ECOLE POLYTECHNIQUE

CENTRE DE MATHÉMATIQUES APPLIQUÉES
UMR CNRS 7641

91128 PALAISEAU CEDEX (FRANCE). Tél: 01 69 33 41 50. Fax: 01 69 33 30 11

http://www.cmap.polytechnique.fr/

Fast direct solvers for some complex symmetric block Toeplitz linear systems.

Natacha Béreux

R.I. N^0 531 *May 2004*

Fast direct solvers for some complex symmetric block Toeplitz linear systems

Natacha Béreux

6th May 2004

Abstract

We consider the solution of a class of complex symmetric block Toeplitz linear systems, arising from integral equations problems. Algorithms that exploit the Toeplitz structure provide considerable savings on the number of arithmetic operations, compared to the classical Cholesky factorization. We propose a fast Schur algorithm adapted to the *complex symmetric* case. We detail blocked variants, that perform better by a wider use of BLAS3 primitives. We also propose a solver, based on an augmented matrix approach, that allows a substantial decrease in the use of memory, by avoiding an explicit assembly of the Cholesky factor. All algorithms have been implemented and numerical results are included to illustrate the effectiveness of our approach.

1 Introduction

The Boundary Element Method (BEM), which is widely used in electromagnetic or acoustic scattering, consists in transforming the original scattering problem, set in an unbounded domain into an integral equation set on the boundary of the scatterer [18]. In many situations, a symmetric formulation of the integral equation is preferred (such as the Electric Field Integral Equation in electromagnetism [3] [8]). The discretisation of such an integral equation *e.g* by Finite Element Methods (FEM) leads to a linear system

$$A\,X = B,$$

where the coefficients matrix A of size N is dense, complex valued, symmetric but non hermitian. It is well known that for a general complex symmetric matrix Cholesky algorithm may break down; hence it is recommended for the complex symmetric matrices to use the Bunch-Kauffman factorization algorithm with diagonal pivoting [7] [23]. Nevertheless, it has been widely observed, although never demonstrated, that for complex symmetric matrices issued from the discretization of Boundary Integral Equations, pivoting is not required: thus, a Cholesky factorization can be used, which substantially speeds up the computation [4]. For very large problems (*e.g* $N \geq 100000$), direct solvers are too costly

and preconditionned iterative algorithms are preferred [9] [10].
Our aim in this paper is to propose a fast direct solver when the coefficients matrix possess an additional block Toeplitz structure. For instance, the numerical analysis of piezo-electric surface acoustic wave (SAW) filters, by a coupled integral equation/finite element formulation [20] leads to a linear system whose coefficients matrix is both complex symmetric and block Toeplitz. This block Toeplitz structure comes from the periodicity of the geometry of the filter.
The concept of displacement structure [15] provides a unifiying framework to describe such matrices, whose entries are not independent: a point Toeplitz matrix of size N can namely be reconstructed from the knowledge of one row and one column, that is its N^2 entries only depend on $2N$ parameters. Fast direct algorithms for solving

$$T\,X \;=\; B,$$

where T is Toeplitz, have been proposed in [12] [17]. These algorithms are Schur-type algorithm, that is they are fast procedures, in only $O(N^2)$ operations, for obtaining the Cholesky factorization of T, which usually requires $O(N^3)$ operations. They have been developped and tested in the hermitian case. We refer to [22] for an analysis of stability of these algorithms.
In this article, we derive several variants of Schur-type algorithms adapted to complex symmetric block Toeplitz matrices [1].
In section 2, we review some definitions about the displacement structure. Then, in Section 3, we propose a fast algorithm with a complexity of $O(n^2m^3)$ for the Cholesky factorization of a block Toeplitz complex symmetric matrix with $n \times n$ blocks, each of size $m \times m$. The key step consists in putting the generator of T in proper form, by means of complex orthogonal Householder transforms. In Section 4, we adapt several accumulation techniques to the complex orthogonal case and provide blocked variants of the algorithm, designed to make a wider use of BLAS 3 primitives. We present in Section 5 a fast solver, based on an augmented matrix approach, which does not explicitly assemble the triangular Cholesky factor, and thus is less memory consuming. Finally, we present in Section 6 the results of numerical experiments that demonstrate the efficiency of the algorithms, especially the blocked versions, both in terms of accuracy and execution time.

2 Displacement structure

We recall some basic definitions on the displacement structure approach introduced first in [11] for Toeplitz matrices, and presented in a systematic way in [15] and [14]. Since we are mainly interested in symmetric matrices, we present the symmetric case first, and then the modifications that are necessary when dealing with non-symmetric matrices.

[1]Like Cholesky algorithm for complex symmetric matrices, our algorithm can be subject to breakdown. Nevertheless, for matrices issued from the discretization of BIE, it has never been observed

2.1 Symmetric displacement

For a given matrix $F \in \mathbb{C}^{n\times n}$, we define a symmetric displacement operator by:

$$\begin{array}{rcl} \nabla_F : \mathbb{C}^{n\times n} & \rightarrow & \mathbb{C}^{n\times n} \\ M & \mapsto & \nabla_F(M) = F\,M + M\,F^T, \end{array} \tag{1}$$

or

$$\begin{array}{rcl} \nabla_F : \mathbb{C}^{n\times n} & \rightarrow & \mathbb{C}^{n\times n} \\ M & \mapsto & \nabla_F(M) = M - F\,M\,F^T, \end{array} \tag{2}$$

The operator defined by (1) is called a *Sylvester-type* operator, and the operator defined by (2) a *Stein-type* operator.
For symmetric matrices with a particular structure, F can be chosen so that ∇_F maps these matrices to low-rank matrices.
Moreover, $\nabla_F(M)$ is called *displacement of* M and its rank is called ∇_F-*displacement rank.*
A *structured* matrix $M \in \mathbb{C}^n$ (for a certain displacement operator ∇_F) is a matrix for which a F can be found such that

$$rank(\nabla_F(M)) \ll n$$

Example 2.1 (Symmetric block Toeplitz matrix). A *block Toeplitz matrix* is a matrix whose block entries are constant along the diagonals. If T is moreover symmetric, it reads:

$$T = \begin{pmatrix} T_0 & T_1^T & \dots & T_{n-1} \\ T_1^T & T_0 & \dots & T_{n-2} \\ \dots & & & \\ T_{n-1}^T & T_{n-2}^T & \dots & T_0 \end{pmatrix},$$

where each $T_k \in \mathbb{C}^{m\times m}$, and $T_0 = T_0^T$.
We choose F as the block shift matrix:

$$F = Z_n = \begin{pmatrix} 0_m & & \dots & 0_m \\ I_m & 0_m & & \vdots \\ \vdots & \ddots & \ddots & \\ 0_m & \dots & I_m & 0_m \end{pmatrix},$$

where I_m denotes the identity of size m and 0_m the zero matrix of size m. The index n recalls the size of Z_n: $n \times n$ blocks. It is ommitted whenever its value is obvious. Then, the displacement of T is

$$\nabla_Z(T) = T - Z\,T\,Z^T = \begin{pmatrix} T_0 & T_1 & \dots & T_{n-1} \\ T_1^T & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ T_{n-1}^T & 0 & \dots & 0 \end{pmatrix}. \tag{3}$$

Its rank is equal to $2m$, and it is much smaller than the size mn of T.

2.2 Generators

The symmetric displacement of a real matrix $M \in \mathbb{R}^{n \times n}$ can be factorized (in a nonunique way) as

$$\nabla_F(M) \,=\, A\,J\,A^T,$$

where A is a real matrix, of size $n \times \alpha$ ($\alpha \,=\, rank(\nabla_F(M))$), and $J \in \mathbb{R}^{\alpha\times\alpha}$ a symmetric real matrix.
When M is a complex matrix, its displacement can be factorized (also in a non unique way) as

$$\nabla_F(M) \,=\, A\,A^T,\; A \,\in\, \mathbb{C}^{n\times\alpha}$$

Example 2.2. For a complex block Toeplitz matrix T, the displacement $\nabla_Z(T)$ is factorized by:

$$\nabla_Z(T) \,=\, T \,-\, Z_n\,T\,Z_n^T \,=\, A\,A^T, \tag{4}$$

where the matrix $A \in \mathbb{C}^{mn\times m}$ is obtained by identification:

$$A \,=\, \begin{pmatrix} L_0 & 0 \\ T_1\,L_0^{-T} & i\,T_1\,L_0^{-T} \\ \vdots & \vdots \\ T_n\,L_0^{-T} & i\,T_n\,L_0^{-T} \end{pmatrix}, \tag{5}$$

and L_0 is the lower Cholesky factor of the first bloc T_0:

$$L_0\,L_0^T \,=\, T_0. \tag{6}$$

The matrix A (or the pair A, J for real matrices) is called *symmetric generator* of M.

Remark 1. The generator is not unique:

$$\forall\,Q \in \mathbb{C}^{\alpha\times\alpha}, \quad Q\,Q^T \,=\, I_\alpha, \quad \nabla_F(M) \,=\, A\,A^T \,=\, (A\,Q)\,(A\,Q)^T$$

and $A\,Q$ is another symmetric generator of M.

Definition 2.1. The symmetric proper generator of M, associated to the displacement operator ∇_Z is the unique matrix A such that

- $A\,A^T \,=\, \nabla_Z(M)$
- $A_{12} \,=\, 0_m$
- A_{11} is lower triangular

Remark 2. The generator defined by (5) is in proper form.

2.3 The non-symmetric case

The definitions introduced in the last section can be stated in a more general way, for matrices that are not symmetric and not even square.
For two given matrices $F \in \mathbb{C}^{n\times n}$ and $G \in \mathbb{C}^{p\times p}$, a (non-symmetric) displacement operator is defined by

$$\begin{array}{rcll} \nabla_{F,G} : \mathbb{C}^{n\times p} & \rightarrow & \mathbb{C}^{n\times p} & \\ M & \mapsto & \nabla_{F,G}(M) \,=\, F\,M + M\,G, & \end{array}$$

or

$$\begin{array}{rcll} \nabla_{F,G} : \mathbb{C}^{n\times p} & \rightarrow & \mathbb{C}^{n\times p} & \\ M & \mapsto & \nabla_{F,G}(M) \,=\, M - F\,M\,G, & \end{array}$$

A *structured* matrix is a matrix for which a pair (F, G) can be found such that

$$\alpha \overset{def}{=} rank(\nabla_{F,G}(M)) \ll rank(M).$$

The displacement of M can then be factorized as:

$$\nabla_{F,G}(M) \,=\, A\,B,$$

where $A \in \mathbb{C}^{n\times\alpha}$ and $B \in \mathbb{C}^{\alpha\times p}$. The pair (A, B) is called *generator* of M.

Remark 3. Note that these definitions reduce to the symmetric case if we take:

$$n \,=\, p, \;\; G \,=\, F^T, \;\; B \,=\, A^T.$$

3 Fast Cholesky algorithm

In this section, we derive a fast algorithm for the Cholesky factorization of a block Toeplitz matrix, T. Since T is only defined by n block entries (instead of $n(n+1)/2$), each of size m^2, the complexity of its Cholesky factorization can be reduced from an order of magnitude and a n^2 complexity is expected.
The key point in writing a fast algorithm is to rewrite the classical algorithm in terms of generator updates, thanks to the displacement equation (4), which expresses that the whole matrix can be reconstructed from the knowledge of the generator. The operations needed are thus applied to roughly n block entries, instead of n^2 and the complexity is decreased.
We refer to [12] and [17], for the derivation of fast Cholesky algorithms for real symmetric block Toeplitz matrices.

