Inverting Block Toeplitz Matrices in Block Hessenberg Form by Means of Displacement Operators: Application to Queueing Problems

Dario Andrea Bini and Beatrice Meini
Dipartimento di Matematica, Università di Pisa
bini@dm.unipi.it, meini@dm.unipi.it

Abstract

The concept of displacement rank is used to devise an algorithm for the inversion of an $n \times n$ block Toeplitz matrix in block Hessenberg form H_n having $m \times m$ block entries. This kind of matrices arises in many important problems in queueing theory. We explicitly relate the first and the last block rows and block columns of H_n^{-1} with the corresponding ones of $H_{\frac{n}{2}}^{-1}$. These block vectors fully define all the entries of H_n^{-1} by means of a Gohberg-Semencul-like formula. In this way we obtain a doubling algorithm for the computation of $H_{2^i}^{-1}$, $i = 0, 1, \ldots, q$, $n = 2^q$, where at each stage of the doubling procedure only few convolutions of block vectors must be computed. The overall cost of this computation is $O(m^2 n \log n + m^3 n)$ arithmetic operations with a moderate overhead constant. The same technique can be used for solving the linear system $H_n \mathbf{x} = \mathbf{b}$ within the same computational cost. The case where H_n is in addition a scalar Toeplitz matrix is analyzed as well. An application to queueing problems is presented and comparisons with existing algorithms are performed showing the higher efficiency and reliability of this approach.

Key words: Block Toeplitz matrices, block Hessenberg, displacement rank, queueing problems, divide-and-conquer.

1 Introduction

Block Toeplitz matrices are frequently encountered in the modeling of problems of the real world where some sort of shift invariance holds. The shift invariance property is captured by the block Toeplitz structure where the block entries $T_{i,j}$ of the matrix T are invariant under shift along the diagonal direction, that is $T_{i,j}$ is a function of $j - i$.

In queueing theory block Toeplitz matrices play a relevant role, in fact they model a wide and important class of problems (the one described by M/G/1 type or by G/M/1 type Markov chains [14], [13]) where the block Toeplitz matrix has also the block Hessenberg structure.

One of the major problems for M/G/1 Markov chains consists in computing the first block row of the inverse of the $n \times n$ block matrix

$$H_n = \begin{bmatrix} A_1 & A_0 & & & \bigcirc \\ A_2 & A_1 & A_0 & & \\ \vdots & \ddots & \ddots & \ddots & \\ A_{n-1} & \ddots & \ddots & A_1 & A_0 \\ A_n & A_{n-1} & \dots & A_2 & A_1 \end{bmatrix} \tag{1}$$

where A_i are $m \times m$ blocks and n generally takes very large values. In fact, for Markov chains endowed with an infinite set of states, the block Toeplitz matrix involved is infinite and we are interested in the asymptotic behavior of the inverse of H_n, for $n \to +\infty$.

Motivated by the large size of the matrices arising in the applications, we are interested in devising fast and efficient algorithms for inverting block Toeplitz matrices in block Hessenberg form.

In [15], [12] G. W. Stewart and G. Latouche have introduced, analyzed and implemented a recursive algorithm, that relies on the Sherman Woodbury Morrison formula, for solving systems in block Hessenberg form. This method, based on a doubling technique, has been adapted to block Toeplitz matrices and used for solving M/G/1 type Markov chains. The resulting cost of the Latouche and Stewart algorithm for solving the system $H_n\mathbf{x} = \mathbf{b}$ is reduced to $O(m^3 n \log^2 n)$ arithmetic operations if Fast Fourier Transform (FFT) is used for performing computations among block Toeplitz matrices. However, this adaptation does not fully exploit the block Toeplitz structure of the matrix H_n. In fact, this structure is only used for the fast computation of the products between Toeplitz matrices and vectors.

The concept of displacement rank, introduced by Kailath et al. in [11], is fundamental in devising and analyzing algorithms related to Toeplitz matrices. An excellent survey on the properties and the applications of the displacement rank, together with a very rich bibliography, is presented by Kailath and Sayed in [10]. Applications to parallel computations, polynomial computations and Toeplitz/Hankel-like matrices can be found in [6], [5].

The invariance under inversion of the displacement rank of a block Toeplitz matrix allows us to fully exploit the structure of H_n.

In fact, by using the block displacement operator $\Delta(H) = ZH - HZ$, where Z is the block down-shift matrix, we provide an inversion formula for the matrix H_n based on the properties of $\Delta(H_n)$. More specifically, we show that the inverse of H_n can be explicitly represented in terms of its first and last block rows $R_n^{(1)}$,

$R_n^{(2)}$, and its first and last block columns $C_n^{(1)}$, $C_n^{(2)}$, by means of the relation

$$H_n^{-1} = L_n(C_n^{(1)})L_n^T(E_n^{(1)} + Z_n R_n^{(1)T} A_0^T) - L_n(C_n^{(2)})L_n^T(Z_n R_n^{(2)T} A_0^T) \qquad (2)$$

where $L_n(W)$ denotes the $n \times n$ block lower triangular block Toeplitz matrix defined by its first block column W, and $E_n^{(1)}$ denotes the $nk \times k$ matrix made up by the first k columns of the identity matrix of dimension nk.

