On computer realization of algorithms for solving special block linear systems

Borislav V. Minchev* Ivan G. Ivanov**

* *Faculty of Mathematics and Informatics, Shoumen University, Shoumen, 9712, Bulgaria*
e-mail: Borko.Minchev@ii.uib.no

** *Faculty of Economics and Business Administration, Sofia University, Sofia 1113, Bulgaria*
e-mail: `i_ivanov@feb.uni-sofia.bg`

Abstract

The solution of Hermitian block circulant tridiagonal linear system is investigated. This special kind of system appears in many applications. We propose new approach of El-sayed's method and develop two new algorithms for solving such kind of systems. Numerical experiments with our algorithms and some classical algorithms are executed. Numerical examples are given to illustrate the effectiveness of the proposed algorithms.

Keywords: linear system, block circulant matrix, matrix equation, Woodburry's formula.

1 Introduction

Many problems in practice lead to the solution of linear systems having special coefficient matrices. Tridiagonal block circulant linear systems arise from a finite difference approximation to Poisson's equation on a rectangular region with periodicity conditions [2, 5], in approximation of periodic functions using splines [1, 10] and etc. It is known that this systems have the form

$$\tilde{M}\ \tilde{x}\ =\ f, \tag{1}$$

where

$$\tilde{M}\ =\ \begin{pmatrix} A & B & & & & & B^* \\ B^* & A & B & & & & \\ & B^* & A & . & & 0 & \\ & & . & . & . & & \\ & & & . & . & . & \\ & 0 & & & . & . & . \\ & & & & & . & . & B \\ B & & & & & B^* & A \end{pmatrix}, \tag{2}$$

is Hermitian block circulant matrix with block size n. A and B are $m \times m$ blockss. Let introduce the following notation $\tilde{x} = \{\tilde{x}_i\}_{i=1,\ldots,n}$, $f = \{f_i\}_{i=1,\ldots,n}$, where $\tilde{x}_i$ and f_i, are block with size $m \times 1$.

The purpose of this paper is to develop new algorithms for solving (1) which are based on method proposed in [3]. According to the results obtained in [9] we propose two new algorithms and compare tham with classical LU factorization and Cholesky factorization [6], concerning time for implementation, number of operations and storage memory. Numerical experiments corroborating the theoretical results are reported.

2 A Modification of LU factorizations

In [3] El-Sayed extends the Rojo's method [11] in case when the coefficient matrix has block structure. He uses a nonlinear matrix equation for solving the linear system (1). According to the algorithms presented in [9], we propose new approach of the El-Sayed's method. New decomposition of the matrix $\tilde{M}$, which decrease the size of the inverse matrix obtain in the Woodbury's formula is proposed. For solving the problem (1) we use the following steps:

Step 1. Solve the parametric linear system (by [3])

$$N\, y \;=\; f, \tag{3}$$

where

$$N \;=\; \begin{pmatrix} X & B & & & & \\ B^* & A & . & & 0 & \\ & . & . & . & & \\ & & . & . & . & \\ & & & . & . & . \\ & 0 & & & . & . & B \\ & & & & B^* & A \end{pmatrix},$$

is tridiagonal matrix with block size n. $y = \{y_i\}_{i=1,\dots,n}$, and $f = \{f_i\}_{i=1,\dots,n}$ are column vector with block size n, y_i and f_i are block with size $m \times 1$.
Matrix N admit following LU factorizations

$$N \;=\; LU \;=\; \begin{pmatrix} I_m & & & \\ B^*X^{-1} & . & & 0 \\ & . & . & \\ 0 & & . & \\ & & B^*X^{-1} & I \end{pmatrix} \begin{pmatrix} X & B & & \\ & . & . & 0 \\ & & . & . \\ 0 & & & . & B \\ & & & & X \end{pmatrix},$$

where I_m is identity matrix with size $m \times m$.
The above decomposition exists when the parameter X, satisfy the nonlinear matrix equation

$$X + B^*X^{-1}B \;=\; A. \tag{4}$$

In this way solving the linear system (3) is equivalent to solving two simpler systems

$$\begin{aligned} L\, z \;&=\; f, \qquad z = \{z_i\}_{i=1,\dots,n} \\ U\, y \;&=\; z, \qquad y = \{y_i\}_{i=1,\dots,n}. \end{aligned}$$

