equations (3),(4) imply that the first row of U is given by the first row of G :

$$U(1,:) = G(1,:).$$

A recursive algorithm for the successive rows of U is then obtained if one can construct the generator $\hat{G}$ for the Schur complement of T :

$$T - U(1,:)^T U(1,:) = \hat{G}^T \Sigma \hat{G}. \tag{5}$$

It is shown in [3] that a simple shift operation with Z applied to the first row of G does the job :

$$\hat{G} \doteq \begin{bmatrix} 0 & \hat{x}_1 & \ldots & \hat{x}_{n-1} \\ 0 & \hat{y}_1 & \ldots & \hat{y}_{n-1} \end{bmatrix} \doteq \begin{bmatrix} 0 & x_0 & \ldots & x_{n-2} \\ 0 & y_1 & \ldots & y_{n-1} \end{bmatrix} = \begin{bmatrix} G(1,:)Z \\ G(2,:) \end{bmatrix}. \tag{6}$$

Eliminating the element $\hat{y}_1$ in $\hat{G}$ then yields the second row of U and so on. This is obtained by constructing a 2×2 transformation Θ such that

$$\Theta^T \Sigma \Theta = \Sigma, \quad \Theta \hat{G} = \tilde{G} \doteq \begin{bmatrix} 0 & \tilde{x}_1 & \tilde{x}_2 & \ldots & \tilde{x}_{n-1} \\ 0 & 0 & \tilde{y}_2 & \ldots & \tilde{y}_{n-1} \end{bmatrix}.$$

Such a transformation can always be found in the form

$$\Theta = \frac{1}{c} \begin{bmatrix} 1 & s \\ s & 1 \end{bmatrix}, \quad s^2 + c^2 = 1, \quad s = -\hat{y}_1/\hat{x}_1.$$

Moreover, it is shown in [1, 4] that the factorization

$$\Theta = \begin{bmatrix} 1 & 0 \\ s & c \end{bmatrix} \cdot \begin{bmatrix} 1/c & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix},$$

is a stable and economical way to implement the transformation Θ. This then yields the

Basic (symmetric) SCHUR algorithm

Initialize G with element $G(2,1)=0$
for $i=1,n-1$
$U(i,:)=G(1,:);$
$G(1,:)=G(1,:)*Z;$
$s=-G(2,i+1)/G(1,i+1); c=sqrt((1-s)*(1+s));$
$G(1,:)=(G(1,:)+s*G(2,:))/c;$
$G(2,:)=s*G(1,:)+c*G(2,:);$
end
$U(n,n)=G(1,n);$

This algorithm requires a total of $3n^2$ floating point operations for obtaining the factorization. We show in the next section how to obtain as a by-product of this factorization the generator of the inverse of T. This can then be exploited for computing the solution $T^{-1}b$ of the system $Tx=b$ via convolution in $n \log n$ operations rather than n^2 [3].

3 Variants of the Schur algorithm

A first remark is that for banded Toeplitz matrices the entries t_i are zero beyond a certain index $i>r$. In such case it is easy to see that the elements x_i and y_i for $i>r$ also equal zero, and it is not needed to carry these zeros in all computations. As a consequence the Cholesky factor will of course also be banded. The complexity of the algorithm then becomes of the order of rn and the generator will be of maximum length r at each step.

A second variant is that of low rank matrices. Suppose the matrix T is of rank r then it is known that for semi-definite Toeplitz matrices the Cholesky factor will be of the type

$$U \doteq \begin{bmatrix} u_{1,1} & \dots & \dots & u_{n,n} \\ & \ddots & & \vdots \\ & & u_{r,r} & u_{r,n} \end{bmatrix}. \tag{7}$$

One shows [2] in this case that the Schur algorithm terminates at stage $r<n:$ one encounters then a value $\hat{y}_r = \pm \hat{x}_r$, which implies that $|s|=1, c=0$, and hence that the transformation Θ does not exist anymore. The generator $\hat{G}$ at that stage satisfies $\hat{G}^T \Sigma \hat{G}=0$ and hence can be dismissed.