3.1 Cholesky algorithm

Let T be a block Toeplitz matrix with $n \times n$ blocks. Let us set

$$T^{(0)} \,=\, T,$$

and define the block Schur complement of order k, $T^{(k)} \in \mathbb{C}^{(n-k).m\times(n-k).m}$ by

$$T^{(k)} \,=\, T_{km+1:nm,km+1:nm} - T_{km+1:nm,1:km}\, T^{-1}_{1:km,1:km}\, T^{T}_{km+1:nm,1:km},$$

or equivalently by recursion from $T^{(k-1)}$ by

$$\begin{aligned} T^{(k)} &= T^{(k-1)}_{m+1:(n-k)m,m+1:(n-k)m} \\ &\quad -T^{(k-1)}_{m+1:(n-k)m,1:m} {T^{(k-1)}}^{-1}_{1:m.1:m} {T^{(k-1)}}^{T}_{m+1:(n-k)m,1:m}. \end{aligned}$$

Cholesky block algorithm consists in computing $n-1$ successive block Schur complements of T, and finally obtain a symmetric fatorization of T as

$$T = L\,L^T,$$

with L lower triangular matrix.
We recall the block right-looking version of Cholesky factorization in Algorithm 1. In this algorithm, the function `llt` computes the Cholesky factorization of a complex symmetric matrix in a non-blocked way.

Algorithm 1 Block right-looking Cholesky factorization

```
Input: Matrix $T$ of size $nm \times nm$
Output: Lower Cholesky factor of $T$, $L$ such that $T = L\,L^T$.
 1: for $k = 1 : n$ do
 2:   $s = (k-1).m + 1$                          {Start of block to factorize}
 3:   $e = k.m$                                  {End of block to factorize}
 4:   $u = e + 1$                                {Start position for update}
 5:   Compute Cholesky factorization of $T_{s:e,s:e}$:
      $L_{s:e,s:e} \leftarrow \texttt{llt}(T_{s:e,s:e})$
 6:   if $e < m$ then
 7:     Update bottom of block column $k$:
        $L_{u:nm,s:e} = T_{u:nm,s:e} / L_{s:e,s:e}$
 8:     Compute Schur complement $T^{(k)}$:
        $T_{u:nm,u:nm} \leftarrow$
                 $T_{u:nm,u:nm} - L_{u:nm,s:e}\,L^T_{u:nm,s:e}$
 9:   end if
10: end for
```

3.2 Schur complementation and generator recursion

We show that the displacement operator is invertible, that is the whole block matrix T solution of

$$T - Z\,T\,Z^T = A\,A^T.$$

can be reconstructed from a generator A.
The next Lemma justifies that the successive Schur complements are themselves structured and satisfy the same kind of displacement equations than T.

Lemma 3.1. *Let T be a matrix of size $nm \times nm$, solution of the homogeneous equation:*

$$T - Z\,T\,Z^T = 0_{mn\times mn}.$$

Then, $T = 0_{mn\times mn}$.

Proof. The first block column and the first block row of $Z\,T\,Z^T$ are equal to zero, forall matrix T. If moreover T is such that $\nabla_Z(T) = 0$, then the first block row and column of T are bound to be zero. There follows immediately that the rest of the matrix T has to be zero. $\square$

Lemma 3.2. *Let T be a complex symmetric block Toeplitz matrix, with $n \times n$ blocks of size $m \times m$. Let us set*

$$T^{(0)} = T.$$

There exists a sequence of proper generators:

$$A^{(k)} \in \mathbb{C}^{(n-k).m \times 2m}, \quad k \in [0 : n-1],$$

such that:

$$\forall\, k \in [0 : n-1], \quad \nabla_{Z_{n-k}}(T^{(k)}) = T^{(k)} - Z_{n-k}\,T^{(k)}\,Z_{n-k}^T = A^{(k)}\,{A^{(k)}}^T. \tag{7}$$

Moreover, if L is the lower Cholesky factor of T such that $T = L\,L^T$, we have:

$$\forall\, k \in [0 : n-1], \quad L(km+1 : nm, km+1 : (k+1).m) = A^{(k)}(:, 1 : m). \tag{8}$$

Proof. The Lemma is shown by induction.
For $k = 0$, the matrix $T = T^{(0)}$ satisfies the displacement equation

$$\nabla_{Z_n}(T^{(0)}) = A^{(0)}\,{A^{(0)}}^T, \tag{9}$$

where the proper generator $A^{(0)} = A$ is defined by (5).
Equality (8) for $k = 0$

$$A^{(0)}(1 : m, :) = L(:, 1 : m), \tag{10}$$

is immediate (by identification) since $A^{(0)}$ is in proper form.
For the sake of readability, we will now show (7) for $k = 1$, instead of justifying that this equation remains true for k, provided that it is true for $k-1$.
We construct a proper generator $A^{(1)}$ such that the Schur complement of order 1 satisfies

$$\nabla_{Z_{n-1}}(T^{(1)}) = A^{(1)}\,{A^{(1)}}^T.$$

Let us partition $T^{(0)}$ in

$$T^{(0)} = \begin{pmatrix} T_{11} & T_{21}^T \\ T_{21} & T_{22} \end{pmatrix},$$

where we have denoted

$$T_{11} = T_{1:m,1:m},\; T_{21} = T_{1:m,m+1:nm},\; T_{22} = T_{m+1:nm,m+1:nm}.$$

We introduce the Cholesky factorization of the first block T_{11}:

$$T_{11} = L_{11}\,L_{11}^T = L_{1:m,1:m}\,L_{1:m,1:m}^T.$$

By (10), we have $L_{11} = A^{(0)}_{1:m,1:m}$, since $A^{(0)}$ is in proper form.
Since the Schur complement of order 1

$$T^{(1)} = T_{22} - T_{21}^T T_{11}^{-1} T_{21},$$

is given by

$$\begin{pmatrix} I_m & 0 \\ 0 & T^{(1)} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ -T_{12} T_{11}^{-1} & I_{(n-1)m} \end{pmatrix} \begin{pmatrix} T_{11} & T_{12}^T \\ T_{12} & T_{22} \end{pmatrix} \begin{pmatrix} L_{11}^{-T} & -T_{11}^{-1} T_{12}^T \\ 0 & I_{(n-1)m} \end{pmatrix},$$

we multiply the displacement equation (9), on the left by the elimination matrix

$$M = \begin{pmatrix} L_{11}^{-1} & 0 \\ -T_{12} T_{11}^{-1} & I_{(n-1)m} \end{pmatrix}$$

and on the right by M^T. Hence we get:

$$\begin{aligned} \begin{pmatrix} I_m & 0 \\ 0 & T^{(1)} \end{pmatrix} - M Z_n M^{-1} \begin{pmatrix} I_m & 0 \\ 0 & T^{(1)} \end{pmatrix} \left(M Z_n M^{-1}\right)^T \\ = M A^{(0)} (M A^{(0)})^T. \end{aligned} \tag{11}$$

We have

$$M Z_n = Z_n, \quad \text{and} \quad M^{-1} = \begin{pmatrix} L_{:,1:m} & \begin{matrix} 0 \\ I_{(n-1).m} \end{matrix} \end{pmatrix}.$$

So

$$M Z_n M^{-1} = Z_n M^{-1} = \begin{pmatrix} 0_m & 0 \\ L_{m+1:(n-1)m,1:m} & Z_{n-1} \end{pmatrix}.$$

We compute next

$$M A^{(0)} = \begin{pmatrix} I_m & 0 \\ 0 & A^{(0)}_{m+1:nm,m+1:2m} \end{pmatrix}.$$

By identifying the $(2,2)$ block term in (11), and remembering (10), we get

$$T^{(1)} - Z_{n-1} T^{(1)} Z_{n-1}^T = a^{(1)} {a^{(1)}}^T,$$

with

$$a^{(1)} = \begin{pmatrix} A_{1:(n-1)m,1:m} & A_{m+1:nm,m+1:2m} \end{pmatrix}. \tag{12}$$

So, $T^{(1)}$ satisfies the same kind of displacement equation than $T^{(0)}$, with Z_n replaced by Z_{n-1}. A symmetric generator of $T^{(1)}$ is given by (12), but it is not in proper form.
We then compute a (complex) orthogonal transform Q, such that

$$(a^{(1)} Q)_{1:m,m+1:2m} = 0_m.$$

The symmetric proper generator associated to $T^{(1)}$ is defined by:

$$A^{(1)} = a^{(1)} Q.$$

Equality (8) is easily obtained by identification from (7), since for all k, the generator $A^{(k)}$ is in proper form. □

3.3 Fast factorization algorithm

In this paragraph, we use Lemmas 3.2 and 3.1 to translate Algorithm 1 to its fast equivalent.
Thanks to Lemma 3.2, we know that the successive Schur complements of T satisfy displacement equations of type:

$$\nabla_{Z_{n-k}}(T^{(k)}) = A^{(k)} A^{(k)^T}$$

Thanks to Lemma 3.1, we know that such a displacement equation is uniquely solvable and characterizes matrix $T^{(k)}$ in a unique way. Moreover, at Schur step number k, the first block row of the proper generator is equal to the $k+1$ block row of the Cholesky factor, by (8).
So, instead of computing the sequence of Schur complements $T^{(k)}$ (classical Cholesky algorithm), we compute the sequence of their proper generators $A^{(k)}$ (fast Cholesky algorithm), as described in Alg. 2.

Algorithm 2 FAST BLOCK CHOLESKY FACTORIZATION

```
Input: First block-column of $T$, of size $m \times nm$
Output: Lower Cholesky factor of $T$, $L$ such that $T = L L^T$.
1: Set up generator $A$ from formula (5).
2: Initialize $L$ from $A$ by $L_{:,1:m} = A_{:,1:m}$.
3: for $k = 2 : n$ do
4:     Determine a generator $a^{(k)}$ for the $k^{th}$ Schur complement, $T^{(k)}$ from formula:
           $a^{(k)} = \begin{pmatrix} A^{(k-1)}_{1:(n-k).m,1:m} & A^{(k-1)}_{m+1:(n-k+1).m,m+1:2m} \end{pmatrix}$
5:     Put $a^{(k)}$ in proper form:
       Determine $Q$ complex orthogonal such that:
       $(a^{(k)} Q)_{1:m,m+1:2m} = 0_{m \times m}$.
       Apply $Q$ to the generator: $a^{(k)} \leftarrow a^{(k)} Q$.
6:     Store $k^{th}$ column of Cholesky factor $L$:
       $L_{(k-1).m+1:nm,(k-1).m+1:km} = a^{(k)}_{:,1:m}$
7: end for
```

4 Implementation issues

In this section, we now detail the implementation of the different steps of Algorithm 2.

