

**A direct method to solve block banded
block Toeplitz systems with non-banded
Toeplitz blocks**

Andrey Chesnokov Marc Van Barel

Report TW 531, September 2008



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A direct method to solve block banded block Toeplitz systems with non-banded Toeplitz blocks

Andrey Chesnokov Marc Van Barel

Report TW531, September 2008

Department of Computer Science, K.U.Leuven

Abstract

A fast solution algorithm is proposed for solving block banded block Toeplitz systems with non-banded Toeplitz blocks. The algorithm constructs the circulant transformation of a given Toeplitz system and then by means of the Sherman-Morrison-Woodbury formula transforms its inverse to an inverse of the original matrix. The block circulant matrix with Toeplitz blocks is converted to a block diagonal matrix with Toeplitz blocks, and the resulting Toeplitz systems are solved by means of a fast Toeplitz solver.

The computational complexity in the case one uses fast Toeplitz solvers is equal to $\xi(m, n, k) = O(mn^3) + O(k^3n^3)$ flops, there are m block rows and m block columns in the matrix, n is the order of blocks, $2k + 1$ is the bandwidth. The validity of the approach is illustrated by numerical experiments.

Keywords : Toeplitz matrices, block Toeplitz Toeplitz block matrices, Sherman-Morrison-Woodbury formula, fast Toeplitz solver.

MSC : Primary : 65F05.

Andrey Chesnokov ^{*,1}, Marc Van Barel ¹

*Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan
200A, 3001 Heverlee, Belgium*

A direct method to solve block banded block Toeplitz systems with non-banded Toeplitz blocks

Abstract

A fast solution algorithm is proposed for solving block banded block Toeplitz systems with non-banded Toeplitz blocks. The algorithm constructs the circulant transformation of a given Toeplitz system and then by means of the Sherman-Morrison-Woodbury formula transforms its inverse to an inverse of the original matrix. The block circulant matrix with Toeplitz blocks is converted to a block diagonal matrix with Toeplitz blocks, and the resulting Toeplitz systems are solved by means of a fast Toeplitz solver.

The computational complexity in the case one uses fast Toeplitz solvers is equal to $\xi(m, n, k) = O(mn^3) + O(k^3n^3)$ flops, there are m block rows and m block columns in the matrix, n is the order of blocks, $2k + 1$ is the bandwidth. The validity of the approach is illustrated by numerical experiments.

Key words: Toeplitz matrices, block Toeplitz Toeplitz block matrices, Sherman-Morrison-Woodbury formula, fast Toeplitz solver
AMS Classification: 65F05, 15A57.

^{*} Corresponding author.

Email addresses: andrey.chesnokov@cs.kuleuven.be (Andrey Chesnokov),
marc.vanbare1@cs.kuleuven.be (Marc Van Barel).

¹ The research was partially supported by the Research Council K.U.Leuven, project OT/05/40 (Large rank structured matrix computations), CoE EF/05/006 Optimization in Engineering (OPTEC), by the Fund for Scientific Research–Flanders (Belgium), project G.0423.05 (RAM: Rational modelling: optimal conditioning and stable algorithms), and by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization). The scientific responsibility rests with the authors.

1 Introduction

The mathematical modelling of problems of the real world often leads to solving linear systems of equations with some intrinsic structure.

A matrix $T \in \mathbb{R}^{n \times n}$ is called Toeplitz, if $T = (t_{ij}) = (a_{j-i})$:

$$T = \begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{n-1} \\ a_{-1} & a_0 & a_1 & \ddots & \vdots \\ a_{-2} & a_{-1} & a_0 & \ddots & a_2 \\ \vdots & \ddots & \ddots & \ddots & a_1 \\ a_{-(n-1)} & \cdots & a_{-2} & a_{-1} & a_0 \end{pmatrix}. \quad (1)$$

This kind of matrices, with their block versions, arises whenever properties of shift invariance are satisfied by some function in the model. They are encountered, in particular, in fields like image processing and in the numerical solution of differential equations where the shift invariance takes different forms. Very often these matrices are block banded block Toeplitz (BBBT) matrices accompanied with a Toeplitz structure of the blocks. A very extensive study of the problems where block banded block Toeplitz matrices arise, is given in [1].