Then we explicitly relate the block vectors $R_n^{(1)}$, $R_n^{(2)}$, $C_n^{(1)}$, $C_n^{(2)}$, to the block vectors $R_{\frac{n}{2}}^{(1)}$, $R_{\frac{n}{2}}^{(2)}$, $C_{\frac{n}{2}}^{(1)}$, $C_{\frac{n}{2}}^{(2)}$, defining $H_{\frac{n}{2}}^{-1}$, and exploit these relations in order to devise a doubling algorithm for the computation of H_n^{-1}.

Since the equations relating block rows and block columns of H_n^{-1} and $H_{\frac{n}{2}}^{-1}$ involve products between block Toeplitz matrices and block vectors, we may implement these relations by using FFT-based techniques. In this way we devise an algorithm for the computation of H_n^{-1} that requires $O(m^2 n \log n + m^3 n)$ arithmetic operations, where the hidden constant in the asymptotic estimate has a moderate size.

Once the representation (2) of H_n^{-1} has been computed, we may solve any linear system $H_n \mathbf{x} = \mathbf{b}$ with an additional cost of $O(m^2 n \log n)$ arithmetic operations.

Moreover, we consider the important case where the matrix H_n is also Toeplitz in the scalar sense. This situation occurs in the modeling of certain queueing problems [9]. We show that, under this further assumption, the doubling algorithm can be strongly simplified since only the first and last rows and columns of H_n^{-1} must be computed. In this way we reduce the computational cost to $O(m^2 \log n + mn \log(mn))$ arithmetic operations.

We have implemented our algorithm in Fortran 90 and compared it to the algorithm of [15] on a problem arising from the modeling of metropolitan networks [1]. Our algorithm, besides being faster than the one of [15] by a factor of 6.8, requires less storage, and therefore allows us to deal with larger values of n, and provides more accurate results.

The paper is organized as follows. In section 2 we recall the equations, based on the Sherman Woodbury Morrison formula, relating the matrices H_n^{-1} and $H_{\frac{n}{2}}^{-1}$, on which the algorithm of [15], [12] relies. In section 3 we introduce the block displacement operator $\Delta(H)$, analyze its properties and obtain the representation formula (2) expressing the matrix H_n^{-1} by means of the block vectors $R_n^{(1)}$, $R_n^{(2)}$, $C_n^{(1)}$, $C_n^{(2)}$. In section 4, after recalling FFT-based procedures for performing computations between block Toeplitz matrices, we describe and analyze our algorithm. In section 5 we show an application to the solution of queueing problems, analyze the modifications needed to deal with the case of scalar Toeplitz matrices and present numerical results and comparisons.

2 A divide and conquer algorithm

Let A_i, $i = 0, 1, \ldots, n$, be $m \times m$ matrices and consider the block Toeplitz matrix in block Hessenberg form H_n of (1). Let us assume for simplicity that $n = 2^q$, for a positive integer q. Suppose that $\det H_{2^j} \neq 0$, $j = 0, 1, \ldots, q$, and partition the matrix H_n in the following way

$$H_n = \begin{bmatrix} H_{\frac{n}{2}} & U_{\frac{n}{2}} V_{\frac{n}{2}}^T \\ T_{\frac{n}{2}} & H_{\frac{n}{2}} \end{bmatrix} \tag{3}$$

where

$$U_{\frac{n}{2}} = \begin{bmatrix} O \\ \vdots \\ O \\ I \end{bmatrix}, V_{\frac{n}{2}} = \begin{bmatrix} A_0^T \\ O \\ \vdots \\ O \end{bmatrix}, \quad T_{\frac{n}{2}} = \begin{bmatrix} A_{\frac{n}{2}+1} & A_{\frac{n}{2}} & \ldots & A_2 \\ A_{\frac{n}{2}+2} & A_{\frac{n}{2}+1} & \ddots & A_3 \\ \vdots & \ddots & \ddots & \vdots \\ A_n & A_{n-1} & \ldots & A_{\frac{n}{2}+1} \end{bmatrix} \tag{4}$$

and I denotes the $m \times m$ identity matrix. By applying the Sherman Woodbury Morrison formula [8] to the decomposition

$$\begin{aligned} H_n &= \begin{bmatrix} H_{\frac{n}{2}} & O \\ T_{\frac{n}{2}} & H_{\frac{n}{2}} \end{bmatrix} + \begin{bmatrix} O & U_{\frac{n}{2}} V_{\frac{n}{2}}^T \\ O & O \end{bmatrix} = S_n + M_n N_n^T, \\ M_n &= \begin{bmatrix} U_{\frac{n}{2}} \\ O \end{bmatrix}, N_n = \begin{bmatrix} O \\ V_{\frac{n}{2}} \end{bmatrix} \end{aligned} \tag{5}$$

we immediately find the following expression for the matrix inverse of H_n:

$$H_n^{-1} = S_n^{-1} - S_n^{-1} M_n (I + N_n^T S_n^{-1} M_n)^{-1} N_n^T S_n^{-1}, \tag{6}$$

where

$$S_n^{-1} = \begin{bmatrix} H_{\frac{n}{2}}^{-1} & O \\ -H_{\frac{n}{2}}^{-1} T_{\frac{n}{2}} H_{\frac{n}{2}}^{-1} & H_{\frac{n}{2}}^{-1} \end{bmatrix}. \tag{7}$$

The above formulae have been used by Stewart [15] in order to devise an efficient doubling method for solving the block Hessenberg system $H_n \mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ and $\mathbf{b}$ are mn-dimensional vectors, and applied by Latouche and Stewart [12] to solving queueing problems.

