New algorithm for solving pentadiagonal CUPL-Toeplitz linear systems

Hcini Fahd[a,*], Yulin Zhang[b]

[a] *University of Tunis El Manar, ENIT-LAMSIN, BP 37, 1002, Tunis, Tunisia*
[b] *Centro de Matematica, Universidade do Minho, 4710-057 Braga, Portugal*

Abstract

In this paper, based on the structure of pentadiagonal CUPL-Toeplitz matrix and Sherman-Morrison-Woodbury formula, we develop a new algorithm for solving nonsingular pentadiagonal CUPL-Toeplitz linear system. Some numerical examples are given in order to illustrate the effectiveness of the proposed algorithms.

Keywords: CUPL-Toeplitz matrix, Sherman-Morrison-Woodbury formula, Toeplitz matrix, low rank matrix, *LU* decomposition.
2022 MSC: 15A23, 35L30

1. Introduction

A $n \times n$ matrix is called column upper-plus-lower (CUPL) Toeplitz matrix if it is of the following form

$$T_{CUPL} = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{1-n} \\ a_1 & a_0 + a_1 & \ddots & \ddots & \vdots \\ a_2 & a_1 + a_2 & \ddots & \ddots & a_{-2} \\ \vdots & \vdots & \ddots & \ddots & a_{-1} \\ a_{n-1} & a_{n-2} + a_{n-1} & \cdots & a_1 + a_2 & a_0 + a_1 \end{bmatrix}, \tag{1}$$

i.e., its entries satisfy

$$a_{ij} = \begin{cases} a_{i-j}, & j = 1 \text{ or } j > i \\ a_{i-j} + a_{i-j+1}, & 2 \leq j \leq i \end{cases}.$$

Obviously, a CUPL-Toeplitz matrix T_{CUPL} can be splitted as the difference of a Toeplitz matrix and a rank one matrix, i.e.,

$$T_{CUPL} = T - ae_1^T, \tag{2}$$

where $T = [a_{k,j}]$, $k, j = 0, 1, ..., n-1$ is a Toeplitz matrix, $a = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & 0 \end{bmatrix}^T$ and $e_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$.

Any Toeplitz Matrix perturbed by a lower rank matrix can be viewed as quasi-Toeplitz matrix. Therefore, the CUPL-Toeplitz matrix can be considered as a special class of quasi-Toeplitz matrix.

Quasi-Toeplitz matrices are widely used in quasi birth-and-death processes, option pricing and signal processing etc. [1, 2]. In [3], Du et al presented an efficient method for finding the solution of tridiagonal quasi Toeplitz linear systems. Later on this method was generalised to solve the block quasi tridiagonal Toeplitz [4]. In [5, 6], a fast method for solving quasi-pentadiagonal Toeplitz linear systems was presented, which was based on solving pentadiagonal Toeplitz linear systems.

*Corresponding author
Email address: hcini.fahd@enit.rnu.tn (Hcini Fahd)

As a special class of quasi-Toeplitz matrix, the CUPL-Toeplitz Matrix brought attention to the researchers in the last decades [7, 8, 9, 10]. For calculating the inverse of CUPL-Toeplitz matrix, an explicit inverses of this matrix were introduced [11]. Recently, based on the decomposition form a CUPL-Toeplitz matrix (2) and the Sherman-Morrison-Woodbury formula, Fu et al [12] presented two algorithms to solve the CUPL-Toeplitz linear system

$$T_{CUPL}x = f, \tag{3}$$

where T_{CUPL} is nonsingular.

In this paper, we propose a new algorithm for finding the solution of the nonsingular pentadiagonal CUPL-Toeplitz linear system

$$Tx = f, \tag{4}$$

where

$$T = \begin{bmatrix} a & b & c & & & & & & \\ d & a+d & b & c & & & & & \\ e & d+e & a+d & b & c & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & e & d+e & a+d & b & c & \\ & & & & e & d+e & a+d & b & \\ & & & & & e & d+e & a+d \end{bmatrix}.$$

The idea is to decompose the pentadiagonal CUPL-Toeplitz matrix into a product of $\mathcal{LU}$ matrices plus a ranks two matrix, then apply the Sherman-Morrison-Woodbury formula to get its inverse. At the and, some numerical examples are given to illustrate the effectiveness of our algorithms.