Due to the large block size of the matrices (for image processing the product of the two sizes is the number of pixels in the image) it is mandatory to exploit both the outer banded Toeplitz structure and the inner Toeplitz structure to devise efficient algorithms for the solution of these systems. Several iterative techniques for the solution of BBBT systems have been introduced in the literature, in particular we recall PCG methods [2–4], the multigrid techniques [5] and the algorithms based on the cyclic reduction [1,6]. There are also some direct methods, such as generalizations of the methods based on the Schur algorithm [7], the generalization of displacement ranks [8] and the deconvolution approach [9].

However the best of these algorithms give $O(kn^4)$ complexity for nonsymmetric $n^2 \times n^2$ BBBT systems with nonbanded Toeplitz blocks and are not easy to program. Our new algorithm reduces the complexity to $O(n^4) + O(k^3n^3)$ and is quite easy to program. Such a direct method may be of particular interest when constructing preconditioners for existing iterative methods like GMRes and BiCGStab.

The proposed algorithm constructs the low-rank circulant transformation of a given BBBT system and then by means of the Sherman-Morrison-Woodbury formula transforms the inverse of such a transformation to the inverse of the

original matrix. The block circulant matrix with Toeplitz blocks is converted to a block diagonal matrix with Toeplitz blocks, and the resulting Toeplitz systems are solved by means of a fast Toeplitz solver.

Our paper is organized in the following way. In Section 2 the main algorithm is given. Theoretical and practical complexity details are studied in Section 3.

2 Main algorithm

2.1 Reduction to the block circulant case

Consider a block banded block Toeplitz system with dense Toeplitz blocks $Bx = b$. We suppose that inner blocks are $n \times n$ -matrices and there are m block columns and m block rows. The bandwidth is equal to $2k + 1$.

A block banded block Toeplitz matrix could be easily transformed to a block circulant matrix with Toeplitz blocks (BCTB matrix) by adding corresponding blocks in its lower-left and upper-right corners. The matrix B in (2) is block Toeplitz and C in (3) is its corresponding block circulant.

$$B = \begin{pmatrix} A_0 & A_1 & \cdots & A_k & 0 & 0 & \cdots & 0 \\ A_{-1} & A_0 & A_1 & \cdots & \cdots & \cdots & \cdots & \vdots \\ \cdots & A_{-1} & A_0 & A_1 & \cdots & \cdots & \cdots & 0 \\ A_{-k} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & A_k \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & A_{-1} & A_0 & A_1 \\ 0 & \cdots & 0 & 0 & A_{-k} & \cdots & A_{-1} & A_0 \end{pmatrix} \quad (2)$$

$$C = \begin{pmatrix} A_0 & A_1 & \cdots & A_k & 0 & A_{-k} & \cdots & A_{-1} \\ A_{-1} & A_0 & A_1 & \cdots & \cdots & \cdots & \cdots & \vdots \\ \cdots & A_{-1} & A_0 & A_1 & \cdots & \cdots & \cdots & A_{-k} \\ A_{-k} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & A_k \\ A_k & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & A_{-1} & A_0 & A_1 \\ A_1 & \cdots & A_k & 0 & A_{-k} & \cdots & A_{-1} & A_0 \end{pmatrix} \quad (3)$$

We can write that $B = C - UV^T$, where

$$U = \begin{pmatrix} I_{kn} & 0 \\ 0 & 0 \\ 0 & P \end{pmatrix}, \quad V^T = \begin{pmatrix} 0 & 0 & Q \\ I_{kn} & 0 & 0 \end{pmatrix}. \quad (4)$$

Matrices P and Q are $kn \times kn$ block triangular matrices that reside in the lower left and upper right corners of C , respectively,

$$P = \begin{pmatrix} A_k & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots \\ A_2 & \cdots & \cdots & 0 \\ A_1 & A_2 & \cdots & A_k \end{pmatrix}, \quad Q = \begin{pmatrix} A_{-k} & \cdots & A_{-2} & A_{-1} \\ 0 & \cdots & \cdots & A_{-2} \\ \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & 0 & A_{-k} \end{pmatrix}. \quad (5)$$

The idea of interchanging the outer and inner structures is thoroughly studied in a general case in the book [10].