4.1 Generator data structure

We have chosen to store A the generator of T, (and hence the sequence of the generators of the successive Schur complements) in two matrices $A1$ and $A2$, one for each block column:

$$A1,\, A2 \in \mathbb{C}^{mn\times m}, \quad A1 \leftarrow A(:,1:m), \quad A2 \leftarrow A(:,m+1:2m)$$

This results in Alg. 3 to set up the generator (based on formula (5)).
At each step k, the determination of a generator of the current Schur comple-

Algorithm 3 SETUP GENERATOR: $(A1, A2) = \texttt{setup\_gen}(T)$

```
1: $A1 = \texttt{zeros}(mn, m)$, $A2 = \texttt{zeros}(mn, m)$
2: $A1 \leftarrow T_{1:mn,1:m}$
3: Compute the Cholesky factorization of the first block of $A1$ *in situ*:
   $A1_{1:m,1:m} \leftarrow \texttt{llt}(A1_{1:m,1:m})$        {Non fast Cholesky factorization}
4: $A1_{m+1:nm,:} = A1_{m+1:nm,:}/A1^T{}_{1:m,1:m}$
5: $A2_{m+1:nm,:} = i * A1_{m+1:nm,:}$
```

ment is based on formula

$$a^{(k)} = \begin{pmatrix} A^{(k-1)}_{1:(n-k).m,1:m} & A^{(k-1)}_{m+1:(n-k+1).m,m+1:2m} \end{pmatrix},$$

which is the equivalent of (12) for a general k (not necessary 1). To avoid copy operations, it is achieved through the update of four indexes s_1, e_1, s_2 and e_2 such that

$$a^{(k)} = \begin{pmatrix} A1_{1:e1,:} \\ A2_{s2:e2,:} \end{pmatrix}$$

as shown in Alg. 4

Algorithm 4 SET CURRENT GENERATOR: $(e_1, s_2, e_2) = \texttt{set\_curr\_gen}(k)$

```
1: $e1 = (n-k+1)*m$
2: $s2 = (k-1)*m+1$
3: $e2 = n*m$
```

4.2 Reduction step

In this paragraph, we detail the main step of Algorithm 2, that is the reduction of generator in proper form:
At step k, the generator $a^{(k)}$ is defined by two matrices of $n-k$ blocks of size $m \times m$:

$$a^{(k)} = A = \begin{pmatrix} A1_{1:e1,1:m} & A2_{s2:e2,1:m} \end{pmatrix}$$

Matrix $A1$ is lower triangular.
To put $a^{(k)}$ in proper form, we have to find a complex orthogonal transform Q, such that

$$(a^{(k)}\,Q)_{1:m,m+1:2m} = 0_{m\times m}.$$

The transform Q is computed as a product of m elementary Householder reflectors.

Remark 4 *(Gram-Schmidt orthogonalization).* Modified Gram-Schmidt (MGS) method ([13] p.232) is an alternative to Householder method to perform the QR factorization of a square given matrix M.
But our purpose is slightly different, because the matrix A is not square: its size is $m \times 2m$.
MGS algorithm allows to build a set of m orthogonal vectors $Q(1, 1 : 2m) \dots Q(m, 1 : 2m)$ spanning the same space as $A(1, :), \dots, A(m, :)$. The matrix $Q \in \mathbb{C}^{m\times 2m}$ then satisfies

$$A = \begin{pmatrix} A1 & A2 \end{pmatrix} = Q\,R$$

for a triangular matrix $R \in \mathbb{C}^{m\times m}$.
But, one can not deduce from Q in a simple way an orthogonal matrix P, so that

$$A\,P = \begin{pmatrix} R & 0 \end{pmatrix}, \quad \text{and } P\,P^T = I_{2m},$$

as would be the case for A square!

That is the reason why we focus now on Householder orthogonalization method.

4.2.1 Householder transforms

The Householder method is the one chosen in LAPACK [1] to achieve the QR factorization of a matrix (routine `xGEQRF`). We refer to [13] (p.224) for a description. But since we are interested into complex *orthogonal* reflectors (and not *unitary*), we have to adapt the standard algorithm.
To annihilate block $a^{(k)}_{1:m,m+1:2m}$, we zero in turn rows s_2 to $s_2 + m - 1$ of $A2$ by applying successively m complex orthogonal Householder elementary reflectors H_j to A. Each reflector H_j is defined through:

$$H_j = I_{2m} - \beta_j\, x_j\, x_j^T, \quad j \in [1 : m],$$

where β_j is a complex scalar ($\beta_j = 2/(x_j^T.x_j)$) and $x_j \in \mathbb{C}^{2m}$ is the Householder vector.
The vector x_j is chosen in such a way that

$$A(:, j)\, H_j = \alpha_j\, e_j,$$

and H_j leaves the $j - 1$ previous rows of A unchanged.
The matrix A can thus be put in proper form by:

$$A_{1:m,1:2m} \leftarrow H_1 \dots H_m\, A_{1:m,1:2m}.$$

The orthogonal transform Q is not explicitly formed, it is instead determined in factored form as:

$$Q = H_1 \dots H_{m-1}\, H_m$$

Let us now detail the determination of the Householder reflector H_j. We impose that $x_j(j) = 1$, hence this term need not be stored. Taking into account the triangular pattern of $A1$, the components of x_j are then given by

$$x_j(i) = \begin{cases} 0 & 1 \leq i \leq j-1 \\ 1 & i = j \\ 0 & j+1 \leq i \leq m \\ x2(i-m) & m+1 \leq i \leq 2m \end{cases},$$

where $x2$ is a vector of size m that contains the significant part of x, called the *essential* part of x.
Alg. 5 computes the essential part of w which reduces to the vector $x2$, and the scalars β and α.

Algorithm 5 HOUSEHOLDER VECTOR: $(x_2, \beta, \alpha) = \texttt{house\_vec}(A1, A2, j_1, j_2)$

```
1: if $A2_{j_2,:}$ is already zero then
2:     $\beta = 0$
3: else
4:     Compute pseudo-norm of $A_{j,j:2m}$:
       $\sigma = A2_{j_2,:} . A2^T_{j_2,:}$
       $\alpha = (A1^2_{j_1,j_1} + \sigma)^{1/2}$
5:     if $|A1_{j_1,j_1} + \alpha| \leq |A1_{j_1,j_1} - \alpha|$ then
6:         $z \leftarrow A1_{j_1,j_1} - \alpha$
7:     else
8:         $z \leftarrow A1_{j_1,j_1} + \alpha$
9:         $\alpha \leftarrow -\alpha$
10:    end if
11:    $x2 \leftarrow A2_{j_2,:}/z$
12:    $\beta \leftarrow 2z^2/(\sigma + z^2)$
13: end if
```

4.3 Update of the generator

In the last paragraph, we described how to determine a set of m Householder reflectors $(H_j)_{j \in [1:m]}$ such that:

$$a^{(k)}_{1:m,1:2m} H_1 \dots H_m$$

is lower triangular.
Updating the generator consists in applying these transforms to the generator, that is:

$$a^{(k)} \leftarrow a^{(k)} H_1 \dots H_m.$$

4.3.1 Sequential update

A straightforward implementation shown in Alg. 6 consists in applying each reflector in turn to the whole generator. Computations are arranged so that

Algorithm 6 Sequential Reduction: $(A1, A2) = \texttt{seq\_reduc}\,(A1, A2, k)$

```
1: for $j = 1 : m$ do
2:     Determine $H_j$:
       $j_1 = j$                                        {row index in $A1$}
       $j_2 = s2 + j - 1$                               {row index in $A2$}
       $(x_2, \beta, \alpha) = \texttt{house\_vec}(A1, A2, j_1, j_2)$
3:     Update $A_{:,j}$ :
       $A1_{j_1,j_1} \leftarrow \alpha, \quad A2_{j_2,:} \leftarrow 0.$
4:     Update $A_{j+1:,:}$ :
       $A_{j+1:,:} \leftarrow A_{j+1:,:}\, H_j.$
5: end for
```

matrix R overwrites matrix $A1$ and matrix $X2$ overwrites matrix $A2$.

Due to the storage of A in two different matrices, step 4 of Alg. 6 is performed in the following way for rows $j+1 : e1$ of $A1$ and rows $s2+j : e2$ of $A2$:

```
1: $u1 = 1 + j$                                  {Start (row) position for update in $A1$}
2: $u2 = s2 + j$                                 {Start (row) position for update in $A2$}
3: $vec \leftarrow A1_{u1:e1,j} + A2_{u2:e2,:}\, x_2$                        {ZGEMV}
4: $A1_{u1:e1,j} \leftarrow A1_{u1:e1,j} - \beta_j\, vec$                    {ZAXPY}
5: $A2_{:,u2:e2} \leftarrow A2_{u2:e2,:} - \beta_j\, vec\, x_2^T$            {ZGERU}
```

It involves Level 2 BLAS operations: `ZGEMV` (matrix-vector multiplication) and `ZGERU` (rank-1 update).

4.3.2 Blocked versions

We describe in this paragraph variants of the reduction step, that can employ BLAS Level 3 kernels, and are thus able to achieve better performances than Alg. 6. These variants are based on aggregating the Householder transforms before applying them to the generator. Several techniques exists for performing the aggregation of transforms, respectively based on the WY representation of Bischof and Van Loan [6], on the storage efficient YTY^T representation of Schreiber and Van Loan [21] (which is used in LAPACK QR factorization routine `xGEQRF`) or its modification by Puglisi [19].

We present in turn these techniques, adapted to the complex orthogonal case. Our presentation closely follows the review in [5].

A product

$$Q_p = H_1\, H_2 \ldots H_p$$

of p Householder reflectors defined by

$$H_j = I_{2m} - \beta_j\, x_j\, x_j^T, \quad j \in [1:p],$$

can be represented as:

$Q_p \,=\, I_p \,+\, X_p\, Y_p^T$	**WY1**-way	[6]
$Q_p \,=\, I_p \,+\, W_p\, X_p^T$	**WY2**-way	[6]
$Q_p \,=\, I_p \,+\, X_p\, T_p\, X_p^T$	**YTY1**-way	[21]
$Q_p \,=\, I_p \,+\, X_p\, (T_p^{-1})^{-1}\, X_p^T$	**YTY2**-way	[19] [24]

Matrix X always stores the Householder vectors:

$$X_p \,\in \mathbb{C}^{2m \times p}.$$

Matrices $Y_j,\, W_j \,\in \mathbb{C}^{2m \times j}$ are updated for $j \,\in\, [1:p]$ by:

$$Y_1 \,=\, -\beta_1\, x_1, \quad Y_j \,=\, [H_j\, Y_{j-1} \quad -\beta_j\, x_j], \tag{13}$$

and

$$W_1 \,=\, -\beta_1\, x_1, \quad W_j \,=\, [W_{j-1} \quad -\beta_j\, Q_{j-1}\, x_j]. \tag{14}$$

Upper triangular matrix $T_j \,\in \mathbb{C}^{j \times j}$ is updated for $j \,\in\, [1:p]$ by:

$$T_1 \,=\, -\beta_1, \quad T_j \,=\, \begin{pmatrix} T_{j-1} & -\beta_j\, T_{j-1}\, X_{j-1}\, x_j \\ 0_{1 \times j-1} & -\beta_j \end{pmatrix} \tag{15}$$

In the YTY2 variant, matrix T_p^{-1} (instead of T_p) is computed by

$$T_p^{-1} \,=\, \texttt{triu}(-X_p^T\, X_p), \quad T_p^{-1}(i,i) \leftarrow T_p^{-1}(i,i)/2, \quad i \,=\, [1:p]. \tag{16}$$

This formula is based on the Woodbury-Morrison formula applied to the orthogonal matrix $I + YTY^T$, which yields $T^{-1} \,+\, T^{-T} \,=\, -X^T\, X$ [19].
The general algorithm corresponding to a block reduction is shown in Alg. 7. In this algorithm, the aggregation of the elementary Householder reflectors, and the block updating have to be chosen among the different available methods. The aggregation and update steps are detailed for each method as summarized below:

Method	Aggregate step	Update step
WY1	Alg. 8	Alg. 9
WY2	Alg. 10	Alg. 11
YTY1	Alg. 12	Alg. 13
YTY2	Alg. 14	Alg. 15

In our case, only the lower part of matrix X needs to be stored, in a $m \times p$ matrix $X2$, since

$$X \,\in \mathbb{C}^{2m \times p}, \quad X \,=\, \begin{bmatrix} I_m \\ X2 \end{bmatrix},$$

and $X2(:,j)$, for $j \,=\, 1:p$, stores the essential part x_2 of the Householder vector, as computed by Alg. 5.