2. Algorithms for solving pentadiagonal CUPL-Toeplitz linear system

In this section, we decompose the CUPL-Toeplitz into the quasi-Toeplitz matrix plus a rank two matrix. Let

$$\mathcal{L} = \begin{bmatrix} 1 & & & & & & & \\ s & 1 & & & & & & \\ e/p & s & 1 & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & e/p & s & 1 & & \\ & & & & e/p & s & 1 & \\ & & & & & e/p & s & 1 \end{bmatrix} \text{ and } \mathcal{U} = \begin{bmatrix} 1 & t & c/p & & & & & \\ & 1 & t & c/p & & & & \\ & & 1 & t & c/p & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & 1 & t & c/p \\ & & & & & & 1 & t \\ & & & & & & & 1 \end{bmatrix}.$$

The

$$p\mathcal{LU} = \begin{bmatrix} p & pt & c & & & & & \\ ps & pst+p & b & c & & & & \\ e & c' & a' & b & c & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & e & c' & a' & b & c \\ & & & & e & c' & a' & b \\ & & & & & e & c' & a' \end{bmatrix},$$

where $c' = d + e$, $a' = a + d$. We will choose p, s and t such that the following non-linear equations are satisfied.

$$\begin{cases} et + ps - c' = 0, \\ p^2(st+1) - a'p + ec = 0, \\ pt + sc - b = 0. \end{cases} \tag{5}$$

Then the matrix T can be written as

$$T = p\mathcal{L}\mathcal{U} + \begin{bmatrix} e_1 & e_2 \end{bmatrix} \begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix} \begin{bmatrix} e_1^T \\ e_2^T \end{bmatrix}. \tag{6}$$

From (5), we observe that the parameter p satisfy the following equation

$$p^6 - a'p^5 + (bc' - ce)p^4 + (2a'ce - b^2e - c'^2c)p^3 + (bcc'e - e^2c^2)p^2 - a'e^2c^2p + e^3c^3 = 0. \tag{7}$$

Theorem 1. *The analytic solution of Eq. (7) can be written as*

$$p_{2i-1} = \frac{q_i + \sqrt{q_i^2 - 4ce}}{2} \quad \text{and} \quad p_{2i} = \frac{q_i - \sqrt{q_i^2 - 4ce}}{2}, \quad i = 1, 2, 3, \tag{8}$$

where

$$\begin{aligned} q_1 &= \mu_1 + \mu_4 + \frac{\mu_1^2 + \mu_2}{\mu_4}, \\ q_2 &= \mu_1 - \frac{\mu_4}{2} - \frac{\mu_1^2 + \mu_2}{2\mu_4} + i\frac{\sqrt{3}}{2}\left(\mu_4 - \frac{\mu_1^2 + \mu_2}{\mu_4}\right), \\ q_3 &= \mu_1 - \frac{\mu_4}{2} - \frac{\mu_1^2 + \mu_2}{2\mu_4} - i\frac{\sqrt{3}}{2}\left(\mu_4 - \frac{\mu_1^2 + \mu_2}{\mu_4}\right), \end{aligned}$$

and $\mu_1 = \frac{a'}{3}$, $\mu_2 = \frac{4ce - bc'}{3}$, $\mu_3 = \frac{4a'ce - cc'^2 - b^2e}{2}$, $\mu_4 = \sqrt[3]{\mu_1^3 + \frac{3\mu_1\mu_2}{2} - \mu_3 + \sqrt{\left(\mu_1^3 + \frac{3\mu_1\mu_2}{2} - \mu_3\right)^2 - (\mu_1^2 + \mu_2)^3}}$.

Proof. We divide the both sides of (7) by p^3, and define $q = p + \frac{c}{e}$, then the equations (7) can be written as

$$q^3 - a'q^2 - (4ce - bc)q + 4a'ce - c'^2c - b^2e = 0. \tag{9}$$

Using the Cardano's method, we can deduce the solution of equation (9), then the final solution of (7) is given by

$$p_{2i-1} = \frac{q_i + \sqrt{q_i^2 - 4ce}}{2} \quad \text{and} \quad p_{2i} = \frac{q_i - \sqrt{q_i^2 - 4ce}}{2}, \quad i = 1, 2, 3,$$

where q_1, q_2, q_3 are defined above, and the proof is complete.