The inverses of B and C are then connected by means of the Sherman-Morrison-Woodbury formula [11]:

$$B^{-1} = (C - UV^T)^{-1} = C^{-1} + C^{-1}U(I - V^TC^{-1}U)^{-1}V^TC^{-1}. \quad (6)$$

Remember that we have to compute $x = B^{-1}b$. By means of the above formula this computation is partitioned into several steps.

At first, solve the BCTB system $Cw = b$ as described in the next section. Let us denote $w = C^{-1}b$. Then by means of a simple matrix-vector multiplication we compute the product $v = V^T w = V^T C^{-1}b$.

In the third step, compute the inner matrix $S = I - V^T C^{-1} U$ and solve the linear system $Sy = v$. To compute S directly we need to solve $2kn$ linear systems with C as the coefficient matrix and the columns of U as right-hand side vectors. The computation of the S matrix could be slightly optimized as follows.

$$V^T C^{-1} U = \begin{pmatrix} 0 & 0 & Q \\ I_{kn} & 0 & 0 \end{pmatrix} \begin{pmatrix} C_{UL} & \dots & C_{UR} \\ \dots & \dots & \dots \\ C_{DL} & \dots & C_{DR} \end{pmatrix} \begin{pmatrix} I_{kn} & 0 \\ 0 & 0 \\ 0 & P \end{pmatrix} = \begin{pmatrix} QC_{DL} & QC_{DR}P \\ C_{UL} & C_{UR}P \end{pmatrix}. \quad (7)$$

Here C_{DL} , C_{DR} , C_{UL} and C_{UR} are the corresponding $kn \times kn$ corner blocks of C^{-1} .

Since C is block-circulant, C^{-1} is also block circulant and thus determined by its first block column. This means that it's enough to solve only n linear systems with the first n columns of a $mn \times mn$ identity matrix as right-hand side vectors. The rest of C^{-1} is then constructed just by a block reordering of its first block column.

In the fourth and the last step, we perform the matrix-vector multiplication $z = Uy$ and finally solve the remaining BCTB system $Cf = z$. Addition of $C^{-1}b$ to $C^{-1}z$ yields $B^{-1}b$.

2.2 Block circulant Toeplitz block case

Let us consider a block circulant matrix with Toeplitz blocks. We suppose that the inner blocks are $n \times n$ -matrices and there are m block columns and m block rows. Let P_1 denote a permutation matrix of size $mn \times mn$ such that it brings rows with numbers $1, n + 1, 2n + 1$, etc, together to the first rows, then all rows with numbers $2, n + 2, 2n + 2$, etc, together, and so on. The same should hold for the columns after the multiplication by P_1^T . An example

of the P_1 matrix for $n = m = 3$ is given in (8).

$$P_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (8)$$

In other words, if C was a BCTB matrix, then $K = P_1 C P_1^T$ would be a BTCB matrix: the inner structure becomes the outer and vice versa. Because the outer structure of K is Toeplitz, there are only $2n + 1$ different circulant blocks of size $m \times m$. These blocks are easily diagonalized by means of a Fourier transform.

Let $F = \bigoplus F_m$ be a direct sum of n Fourier matrices of order m . Then $L = F K F^* = F P_1 C P_1^T F^*$ is a block Toeplitz matrix with diagonal blocks. Let us denote by P_2 a permutation matrix which is very like P_1 , but brings first all rows with numbers $1, m + 1, 2m + 1$ together, then all rows with numbers $2, m + 2, 2m + 2$ together and so on. Then $M = P_2 F P_1 C P_1^T F^* P_2^T$ is a block diagonal matrix with Toeplitz blocks.