Another variant is that of the block Toeplitz case. Let $T \in \mathbb{R}^{nk,nk}$ be a positive definite Toeplitz matrix with blocks of size $k \times k$:

$$T = \begin{bmatrix} T_0 & T_1 & \dots & T_{n-1} \\ T_1^T & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_1 \\ T_{n-1}^T & \dots & T_1^T & T_0 \end{bmatrix}, \tag{8}$$

then essentially the same algorithm holds except that the shift operator $Z \in \mathbb{R}^{nk,nk}$ now becomes

$$Z = \begin{bmatrix} 0 & I_k & & \\ & \ddots & \ddots & \\ & & \ddots & I_k \\ & & & 0 \end{bmatrix}.$$

The displacement rank of ∇T can now be as large as $2k$ and we thus have a general factorization of the type

$$\nabla T = G^T \Sigma G, \quad \Sigma \doteq \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}, \tag{9}$$

where $p + q \leq 2k$. The banded case can be exploited in much the same way as for the scalar Toeplitz case but the low rank case is more complicated. The rank profile of a block Toeplitz matrix can be arbitrary and more care is needed to implement the steps implying a value $c = 0$. In the block case this implies that two rows of the current generator can be dismissed, but one has to continue the Schur algorithm with the remaining rows. We refer to [2] for more details.

4 Generators and displacement of Schur complements

So far we only explained how to use the Schur algorithm to obtain the Cholesky factor of T. But the same algorithm can as well be used to obtain the Cholesky factor of the inverse T^{-1}, to find the factors of a QR factorization of T or even the solution of a system of equations with Toeplitz structure. We explain here the basic ideas and apply them directly to the block Toeplitz case.

Let T be a positive definite block Toeplitz matrix, then consider the bordered matrices

$$M \doteq \begin{bmatrix} T & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix}, \quad Z_{ml} \doteq \begin{bmatrix} Z & 0 \\ 0 & Z_l \end{bmatrix}, \quad Z_{mr} \doteq \begin{bmatrix} Z & 0 \\ 0 & Z_r \end{bmatrix}.$$

(We assume in all generality that the matrix M is non-symmetric although T is symmetric.) One could apply the first steps of the Cholesky factorization corresponding to the leading block $T = U^T.U$ to obtain the decomposition

$$M = \begin{bmatrix} U^T \\ M_{2,1}U^{-1} \end{bmatrix} \begin{bmatrix} U & U^{-T}M_{1,2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & M_T \end{bmatrix}$$

where M_T is the Schur complement of M with respect to T :

$$M_T \doteq M_{2,2} - M_{2,1}T^{-1}M_{1,2}.$$

We now show how this can be done via the use of the Schur algorithm. It is easy to see that the displacement of M :

$$\nabla M \doteq M - Z_{ml}^T M Z_{mr}$$

has a (not necessarily symmetric) factorization

$$\nabla M = G_l^T \Sigma G_r$$

which can be partitioned as follows :

$$G_l \doteq \begin{bmatrix} G_{l1} \\ G_{l2} \end{bmatrix}, G_r \doteq \begin{bmatrix} G_{r1} \\ G_{r2} \end{bmatrix},$$

and that

$$\nabla T = G_{l1}^T \Sigma G_{r1}.$$

In other words, the displacement factorization of the submatrix T is embedded in that of M and so are the left and right generators. If we now run the (non-symmetric) Schur algorithm for the first steps corresponding to the leading matrix T, we will have computed the columns of

$$\begin{bmatrix} U^T \\ M_{2,1}U^{-1} \end{bmatrix}$$

and the rows of

$$\begin{bmatrix} U & U^{-T}M_{1,2} \end{bmatrix}.$$

Moreover, the resulting generators will be of the type :