In fact, partitioning $\mathbf{x}$ and $\mathbf{b}$ in two $nm/2$ dimensional vectors such that $\mathbf{x}^T = [\mathbf{x}_1^T, \mathbf{x}_2^T]$, $\mathbf{b}^T = [\mathbf{b}_1^T, \mathbf{b}_2^T]$, and denoting $\mathbf{y}_1 = H_{\frac{n}{2}}^{-1} \mathbf{b}_1$, $\mathbf{y}_2 = H_{\frac{n}{2}}^{-1} (\mathbf{b}_2 - T_{\frac{n}{2}} \mathbf{y}_1)$, $Y_3 = H_{\frac{n}{2}}^{-1} U_{\frac{n}{2}}$, $Y_4 = -H_{\frac{n}{2}}^{-1} T_{\frac{n}{2}} Y_3$, from (6), (7) we easily find that

$$\mathbf{x} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} - \begin{bmatrix} Y_3 \\ Y_4 \end{bmatrix} (I + V_{\frac{n}{2}}^T Y_4)^{-1} V_{\frac{n}{2}}^T \mathbf{y}_2. \tag{8}$$

In this way the solution of an $n \times n$ block Hessenberg block Toeplitz system is reduced to solving the two $\frac{n}{2} \times \frac{n}{2}$ block Hessenberg block Toeplitz systems

yielding $\mathbf{y}_1$ and $\mathbf{y}_2$, to solving the block Hessenberg block Toeplitz systems $H_{\frac{n}{2}} Y_3 = U_{\frac{n}{2}}$ and $H_{\frac{n}{2}} Y_4 = -T_{\frac{n}{2}} Y_3$. Observe that all the above systems are associated with the same matrix $H_{\frac{n}{2}}$ and that the latter two block systems do not depend on the vector $\mathbf{b}$.

Now, denote with p_n the cost of computing Y_3 and Y_4, with s_n the arithmetic cost of computing $\mathbf{x}$ by means of (8) once the block vectors Y_3 and Y_4 have been computed, finally with c_n the overall cost of computing $\mathbf{x}$. In this way from the equations (6), (8) and (7) we have

$$\begin{aligned} c_n &= s_n + p_n + O(m^2 n + m^3) \\ s_n &= 2s_{\frac{n}{2}} + O(m^2 n \log n) \\ p_n &= 2ms_{\frac{n}{2}} + p_{\frac{n}{2}} + O(m^2 n \log n + m^3 n), \end{aligned}$$

where the products between block Toeplitz matrices and (block) vectors are computed by means of FFT's, as shown in section 4, for the cost $O(m^2 n \log n + m^3 n)$.

From the above relations we obtain $s_n = O(m^2 n \log^2 n)$, whence $p_n = p_{\frac{n}{2}} + O(m^3 n \log^2 n)$. Therefore we deduce that $p_n = O(m^3 n \log^2 n)$ and $c_n = O(m^3 n \log^2 n)$.

In the following section, by using a suitable displacement operator, we show how the above doubling technique can be modified in order to obtain a simpler algorithm having a computational cost of $O(m^3 n + m^2 n \log n)$ operations.

3 Displacement properties

Let us recall some properties of displacement operators which are particularly suitable for dealing with the inversion of the matrix H_n. For more details on such operators we refer the reader to the pioneering paper [11], to the excellent survey given by Kailath and Sayed [10], and more specifically, concerning the operator that we use and adjust to this application, to the book [6] and to [2], [5].

Define the $n \times n$ *block down-shift* matrix

$$Z_n = \begin{bmatrix} O & & & \\ I & O & & \\ & \ddots & \ddots & \\ & & I & O \end{bmatrix},$$

where the blocks have dimension m, and consider the following *block displacement operator*

$$\Delta(H) = Z_n H - H Z_n$$

defined for any $n \times n$ block matrix H.

Let us also introduce the following notations: the block vectors $E_n^{(1)} = [I, O, \ldots, O]^T$, $E_n^{(2)} = [O, \ldots, O, I]^T$ denote $mn \times m$ matrices made up by the first and the last m columns of the identity matrix of order mn. Throughout we call n-dimensional *block column vector* any $mn \times m$ matrix, similarly we call n-dimensional *block row vector* any $m \times mn$ matrix. Moreover we say that the block matrix H has block displacement rank r if r is the minimum integer such that there exist block column vectors $U^{(i)}$, and block row vectors $V^{(i)}$, $i = 1, \ldots, r$, satisfying the equation $\Delta(H) = \sum_{i=1}^{r} U^{(i)} V^{(i)}$.

For $m = 1$ the block displacement rank of H is reduced to the well-known concept of displacement rank that coincides with the rank of $\Delta(H)$, where Δ is the (scalar) displacement operator.

It is immediate to observe that for the matrix H_n of (1) it holds

$$\Delta(H_n) = \begin{bmatrix} I \\ O \\ \vdots \\ O \end{bmatrix} [A_0, O, \ldots, O] - \begin{bmatrix} O \\ \vdots \\ O \\ I \end{bmatrix} [O, \ldots, O, A_0]. \tag{9}$$

Observe that $\Delta(H_n)$ only depends on the block entry A_0 of the matrix H_n and is strongly structured. This fact in principle might seem a draw-back of this operator since the informations contained in the remaining blocks A_i are lost. However, it can be used, in the light of theorem 1, to give a simple representation of H_n^{-1}, well suited for block Hessenberg matrices.