Algorithm 7 BLOCK REDUCTION: $(A1, A2) = \texttt{block\_reduc}\,(A1, A2, k)$

```
Input: $p$: blocking factor
1: for $s_b = 1 : p : m$ do
2:    $e_b = \texttt{min}(s_b + p - 1, m)$    {end position for current bloc}
3:    $u1 = e_b + 1$    {update position for $A1$}
4:    $u2 = e_b + s2$    {update position for $A2$}
5:    $p_{eff} = \texttt{min}(p, m - sb + 1)$    {effective size of the block}
6:    for $j = 1 : p_{eff}$ do
7:       $j_1 = s_b + j - 1$    {column index for $A1$}
8:       $j_2 = s2 + s_b + j - 2$    {column index for $A2$}
9:       Determine $H_j$:
         $(X2_{:,j}, \beta, \alpha) = \texttt{house\_vec}(A1, A2, j_1, j_2)$
10:      Update $A(j, :)$:
         $A1_{j_1,j_1} \leftarrow \alpha, \quad A2_{j_2,:} \leftarrow 0.$
11:      Update current block:
         $\begin{pmatrix} A1_{j_1+1:e_b,:} & A2_{j_2+1:e_b+s_2-1,:} \end{pmatrix} \leftarrow \begin{pmatrix} A1_{j_1+1:e_b,:} & A2_{j_2+1:e_b+s_2-1,:} \end{pmatrix} H_j$
12:      Aggregate $H_j$
13:   end for
14:   Update $A1_{u1:e1,:}$ and $A2_{u2:e2,:}$ :
      $(A1, A2) = \texttt{block\_update}\,(A1, u_1, e_1, A2, u_2, e_2)$
15: end for
```

Algorithm 8 $(Y1, Y2) = \texttt{wy1\_aggregate}\,(H_j, j_1, j_2)$ see (13)

```
1:
   $Y1_{j_1,j} \leftarrow -\beta$
   $Y2_{:,j} \leftarrow -\beta\, X2_{1:m,j}$
   $vec_{1:j-1} = -\beta\, X2^T_{1:m,j}\, Y2_{1:m,1:j-1}$    {ZGEMV}
   $Y1_{j_1,1:j-1} \leftarrow Y1_{j_1,1:j-1} + vec_{1:j-1}$    {ZAXPY}
   $Y2_{1:m,1:j-1} \leftarrow Y2_{1:m,1:j-1} + X2_{1:m,j}\, vec_{1:j-1}$    {ZGERU}
```

Matrices Y and W have the form

$$Y, W \in \mathbb{C}^{2m\times p}, \quad Y = \begin{bmatrix} Y1 \\ Y2 \end{bmatrix}, \quad W = \begin{bmatrix} W1 \\ W2 \end{bmatrix},$$

and $Y1$ upper triangular, $W1$ lower triangular.

Algorithm 9 $(A1, A2) = $ `wy1_update` $(A1, u1, e1, A2, u2, e2,)$

```
$mat = A1_{u1:e1,s_b:e_b} + A2_{u2:e2,1:m}\, X2_{1:m,1:p_{eff}}$    {ZGEMM}
$A1_{u1:e1,s_b:e_b} \leftarrow A1_{u1:e1,s_b:e_b} + mat_{:,1:p_{eff}}\, Y1^T_{s_b:e_b,1:p_{eff}}$    {ZTRMM}
$A2_{u2:e2,1:m} \leftarrow A2_{u2:e2,1:m} + mat_{:,1:p_{eff}}\, Y2^T_{1:m,1:p_{eff}}$    {ZGEMM}
```

Algorithm 10 $(W1, W2) = $ `wy2_aggregate` (H_j, j_1, j_2) see (14)

```
1: $vec_{1:j-1} = -\beta\, X2^T_{1:m,1:j-1}\, X2_{1:m,j}$    {ZGEMV}
   $W1_{j_1,j} = -\beta$
   $W2_{:,j} = -\beta\, X2_{:,j}$
2: if $j > 1$ then
3:    $W1_{s_b:j_1-1,j} \leftarrow W1_{s_b:j_1-1,j}\, vec_{1:j-1}$    {ZTRMV}
4:    $W2_{:,j} \leftarrow W2_{:,j} + W2_{1:m,1:j-1}\, vec_{1:j-1})$    {ZGEMV}
5: end if
```

Algorithm 11 $(A1, A2) = $ `wy2_update` $(A1, u1, e1, A2, u2, e2,)$

```
$mat_{:,1:p_{eff}} = A1_{u1:e1,s_b:e_b}\, W1_{s_b:e_b,1:p_{eff}}$    {ZTRMM}
    $+ A2_{u2:e2,1:m}\, W2_{1:m,1:p_{eff}}$    {ZGEMM}
$A1_{u1:e1,s_b:e_b} \leftarrow A1_{u1:e1,s_b:e_b} + mat_{:,1:p_{eff}}$    {ZAXPY}
$A2_{u2:e2,1:m} \leftarrow A2_{u2:e2,1:m} + mat_{:,1:p_{eff}}\, X2_{1:m,1:p_{eff}}$    {ZGEMM}
```

5 Fast direct solver

We have developped fast techniques to compute the Cholesky factorization of a complex symmetric block Toeplitz matrix T. Since the n^2 block entries of T are described by the $2n$ blocks (of size $m \times m$) of the generator, the factorization complexity is reduced by an order of magnitude. But, the Cholesky factor is not structured and its storage requires $n(n+1)/2$ blocks. If one's aim is to solve the linear system:

$$T\,X = B,$$

Algorithm 12 $(T) = $ `yty1_aggregate` (H_j, j_1, j_2) see (15)

```
$T(j,j) = -\beta$
$vec_{1:j-1} = -\beta\, X2^T_{1:m,1:j-1}\, X2_{1:m,j}$    {ZGEMV}
$T_{j,1:j-1} \leftarrow T_{1:j-1,1:j-1}\, vec_{1:j-1}$    {ZTRMV}
```

Algorithm 13 $(A1, A2) = $ `yty1_update` $(A1, u1, e1, A2, u2, e2,)$

```
$mat_{:,1:p_{eff}} = A1_{u1:e1,s_b:e_b} + A2_{u2:e2,1:m}\, X2_{1:m,1:p_{eff}}$    {ZGEMM}
$mat_{:,1:p_{eff}} \leftarrow mat_{:,1:p_{eff}}\, T_{1;p_{eff},1:p_{eff}}$    {ZTRMM}
$A1_{u1:e1,s_b:e_b} \leftarrow A1_{u1:e1,s_b:e_b} + mat_{:,1:p_{eff}}$    {ZAXPY}
$A2_{u2:e2,1:m} \leftarrow A2_{u2:e2,1:m} + mat_{:,1:p_{eff}}\, X2^T_{1:m,1:p_{eff}}$    {ZGEMM}
```

Algorithm 14 $(invT) = $ `yty2_aggregate` (H_j, j_1, j_2) see (16)

```
1: $invT(:,:) \leftarrow -$triu$(I + X2^T\, X2)$    {ZSYRK}
2: for $j = 1 : p_{eff}$ do
3:     $invT(j,j) \leftarrow invT(j,j)/2$
4: end for
```

Algorithm 15 $(A1, A2) = $ `yty2_update` $(A1, u1, e1, A2, u2, e2,)$

```
$mat_{:,1:p_{eff}} = A1_{u1:e1,s_b:e_b} + A2_{u2:e2,1:m}\, X2_{1:m,1:p_{eff}}$    {ZGEMM}
$mat_{:,1:p_{eff}} \leftarrow mat_{:,1:p_{eff}} \,/\, invT_{1:p_{eff},1:p_{eff}}$    {ZTRMM}
$A1_{u1:e1,s_b:e_b} \leftarrow A1_{u1:e1,s_b:e_b} + mat_{:,1:p_{eff}}$    {ZAXPY}
$A2_{u2:e2,1:m} \leftarrow A2_{u2:e2,1:m} + mat_{:,1:p_{eff}}\, X2^T_{1:m,1:p_{eff}}$    {ZGEMM}
```

associated to the matrix T, one can use an alternative method (shown in [17] and [12] for the real case), which avoids the storage of the Cholesky factor. Instead of factoring T and then solving the linear system by a sequence of backward and forward substitution, we proceed through n steps of block factorization of an augmented structured matrix. This method is called the augmented matrix approach.

5.1 The augmented system

Lemma 5.1. *Let $M \in \mathbb{C}^{n\times n}$ be a matrix with complex entries. Then, for $B \in \mathbb{C}^{n\times nrhs}$, solving*

$$M\,X \,=\, B,$$

is equivalent to compute the Schur complement of M in the augmented matrix

$$\mathcal{M} \,\in\, \mathbb{C}^{2n\times(n+nrhs)}, \quad \mathcal{M} \,=\, \begin{pmatrix} M & -B \\ I_n & 0_{n\times nrhs} \end{pmatrix}.$$

Proof. The Schur complement of M in $\mathcal{M}$ is defined by

$$0_{n\times nrhs} - I_n\,M^{-1}\,(-B) \,=\, M^{-1}\,B \,=\, X.$$

□

If M is structured, its associated augmented matrix $\mathcal{M}$ is also structured. We choose M as the complex symmetric block Toeplitz matrix T, with $n \times n$ blocks of size $m \times m$. The augmented matrix $\mathcal{T}$ is the (non-square) matrix defined by

$$\mathcal{T} \,=\, \left(\begin{array}{cc} T & -B \\ I_{nm} & 0_{nm,nrhs} \end{array} \right) \tag{17}$$

and we have the following Lemma:

Lemma 5.2. *The augmented matrix $\mathcal{T}$ is structured with respect to the (non symmetric) displacement operator*

$$\begin{array}{rcl} \nabla_{Z_n^l,Z_n^r} : \mathbb{C}^{2nm\times(n+nrhs)} & \longrightarrow & \mathbb{C}^{2nm\times(n+nrhs)} \\ M & \mapsto & \nabla_{Z_n^l,Z_n^r}(M) \,=\, M - Z_n^l\,M\,Z_n^r, \end{array}$$

where the (square) matrices $Z_n^l \in \mathbb{C}^{nm\times nm}$ and $Z_n^r \in \mathbb{C}^{(nm+nrhs)\times(nm+nrhs)}$ are given by:

$$Z_n^l \,=\, \left(\begin{array}{cc} Z_n & 0_{nm,nm} \\ 0_{nm,nm} & Z_n \end{array} \right) \qquad Z_n^r \,=\, \left(\begin{array}{cc} Z_n^T & 0_{nm,nrhs} \\ 0_{nrhs,nm} & 0_{nrhs,nrhs} \end{array} \right).$$