$$\hat{G}_l \doteq \begin{bmatrix} 0 \\ \hat{G}_{l2} \end{bmatrix}, \hat{G}_r \doteq \begin{bmatrix} 0 \\ \hat{G}_{r2} \end{bmatrix},$$

and will correspond to the displacement of the Schur complement [3] :

$$\hat{G}_{l2}^T \Sigma \hat{G}_{r2} = \nabla M_T \doteq M_T - Z_l^T M_T Z_r.$$

If we apply this to the following matrix :

$$M = \begin{bmatrix} T & I \\ I & 0 \end{bmatrix},$$

with $Z_{ml} = Z_{mr} = Z \oplus Z$, then ∇M has rank $2k$ and possesses a factorization of the type

$$M - Z_{mr}^T M Z_{mr} = G^T \Sigma G, \quad \Sigma = I_k \oplus -I_k. \tag{10}$$

The generator is given by

$$G = \begin{bmatrix} X_0 & X_1 & \dots & X_{n-1} & R_0^{-T} & 0 & \dots & 0 \\ 0 & Y_1 & \dots & Y_{n-1} & 0 & 0 & \dots & 0 \end{bmatrix}, \tag{11}$$

where $X_i = Y_i = R_0^{-T} T_i$ and $T_0 = R_0^T R_0$ is the Cholesky decomposition of $T_0 > 0$.

Using this generator we can now consider the problem of computing simultaneously the Cholesky factorizations for T and T^{-1}. Let U denote the upper Cholesky factor of T and L the lower (or reversed) Cholesky factor of T^{-1} such that $T = U^T U$ and $T^{-1} = L^T L$. Then the first part of the Cholesky decomposition of M clearly yields

$$M = \begin{bmatrix} U^T \\ L^T \end{bmatrix} \begin{bmatrix} U & L \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -T^{-1} \end{bmatrix}.$$

In other words, this constructs both factors U and L as well as the generator for T^{-1}. If applying the same ideas to the non-symmetric matrix

$$M = \begin{bmatrix} T & -b \\ I & 0 \end{bmatrix},$$

one easily checks that the Schur complement is $T^{-1}b$, i.e. the solution of the system of equations

$$Tx = b.$$

The subroutine `TOEPI` (see appendix 7) uses the above ideas to compute the Cholesky factors of T, T^{-1} and a generator for T^{-1}.

Example 1 A Matlab version of the subroutine `TOEPI` was applied to several block Toeplitz matrices with normally distributed random entries ($\sim N(0,1)$). T_0 was always strictly diagonal dominant enough to guarantee positive definiteness of T.

The following errors were computed

$$\begin{aligned} e_U &= \|U^TU - T\|/\|T\|, \\ e_L &= \|LTL^T - I\|, \\ e_I &= \|T_iT - I\|, \quad (T_i \text{ is the inverse of } T \text{ computed from its generator}), \\ e_{\text{chol}} &= \|U^TU - T\|/\|T\|, \quad (\text{using } \texttt{chol} \text{ of Matlab}© \text{ to compute } U). \end{aligned}$$

Furthermore, the execution time of `TOEPI` in comparison with the LAPACK routine `DPOTRF`.

k	n	e_U	e_L	e_I	e_{chol}	t_{TOEPI}	t_{DPOTRF}
1	1000	1.14e-13	4.68e-15	5.53e-15	1.13e-15	0.72s	4.05s
2	500	1.07e-13	4.32e-15	2.01e-14	2.03e-15	0.74s	4.05s
20	50	5.17e-13	3.22e-15	1.48e-14	2.50e-14	1.13s	4.04s
50	20	1.32e-12	4.89e-15	3.14e-14	4.90e-14	1.70s	4.06s

5 QR Decomposition of block Toeplitz and Hankel matrices

A related question is that of finding the QR factorization of a (block) Toeplitz or (block) Hankel matrix :

$$T = QR, \quad H = QR.$$

Both problems are equivalent since a permutation (i.e. an orthogonal row transformation) transforms a (block) Toeplitz matrix into a (block) Hankel matrix and vice-versa. Clearly, symmetry is not an issue anymore and the blocks need not be square anymore as well. Since both problems are equivalent, we only consider the Toeplitz case below.