A nice property of $\Delta(H)$ is that the block displacement of the inverse matrix can be explicitly related to the block displacement of the matrix itself, in fact, if H is nonsingular, then

$$\Delta(H^{-1}) = -H^{-1} \Delta(H) H^{-1}. \tag{10}$$

The following result can be easily proved by extending to block matrices the same proof given in [6].

Theorem 1 *Let K_n be an $n \times n$ block matrix such that $\Delta(K_n) = \sum_{i=1}^{r} U^{(i)} V^{(i)}$ where $U^{(i)}$ and $V^{(i)}$ are n-dimensional block column and block row vectors, respectively. Then we have*

$$K_n = L_n(K_n E_n^{(1)}) - \sum_{i=1}^{r} L_n(U^{(i)}) L_n^T(Z_n V^{(i)T})$$

where $L_n(W)$ denotes the $n \times n$ block lower triangular block Toeplitz matrix defined by its first block column W.

The above result allows one to represent any matrix K_n as a sum of products of block lower and upper triangular block Toeplitz matrices defined by the first

block column of K_n and by the block vectors $U^{(i)}$, $V^{(i)}$ associated with the block displacement of K_n.

If the matrix K_n is nonsingular then the above representation theorem can be applied to K_n^{-1} in the light of (10) yielding the following result

$$K_n^{-1} = L_n(K_n^{-1}E_n^{(1)}) + \sum_{i=1}^{r} L_n(K_n^{-1}U^{(i)})L_n^T(Z_nK_n^{-T}V^{(i)T}). \qquad (11)$$

For a block Toeplitz matrix in block Hessenberg form the equation (11) can be much simplified. In fact, due to the special structure (9) of $\Delta(H_n)$ we may easily prove that

$$H_n^{-1} = L_n(C_n^{(1)})L_n^T(E_n^{(1)} + Z_nR_n^{(1)T}A_0^T) - L_n(C_n^{(2)})L_n^T(Z_nR_n^{(2)T}A_0^T) \qquad (12)$$

where $C_n^{(1)}$, $C_n^{(2)}$ denote the first and the last block column, respectively, of H_n^{-1}, while $R_n^{(1)}$, $R_n^{(2)}$ denote the first and the last block row, respectively, of the matrix H_n^{-1}.

In this way the inverse of H_n is explicitly determined as a sum of products of block triangular block Toeplitz matrices defined by the block A_0, by the block rows $R_n^{(1)}$, $R_n^{(2)}$, and the block columns $C_n^{(1)}$, $C_n^{(2)}$.

4 Inverting the matrix H_n

The results reported in sections 2 and 3 can be merged together in order to derive a doubling algorithm for the inversion of H_n. More precisely we determine explicit relations between the block vectors $C_n^{(1)}$, $C_n^{(2)}$, $R_n^{(1)}$, $R_n^{(2)}$ defining the matrix H_n^{-1} and the block vectors $C_{\frac{n}{2}}^{(1)}$, $C_{\frac{n}{2}}^{(2)}$, $R_{\frac{n}{2}}^{(1)}$, $R_{\frac{n}{2}}^{(2)}$ defining the matrix $H_{\frac{n}{2}}^{-1}$. Moreover, since such relations only involve operations between block Toeplitz matrices and block vectors, we may devise an efficient scheme for their implementation based on performing few FFT's. In this way we arrive at an algorithm for the computation of the inverses of H_{2^i}, $i = 0, 1, \ldots, q$, requiring $O(m^3n + m^2n\log n)$ arithmetic operations, for $n = 2^q$, and a low storage cost; in fact, only four auxiliary block vectors need to be allocated in order to carry out our algorithm. Another interesting feature is that in the PRAM model of parallel computation the algorithm can be carried out in $O(\log^2 n)$ parallel steps using $O(m^3n)$ processors.

The following result holds

Theorem 2 *The first and the last block columns $C_n^{(1)}$, $C_n^{(2)}$, and the first and*

the last block rows $R_n^{(1)}$, $R_n^{(2)}$ of H_n^{-1} satisfy the following relations

$$
\begin{aligned}
C_n^{(1)} &= \begin{bmatrix} C_{\frac{n}{2}}^{(1)} \\ -H_{\frac{n}{2}}^{-1} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(1)} \end{bmatrix} - \begin{bmatrix} C_{\frac{n}{2}}^{(2)} \\ -H_{\frac{n}{2}}^{-1} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(2)} \end{bmatrix} P R_{\frac{n}{2}}^{(1)} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(1)} \\
C_n^{(2)} &= \begin{bmatrix} O \\ C_{\frac{n}{2}}^{(2)} \end{bmatrix} + \begin{bmatrix} C_{\frac{n}{2}}^{(2)} \\ -H_{\frac{n}{2}}^{-1} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(2)} \end{bmatrix} P R_{\frac{n}{2}}^{(1)} E_{\frac{n}{2}}^{(2)} \qquad (13) \\
R_n^{(1)} &= A_0 [R_{\frac{n}{2}}^{(1)}, O] - A_0 R_{\frac{n}{2}}^{(1)} E_{\frac{n}{2}}^{(2)} P [R_{\frac{n}{2}}^{(1)} T_{\frac{n}{2}} H_{\frac{n}{2}}^{-1}, -R_{\frac{n}{2}}^{(1)}] \\
R_n^{(2)} &= A_0 [-R_{\frac{n}{2}}^{(2)} T_{\frac{n}{2}} H_{\frac{n}{2}}^{-1}, R_{\frac{n}{2}}^{(2)}] + A_0 R_{\frac{n}{2}}^{(2)} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(2)} P [R_{\frac{n}{2}}^{(1)} T_{\frac{n}{2}} H_{\frac{n}{2}}^{-1}, -R_{\frac{n}{2}}^{(1)}],
\end{aligned}
$$

where $P = (I + A_0 R_{\frac{n}{2}}^{(1)} T_{\frac{n}{2}} C_{\frac{n}{2}}^{(2)})^{-1} A_0$.