Proof. The displacement of $\mathcal{T}$ is given by:

$$\nabla_{Z_n^l,Z_n^r}(\mathcal{T}) \,=\, \begin{pmatrix} \nabla_{Z_n}(T) & -B \\ \nabla_{Z_n}(I_{nm}) & 0 \end{pmatrix}.$$

We have

$$\nabla_{Z_n}(I_{nm}) = \begin{pmatrix} I_m & 0_m & \dots & 0_m \\ 0_m & 0_m & & \\ \vdots & & \ddots & \\ 0_m & & \dots & 0_m \end{pmatrix},$$

and its rank is equal to the block size m.
The displacement rank of $\mathcal{T}$ is thus equal to the displacement rank of T, and if T is structured, $\mathcal{T}$ is also structured. We recall that the displacement of T is defined by (3). □

The displacement of $\mathcal{T}$ can be factorized as

$$\nabla_{Z_n^l, Z_n^r}(\mathcal{T}) = \mathcal{A}\mathcal{B},$$

where $\mathcal{A} \in \mathbb{C}^{2nm \times (2m+nrhs)}$ and $\mathcal{B} \in \mathbb{C}^{(2m+nrhs)\times(nm+nrhs)}$ are constructed by identification:

$$\mathcal{A} = \left(\begin{array}{cc|c} \multicolumn{2}{c|}{A} & -B \\ \hline L_0^{-T} & i\,L_0^{-T} & \\ 0_{(n-1)m,m} & 0_{(n-1)m,m} & 0_{nm,nrhs} \end{array}\right) \tag{18}$$

and

$$\mathcal{B} = \left(\begin{array}{c|c} A^T & 0_{nm,nrhs} \\ \hline 0_{nrhs,nm} & I_{nrhs} \end{array}\right), \tag{19}$$

where A and L_0 are defined in (5) and (6).
We say that $(\mathcal{A}, \mathcal{B})$ is a generator *in proper form* of $\mathcal{T}$, associated to the displacement operator $\nabla_{Z_n^l, Z_n^r}$ if

- $\mathcal{A}, \mathcal{B} = \nabla_{Z_n^l, Z_n^r}(\mathcal{T})$,
- $\mathcal{A}_{12} = \mathcal{B}_{21} = 0_m$,
- $\mathcal{A}_{11}$ is lower triangular,
- $\mathcal{B}_{11}$ is upper triangular.

A proper generator is unique.
The next Lemma justifies that the successive block Schur complements of $\mathcal{T}$, denoted by $\mathcal{T}^{(k)}$, $k = [0 : n-1]$, are structured for the same kind of displacement operator than $\mathcal{T}$. It is the equivalent for $\mathcal{T}$ of Lemma 3.2 for T.

Lemma 5.3. *Let $\mathcal{T}$ the augmented matrix associated to T by (17). Let us set*

$$\mathcal{T}^{(0)} = \mathcal{T}.$$

There exists a sequence of proper generators:

$$(\mathcal{A}^{(k)},\, \mathcal{B}^{(k)}) \in \mathbb{C}^{(2n-k).m\times 2m} \times \mathbb{C}^{2m\times(n-k).m+nrhs}, \quad k \in [0:n-1],$$

such that:

$$\begin{aligned} \forall\, k \in [0:n-1], \quad \nabla_{Z^l_{n-k}, Z^r_{n-k}}(\mathcal{T}^{(k)}) \;&=\; \mathcal{T}^{(k)} - Z^l_{n-k}\, \mathcal{T}^{(k)}\, Z^r_{n-k} \\ &=\; \mathcal{A}^{(k)}\, \mathcal{B}^{(k)}, \end{aligned} \tag{20}$$

where Z^l_{n-k} and Z^r_{n-k} are defined by

$$Z^l_{n-k} \;=\; Z^l_{km+1:,km+1:}\,, \quad Z^r_{n-k} \;=\; Z^r_{km+1:,km+1:}\,.$$

Proof. The Lemma is shown by induction. The proof is very similar to that of Lemma 3.2. To simplify the notations, we show the result for $k = 0$ and deduce that it then remains true for $k = 1$.
For $k = 0$, the proper generator

$$(\mathcal{A}^{(0)}, \mathcal{B}^{(0)}) \;=\; (\mathcal{A}, \mathcal{B}),$$

is defined by (18) and (19). Since it is in proper form, we have the identity:

$$\mathcal{A}^{(0)}(:, 1:m) \;=\; \mathcal{T}^{(0)}(:, 1:m), \qquad \mathcal{B}^{(0)}(1:m, :) \;=\; \mathcal{T}^{(0)}(1:m, :). \tag{21}$$

Moreover, since T is symmetric,

$$\mathcal{A}^{(0)}(1:nm, 1:2m) \;=\; \mathcal{B}^{(0)}(1:2m, 1:nm)^T.$$

We now construct the proper generator $(\mathcal{A}^{(1)}, \mathcal{B}^{(1)})$, such that the Schur complement of order 1 satisfies:

$$\nabla_{Z^l_{n-1}, Z^r_{n-1}}(\mathcal{T}^{(1)}) \;=\; \mathcal{A}^{(1)}\, \mathcal{B}^{(1)}.$$

Let us partition $\mathcal{T}^{(0)}$ in:

$$\mathcal{T}^{(0)} \;=\; \begin{pmatrix} \mathcal{T}_{11} & \mathcal{T}_{12} \\ \mathcal{T}_{21} & \mathcal{T}_{22} \end{pmatrix},$$

where we have denoted

$$\begin{aligned} &\mathcal{T}_{11} \;=\; \mathcal{T}_{1:m,1:m},\ \mathcal{T}_{12} \;=\; \mathcal{T}_{1:m,m+1:nm+nrhs}, \\ &\mathcal{T}_{22} \;=\; \mathcal{T}_{m+1:2nm,m+1:nm+nrhs},\ \mathcal{T}_{21} \;=\; \mathcal{T}_{m+1:2nm,1:m}, \end{aligned}$$

We introduce the Cholesky factorization of the first block $\mathcal{T}_{11}$, which is also the first block of T:

$$\mathcal{T}_{11} \;=\; L_{11}\, L_{11}^T.$$

We define two (left and right) elimination matrices

$$M_l \;=\; \begin{pmatrix} L_{11}^{-1} & 0 \\ -\mathcal{T}_{21}\,\mathcal{T}_{11}^{-1} & I_{(2n-1).m} \end{pmatrix}, \qquad M_r \;=\; \begin{pmatrix} L_{11}^{-T} & -\mathcal{T}_{11}^{-1}\,\mathcal{T}_{12} \\ 0 & I_{(n-1).m+nrhs} \end{pmatrix},$$

such that

$$M_l\,\mathcal{T}^{(0)}\,M_r \;=\; \begin{pmatrix} I_m & 0_{(n-1).m+nrhs} \\ 0_{(2n-1).m} & \mathcal{T}^{(1)} \end{pmatrix}.$$

By multiplying the displacement equation (20) by M_l on the left and by M_r on the right, we get:

$$\begin{aligned} & \begin{pmatrix} I_m & 0_{m\times(n-1).m+nrhs} \\ 0_{(2n-1).m\times m} & \mathcal{T}^{(1)} \end{pmatrix} \\ + \quad & M_l\,Z_n^l\,M_l^{-1} \begin{pmatrix} I_m & 0_{m\times(n-1).m+nrhs} \\ 0_{(2n-1).m\times m} & \mathcal{T}^{(1)} \end{pmatrix} M_r^{-1}\,Z_n^r\,M_r \\ = \quad & (M_l\,\mathcal{A}^{(0)})\,(\mathcal{B}^{(0)}\,M_r). \end{aligned} \tag{22}$$

We compute

$$\begin{aligned} M_l\,Z_n^l\,M_l^{-1} \;=\; Z_n^l\,M_l^{-1} \quad &= \quad \begin{pmatrix} Z_n^l \begin{pmatrix} L_{11} \\ \mathcal{T}_{21}\,L_{11}^{-T} \end{pmatrix} & \begin{matrix} 0_{m\times(n-1).m} \\ Z_{n-1}^l \end{matrix} \end{pmatrix} \\ &= \quad \begin{pmatrix} Z_n^l\,\mathcal{T}^{(0)}(:,1:m) & \begin{matrix} 0_{m\times(n-1).m} \\ Z_{n-1}^l \end{matrix} \end{pmatrix} \end{aligned}$$

and

$$\begin{aligned} M_r^{-1}\,Z_n^r\,M_r \;=\; M_r^{-1}\,Z_n^r \quad &= \quad \begin{pmatrix} Z_n^r\,\left(L_{11}^T\,L_{11}^{-1}\,\mathcal{T}_{12}\right) \\ 0_{(n-1).m\times m} \quad Z_{n-1}^r \end{pmatrix} \\ &= \quad \begin{pmatrix} Z_n^r\,\mathcal{T}^{(0)}(1:m,:) \\ 0_{(n-1).m\times m} \quad Z_{n-1}^r \end{pmatrix}. \end{aligned}$$

We also have

$$M^l\,\mathcal{A}^{(0)} \;=\; \begin{pmatrix} I_m & 0_m \\ 0_{(2n-1).m\times m} & \mathcal{A}^{(0)}(m+1:,m+1:) \end{pmatrix}$$

and

$$\mathcal{B}^{(0)}\,M^r \;=\; \begin{pmatrix} I_m & 0_{(n-1).m+nrhs} \\ 0_m & \mathcal{B}^{(0)}_{m+1:,m+1:} \end{pmatrix},$$

since the generator $(\mathcal{A}^{(0)},\mathcal{B}^{(0)})$ is in proper form. Identifying the $(2,2)$ term in (22) leads to:

$$\mathcal{T}^{(1)} \;-\; Z_{n-1}^l\,\mathcal{T}^{(1)}\,Z_{n-1}^r \;=\; a^{(1)}\,b^{(1)},$$

with

$$a^{(1)} \;=\; \Big((Z_n^l\;\mathcal{T}^{(0)}_{:,1:m})(m+1:,1:m) \quad \mathcal{A}^{(0)}(m+1:,m+1:2m+nrhs) \Big)$$

and

$$b^{(1)} \;=\; \begin{pmatrix} (\mathcal{T}^{(0)}_{1:m,:}\;Z_n^r)\,(1:m,m+1:) \\ \mathcal{B}^{(0)}_{m+1:2m+nrhs,m+1:} \end{pmatrix}$$

By (21), $a^{(1)}$ and $b^{(1)}$ are also directly defined from $\mathcal{A}^{(0)}$ and $\mathcal{B}^{(0)}$ by:

$$a^{(1)} = \Big((Z_n^l\, \mathcal{A}^{(0)}_{:,1:m})(m+1:, 1:m) \quad \mathcal{A}^{(0)}_{m+1:,m+1:2m+nrhs} \Big)$$

and

$$b^{(1)} = \begin{pmatrix} (\mathcal{B}^{(0)}_{1:m,:}\, Z_n^r)\,(1:m, m+1:) \\ \mathcal{B}^{(0)}_{m+1:2m+nrhs,m+1:} \end{pmatrix}$$