Let $T \in \mathbb{R}^{mk,nl}$ be a block Toeplitz matrix with blocks of size $k \times l$:

$$n \geq m: \quad T \quad = \quad \left[\begin{array}{ccccccc} T_0 & T_1 & \dots & T_{n-m} & \dots & \dots & T_{n-1} \\ T_{-1} & \ddots & \ddots & & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \ddots & \vdots \\ T_{-m+1} & \dots & T_{-1} & T_0 & T_1 & \dots & T_{n-m} \end{array} \right],$$

$$n \leq m: \quad T \quad = \quad \left[\begin{array}{cccc} T_0 & T_1 & \dots & T_{n-1} \\ T_{-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_1 \\ T_{-m+n} & & \ddots & T_0 \\ \vdots & \ddots & & T_1 \\ \vdots & & \ddots & \vdots \\ T_{-m+1} & \dots & \dots & T_{-m+n} \end{array} \right].$$

We consider the problem of computing the QR factorization of the matrix T, where $Q^TQ = I$ and R is upper trapezoidal and of full row rank. It is known that the matrix $T^T.T$ has a low displacement rank [3], and hence we can apply the Schur algorithm to it. Since

$$T^T.T = R^T.Q^T.Q.R = R^T.R,$$

this would only yield the factor R. In order to get both factors we consider the bordered matrix

$$M = \left[\begin{array}{cc} T^TT & T^T \\ T & I \end{array} \right] = \left[\begin{array}{c} T^T \\ I \end{array} \right] \left[\begin{array}{cc} T & I \end{array} \right],$$

which is obviously of low rank and will be shown to have displacement rank $2(\ell + k)$ bounded by $2(l + k)$, i.e.,

$$\operatorname{rank}(M - Z^TMZ) = 2(\ell + k), \quad \ell \leq l \tag{12}$$

where

$$Z = \left[\begin{array}{cccc} 0 & I_l & & \\ & \ddots & \ddots & \\ & & \ddots & I_l \\ & & & 0 \end{array} \right] \oplus \left[\begin{array}{cccc} 0 & I_k & & \\ & \ddots & \ddots & \\ & & \ddots & I_k \\ & & & 0 \end{array} \right] \doteq Z_1 \oplus Z_2 \in \mathbb{R}^{nl,nl} \oplus \mathbb{R}^{mk,mk}.$$

It easily follows from $T = QR$ that the Cholesky factorization of M terminates early and yields the factors

$$M = \left[\begin{array}{c} R^T \\ Q \end{array} \right] \left[\begin{array}{cc} R & Q^T \end{array} \right].$$

Since M is symmetric it should be possible to construct a generator G which satisfies the displacement equation

$$M - Z^TMZ = G^T\Sigma G, \quad \text{with } G \in \mathbb{R}^{2(\ell+k),nl+mk}, \quad \Sigma = I_{\ell+k} \oplus -I_{\ell+k}. \tag{13}$$

The following theorem extends known formulas for the scalar case [3] to the block case :

Theorem 2 *Given the QR factorization of the first block column of a block Toeplitz matrix $T \in \mathbb{R}^{mk,nl}$:*

$$\left[\begin{array}{c} T_0 \\ \vdots \\ T_{-m+1} \end{array} \right] = \left[\begin{array}{c} C_0^T \\ \vdots \\ C_{m-1}^T \end{array} \right] .R_0, \tag{14}$$

where R_0 (can be assumed upper trapezoidal and) has full row rank $\ell \leq l$, and the product

$$\left[\begin{array}{ccc} S_0 & \ldots & S_{n-1} \end{array} \right] = \left[\begin{array}{ccc} C_0 & \ldots & C_{m-1} \end{array} \right] T, \tag{15}$$