Proof. The formulae follow directly from (5–7). □

In order to describe the algorithm for the computation of H_n^{-1}, represented in terms of the block vectors $R_n^{(1)}$, $R_n^{(2)}$, $C_n^{(1)}$, $C_n^{(2)}$, based on Theorem 2, we need to recall some procedures for performing computations involving block Toeplitz matrices. For this purpose we denote with $\mathbf{y} = (y_0, \ldots, y_{n-1}) = \mathrm{DFT}(\mathbf{x})$ the Discrete Fourier Transform of order n of the vector $\mathbf{x} = (x_0, \ldots, x_{n-1})$ defined by $y_i = \sum_{j=0}^{n-1} \omega^{ij} x_j$, $i = 0, \ldots, n-1$, where $\omega = \cos\frac{2\pi}{n} + \mathbf{i} \sin\frac{2\pi}{n}$, and $\mathbf{i}$ is the imaginary unit such that $\mathbf{i}^2 = -1$. Analogously $\mathbf{x} = \mathrm{IDFT}(\mathbf{y})$ denotes the Inverse Discrete Fourier Transform of order n such that $x_i = \frac{1}{n} \sum_{j=0}^{n-1} \omega^{-ij} y_j$, $i = 0, \ldots, n-1$. We recall that the computation of a DFT of order $n = 2^q$ of a real vector can be performed by means of the base 2 FFT algorithm in about $\frac{5}{2} n \log_2 n$ arithmetic operations between real numbers [7]. Similarly, the cost of an IDFT is about $\frac{5}{2} n \log_2 n + n$ real arithmetic operations in the case where the transformed vector is real.

The above notations can be extended to block vectors. Given the block vector $X = (X_0, \ldots, X_{n-1})$, where $X_s = (x_{i,j}^{(s)})$ are $m \times m$ matrices, we define $Y = (Y_0, \ldots, Y_{n-1}) = \mathrm{DFT}(X)$ the block vector such that $Y_s = (y_{i,j}^{(s)})$ are $m \times m$ matrices and $(y_{i,j}^{(0)}, \ldots, y_{i,j}^{(n-1)}) = \mathrm{DFT}(x_{i,j}^{(0)}, \ldots, x_{i,j}^{(n-1)})$, $i, j = 1, \ldots, m$. Analogously we set $X = \mathrm{IDFT}(Y)$ if $(x_{i,j}^{(0)}, \ldots, x_{i,j}^{(n-1)}) = \mathrm{IDFT}(y_{i,j}^{(0)}, \ldots, y_{i,j}^{(n-1)})$. In this way the computation of $\mathrm{DFT}(X)$ and $\mathrm{IDFT}(Y)$ is reduced to the computation of m^2 DFT's and IDFT's, respectively, for the cost of about $\frac{5}{2} m^2 n \log_2 n$ and $\frac{5}{2} m^2 n \log_2 n + mn$ real operations, respectively.

Remark 3 *Observe that the product of a block Toeplitz matrix and a vector can be reduced to computing the product of two matrix polynomials, that is, polynomials whose coefficients are matrices. In fact, if* $P(z) = \sum_{i=0}^{n-1} P_i z^i$, $Q(z) = \sum_{i=0}^{n/2-1} Q_i z^i$, *are matrix polynomials, where* P_i, Q_i *are* $m \times m$ *matrices, then, by comparing the terms of degree* $n/2, \ldots, n-1$ *in the equation* $R(z) =$

$P(z)Q(z) \bmod (z^n - 1)$, *we arrive at the following equation involving a block Toeplitz matrix*

$$\begin{bmatrix} R_{n/2} \\ \vdots \\ R_{n-1} \end{bmatrix} = \begin{bmatrix} P_{n/2} & \cdots & P_1 \\ \vdots & \ddots & \vdots \\ P_{n-1} & \cdots & P_{n/2} \end{bmatrix} \begin{bmatrix} Q_0 \\ \vdots \\ Q_{n/2-1} \end{bmatrix}.$$

The product of two matrix polynomials can be computed by the following algorithm.

Algorithm 1. *Computation of matrix polynomial product modulo* $z^n - 1$.

Input. Positive integers q, m and the $m \times m$ matrices $P_0, \ldots, P_{n-1}$, $Q_0, \ldots, Q_{n-1}$, coefficients of the matrix polynomials $P(z) = \sum_{i=0}^{n-1} P_i z^i$, $Q(z) = \sum_{i=0}^{n-1} Q_i z^i$, where $n = 2^q$.

Output. The $m \times m$ matrices $R_0, \ldots, R_{n-1}$, coefficients of the matrix polynomial $R(z) = \sum_{i=0}^{n-1} R_i z^i$ such that $R(z) = P(z)Q(z) \bmod (z^n - 1)$.

Computation.