Their solution are respectively

$$\begin{aligned} &z_1 = f_1 \\ &z_i = f_i - B^*X^{-1}z_{i-1}, \qquad i = 2, 3, \dots, n, \\ &y_n = X^{-1}z_n \\ &y_i = X^{-1}(z_i - By_{i+1}), \qquad i = n-1, n-2, \dots, 1. \end{aligned} \tag{5}$$

Step 2. Solve tridiagonal block Teoplitz linear system (by [3])

$$M\, x \;=\; f, \tag{6}$$

where

$$M \;=\; \begin{pmatrix} A & B & & & & \\ B^* & A & B & & 0 & \\ & B^* & A & . & & \\ & & . & . & . & \\ & & & . & . & . \\ & & & & . & . & . \\ & 0 & & & . & . & B \\ & & & & B^* & A \end{pmatrix}, \tag{7}$$

is Hermitian tridiagonal block Teoplitz matrix with block size n. $x = \{x_i\}_{i=1,\dots,n}$ and $f = \{f_i\}_{i=1,\dots,n}$ are column vectors with block size n , x_i and f_i are block with size $m \times 1$.
Matrices M and N are related by the connection

$$M = N + E_1 V_1^T,$$

where $E_1^T = \begin{pmatrix} I_m & 0 & \dots & 0 \end{pmatrix}^T$, $V_1^T = \begin{pmatrix} A - X & 0 & \dots & 0 \end{pmatrix}$.
Using the Woodbury's formula we have

$$M^{-1} = N^{-1} - N^{-1} E_1 \left[I_m + V_1^T N^{-1} E_1 \right]^{-1} V_1^T N^{-1}, \tag{8}$$

Therefore, for the solution x of (6) we have

$$\begin{aligned} x &= M^{-1} f \\ &= y - N^{-1} E_1 \left[I_m + (A - X) E_1^T N^{-1} E_1 \right]^{-1} (A - X) y_1. \end{aligned} \tag{9}$$

The coordinates of $N^{-1}E_1$ can be computed by the next algorithm, proposed in [9].
Algorithm R Recurrent computation of blocks $(N^{-1}E_1)_i$

- Find the cells $Y_i = (-1)^{n-i} X^{-1} P^{n-i}$ $\quad for\ i = 1, \dots, n.$
- Computed the blocks $(N^{-1}E_1)_i$ by the formulas

$$\begin{aligned} (N^{-1}E_1)_n &= Y_n \\ (N^{-1}E_1)_i &= Y_i - Q(N^{-1}E_1)_{i+1} \qquad for\ \ i = n-1, \dots, 1, \end{aligned}$$

where $P = B^* X^{-1}$ and $Q = X^{-1} B$.

In the next step according to the algorithm R we propose two new approaches for solving (1).

Step 3. Solve the system (1)

3.1 The matrix $\tilde{M}$ satisfy

$$\tilde{M} = M + \tilde{U} \tilde{V}^T,$$

where

$$\tilde{U} = \begin{pmatrix} I_m & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & I_m \end{pmatrix}, \quad \tilde{V}^T = \begin{pmatrix} 0 & 0 & \dots & B^* \\ B & 0 & \dots & 0 \end{pmatrix}$$

are the matrices with block size $n \times 2$ and $2 \times n$. respectively
Using the Woodbury's formula we have

$$\tilde{M}^{-1} = M^{-1} - M^{-1} \tilde{U} \left[I_{2m} + \tilde{V}^T M^{-1} \tilde{U} \right]^{-1} \tilde{V}^T M^{-1},$$

where I_{2m} is identity matrix with size $2m \times 2m$.
Solution $\tilde{x}$ of (1) holds

$$\tilde{x} = \tilde{M}^{-1} f = x - M^{-1} \tilde{U} \left[I_{2m} + \tilde{V}^T M^{-1} \tilde{U} \right]^{-1} \tilde{V}^T x. \tag{10}$$