then a generator for (13) is given by

$$G = \left[\begin{array}{cc} G_R & G_Q \end{array} \right], \quad \Sigma = I_{\ell+k} \oplus -I_{\ell+k} \tag{16}$$

where

$$G_R = \left[\begin{array}{cccc} S_0 & S_1 & \ldots & S_{n-1} \\ 0 & T_1 & \ldots & T_{n-1} \\ \hline 0 & S_1 & \ldots & S_{n-1} \\ 0 & T_{-m+1} & \ldots & T_{-m+n-1} \end{array} \right], \quad G_Q = \left[\begin{array}{cccc} C_0 & C_1 & \ldots & C_{m-1} \\ I_k & 0 & \ldots & 0 \\ \hline C_0 & C_1 & \ldots & C_{m-1} \\ 0 & 0 & \ldots & 0 \end{array} \right].$$

Proof :
In order to prove the result we consider the displacement

$$\nabla \left[\begin{array}{cc} T^T T & T^T \\ T & I \end{array} \right] = \left[\begin{array}{cc} T^T T - Z_1^T T^T T Z_1 & T^T - Z_1^T T^T Z_2 \\ T - Z_2^T T Z_1 & I - Z_2^T Z_2 \end{array} \right]$$

which ought to be equal to

$$\left[\begin{array}{cc} G_R^T \Sigma G_R & G_R^T \Sigma G_Q \\ G_Q^T \Sigma G_R & G_Q^T \Sigma G_Q \end{array} \right].$$

The identity $I - Z_2^T Z_2 = G_Q^T \Sigma G_Q$ is trivial because the I_k block in the second row of G_Q yields the only nonzero block in the product $G_Q^T \Sigma G_Q$. The equality $T - Z_2^T T Z_1 = G_Q^T \Sigma G_R$ also easily follows since

$$G_Q^T \Sigma G_R = \left[\begin{array}{cccc} C_0^T R_0 & T_1 & \ldots & T_{n-1} \\ C_1^T R_0 & 0 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ C_{m-1}^T R_0 & 0 & \ldots & 0 \end{array} \right] = \left[\begin{array}{cccc} T_0 & T_1 & \ldots & T_{n-1} \\ T_{-1} & 0 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ T_{-m+1} & 0 & \ldots & 0 \end{array} \right].$$

Finally,

$$G_R^T \Sigma G_R = \left[\begin{array}{cccc} S_0^T S_0 & S_0^T S_1 & \ldots & S_0^T S_{n-1} \\ S_1^T S_0 & & & \\ \vdots & & A^T A - B^T B & \\ S_{n-1}^T S_0 & & & \end{array} \right]$$

where

$$A = \left[\begin{array}{ccc} T_1 & \ldots & T_{n-1} \end{array} \right], \quad B = \left[\begin{array}{ccc} T_{-m+1} & \ldots & T_{-m+n-1} \end{array} \right].$$

It now follows from (14,15) that $S_0 = R_0$ and hence

$$S_0^T \left[\begin{array}{ccc} S_0 & \dots & S_{n-1} \end{array} \right] = R_0^T \left[\begin{array}{ccc} C_0 & \dots & C_{m-1} \end{array} \right] .T,$$

which is the first block row of the product $T^T T$. This thus verifies the first block row and block column of the identity $G_R^T \Sigma G_R = T^T T - Z_1^T T^T T Z_1$. The rest easily follows from the block Toeplitz structure of T. ∎

Note that if the first block column of T has full rank then R_0 is square invertible and $\ell = l$. If moreover the whole matrix T has full column rank, then the Schur algorithm will not encounter singularities in the first $nl - 1$ steps. Singularities in the Schur algorithm can be encountered otherwise, but in either case R and Q are produced by the algorithm. The subroutine `TOEPQR` (see appendix 8) uses this theorem to compute the QR decomposition of T.