The last step of the proof consists in putting the generator $(a^{(1)}, b^{(1)})$ in proper form.
We construct a transform $\mathcal{L}$, such that the pair

$$(a^{(1)}\,\mathcal{L}, \mathcal{L}^{-1}\,b^{(1)}) \overset{def}{=} (\mathcal{A}^{(1)}, \mathcal{B}^{(1)}), \tag{23}$$

is in proper form.
Let $Q \in \mathbb{C}^{2m\times 2m}$ be an orthogonal transform such that

$$(a^{(1)}_{:,1:2m}\,Q)(m+1:2m) = 0_m,$$

defined in a similar way as in the proof of Lemma 3.2. The transform $\mathcal{L}$ is defined in matrix form by:

$$\mathcal{L} \in \mathbb{C}^{(2m+nrhs)\times(2m+nrhs)}, \qquad \mathcal{L} = \begin{pmatrix} Q & \begin{matrix} -\alpha^{-1}\,\gamma \\ 0_{m\times nrhs} \end{matrix} \\ 0_{nrhs\times 2m} & I_{nrhs} \end{pmatrix},$$

where the blocks α and γ are defined by:

$$\alpha = a^{(1)}(1:m, 1:m), \quad \gamma = a^{(1)}(2m+1:2m+nrhs, 1:m).$$

When we apply $\mathcal{L}$ to $a^{(1)}$, the block $a^{(1)}(1:m, m+1:2m)$ is annihilated by the orthogonal transform Q, whereas the block $a^{(1)}(2m+1:2m+nrhs, 1:m)$ is annihilated by a simple Gaussian elimination.
Now the inverse of $\mathcal{L}$

$$\mathcal{L}^{-1} \in \mathbb{C}^{(2m+nrhs)\times(2m+nrhs)}, \qquad \mathcal{L} = \begin{pmatrix} Q^T & \begin{matrix} \alpha^{-1}\,\gamma \\ 0_{m\times nrhs} \end{matrix} \\ 0_{nrhs\times 2m} & I_{nrhs} \end{pmatrix},$$

has to be applied to $b^{(1)}$ from the left (see formula (23). Note that $b^{(1)}(2m+1: 2m+nrhs, 1:m)$ was already zero!
So, we have constructed a generator in proper form of $\mathcal{T}^{(1)}$, which ends the proof by induction. □

5.2 Fast direct solver

The fast direct solver described by Alg. 16 is based on Lemma 3.2. We compute the sequence of the proper generators $(\mathcal{A}^{(k)}, \mathcal{B}^{(k)})$ of the successive Schur

complements $\mathcal{T}^{(k)}$.
Therefore, for $k = 1 : n-1$, we set in proper form the generators:

$$\begin{aligned} a^{(k)} &= \Big((Z^l_{n-k+1}\, \mathcal{A}^{(k-1)}_{:,1:m})(m+1:,1:m) \quad \mathcal{A}^{(k-1)}_{m+1:,m+1:2m+nrhs} \Big) \\ b^{(k)} &= \begin{pmatrix} (\mathcal{B}^{(k-1)}_{1:m,:}\, Z^r_{n-k+1})\,(1:m, m+1:) \\ \mathcal{B}^{(k-1)}_{m+1:2m+nrhs,m+1:} \end{pmatrix} \end{aligned} \tag{24}$$

Remark 5. In practice, it is enough to put $a^{(k)}$ in proper form at each step of the algorithm. The other matrix $\mathcal{B}^{(k)}$ is deduced from $\mathcal{A}^{(k)}$ by inspection:

$$\mathcal{B}^{(k)} = \begin{pmatrix} {\mathcal{A}^{(k)}}^T_{1:(n-k).m,1:m} & x \\ {\mathcal{A}^{(k)}}^T_{1:(n-k).m,m+1:2m} & 0_{m\times nrhs} \\ 0_{m\times(n-k).m} & I_{nrhs} \end{pmatrix},$$

where x is some non-zero, but not important value: it is namely not neeeded to construct the next generator $(a^{(k+1)}, b^{(k+1)})$.

Hence, after n steps of fast Schur complementation, we get a generator of the solution X of the linear system:

$$T\,X = B,$$

and X is given by:

$$X = A^{(n-1)}(1:nm, 2m+1:2m+nrhs). \tag{25}$$

Algorithm 16 Fast Direct Solver

```
Input:  First block-row of $T$, of size $nm \times m$,
        Right hand side $B$, of size $nm \times nrhs$
Output: Solution $X$ of $T\,X = B$.
1: Set up generator $(\mathcal{A}, \mathcal{B})$ from formula (18) and (19).
2: for $k = 2 : n$ do
3:    Determine a generator $(a^{(k)}, b^{(k)})$ for the $k^{th}$ Schur complement, $\mathcal{T}^{(k)}$
      from formula (24).
4:    Put $(a^{(k)}, b^{(k)})$ in proper form:
      Determine $Q$ complex orthogonal such that:
      $(a^{(k)}\,Q)(1:m, m+1:2m) = 0_{m\times m}$.
      Apply $Q$ to the generator: $a^{(k)} \leftarrow a^{(k)}\,Q$.
      Gaussian elimination of $a^{(k)}_{1:m,2m+1:2m+nrhs}$.
5: end for
6: Get $X$ from formula (25).
```

6 Numerical Experiments

We have implemented our algorithms in Fortran90 and performed numerical experiments to investigate the behavior of the different options:

- blocked/unblocked methods
- with/without assembling the Cholesky factor

We compare their efficiency both in terms of accuracy and execution time to the other available algorithms: the classical Cholesky factorization and the Levinson solver for block complex general Toeplitz matrices [2], available from Netlib.
Throughout this section, we adopt the following naming scheme:

- `no_struct`: non-structured Cholesky method,
- `seq`: structured factorization with non-blocked Householder transforms,
- `wy1`, `wy2`, `yty1`, `yty2`: structured, factorization with blocked Householder transforms, for the corresponding block schemes.
- `levin`: Levinson solver from Netlib

We use Intel fortran compiler `ifc` (version 7.1) and Math Kernel Library optimized implementation of BLAS. Numerical experiments have been performed in double precision arithmetic. The tests were carried out on a bi-processor Intel Pentium 4 XEON, with 2 CPU at 3.05GHz, and a cache size of 512 KB, running Linux.
We consider complex symmetric block-Toeplitz matrices, generated with the BEM-based code `transdper` [20].

6.1 Cholesky factorization

We compute the Cholesky factor L of block-Toeplitz matrices, with blocks of size 20, by structured and non-structured methods.When a blocking scheme is used, the blocking factor (p) is chosen equal to the size of the Toeplitz blocks (m).
We show on Figure 1 the size of the relative residual

$$R = \frac{||L\,L^T - T||_F}{||T||_F}$$

for the different structured approaches, in Frobenius norm. We see that their accuracy slightly depends on the (blocked/unblocked) method for applying Householder transforms, but they all reasonably low.
Nevertheless, when we compare any structured method with the classsical Cholesky method, we note that the relative size of the factorization residual obtained via the structured method is significantly higher. This confirms the accuracy results obtained by Kressner in [16], for the real case. But in spite of this, the structured methods give sufficiently accurate factorizations. The execution time is shown on Figure 2.

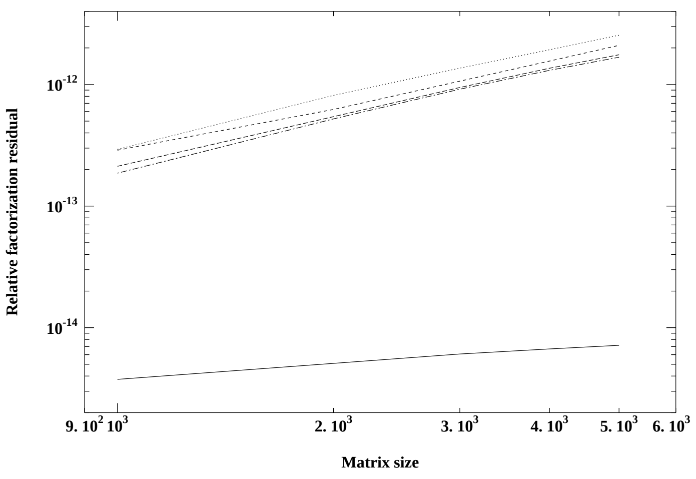

Figure 1: Relative residual of the factorization (Frobenius norm)

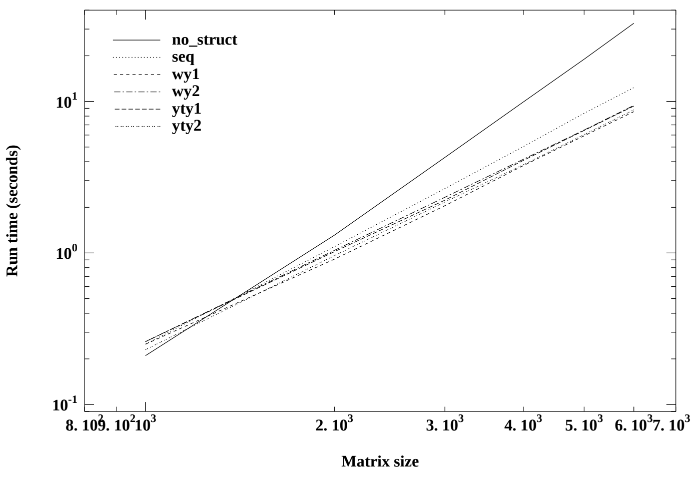

Figure 2: Execution time for the factorization

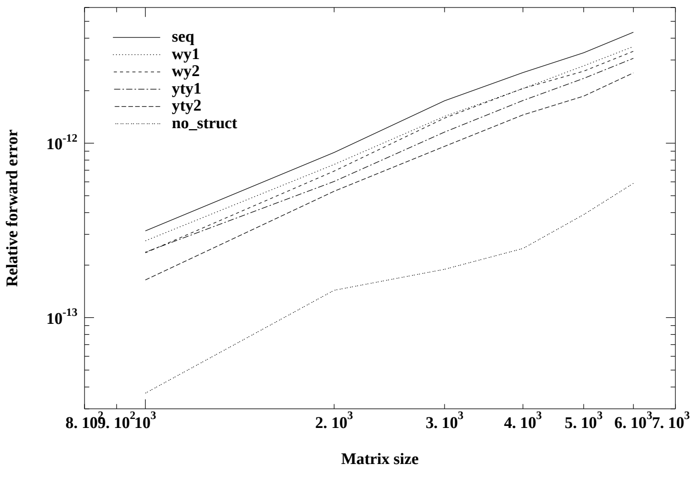

Figure 3: Relative forward error, with forming the Cholesky factor

6.2 Solution of the linear system, with an explicit factorization of the coefficients matrix

In this section, we solve linear systems

$$T\,X \,=\, B. \tag{26}$$

associated to Toeplitz matrices of different sizes. We proceed as follows:

- Cholesky factorization of T, by the various structured/non-structured methods displayed in the last paragraph.
- solution of the system by forward/backward substitutions.