Denote the columns block of $\tilde{U}$ with

$$E_1^T = \begin{pmatrix} I_m & 0 & \dots & 0 \end{pmatrix}^T, \quad E_n^T = \begin{pmatrix} 0 & 0 & \dots & I_m \end{pmatrix}^T.$$

Compute $M^{-1}\tilde{U}$ by successive calculations of $M^{-1}E_1$ and $M^{-1}E_n$.

$$\begin{aligned} M^{-1}E_1 &= N^{-1}E_1 - N^{-1}E_1 \left[I_m + V_1^T N^{-1} E_1 \right]^{-1} V_1^T N^{-1} E_1 \\ M^{-1}E_n &= N^{-1}E_n - N^{-1}E_1 \left[I_m + V_1^T N^{-1} E_1 \right]^{-1} V_1^T N^{-1} E_n. \end{aligned} \tag{11}$$

Since we already compute (in Step 2.) the elements $N^{-1}E_1$, $\left[I_m + V_1^T N^{-1} E_1\right]^{-1}$. We recommend using the formulas (11) instead of solving of $2m$ linear system of kind (6) with right hand side the corresponding different column vectors of $E_1$and E_2. It is easy to observe that separate block of $N^{-1}E_n$ satisfy

$$(N^{-1}E_n)_i \;=\; Y^*_{n+1-i} \qquad for \;\; i = 1, \ldots, n, \tag{12}$$

where Y_i for $i = 1, \ldots, n$ are blocks from algorithm R.

3.2 Like in [4], in order to decrease the size of the inverse matrix obtain in the Woodbury's formula we propose the following decomposition of matrix $\tilde{M}$

$$\tilde{M} \;=\; \begin{pmatrix} M & \tilde{W} \\ \tilde{W}^* & A \end{pmatrix},$$

where M is block tridiagonal Teoplitz Matrix with block size $n-1 \times n-1$ and $\tilde{W}^* = \begin{pmatrix} B & \ldots & B^* \end{pmatrix}$.
Let denote

$$\hat{x} \;=\; \begin{pmatrix} \tilde{x}_1, & \ldots, & \tilde{x}_{n-1} \end{pmatrix}, \quad \tilde{x} \;=\; \begin{pmatrix} \hat{x} \\ \tilde{x}_n \end{pmatrix},$$

$$\hat{f} \;=\; \begin{pmatrix} \tilde{f}_1, & \ldots, & \tilde{f}_{n-1} \end{pmatrix}, \quad f \;=\; \begin{pmatrix} \hat{f} \\ f_n \end{pmatrix}.$$

Then system (1) can be written in the form

$$\begin{pmatrix} M & \tilde{W} \\ \tilde{W}^* & A \end{pmatrix} \begin{pmatrix} \hat{x} \\ \tilde{x}_n \end{pmatrix} \;=\; \begin{pmatrix} \hat{f} \\ f_n \end{pmatrix},$$

which is equivalent to

$$\left| \begin{array}{l} G\hat{x} \;=\; r \\ \tilde{x}_n \;=\; A^{-1}\left(f_n - \tilde{W}^*\hat{x}\right) \end{array} \right. ,$$

where $G = M - \tilde{W}A^{-1}\tilde{W}^*, \quad r = \hat{f} - \tilde{W}A^{-1}f_n$.
By Woodbury's formula we have

$$G^{-1} \;=\; M^{-1} - M^{-1}\tilde{W}\left[A - \tilde{W}^*M^{-1}\tilde{W}\right]^{-1}\tilde{W}^*M^{-1}.$$