Example 3 A Matlab version of the subroutine `TOEPQR` was applied on several block Toeplitz matrices with normally distributed random entries ($\sim N(0,1)$).

The following errors were computed

$$\begin{aligned} e_R &= \|T^T T - R^T R\| / \|T^T T\|, \\ e_{QR} &= \|T - QR\| / \|T\|, \\ e_Q &= \|I - Q^T Q\|, \\ e_{\text{qr}} &= \|T - QR\| / \|T\|, \quad \text{(using qr of Matlab© to compute } Q \text{ and } R). \end{aligned}$$

Furthermore, the execution time of `TOEPQR` in comparison with the LAPACK routines `DGEQRF` and `DORG2R`.

k	l	m	n	e_R	e_{QR}	e_Q	e_{qr}	t_{TOEPQR}	t_{LAPACK}
1	1	1000	1000	5.38e-15	3.07e-15	1.99e-09	3.19e-15	3.36s	56.67s
10	100	100	10	1.62e-15	2.28e-15	5.00e-09	1.57e-15	20.40s	55.39s
100	10	10	100	3.87e-15	1.33e-15	5.94e-08	3.39e-15	28.47s	56.49s
100	100	10	10	3.14e-15	2.83e-15	2.07e-10	2.83e-15	39.62s	56.65s

Another variant of this uses the embedding

$$M = \left[\begin{array}{cc} T^T T & T^T \\ T & 0 \end{array} \right],$$

which has the incomplete factorization

$$M = \left[\begin{array}{c} R^T \\ Q \end{array} \right] \left[\begin{array}{cc} R & Q^T \end{array} \right] - \left[\begin{array}{cc} 0 & 0 \\ 0 & -Q.Q^T \end{array} \right].$$

The generator of this will have a displacement rank that is slightly higher than the one proposed above [3], but it can be used to solve systems of least squares solutions since

$$\min_x \|Tx - b\|_2$$

is equivalent to the embedded system of equations

$$\left[\begin{array}{cc} T^T T & T^T \\ T & 0 \end{array} \right] \left[\begin{array}{c} x \\ -b \end{array} \right] = \left[\begin{array}{c} 0 \\ b \end{array} \right].$$

6 Concluding remarks

This report describes several variants of the classical Schur algorithm which we implemented for the computation of various factorizations of Toeplitz and block Toeplitz matrices. The problems considered in this report always implicitly rely on the Cholesky factorization of semidefinite matrices, for which the Schur algorithm has been shown to be numerically reliable. We also included numerical results substantiating these claims.

References

[1] A. Bojanczyk, R. Brent, P. Van Dooren, F. de Hoog, "A note on downdating the Cholesky factorization", *SIAM J. Sci. & Stat. Comp.*, Vol. 8, pp. 210–221, May 1987.

[2] K. Gallivan, S. Thirumalai, P. Van Dooren, V. Vermaut, "High performance algorithms for Toeplitz and block Toeplitz matrices", *Lin. Alg. & Appl.*, Vol. 241-243, pp. 343-388, 1996.

[3] T. Kailath, A. Sayed, *Fast Reliable Algorithms for Matrices with Structure* SIAM Publ., Philadelphia, 1999.

[4] M. Stewart, P. Van Dooren, "Stability Issues in the Factorization of Structured Matrices", *SIAM J. Matrix Anal. & Appl.*, Vol. 18-1, pp. 104–108, 1997.