1. (Evaluation) Compute the entries of the $2n$ matrices $U_s = P(\omega^s)$, $V_s = Q(\omega^s)$, $s = 0, \ldots, n-1$, $\omega = \cos\frac{2\pi}{n} + \mathbf{i}\,\sin\frac{2\pi}{n}$, in the following way
$$(U_0, \ldots, U_{n-1}) = \mathrm{DFT}(P_0, \ldots, P_{n-1}),$$
$$(V_0, \ldots, V_{n-1}) = \mathrm{DFT}(Q_0, \ldots, Q_{n-1}).$$
2. Compute the n matrix products $W_s = U_s V_s$, $s = 0, \ldots, n-1$, such that $W_s = P(\omega^s)Q(\omega^s) = R(\omega^s)$.
3. (Interpolation) Compute the entries of the matrices $R_0, \ldots, R_{n-1}$ by means of the equation $(R_0, \ldots, R_{n-1}) = \mathrm{IDFT}(W_0, \ldots, W_{n-1})$.

The cost of Algorithm 1 is $O(m^3 n + m^2 n \log n)$ ops. More precisely, for real input matrices the cost is less than $\frac{5}{2}m^2 n \log_2 n + n(3m^3 + 2m^2)$ (compare [4], [3]).

Remark 4 *By using equation (12) and Algorithm 1 it is possible to compute the product of the matrix* H_n^{-1} *and a block column vector by means of* $10m^2$ *FFT's of order* $2n$ *and about* $4n$ *complex* $m \times m$ *matrix products. This algorithm can be easily adjusted to the computation of the product of a block row vector and* H_n^{-1}. *Moreover, for the computation of the products of* H_n^{-1} *with* k *block row and/or column vectors the number of FFT's can be reduced to* $6k + 4$ *while the number of complex matrix products is* $4kn$. *In fact, the DFT's of the four block vectors defining* H_n^{-1} *can be computed once for all.*

Now we are ready to describe the main algorithm of this paper.

Algorithm 2. *Computation of the inverse of a block Hessenberg block Toeplitz matrix* H_n.

Input. A positive integer q and the $m \times m$ blocks A_i, $i = 0, 1, \ldots, n$, $n = 2^q$, such that $\det H_{2^j} \neq 0$, $j = 0, 1, \ldots, q$.

Output. The block vectors $C_{2^j}^{(1)}$, $C_{2^j}^{(2)}$, $R_{2^j}^{(1)}$, $R_{2^j}^{(2)}$, defining the first and the last block columns and block rows, respectively, of the matrix $H_{2^j}^{-1}$, for $j = 0, 1, \ldots, q$.

Computation.

1. Compute $H_1^{-1} = A_1^{-1}$ and set $R_1^{(1)} = R_1^{(2)} = C_1^{(1)} = C_1^{(2)} = A_1^{-1}$.
2. For $j = 0, 1, \ldots, q-1$ compute $C_{2^{j+1}}^{(1)}$, $C_{2^{j+1}}^{(2)}$, $R_{2^{j+1}}^{(1)}$, $R_{2^{j+1}}^{(2)}$ by means of Theorem 2, by calculating:
   (a) $X^{(1)} = T_{2^j} C_{2^j}^{(1)}$, $X^{(2)} = T_{2^j} C_{2^j}^{(2)}$, $Y^{(1)} = R_{2^j}^{(1)} T_{2^j}$, $Y^{(2)} = R_{2^j}^{(2)} T_{2^j}$;
   (b) $H_{2^j}^{-1} X^{(1)}$, $H_{2^j}^{-1} X^{(2)}$, $Y^{(1)} H_{2^j}^{-1}$, $Y^{(2)} H_{2^j}^{-1}$;
   (c) $P = (I + A_0 R_{2^j}^{(1)} X^{(2)})^{-1} A_0$;
   (d) $C_{2^{j+1}}^{(1)}$, $C_{2^{j+1}}^{(2)}$, $R_{2^{j+1}}^{(1)}$, $R_{2^{j+1}}^{(2)}$ by means of (13).

For the computation of the products $T_{2^j} C_{2^j}^{(1)}$, $T_{2^j} C_{2^j}^{(2)}$, $R_{2^j}^{(1)} T_{2^j}$, $R_{2^j}^{(2)} T_{2^j}$, at stage 2a of Algorithm 2, involving the block Toeplitz matrix T_{2^j}, we use Algorithm 1 where the FFT of the matrix T_{2^j}, that appears in each product, is computed once. In this way we have to compute $9m^2$ FFT's of order 2^{j+1} and about 2^{j+2} products of $m \times m$ complex matrices.

For the computation of the four products involving the matrix $H_{2^j}^{-1}$ at stage 2b of Algorithm 2, we use again Algorithm 1 and (12) in the light of Remark 4, where the four FFT's associated with the matrix $H_{2^j}^{-1}$ are computed only once. In this way the computation is reduced to performing $12m^2$ FFT's of order 2^{j+1} and about $10 \cdot 2^j$ complex $m \times m$ matrix products.

Since the cost of an FFT of a real vector of order n is about $\frac{5}{2} n \log_2 n$ real arithmetic operations and the product of two complex matrices can be performed by means of three products of real matrices, we find that the overall cost of Algorithm 2 is roughly given by $42m^3 n + 100m^2 n \log_2 n$ arithmetic operations.

Once the block vectors $R_n^{(1)}$, $R_n^{(2)}$, $C_n^{(1)}$, $C_n^{(2)}$ have been computed, the solution $\mathbf{x}$ of the system $H_n \mathbf{x} = \mathbf{b}$, can be obtained by means of (12) and remark 3, by performing $O(m^2)$ FFT's of order n, thus, leaving unchanged the asymptotic cost $O(m^3 n + m^2 n \log n)$.