Then

$$\hat{x} \;=\; G^{-1}r = u - M^{-1}\tilde{W}\left[A - \tilde{W}^*M^{-1}\tilde{W}\right]^{-1}\tilde{W}^*u. \tag{13}$$

where $u \;=\; M^{-1}r$.
For $M^{-1}\tilde{W}$ according (8) we have

$$M^{-1}\tilde{W} \;=\; N^{-1}\tilde{W} - N^{-1}E_1\left[I_m + V_1^T N^{-1} E_1\right]^{-1} V_1^T N^{-1}\tilde{W}. \tag{14}$$

Separate blocks of $N^{-1}\tilde{W}$ satisfy

$$(N^{-1}\tilde{W})_i \;=\; (N^{-1}E_1)_i B^* + Y^*_{n-i}B \qquad for \;\; i = 1, \ldots, n-1, \tag{15}$$

where Y_i for $i = 1, \ldots, n$ are blocks from algorithm R.

3 Algorithms

For realization of Step 1. and Step 2. we use the algorithm M_R from [9] which is:

Algorithm M_R

1. *Solve the matrix equation (4)*
2. *Find the vector* $y \;=\; N^{-1}f$ *by formulas (5).*

3. Compute the matrix $N^{-1}E_1$ by algorithm R.

4. Receive the solution x of (6) by formula (9) with successive calculation :

$$C = (A - X)(N^{-1}E_1)_1 \ ; \ (I + C)^{-1} \ ; \ (A - X)y_1;$$

$$z = (I + C)^{-1}(A - X)y_1 \ ; \ x \ = \ y - N^{-1}E_1 z.$$

end.

If the matrices A and B are real, this algorithm requires respectably $O(12k * m^3 + 8nm^2 + 4nm^3 + 6m^3) \ = \ O([4n + 12k]m^3)$ flops and memory space of $(n + 11)m^2 + m$ real numbers, where k is number of iterations (4).

For solving system (1) according to Step 3. we propose the following two algorithms:

Algoritm M_R(1m)

1. Find vector $r \ = \ \hat{f} - \tilde{W}A^{-1}f_n$

2. Solve the linear system $M^{'}u \ = \ r$ by algorithm M _ R

3. Compute $N^{-1}\tilde{W}$ by (15).

4. Find $M^{-1}\tilde{W}$ by (14)

5. Find the vector $\hat{x}$ by (13)

6. Compute $\tilde{x}_n \ = \ A^{-1}[f_n - \tilde{W}^*\hat{x}]$

end.

If the matrices A and B are real, this algorithm requires respectably $O(2m^3+[4(n-1)+12k]m^3+4nm^3 + 2nm^3 + 6m^3 + 6m^2) \ = \ O([10n + 12k]m^3)$ flops and memory space of $(3n + 16)m^2 + (n + 2)m$ real numbers, where k is number of iterations (4).

Algorithm M_R(2m)

1. Solve the linear system $Mx \ = \ f$ by algorithm M _ R

2. Compute $M^{-1}E_1$ and $M^{-1}E_n$ by (11)

3. Find the solution $\tilde{x}$ of (1) by (10)

end.

If the matrices A and B are real, this algorithm requires respectably $O([4n + 12k]m^3 + 5nm^3 + 16m^3) \ = \ O([9n + 12k]m^3)$ flops and memory space of $(3n + 19)m^2 + (n + 3)m$ real numbers, where k is number of iterations (4).

For comparison we describe the next algorithm which realize the idea proposed in [3].

Algorithm M_RF(2m)

1. Solve the linear system $Mx \ = \ f$ by algorithm M _ R

2. Compute $M^{-1}E_1$ and $M^{-1}E_n$ by successive calculation of $2m$ linear system of the form (7) with right hend side different columns of E_1 and E_n.

3. Find the solution $\tilde{x}$ of (1) by (10)

end.