7 The FORTRAN 77 subroutine TOEPI

The FORTRAN 77 subroutine `TOEPI` has the following calling sequence and in-line documentation.

```
      SUBROUTINE TOEPI(JOB, K, N, T, LDT, G, LDG, L, LDL, R, LDR,
     $           DWORK, LDWORK, INFO)
C
C     PURPOSE
C
C     Given the first row T of a symmetric positive definite block
C     Toeplitz Matrix this routine will compute the Cholesky
C     factor and the generator/Cholesky factor of the inverse.
C     Transformation information will be stored.
C
C     ARGUMENTS
C
C     Mode Parameters
C
C     JOB     CHARACTER*1
C             Specifies the output of the routine as follows:
C             = 'G':  only computes the generator G of the inverse
C             = 'R':  computes the generator G of the inverse and the
C                     upper Cholesky factor R of T
C             = 'L':  computes the generator G and the lower Cholesky
C                     factor L of the inverse
C             = 'A':  computes the generator G, the lower Cholesky
C                     factor L of the inverse and the upper Cholesky
C                     factor R of T
C             = 'O':  only compute the upper Cholesky factor R of T
C
C     Input/Output Parameters
C
C     K       (input)  INTEGER
C             The number of rows in T, which should be equal to the
C             blocksize. K >= 0
C
C     N       (input)  INTEGER
C             The number of blocks in T
C
C     T       (input/output)  DOUBLE PRECISION array, dimension (LDT,N*K)
C             On entry, the leading K-by-N*K part of this array contains
C             the first row of the block Toeplitz matrix.
C             On exit, the leading K-by-N*K part of this array contains
C             in the first K-by-K block the upper Cholesky factor of
```

```
C           T(1:K,1:K) and in the rest Householder transformations,
C           applied during the process.
C
C    LDT    INTEGER
C           The leading dimension of the array T. LDT >= K
C
C    G      (output)  DOUBLE PRECISION array, dimension (LDG,N*K)
C           If INFO = 0 and JOB IN ['G','R','L','A'], then
C           the leading 2*K-by-N*K part of this array contains the
C           generator of the inverse of T.
C
C    LDG    INTEGER
C           The leading dimension of the array G.
C           IF JOB IN ['G','R','L','A'], LDG >= 2*K
C
C    L      (output)  DOUBLE PRECISION array, dimension (LDL,N*K)
C           If INFO = 0 and JOB IN ['L','A'], then
C           the leading N*K-by-N*K part of this array contains the
C           lower Cholesky factor of the inverse of T.
C           Note that elements in the upper triangular part will not
C           be referenced.
C
C    LDL    INTEGER
C           The leading dimension of the array L.
C           IF JOB IN ['L','A'], LDL >= N*K
C
C    R      (output)  DOUBLE PRECISION array, dimension (LDR,N*K)
C           If INFO = 0 and JOB IN ['R','A','O'], then
C           the leading N*K-by-N*K part of this array contains the
C           upper Cholesky factor of T.
C           Note that elements in the lower triangular part will not
C           be referenced.
C
C    LDR    INTEGER
C           The leading dimension of the array R.
C           IF JOB IN ['R','A','O'], LDR >= N*K
C
C    DWORK  DOUBLE PRECISION array, dimension (LDWORK)
C           Working space.
C           On exit, if INFO = 0, the first 3*(N-1)*K  elements
C           contain information about the hyperbolic rotations
C           and Householder transformations applied during the
C           process. DWORK(3*(N-1)*K+1)  returns the optimal
C           value of LDWORK.
C           On exit, if  INFO = -13,  DWORK(1)  returns the minimum
```

```
C           value of LDWORK.
C
C     LDWORK INTEGER
C           The length of the array DWORK.
C           LDWORK >= 3*N*K
C           ( 3*(N-1)*K for transformation information and (N-1)*K
C             for the routine APPLYG )
C
C     Error Indicator
C
C     INFO   INTEGER
C           = 0:  successful exit
C           < 0:  if INFO = -i, the i-th argument had an illegal
C                 value
C           = 1:  the reduction algorithm failed. The Toeplitz matrix
C                 associated with T is not (numerically)
C                 positive definite.
```