Remember that we have to solve a linear system $Cx = b$, where the inner blocks are $n \times n$ -matrices and there are m block rows and m block columns. By means of the transforms described above we have converted the original BCTB problem to a solution of a block diagonal Toeplitz block system:

$$Cx = b \Leftrightarrow P_2 F P_1 C x = P_2 F P_1 b \Leftrightarrow (P_2 F P_1 C P_1^T F^* P_2^T)(P_2 F P_1 x) = P_2 F P_1 b \Leftrightarrow M \hat{x} = \hat{b}, \quad (9)$$

where

$$M = P_2 F P_1 C P_1^T F^* P_2^T, \quad \hat{x} = P_2 F P_1 x, \quad \hat{b} = P_2 F P_1 b. \quad (10)$$

This block Toeplitz system splits up into m ($n \times n$)-Toeplitz systems. These systems could be solved by means of any conventional Toeplitz solver, like fast solvers [12–16].

Transition from \hat{x} to x is obvious and consists of two reorderings and n inverse

FFT's.

3 Complexity

3.1 Theoretical estimation

Let us look at the formula (6) and consequently calculate its computational cost. Denote the total computational cost by $\xi(m, n, k)$.

- (1) Computing $w = C^{-1}b$ is exactly solving a corresponding linear system. Let us denote the complexity of the solution of a BCTB linear system with m block rows and m block columns, each block is of size $n \times n$, by $\nu(m, n)$. Thus we have $\nu(m, n)$.
- (2) Computing $v = V^T w$ is in fact one $kn \times kn$ matrix-vector product due to a special structure of V^T . Thus we have $O(k^2 n^2)$.
- (3) Computing $V^T C^{-1} U$ by means of a technique (7) involves n operations of complexity $\nu(m, n)$ and then four matrix-matrix products of order $kn \times kn$. Thus we have $n \cdot \nu(m, n) + O(k^3 n^3)$. Recalling that matrices P and Q are block Toeplitz can slightly reduce this estimate.
- (4) Solution of a linear system with the matrix $I - V^T C^{-1} U$ as a coefficient matrix and vector v as a right-hand side takes $O(k^3 n^3)$ operations since the matrix is unstructured.
- (5) Multiplication by matrix U on the left takes $O(k^2 n^2)$ operations and yields an $mn \times 1$ vector z .
- (6) And finally one has to solve the linear system with C matrix and vector z . It takes another $\nu(m, n)$ operations.

In total after gobbling up all low-order terms we get $\xi(m, n, k) = n \cdot \nu(m, n) + O(k^3 n^3)$.

Let us now estimate $\nu(m, n)$. On the basis of the formulas given in Subsection 2.2, we have the following essential steps.

- (1) Multiplications with P_1, P_2, P_1^T and P_2^T are in fact just reorderings.
- (2) There are n Fourier transforms of size m , which gives $nm \log m$ operations.
- (3) We have to solve m Toeplitz systems of order n . This could be done by any conventional method, like Gaussian elimination ($O(mn^3)$ in total) or existing fast methods ([12–14]) ($O(mn^2)$ in total), or even by superfast methods ([17]).

In the case one uses fast Toeplitz solvers the total complexity will be $\xi(m, n, k) = O(mn^3) + O(k^3n^3)$ flops. For superfast solvers the total complexity will be equal to $\xi(m, n, k) = O(mn^2 \log^2 n) + O(k^3n^3)$

3.2 Numerical experiments

The experiments were performed on a machine with 2Gb memory and Xeon 2.3 GHz processor, running Kubuntu Linux, kernel 2.6.22, in Matlab 7.2.0.294. We have chosen as Toeplitz solvers 1) the fast Hankel solver developed by Van Barel and Kravanja [12] and 2) the simple backslash operator for comparison.

3.3 Well-conditioned matrices

Toeplitz matrices and right-hand side vectors were random.

The results are times in seconds. For each table cell there were five runs and the resulting time was averaged. The last column represents the time required to solve the dense original system directly with a backslash operator. OoM stands for Out Of Memory Matlab message.