If the matrices A and B are real, this algorithm requires respectably $O([4n+12k]m^3+20nm^3+16m^3) \ = \ O([24n + 12k]m^3)$flops and memory space of $(3n + 15)m^2 + (2n + 3)m$ real numbers, where k is number of iterations (4).

4 Numerical experiments

We cared out the numerical experiments for solving linear system (1) with exact solution $\tilde{x} = (1, 1, \ldots, 1)^T$ and real symmetric matrix M, with different block size n. The cells A and B are chosen in such a way that guarantied existence the solution of the matrix equation (4). The codes of the programs was

written in MATLAB and computations was done on a PENTIUM computer. The results from the experiments are given in separate tables for each example.

The following notations are used: LU is a program, which realize classical LU factorization; CHOL is a program, which realize classical Cholesky factorizatio; $Iter$ is the number of iteration for solving the matrix equation (4) and $Err. = \|\tilde{x} - \tilde{\tilde{x}}\|_\infty$, where $\tilde{\tilde{x}}$ is the computed solution.

Table 1 reports flops and memory space for each programs.

Table 1. Flops and memory space

Algorithm	flops	memory space
LU	$\frac{41n}{3}m^3 + O(nm^2)$	$(5n-3)m^2 + 2nm$
CHOL	$\frac{31n}{3}m^3 + O(nm^2)$	$\frac{5n}{2}m^2 + \frac{3n}{2}m$
M_R(1m)	$(10n + 12Iter)m^3 + O(nm^2)$	$(3n+16)m^2 + (n+2)m$
M_R(2m)	$(9n + 12Iter)m^3 + O(nm^2)$	$(3n+19)m^2 + (n+3)m$
M_RF(2m)	$(24n + 12Iter)m^3 + O(nm^2)$	$(3n+15)m^2 + (2n+3)m$

Example 1.

Let $A = circ(20, -8, 1, \ldots, 1, -8)$, $B = I$.
Here $\|\tilde{B}\|_2 = \|A^{-\frac{1}{2}}BA^{-\frac{1}{2}}\|_2 = 0.1667$. To reach the requested accuracy algorihm M_R needs of 5 iterations.

Table 2.

$n = 2^6 = 64$						
Algorithm	$m = 5$		$m = 7$		$m = 10$	
	Err.	time	Err.	time	Err.	time
LU	6.6613e-016	0.14	7.7716e-016	0.19	1.3323e-015	0.18
CHOL	1.1102e-015	0.11	6.6613e-016	0.16	8.8818e-016	0.16
M_R(1m)	4.4892e-015	0.08	4.4603e-015	0.11	1.1829e-014	0.11
M_R(2m)	4.3780e-015	0.06	4.4033e-015	0.06	1.1764e-014	0.08
M_RF(2m)	4.3780e-015	0.16	4.4033e-015	0.17	1.1775e-014	0.23
$n = 2^8 = 256$						
LU	6.6613e-016	0.43	7.7716e-016	0.55	1.3323e-015	0.72
CHOL	1.1102e-015	0.38	6.6613e-016	0.44	8.8818e-016	0.60
M_R(1m)	1.0148e-014	0.33	9.7788e-015	0.36	2.4572e-014	0.41
M_R(2m)	1.0099e-014	0.28	9.7529e-015	0.30	2.4541e-014	0.35
M_RF(2m)	1.0099e-014	0.77	9.7529e-015	1.05	2.4546e-014	1.54
$n = 2^{10} = 1024$						
LU	6.6613e-016	1.97	7.7716e-016	2.53	1.3323e-015	4.78
CHOL	1.1102e-015	1.59	6.6613e-016	1.82	8.8818e-016	3.24
M_R(1m)	2.0840e-014	1.26	1.9964e-014	1.39	4.9590e-014	1.98
M_R(2m)	2.0816e-014	1.21	1.9951e-014	1.26	4.9575e-014	1.71
M_RF(2m)	2.0816e-014	5.22	1.9951e-014	7.86	4.9577e-014	13.61
$n = 2^{12} = 4096$						
LU	6.6613e-016	10.820	7.7716e-016	13.35	1.3323e-015	37.52
CHOL	1.1102e-015	7.74	6.6613e-016	8.63	8.8818e-016	22.08
M_R(1m)	4.1947e-014	6.32	4.0128e-014	6.87	9.9402e-014	11.09
M_R(2m)	4.1936e-014	6.10	4.0121e-014	6.45	9.9394e-014	9.33
M_RF(2m)	4.1936e-014	51.41	4.0121e-014	92.0	9.9396e-014	205.35