5 Application to queueing problems

Algorithm 2 proposed in Section 4 can be applied to solve nonlinear matrix equations arising in queueing problems. Specifically, it is fundamental for solving Markov chains of M/G/1 type (for details on M/G/1 type Markov chains we

refer the reader to [14]) the computation of the minimal nonnegative solution G of the nonlinear matrix equation

$$X = \sum_{i=0}^{\infty} X^i B_i, \tag{14}$$

where B_i, $i \geq 0$, are nonnegative $m \times m$ matrices such that $\sum_{i=0}^{\infty} B_i$ is column stochastic, and X is an $m \times m$ matrix. In [12] it is observed that the seeked solution G satisfies the following equation involving an infinite block Hessenberg block Toeplitz matrix

$$[G, G^2, G^3, \ldots] \begin{bmatrix} I - B_1 & -B_0 & & & \bigcirc \\ -B_2 & I - B_1 & -B_0 & & \\ -B_3 & -B_2 & I - B_1 & -B_0 & \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} = [B_0, O, O, \ldots]. \tag{15}$$

Moreover, if we consider the solution of the block $n \times n$ truncated system

$$[X_1^{(n)}, X_2^{(n)}, \ldots, X_n^{(n)}] H_n = [B_0, O, \ldots, O]$$

$$H_n = \begin{bmatrix} I - B_1 & -B_0 & & & \bigcirc \\ -B_2 & I - B_1 & -B_0 & & \\ \vdots & \ddots & \ddots & \ddots & \\ -B_{n-1} & \ddots & \ddots & I - B_1 & -B_0 \\ -B_n & -B_{n-1} & \ldots & -B_2 & I - B_1 \end{bmatrix} \tag{16}$$

then $\{X_1^{(n)}\}_n$ is a sequence of nonnegative matrices that monotonically converges to G [12]. In order to solve the infinite system (15) it is suggested to solve the above systems, for n equal to 2, 4, ..., 2^q for a sufficiently large q.

Since the solution of (16) is given by the first block row of H_n^{-1} multiplied on the left by B_0, we may apply Algorithm 2 of section 4 for the solution of the matrix equation (14). The cost of this computation amounts to $O(m^3 n + m^2 n \log n)$ real arithmetic operations.

In certain important applications [9] H_n is a scalar Toeplitz matrix, consequently, B_0 is a lower triangular Toeplitz matrix that we may assume nonsingular (its non-singularity depends on the choice of the block size m). In this case it has been proved by Gail et al. [9], that the matrix G can be represented as $G = F^m$, where F is a suitable Frobenius matrix. In particular, G is such that $\mathrm{rank}(\Delta(G)) = 1$, that is, its displacement rank is 1, according to the operator $\Delta(A) = ZA - AZ$, where Z is the scalar down-shift matrix.

Here we show that, for a scalar Toeplitz matrix H_n in block Hessenberg form, Algorithm 2 can be strongly simplified. In fact, since H_n is Toeplitz we may write H_n^{-1} as $H_n^{-1} = L(\mathbf{u}_n^{(1)}) L^T(\mathbf{v}_n^{(1)}) - L(\mathbf{u}_n^{(2)}) L^T(\mathbf{v}_n^{(2)})$ for suitable mn

dimensional vectors $\mathbf{u}_n^{(i)}$, $\mathbf{v}_n^{(i)}$, $i = 1, 2$ (say, by using the Gohberg Semencul formula [10], or the scalar version of the formula (12)) and $L(\mathbf{u})$ denotes the lower triangular Toeplitz matrix defined by its first column $\mathbf{u}$..

In this way it is sufficient to relate the vectors $\mathbf{u}_n^{(i)}$, $\mathbf{v}_n^{(i)}$, with $\mathbf{u}_{\frac{n}{2}}^{(i)}$, $\mathbf{v}_{\frac{n}{2}}^{(i)}$, for $i = 1, 2$. This relation is implicitly provided by (5–7). We observe that the computation of $\mathbf{u}_n^{(i)}$, $\mathbf{v}_n^{(i)}$ is reduced to performing a finite number of products of scalar Toeplitz matrices and vectors of dimension nm except for the computation of the inverse of $I + N_n^T S_n^{-1} M_n$. For this purpose we have the following result:

Theorem 5 *The displacement rank of $I + N_n^T S_n^{-1} M_n$ is at most 6, i.e., there exist vectors $\mathbf{u}^{(i)}$, $\mathbf{v}^{(i)}$ such that $\Delta(I + N_n^T S_n^{-1} M_n) = \sum_{i=1}^{6} \mathbf{u}^{(i)} \mathbf{v}^{(i)T}$.*