Assuming $p \neq 0$ and $p \neq ce$, after obtaining p, it is easy to deduce that

$$s = \frac{c'p - be}{p^2 - ce} = \frac{(d+e)p - be}{p^2 - ce} \text{ and } t = \frac{bp - c'c}{p^2 - ce} = \frac{bp - (d+e)c}{p^2 - ce}.$$

Next, for the convenience of the readers, we transcribe the Sherman-Morrison-Woodbury formula here, which can be found in [13], Sherman-Morrison-Woodbury formula: Let A be an $n \times n$ nonsingular matrix, $n \times k$ matrix U and $k \times n$ matrix V and let B be an $n \times n$ matrix such that $B = A + UV$. Then, assuming a nonsingular matrix $(I_k + VA^{-1}U)$, we have

$$B^{-1} = A^{-1} - A^{-1}U\left(I_k + VA^{-1}U\right)^{-1}VA^{-1}.$$

According to the Sherman-Morrison-Woodbury formula, the inverse of the matrix T can be written as

$$T^{-1} = \frac{1}{p}\left[(\mathcal{LU})^{-1} - \frac{1}{p}(\mathcal{LU})^{-1}[\mathbf{e}_1, \mathbf{e}_2]\begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix}\mathcal{N}^{-1}\begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^t \end{bmatrix}(\mathcal{LU})^{-1}\right], \tag{10}$$

where

$$\mathcal{N} = I_2 + \frac{1}{p}\begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{bmatrix}(\mathcal{LU})^{-1}[\mathbf{e}_1, \mathbf{e}_2]\begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix}. \tag{11}$$

Let $y = (y_1, y_2, \ldots, y_n)^T$, $z = (z_1, z_2, \ldots, z_n)^T$ and $w = (w_1, w_2, \ldots, w_n)^T$ be the solutions of the following systems: $\mathcal{LU}y = f/p$, $\mathcal{LU}z = e_1$ and $\mathcal{LU}w = e_2$, which will be calculated by Algorithm 1.

Algorithm 1 An algorithm for solving $\mathcal{LU}y = \tilde{f}$, $\tilde{f} = f/p$

Input: s, t, p, e, c and $\tilde{f}$.

1. (solving $\mathcal{L}\tilde{z} = \tilde{f}$) $\tilde{z}_1 = \tilde{f}_1$, $\tilde{z}_2 = \tilde{f}_2 - s\tilde{z}_1$, $\tilde{z}_i = \tilde{f}_i - s\tilde{z}_{i-1} - \frac{e}{p}\tilde{z}_{i-2}$, $i = 3$ to n .
2. (solving $\mathcal{U}y = \tilde{z}$) $y_n = \tilde{z}_n$, $y_{n-1} = \tilde{z}_{n-1} - ty_n$, $y_i = \tilde{z}_i - ty_{i+1} - \frac{c}{p}y_{i+2}$, $i = n-2$ to 1.

Output: $y = [y_1, y_2 \ldots, y_n]^T$.

Thus, the matrix $\mathcal{N}$ becomes

$$\begin{aligned}
\mathcal{N} =& I_2 + \frac{1}{p}\begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \end{bmatrix}[z, w]\begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix} \\
=& \begin{bmatrix} 1 + (\frac{ec}{p^2} + st - \frac{d}{p})z_1 + (\frac{et}{p} - \frac{e}{p})w_1 & \frac{sc}{p}z_1 + \frac{ec}{p^2}w_1 \\ (\frac{ec}{p^2} + st - \frac{d}{p})z_2 + (\frac{et}{p} - \frac{e}{p})w_2 & 1 + \frac{sc}{p}z_2 + \frac{ec}{p^2}w_2 \end{bmatrix} \\
=& \begin{bmatrix} \theta_1 & \theta_2 \\ \theta_3 & \theta_4 \end{bmatrix}.
\end{aligned}$$

Then, the final solution of (4) is obtained by multiplying equation (10) by f as follows:

$$x = y - [z, w]\begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix}\mathcal{N}^{-1}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{12}$$

$$= y - [z, w]\begin{bmatrix} \frac{ec}{p} + pst - d & sc \\ et - e & \frac{ec}{p} \end{bmatrix}\frac{1}{(\theta_1\theta_4 - \theta_2\theta_3)p}\begin{bmatrix} \theta_4 & -\theta_2 \\ -\theta_3 & \theta_1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{13}$$

$$= y - \alpha z - \beta w. \tag{14}$$

where

$$\alpha = \frac{1}{(\theta_1\theta_4 - \theta_2\theta_3)p}\left((\frac{ec}{p} + pst - d)\theta_4 y_1 - sc\theta_3 y_1 - (\frac{ec}{p} + pst - d)\theta_2 y_2 + sc\theta_1 y_2\right), \tag{15}$$

and

$$\beta = \frac{1}{(\theta_1\theta_4 - \theta_2\theta_3)p}\left((et - e)\theta_4 y_1 - \frac{ec}{p}\theta_3 y_1 + (e - et)\theta_2 y_2 + \frac{ec}{p}\theta_1 y_2\right). \tag{16}$$

The execution algorithm steps are summarized as follows:

Algorithm 2 An algorithm for solving $Tx = f$

Input: a, b, c, d, e and f.