Example 1: Time for implementation (in seconds) and errors

Example 2.

Let cells of the matrix M are the matrices of example 5.2 from [8] i.e. $A = I$ and B is symmetric, nonnegative and such that $Be = (1/2 - \alpha)e$, where e is the vector heaving all the entries equal to 1. $\|\tilde{B}\|_2 = \|A^{-\frac{1}{2}}BA^{-\frac{1}{2}}\|_2 = 1/2 - \alpha$. For $\alpha = 0.4$; $\|\tilde{B}\|_2 = 0.1$. To reach the requested accuracy algorihm M_R needs of 4 iterations.

Table 3.

$n = 2^6 = 64$						
Algorithm	$m = 3$		$m = 5$		$m = 10$	
	Err.	time	Err.	time	Err.	time
LU	6.6613e-016	0.11	4.4409e-016	0.11	6.6613e-016	0.22
CHOL	4.4409e-016	0.05	5.5511e-016	0.11	8.8818e-016	0.16
M_R(1m)	2.7104e-015	0.05	3.3381e-015	0.11	7.9005e-015	0.11
M_R(2m)	2.5847e-015	0.11	3.0707e-015	0.05	7.6902e-015	0.06
M_RF(2m)	2.5847e-015	0.16	3.0707e-015	0.16	7.6878e-015	0.22
$n = 2^8 = 256$						
LU	6.6613e-016	0.39	4.4409e-016	0.50	6.6613e-016	0.77
CHOL	4.4409e-016	0.33	5.5511e-016	0.39	8.8818e-016	0.66
M_R(1m)	5.5689e-015	0.28	6.0971e-015	0.28	1.5565e-014	0.44
M_R(2m)	5.5088e-015	0.27	5.9550e-015	0.27	1.5460e-014	0.38
M_RF(2m)	5.5088e-015	0.60	5.9550e-015	0.77	1.5458e-014	1.53
$n = 2^{10} = 1024$						
LU	6.6613e-016	1.70	4.4409e-016	2.52	6.6613e-016	4.83
CHOL	4.4409e-016	1.37	5.5511e-016	1.65	8.8818e-016	3.24
M_R(1m)	1.1211e-014	1.18	1.1887e-014	1.27	3.1012e-014	1.92
M_R(2m)	1.1181e-014	1.12	1.1815e-014	1.21	3.0959e-014	1.76
M_RF(2m)	1.1181e-014	3.30	1.1815e-014	5.27	3.0958e-014	13.35
$n = 2^{12} = 4096$						
LU	6.6613e-016	9.29	4.4409e-016	10.54	6.6613e-016	36.53
CHOL	4.4409e-016	6.87	5.5511e-016	7.75	8.8818e-016	22.13
M_R(1m)	2.2457e-014	14.18	2.3618e-014	6.37	6.1964e-014	10.93
M_R(2m)	2.2442e-014	13.50	2.3582e-014	6.15	6.1937e-014	9.39
M_RF(2m)	2.2442e-014	25.27	2.3582e-014	0.75	6.1937e-014	207.24

Example 2: ($\alpha = 0.4$) Time for implementation (in seconds) and errors

Example 3.