8 The FORTRAN 77 subroutine TOEPQR

The FORTRAN 77 subroutine `TOEPQR` has the following calling sequence and in-line documentation.

```
      SUBROUTINE TOEPQR(JOB, K, L, M, N, TC, LDTC, TR, LDTR, Q, LDQ,
     $                  R, LDR, DWORK, LDWORK, INFO)
C
C     PURPOSE
C
C
C     Let T be a K*M-by-L*N block Toeplitz matrix with blocks of
C     size (K, L). The first column of T shall be denoted by TC
C     and the first row by TR. Under the assumption that the first
C     MIN(M*K, N*L) columns of T have full column rank this routine
C     will compute isometric Q and upper triangular R such that
C
C                                  T
C                        T  =  Q  R.
C
C     ARGUMENTS
C
C     Mode Parameters
C
C     JOB     CHARACTER*1
C             Specifies the output of the routine as follows:
C             = 'N':  only computes R
C             = 'Q':  computes Q and R
C
C     Input/Output Parameters
C
C     K       (input)  INTEGER
C             The number of rows in one block of T. K >= 0
C
C     L       (input)  INTEGER
C             The number of columns in one block of T. L >= 0
C
C     M       (input)  INTEGER
C             The number of blocks in one column of T. M >= 0
C
C     N       (input)  INTEGER
C             The number of blocks in one row of T. M >= 0
C
C     TC      (input) DOUBLE PRECISION array, dimension (LDTC, (M-1)*K)
C             The leading L-by-(M-1)*K part of this array contains the
```

```
C           transposed of the first column of T without the upper
C           K-by-L block.
C
C   LDTC    INTEGER
C           The leading dimension of the array TC. LDTC >= L
C
C   TR      (input) DOUBLE PRECISION array, dimension (LDTR, N*L)
C           The leading K-by-N*L part of this array contains the
C           first row of T.
C
C   LDTR    INTEGER
C           The leading dimension of the array TR. LDTR >= K
C
C   Q       (output)   DOUBLE PRECISION array, dimension (LDQ, M*K)
C           If JOB = 'Q', then the leading MIN(M*K, N*L)-by-M*K part
C           of this array contains the isometric factor Q.
C
C   LDQ     INTEGER
C           The leading dimension of the array Q.
C           If JOB = 'Q',  LDQ >= MIN(M*K, N*L)
C
C   R       (output)   DOUBLE PRECISION array, dimension (LDR, N*L)
C           The leading MIN(M*K, N*L)-by-N*L part of this array
C           contains the upper triangular factor R.
C           Note that elements in the lower triangular part will not
C           be referenced.
C
C   LDR     INTEGER
C           The leading dimension of the array R.
C           LDR >= MIN(M*K, N*L)
C
C   DWORK   DOUBLE PRECISION array, dimension (LDWORK)
C           Working space.
C           On exit, if INFO = 0 and JOB = 'Q', the first
C           (M*K+(N-1)*L) * (2*K+L) + 3 * MIN(M*K,N*L) elements
C           of DWORK contain information about transformations
C           applied during the process and furthermore the
C           generators at the n-th step of the Schur algorithm.
C           If JOB = 'N', the first ((N-1)*L) * (2*K+L) + 3 *
C           MIN(M*K,N*L) elements contain this information.
C           DWORK(((N-1)*L) * (2*K+L) + 3 * MIN(M*K,N*L) + 1)
C           returns the optimal value of LDWORK.
C           On exit, if INFO = -13,  DWORK(1)  returns the minimum
C           value for LDWORK.
C
```

```
C     LDWORK INTEGER
C            The length of the array DWORK.
C            JOB = 'Q':  LDWORK >= (M*K+(N-1)*L) * (2*K+L) +
C                                 3*MIN(M*K,N*L) + MAX((N-1)*L, M*K))
C            JOB = 'N':  LDWORK >= ((N-1)*L) * (2*K+L+1) +
C                                 3*MIN(M*K,N*L)
C
C     Error Indicator
C
C     INFO   INTEGER
C            = 1:  the first MIN(M*K, N*L) columns of T have no full
C                  column rank
C            = 0:  successful exit
C            < 0:  if INFO = -i, the i-th argument had an illegal
C                  value
```