Proof. It is sufficient to compute the rank of $\Delta(I + N_n^T S_n^{-1} M_n)$, i.e., $\mathrm{rank}(\Delta(N_n^T S_n^{-1} M_n))$. Due to the structure of N_n and M_n, we have $N_n^T S_n^{-1} M_n = A_0 (S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}$, where $(S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}$ is the $(\frac{n}{2}+1, \frac{n}{2})$ block entry of S_n^{-1}. Now, from the definition of displacement rank it follows that $\mathrm{rank}\Delta((S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}) \leq \mathrm{rank}\Delta(S_n^{-1})+2$, moreover, since S_n is a 2×2 block triangular block Toeplitz matrix with Toeplitz blocks, we have $\mathrm{rank}\Delta(S_n) \leq 4$, and therefore $\mathrm{rank}\Delta(S_n^{-1}) \leq 4$. Whence we deduce that $\mathrm{rank}\Delta((S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}) \leq 6$. Since A_0 is a lower triangular Toeplitz matrix then it has null displacement rank and from the relation $\Delta(AB) = A\Delta(B) + \Delta(A)B$, that can be easily verified, it follows $\mathrm{rank}\Delta(A_0 (S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}) \leq \mathrm{rank}\Delta((S_n^{-1})_{\frac{n}{2}+1,\frac{n}{2}}) \leq 6$. □

From the above theorem and from the equations (6–7) applied in the scalar case, it follows that the inverse of $I+N_n^T S_n^{-1} M_n$ can be represented as the sum of at most 7 products of lower and upper triangular Toeplitz matrices respectively. Thus its inverse can be computed by means of fast algorithms which require $O(m^2)$ arithmetic operations [10]. In this way, due to the above properties, each doubling step of size 2^j of Algorithm 2 applied to a scalar Toeplitz matrix is performed with a finite number of FFT's of order $m2^j$ and with the inversion of a matrix having displacement rank at most 4. Whence this computation can be carried out at the arithmetic cost $O(m2^j \log(m2^j)) + O(m^2)$, and the overall cost is $O(m^2 \log n) + O(mn \log(mn))$.

Algorithm 2 has been implemented in Fortran 90 and compared with the Stewart implementation of the algorithm given in [15], on a problem arising from the modeling of a metropolitan network [1], where the goal of the computation is solving the matrix equation (14). The size of the blocks in this problem is $m = 16$ and the maximum value of n that we have tested for our algorithm is $n = 4096$.

The experiments have been performed on an Alpha server. The following table reports, for the values of n ranging from 256 to 4096, the number of cpu seconds, required by our algorithm (algorithm DR: Displacement Rank) and the residual error; it reports also the time required by the Latouche–Stewart algorithm (denoted by LS) to reach the same residual error of our algorithm,

and the ratio between the times. The symbol "*" denotes that we were not able to reach the required accuracy for lack of memory.

DR			LS		
n	time	residual	time	residual	ratio
256	8.7	$2.5 \cdot 10^{-3}$	38.3	$2.5 \cdot 10^{-3}$	4.4
512	18.1	$4.0 \cdot 10^{-4}$	103.5	$4.0 \cdot 10^{-4}$	5.7
1024	39.2	$1.7 \cdot 10^{-5}$	264.9	$1.7 \cdot 10^{-5}$	6.8
2048	89.3	$3.4 \cdot 10^{-8}$	*	*	*
4096	193.4	$1.3 \cdot 10^{-13}$	*	*	*

Table 1: Comparison of the algorithms

Our algorithm, besides being faster than Stewart's algorithm, requires less storage and therefore allows us to deal with larger values of n, and provides more accurate results.

References

[1] G. Anastasi, L. Lenzini, B. Meini, "Performance evaluation of a worst case model of the Metaring MAC Protocol with global fairness", *Performance Evaluation,* to appear.

[2] D. Bini, "On a class of matrices related to Toepliz matrices", Technical Report 83-5, SUNY University, Albany, New York 1983.

[3] D. Bini, B. Meini, "On cyclic reduction applied to a class of Toeplitz-like matrices arising in queueing problems", *Computations with Markov Chains*, W. J. Stewart Ed., pp. 21–38, Kluwer Academic Publisher s, 1996.

[4] D. Bini, B. Meini, "On the solution of a nonlinear matrix equation arising in queueing problems", *SIAM J. Matrix Anal. Appl.,* 17 (1996), pp. 906–926.

[5] D. Bini, V. Pan, "Improved parallel computations with Toeplitz-like and Hankel matrices", *Linear Algebra Appl.,* 188,189: 3–29 (1993).

[6] D. Bini, V. Pan, *Matrix and Polynomial Computations, Vol. 1: Fundamental Algorithms*, Birkhäuser, Boston 1994.

[7] D.F. Elliott, K.R. Rao, *Fast Transform Algorithms, Analyses, Applications,* Academic Press, New York, 1982.

[8] G.H. Golub, C.F. van Loan, *Matrix Computations,* The Johns Hopkins University Press, Baltimore, 1989.

[9] H.R. Gail, S.L. Hantler, B.A. Taylor, "Non-skip-free M/G/1 and G/M/1 Type Markov Chains", *Adv. Appl. Prob.,* to appear.

[10] T. Kailath, A.H. Sayed, "Displacement structure: theory and applications", *SIAM Rev.,* 37 (1995), pp. 297–386.

[11] T. Kailath, A. Viera, M. Morf, "Inverses of Toeplitz operators, innovations and orthogonal polynomials", *SIAM Rev.,* 20, (1978), pp. 106–119.

[12] G. Latouche, G.W. Stewart, "Numerical methods for M/G/1 type queues", *Computations with Markov Chains*, W. J. Stewart Ed., pp. 571–581, Kluwer Academic Publishers, 1996.

[13] M.F. Neuts, *Matrix-Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, 1981.

[14] M.F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications,* Dekker Inc., New York, 1989.

[15] G. W. Stewart, "On the solution of block Hessenberg systems", *Numer. Lin. Algebra with Appl.*, 2 (1995), pp. 287–296.