The cells A and B are chosen like in example 7.3 from [7] i.e.

$$A = \begin{pmatrix} 1.20 & -0.30 & 0.10 \\ -0.30 & 2.10 & 0.20 \\ 0.10 & 0.20 & 0.65 \end{pmatrix}, \quad B = \begin{pmatrix} 0.37 & 0.13 & 0.12 \\ -0.30 & 0.34 & 0.12 \\ 0.11 & -0.17 & 0.29 \end{pmatrix}.$$

In this case $\|\tilde{B}\|_2 = \|A^{-\frac{1}{2}}BA^{-\frac{1}{2}}\|_2 = 0.511$. To reach the requested accuracy algorihm M_R needs of 10 iterations.

Table 4.

$m = 3$				
Algorithm	$n = 2^6 = 64$		$n = 2^{10} = 1024$	
	Err.	time	Err.	time
LU	2.4647e-014	0.11	2.0384e-013	1.76
CHOL	2.7978e-014	0.11	2.2382e-013	1.42
M_R(1m)	2.9759e-013	0.06	1.5230e-012	1.20
M_R(2m)	9.1983e-013	0.06	1.0394e-012	1.15
M_RF(2m)	2.7810e-013	0.17	9.8328e-013	3.24
	$n = 2^8 = 256$		$n = 2^{12} = 4096$	
LU	2.8644e-014	0.38	2.5291e-013	9.88
CHOL	4.1522e-014	0.33	2.6268e-013	7.36
M_R(1m)	5.4734e-013	0.28	1.5286e-011	6.41
M_R(2m)	9.9020e-013	0.27	1.4844e-011	5.93
M_RF(2m)	7.0008e-013	0.65	1.4831e-011	24.77

Example 3: Time for implementation (in seconds) and errors

5 Conclusions

The proposed new algorithms M_R(1m) and M_R(2m) are most effective. We recomend the algorithm M_R(2m) for solving (1) because for realization Step 3 it essentially uses the results obtained in Step 2, which leads to decreasing the flops and time for implementation.

Acknowledgement. This work was partially supported by Shoumen University under contract N 17.

References

[1] Ahlberg, J.H., Nilson, E.N., Walsh, J.L. (1967) *The Theory of Splines and Their Applications*, Academic Press, New York, USA.

[2] Buzbee, B.L., Golub, G.H., Nielson, S.W. (1970) On the direct methods for solvind Poisson's equations *SIAM J. Numer. Analysis,* **7** 627-656.

[3] El-Sayed, S.M. A Direct Method for Solving Circulant Tridiagonal Block System of Linear Equations, *J. Australian Mathematical Society, Series B, to appear.*

[4] El-Sayed, S.M., Ivanov, I.G., Petkov, M.G. (1998) A new modification of the Rojo Method for solving symmetric circulant five-diagonal systems of linear equations. *Computers Math. Applic.,* **35** 35-44.

[5] Fiorentino, G., Serra, S. (1996) Multigrid methods for symmetric positive definite block Teoplitz matrices with nonnegative generating functions *SIAM J. Sci. Computing,* **17** 1068-1081.

[6] Golub, G., Van Loan, C. (1989) *Matrix Computation,* The John Hopkins University Press, Baltimore.

[7] Guo, C.-H., Lancaster, P. (1999) Iterative solution of two matrix equations, *The Mathematics of Computation* **68** 1589-1603.

[8] Meini, B. (2000) Matrix Equations and Structures : Efficient Solution of Special Discrete Algebraic Riccati Equations,*Numerical Analysis and Its Applications*, Second Conference on Numerical Analysis and Applications, eds Lubin Vulkov, Jerzy Wasniewski and Plamen Yalamov, Rouse 2000, 578-585.

[9] Minchev, B. Some Algorithms for Solving Special Tridiagonal Block Teoplitz Linear Systems, *submited to Numerical Algorithms.*

[10] Mingkui, C. (1987) On the solution of circulant linear systems, *SIAM J. Numer. Analysis,* **24** 668-683.

[11] Rojo, O. (1990) A new method for solving symmetric circulant tridiagonal systems of linear systems. *Computers Math. Applic.* **20** 61-67.