We build a random exact solution X_{exact} and define the right-handside from X_{exact} by:

$$B \,=\, T\,X_{exact}.$$

We then present the relative forward error, defined in 2-norm by

$$Ferr \,=\, \frac{||X - X_{exact}||_2}{||X_{exact}||_2},$$

where X denotes the computed solution of the linear system (26). The matrices we consider have (Toeplitz) blocks of size 20. The blocking factor, used to accumulate the Householder transforms is taken equal to the size of the Toeplitz blocks.
We present on Figure 3 the relative forward error, for each method, structured and non-structured. We see that the accuracy is reasonable, though inferior to the non-structured method. This confirms what we observed on the factorization

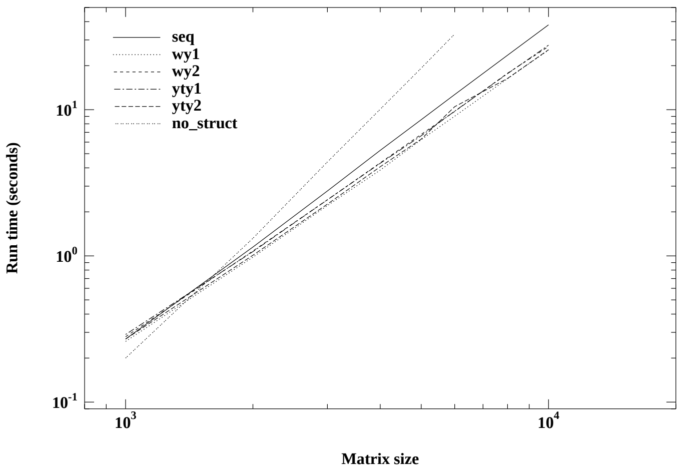

Figure 4: Execution time, with forming the Cholesky factor

residuals, in the last paragraph.
The timing results are shown on Figure 4. The difference in the slopes of the curves corresponding to the structured/non-structured methods represents the difference in the complexity of the algorithms. For sufficiently large systems, the structured methods are clearly faster, especially the blocked implementations. The tests we performed do not show a significant advantage of one of the blocked methods.

6.3 Solution of the linear system, without an explicit factorization of the coefficients matrix

In this section, we test the fast solver presented in section 5. This method allows to solve the linear system

$$T\,X \,=\, B,$$

without explicitly forming the Cholesky factor of T. Different techniques are investigated, corresponding to the way the generator of the augmented system is put in proper form. It is especially attractive when large systems are to be solved, since it only demands a small amount of memory.
We build a random exact solution X_{exact} and define the right-handside from X_{exact} by:

$$B \,=\, T\,X_{exact}.$$

The matrices we consider have (Toeplitz) blocks of size 20. The blocking factor, used to accumulate the Householder transforms is taken equal to the size of the Toeplitz blocks.
The relative forward error is shown on Figure 5. It has the same order of magnitude than the error obtained with the solver tested in the last paragraph

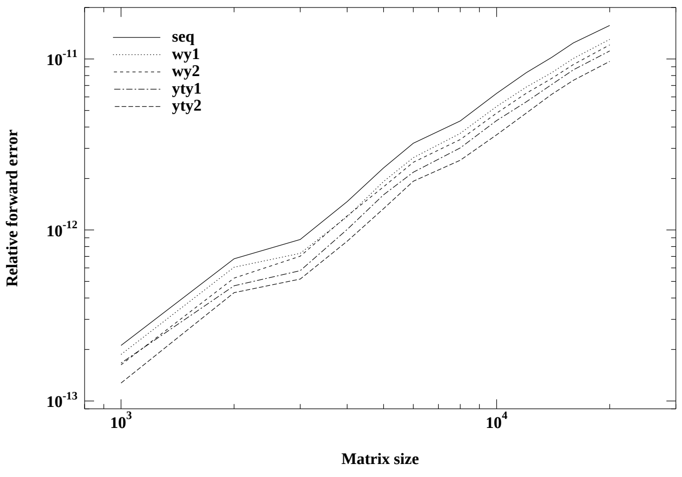

Figure 5: Relative forward error, without forming the Cholesky factor

(with assembling the Cholesky factor, before solving the system). But, since it requires less memory, it has been tested on larger matrices. The timing results are reported on Figure 6. The non-blocked implementation again appears slower than the blocked implementations.

6.4 When is it worth considering the structure of a matrix?

When is a "fast" structured method really faster than a non-structured one? Let us again consider the linear problem:

$$T\,X = B$$

for a block Toeplitz matrix T of $n \times n$ blocks, each of size $m \times m$.
When $n = 1$, the matrix is in fact only symmetric (not Toeplitz) and the proper way to solve (6.4) is a non-structured Cholesky factorization. We guess that when n is small, that is the matrix has only a few blocks, a structured method is not very efficient: it is clear on the beginning of Figure 4. On the opposite, $m = 1$ means that the matrix is point Toeplitz, and that a structured method is probably effective.
We verify numerically this idea on two examples. Therefore, we fix a block size m, and plot the CPU time required to solve (6.4) by structured and unstructured methods, for different values of n. We have chosen the structured solvers based on an explicit assembling of the Cholesky factor, to have a fair comparison with the non-structured method. The result for block sizes $m = 100$, $m = 199$, and $m = 451$ is respectively shown on Figures 7, 8 and 9. We recall that the case of blocks of size 20 corresponds to Figure 4. We see that when n increases, structured methods become attractive. The blocked structured algorithms achieve

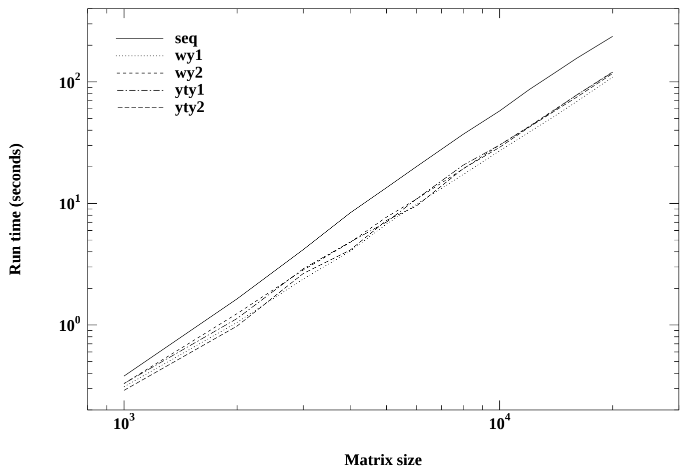

Figure 6: Execution time, without forming the Cholesky factor

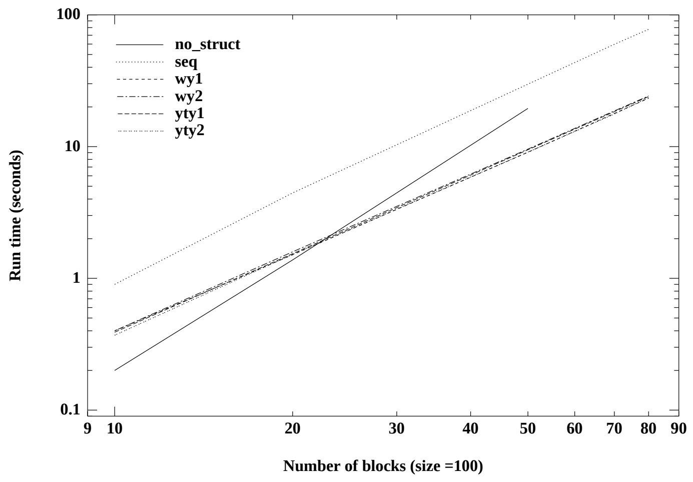

Figure 7: Execution time, matrices with blocks of size 100

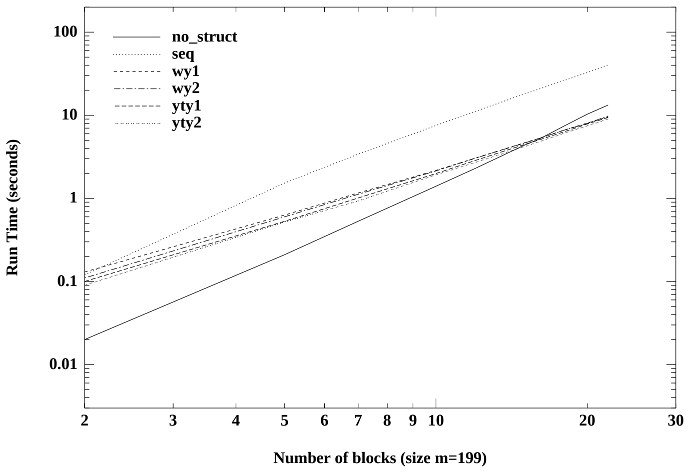

Figure 8: Execution time, matrices with blocks of size 199

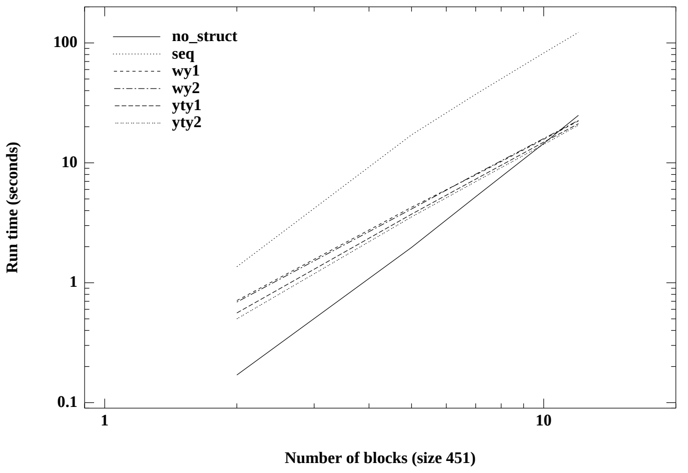

Figure 9: Execution time, matrices with blocks of size 451

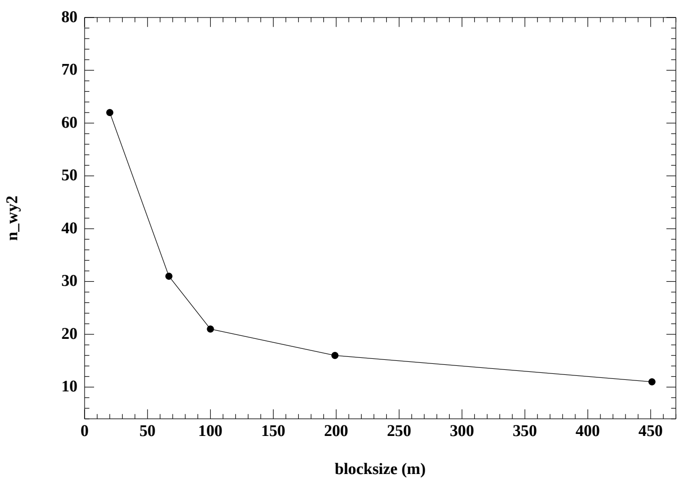

Figure 10: The function $m \mapsto n_{wy2}(m)$

crossover points at which they beat the non-structured algorithm.
Let us now define, for one of the blocked structured method (for instance $wy2$), the mapping:

$$m \mapsto n_{wy2}(m) \;=\; \min\{n,\; T_{wy2}(n,m) \leq T_{no\_struct}(n,m)\}$$

where $T_{wy2}(n,m)$ (resp. $T_{no\_struct}(n,m)$) is the CPU time needed to solve the linear sytem associated to a Toeplitz matrix of $n \times n$ blocks, each of size $m \times m$ by the $wy2$-blocked structured method (resp. by the non-structured method). We plot this function on Figure 10, and see that it is decreasing.