k	m	n	Time $O(n^3)$ T-solver	Time $O(n^2)$ T-solver	Time \ operator
2	32	32	0.48	6.1	0.27
2	32	64	2.57	27.2	1.5
2	32	128	18.81	132.3	9.9
2	32	256	196.7	762.1	78.8
2	32	512	2169	4546	OoM
2	64	32	0.9	12.0	1.6
2	128	32	1.65	23.1	10
2	256	32	3.20	49.0	80
2	512	32	6.41	94.3	OoM
2	4096	32	52.05	766.6	OoM
2	64	64	5.01	54.1	10.1
4	64	64	5.01	54.2	10.1
8	64	64	5.56	54.2	10.1
16	64	64	8.40	62.1	10.1
2	128	128	80.1	530	OoM
4	128	128	80.1	532	OoM
8	128	128	81.0	550	OoM
16	128	128	102.1	597	OoM

The reason why the “fast” solver works slower than the backslash operator lies in the fact that we use quite a straightforward Matlab program as a fast solver, while the backslash operator represents a fine-grained precompiled code. Such a fast solver includes lots of elementwise vector operations and is proved to work much faster when coded in compiled programming language like FORTRAN or C.

So we have to look at the ratios of the neighbouring values in each column. We can see that the results do agree with the estimate. The method is linear in m and is cubic in n when the fast T-solver is applied, and is quartic in n when Gaussian elimination is applied to Toeplitz systems. For larger n the absolute time difference between these two types of solvers would be more significant and we recommend using any of the fast solvers available.

The bandwidth has its effect when $k^3 \gg m$. While $k^3 \leq m$, the bandwidth

increase does not have any impact on time.

The residual size was of the same order for the fast solver and for the backslash operator applied to full dense matrices. For large matrices where the OoM message appeared, the residual was of the order $10^{-10} - 10^{-13}$.

3.4 *Ill-conditioned matrices*

Note that we replace the BBBT matrix B with the BCTB matrix C in the solution process. This has positive and negative effects, as we will show. Usually in the case of an ill-conditioned B or C it is useful to perform one or more steps of an iterative refinement (IR) process:

$$R_i = b - Bx_i, \quad d_i = B^{-1}R_i, \quad x_{i+1} = x_i + d_i, \quad i = 0, 1, \dots, \quad (11)$$

x_0 is some initial approximation. To perform one IR step after the original system was solved is cheaper than to solve it again: the most expensive step 3 (Section 3.1) of computing the inner matrix in the Sherman-Morrison-Woodbury formula does not have to be repeated. The total cost of an IR step is then $\nu(m, n) + O(k^3n^3)$.

3.4.1 *Ill-conditioned B*

In this case the change to the circulant transformation C has a certain regularizing effect. We have constructed several ill-conditioned BBBT matrices and compared their condition numbers with the numbers of their circulant transformations. The results are given in the following table. The residual $b - Bx$ was of the same order as the residual $b - B\hat{x}$, where \hat{x} was the result of the Matlab command $B \setminus b$. For the matrices with the condition numbers greater than 10^{10} we applied one IR step.

cond(B)	cond(C)
10^8	17
10^{11}	32
10^{14}	20
10^{18}	8

However in the last case ($\text{cond}(B) = 10^{18}$) we were not able to get any reasonable result – the matrix was numerically singular.

Ill-conditioned BBBT matrices were constructed in the following way. We took a well-conditioned matrix and then scaled its elements: the elements further away from the main diagonal had bigger values.

3.4.2 Ill-conditioned C

It could happen that even for a well-conditioned B the matrix C would be ill-conditioned or singular. This is illustrated by the following example:

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \quad (12)$$

We can handle this situation by adding a small BBBT noise to the matrix B before it is transformed into the BCTB matrix. Our experiments have shown that it's enough to add just block-diagonal Toeplitz noise and not to affect the strictly lower and upper block triangles of the BBBT matrix.

We illustrate this by the following example: matrix B was fixed in such a way that its circulant extension was singular, $\text{cond}(B) = 15$. We were adding some block-diagonal noise, the 2-norm of this noise is given in the second column. The condition number of (permuted) C matrix is given in the first column. The 2-norm of the residual $b - Bx$ is given in the last column.

cond(C)	noise size	residual
10^5	10^{-4}	10^{-4}
10^9	10^{-6}	10^{-6}
10^9	10^{-8}	10^{-8}
10^{10}	10^{-9}	10^{-6}
10^{13}	10^{-10}	10^{-4}
10^{13}	10^{-12}	10^{-3}
10^{15}	10^{-14}	10^{-1}
10^{18}	10^{-16}	100

It means that small noise does not have enough regularizing effect and large noise moves the perturbed matrix too far from the original. After one IR step with the noise size $O(10^{-8})$ we got the machine precision.

We have also mentioned that the ill-conditioned C gives the very ill-conditioned

inner matrix in the Sherman-Morrison formula (step 3, Section 3.1), as well as ill-conditioned small Toeplitz systems.

We may use all this facts to build a generic approach.

- Start the algorithm as described.
- If one of the small Toeplitz systems of order n or the inner matrix in the Sherman-Morrison-Woodbury formula would appear to be ill-conditioned (this is reported by a conventional solver used), then we have ill-conditioned C . Return to the previous step and add some noise, then repeat the procedure.
- Compute the residual $(b - Bx)/\|b\|_2$. If its accuracy is not enough, then perform one or more steps of iterative refinement.

4 Acknowledgments

The authors would like to thank Kh. D. Ikramov for his comments and useful remarks.

References

- [1] D. A. Bini, B. Meini, Solving block banded block Toeplitz systems with structured blocks: algorithms and applications, *Advances In Computation: Theory And Practice. Structured matrices: recent developments in theory and computation*, Nova Science Publishers, Inc., Commack, NY, USA, 2001, pp. 21–41.
- [2] R. H. Chan, M. K. Ng, Conjugate gradient methods for Toeplitz systems, *SIAM Review* 38 (1996) 427–482.
- [3] S. Serra-Capizzano, Asymptotic results on the spectra of block Toeplitz preconditioned matrices, *SIAM Journal on Matrix Analysis and Applications* 20 (1) (1999) 31–44.
- [4] S. Serra-Capizzano, New PCG based algorithms for the solution of Hermitian Toeplitz systems, *Calcolo* 32 (3-4) (1995) 153–176.
- [5] G. Fiorentino, S. Serra-Capizzano, Multigrid methods for indefinite Toeplitz matrices, *Calcolo* 33 (3-4) (1996) 223–236.
- [6] D. A. Bini, B. Meini, Improved cyclic reduction for solving queueing problems, *Numerical Algorithms* 15 (1) (1997) 55–74.
- [7] D. A. Bini, B. Meini, Effective methods for solving banded Toeplitz systems, *SIAM Journal on Matrix Analysis and Applications* 20 (3) (1999) 700–719.

- [8] S. J. Reeves, Fast algorithm for solving block banded Toeplitz systems with banded Toeplitz blocks, in: Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, 2002, pp. 3325–3328.
- [9] A. E. Yagle, A fast algorithm for Toeplitz-block-Toeplitz linear systems, in: Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 3, 2001, pp. 1929–1932.
- [10] V. V. Voevodin, E. E. Tyrtyshnikov, Computational processes with Toeplitz matrices, Nauka, 1987, (In Russian).
- [11] G. H. Golub, C. F. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, Maryland, USA, 1996.
- [12] P. Kravanja, M. Van Barel, A fast Hankel solver based on an inversion formula for Loewner matrices, *Linear Algebra and its Applications* 282 (1) (1998) 275–295.
- [13] M. H. Gutknecht, M. Hochbruck, Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems, *Numerische Mathematik* 70 (1995) 181–227.
- [14] E. E. Tyrtyshnikov, New cost-effective and fast algorithms for special classes of Toeplitz systems, *Sov. J. Numer. Anal. Math. Modelling* 3 (1) (1988) 63–76.
- [15] P. G. Martinsson, V. Rokhlin, M. Tygert, A fast algorithm for the inversion of general Toeplitz matrices, *Computers & Mathematics with Applications* 50 (2005) 741–752.
- [16] T. F. Chan, P. C. Hansen, A stable Levinson algorithm for general Toeplitz systems, CAM Report 90-11, UCLA (May 1990).
- [17] M. Van Barel, G. Heinig, P. Kravanja, A stabilized superfast solver for nonsymmetric Toeplitz systems, *SIAM Journal on Matrix Analysis and Applications* 23 (2) (2001) 494–510.
URL <http://file:/home/marc/OZOEK/ARTICLES/ma033/info.html>