6.5 Comparison with the general Levinson solver `tgslz`

We compare the solver based on the `yty2`-blocked version of Algorithm 16 (without an explicit assembly of the Cholesky factor) with the structured solver `tgslz` for general block Toeplitz systems [2] available on Netlib. This solver is of Levinson type, that is it is based on a factorization of T^{-1} (and not T, as in Schur type algorithms). This solver was previously used in the BEM-based code `transdper` [20]. Note that `tgslz` deals with general matrices whereas the new Schur type algorithm is adapted to symmetric matrices and should therefore be at least twice faster.
Timing results are reported on Figure 11 and on Figure 12. On Figure 11, we have plotted the CPU time required to solve the linear system associated to a block Toeplitz matrix, with n (variable) blocks of (fixed) size m. On Figure 12, we consider matrices of n (fixed) blocks of (variable) size m. Hence, Figure 11 points out the behaviour of the algorithm with respect to the *number* of blocks n, whereas Figure 12 points out the behaviour of the algorithm with respect to the *size* of blocks m. We see on Figure 11 that except for the smallest size of

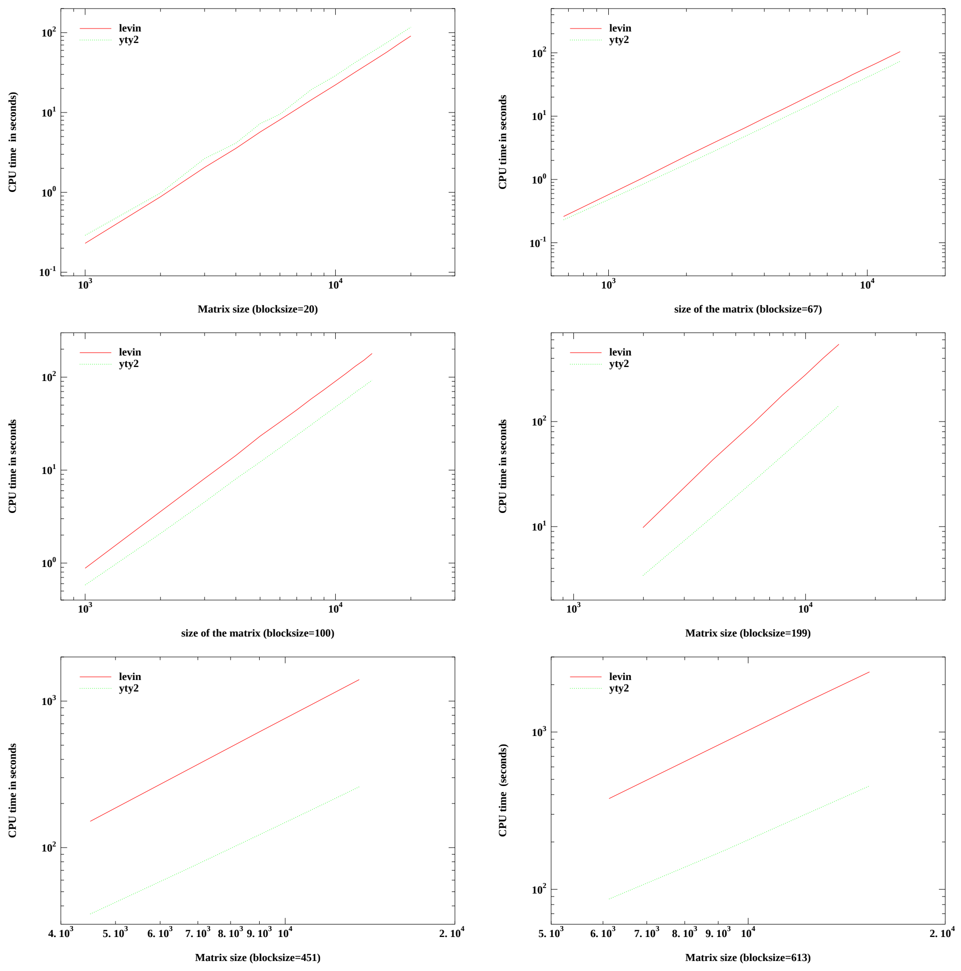

Figure 11: Levinson/symmetric Schur algorithm: Run time w.r.t m (block size)

m	n	Relative forward error Levinson algorithm	Relative forward error yty2 – Schur algorithm
67	50	$8.33\,10^{-13}$	$3.91\,10^{-12}$
	100	$4.05\,10^{-12}$	$8.06\,10^{-12}$
	150	$1.07\,10^{-11}$	$1.29\,10^{-11}$
100	50	$6.74\,10^{-13}$	$2.20\,10^{-12}$
	100	$3.45\,10^{-12}$	$5.29\,10^{-12}$
	140	$7.13\,10^{-12}$	$8.19\,10^{-12}$
451	10	$2.54\,10^{-12}$	$5.10\,10^{-11}$
	30	$7.26\,10^{-12}$	$1.27\,10^{-11}$

Table 1: Levinson/symmetric Schur algorithm: relative forward error

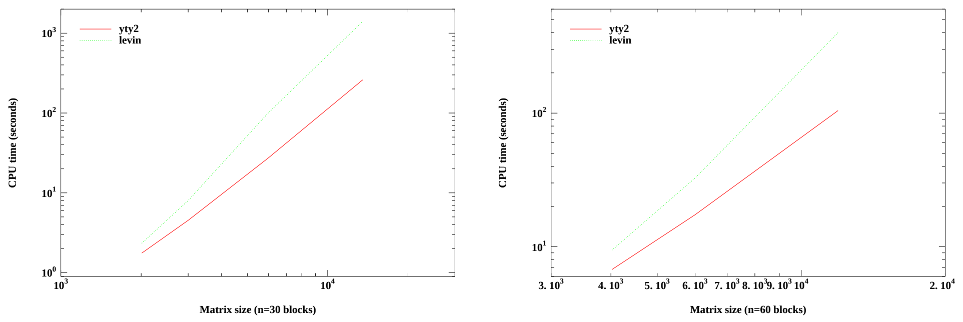

Figure 12: Levinson/symmetric Schur algorithm: Run time w.r.t n (number of blocks)

blocks, the new algorithm outperforms the old one. The larger the blocksize, the better the new algorithm behaves: it is slower for blocks of size 20, but for blocks of size 451, it is about 5 times faster! The slope of the timing curves are identical: both algorithms have the same dependence with respect to n. The accuracy remains the same: Table 1 compares the relative forward error for both methods. Figure 12 confirms what we observed on Figure 11: the superiority of our symmetric Schur algorithm becomes obvious when m increases. This time, the slopes of the timing curves are different: the curve corresponding to our algorithm has a smaller slope, which means that its order with respect to m is lower than Levinson algorithm's. This explains why Schur algorithm is faster for m large.

7 Conclusions

We proposed fast algorithms for the solution of dense complex symmetric block Toeplitz systems. These linear systems arise for instance in the analysis of electromagnetic radiation and scattering problems by Boundary Element Methods.

Memory and CPU-time requirements by classical direct (Cholesky) method are quite high. Therefore, exploiting the structure of the matrix, when it exists due to a periodic geometry, is very attractive. Our numerical experiments demonstrate the efficiency of fast algorithms both in terms of accuracy and run time for several matrices issued from the discretization of Boundary Integral Equations. Moreover, we believe that Algorithm 16 is particularly interesting, since it avoids an explicit assembly of the Cholesky factor, and thereby demands less memory space. Therefore, using this algorithm for the numerical solution of the discretized BEM problems both allows to consider problems of larger size and to speed up their solution. We also pointed out that blocked algorithms should be preferred since they are significantly faster than their unblocked version.

References

[1] E. Anderson, Z. Bai, C.Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, third edition, 1999.

[2] O. Arushanian, M. Samarin, V. Voevoedin, E. Tyrtyshnikov, B. Garbow, J. Boyle, W. Coweli, and K. Dritz. The toeplitz package users' guide. Technical report, Argonne National Laboratory, 1983.

[3] A. Bendali. Numerical analysis of the exterior boundary value problem for the time-harmonic maxwell equations by a boundary finite element method, part 1: The continuous problem, part 2: The discrete problem. *Mathematics of Computation*, 43(167):29–68, 1984.

[4] N. Béreux. A cholesky algorithm for some complex symmetric systems. Technical Report 515, CMAP, École Polytechnique, October 2003.

[5] C. Bischof. A summary of block schemes for reducing a general matrix to hessenberg form. Technical Report 175, Argonne National Laboratory, 1993.

[6] C. Bischof and C. Loan. The wy representation for products of householder matrices. *SIAM J. Scientific and Statistical Computing*, 8:2–13, 1987.

[7] J. R. Bunch and L. Kauffman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31:162–179, 1977.

[8] S. H. Christiansen. Discrete fredholm properties and convergence estimates for the electric field integral equation. *Mathematics of Computation*, 73:143–167, 2004.

[9] S. H. Christiansen and J.-C. Nédélec. A preconditioner for the electric field integral equation based on calderon formulas. *SIAM Journal on Numerical Analysis*, 40(3):1100–1135, 2002.

[10] E. Darve. The fast multipole method (i) : Error analysis and asymptotic complexity. *SIAM Numerical Analysis*, 38(1):98–128, 2000.

[11] B. Friedlander, M. Morf, T. Kailath, and L. Ljung. New inversion formulas for matrices classified in terms of their distance from toeplitz matrices. *Linear Algebra and Applications*, 27:31–60, 1979.

[12] K. Gallivan, S. Thirumalai, P. V. Dooren, and V. Vermaut. High performance algorithms for toeplitz and block toeplitz matrices. *Linear Algebra and its Applications*, 241/243:343–388, 1996.

[13] G. H. Golub and C. F. V. Loan. *Matrix computations.* John Hopkins University Press, third edition, 1996.

[14] T. Kailath and A. Sayed, editors. *Fast Reliable Algorithms for Matrices with Structure.* SIAM, 1999.

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

[16] D. Kressner. *Numerical methods for structured matrix factorizations.* PhD thesis, Technische Universität Chemnitz, 2001.

[17] D. Kressner and P. V. Dooren. Factorizations and linear system solvers for matrices with toeplitz structure. Technical report, SLICOT working note, 2001.

[18] J.-C. Nédélec. *Acoustic and Electromagnetic Equations : Integral Representations for Harmonic Problems.* Springer Verlag, 2001.

[19] C. Puglisi. Modification of the householder method based on the compact *wy* representation. *Siam J. Matrix Anal. Appl.*, 3:723–726, 1992.

[20] J. Ribbe. *On the coupling of integral equations and finite elements/Fourier modes for the simulation of piezoelectric surface acoustic wave components.* Thèse de doctorat, École Polytechnique, France, 2002.

[21] R. Schreiber and C. Loan. A storage efficient wy representation for products of householder transformations. *SIAM J. Scientific and Statistical Computing*, 10:53–57, 1989.

[22] M. Stewart and P. V. Dooren. Stability issues in the factorization of structured matrices. *SIAM Journal on Matrix Analysis and Applications*, 18(1):104–118, 1997.

[23] P. E. Strazdins. A dense complex symmetric indefinite solver for the Fujitsu AP3000. Technical Report TR-CS-99-01, Canberra 0200 ACT, Australia, 1999.

[24] H. Walker. Implementation of the gmres method using householder transformations. *SIAM J. Scientific and Statistical Computing*, 1, 152–163 1988.