1. Choose an arbitrary root p of Eq. (7) which satisfies $p^2 - ce \neq 0$.
2. Compute $s = \frac{(d+e)p-be}{p^2-ce}$ and $t = \frac{bp-(d+e)c}{p^2-ce}$.
3. Solving $\mathcal{LU}y = f/p$, $\mathcal{LU}z = e_1$ and $\mathcal{LU}w = e_2$ by Algorithm 1.
4. Compute α, β.

Output: The solution of system (4), $x = y - \alpha z - \beta w$.

The study of the stability of our Algorithm 2 depends on the Algorithm 1, precisely, the solving the upper and lower triangular linear systems $\mathcal{LU}y = f/p$, $\mathcal{LU}z = e_1$ and $\mathcal{LU}w = e_2$. More precisely, two recursive iterations such as $y_i = sz_{i-1} - \frac{e}{p}z_{i-2}$ and $x_{n+1-i} = z_{n+1-i} - tx_{n+2-i} - \frac{c}{p}x_{n+3-i}$ for $i = 3, 4, ..., n$ corresponding to the forward and backward substitutions as in Algorithm 1 are essential for Algorithm 2. Still, if all the roots of their characteristic equations $\rho^2 + s\rho + \frac{e}{p} = 0$ and $\rho^2 + t\rho + \frac{c}{p} = 0$ are all less than unity in magnitude, errors of z_i and x_{n+1-i} will smaller than the errors of previous values z_{i-1} and x_{n+2-i}, respectively, in which case Algorithm 2 is stable.

3. Numerical Experiments

In this section, we give the results of some simple numerical experiments to demonstrate the effectiveness of our algorithms. All tests are implemented in MATLAB R2018a and the computations are done on an Intel PENTIUM computer, (2.2 GHz), 6 GB memory.

3.1. Experiment 1

In this experiment, we used five artificial examples. Values of a, b, c, d and e corresponding to each example are presented in Table 1. We initialized the exact solution $x^* = [1, 1, ..., 1]^T$. The right-hand side vector was set to be $f = Ax^*$. The absolute errors $\|x - x^*\|$, residuals $\|Ax - f\|$, and CPU time will be listed. Here, $\|.\|$ is the euclidean vector norm.
The compared results between our algorithm and algorithme [12], algorithme [14] and MATLAB Solver for all tests examples are presented in Tables 2–6.

Table 1: Test examples.

	a	b	c	d	e
Example 1	7	-1	5	2	-1.5
Example 2	0.80	0.70	0.65	-0.4	-0.2
Example 3	5.5	2.7	2.6	2.25	-5.25
Example 4	10	-2	1	0.54	1
Example 5	6	-1	-1.5	1	-2

Table 2: Numerical results of Example 1

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$1.1512e-14$	$1.1512e-14$	$1.1512e-14$	$1.1512e-14$
	$\|x-x^*\|$	$1.2462e-15$	$1.2462e-15$	$1.2462e-15$	$1.2462e-15$
	CPU(s)	0.0010	0.018	0.081	0.85
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$1.4862e-14$	$1.4862e-14$	$1.4862e-14$	$1.4862e-14$
	$\|x-x^*\|$	$1.3276e-15$	$1.3276e-15$	$1.3276e-15$	$1.3276e-15$
	CPU(s)	0.0037	0.021	0.17	1.65
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$1.9860e-14$	$4.2596e-14$	$1.2655e-13$	*Failure*
	$\|x-x^*\|$	$2.3708e-15$	$5.2733e-15$	$1.5801e-14$	*Failure*
	CPU(s)	0.0028	0.024	7.30	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$1.5486e-14$	$4.0740e-14$	$1.2593e-13$	*Failure*
	$\|x-x^*\|$	$1.0934e-15$	$2.5966e-15$	$7.8873e-15$	*Failure*
	CPU(s)	0.0046	0.054	1.06	*Failure*

Table 3: Numerical results of Example 2

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$3.6422e-15$	$1.0987e-14$	$3.4541e-14$	$1.0916e-13$
	$\|x-x^*\|$	$4.9214e-15$	$1.1958e-14$	$3.6418e-14$	$1.1471e-13$
	CPU(s)	0.0010	0.0083	0.081	0.82
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$3.8751e-15$	$1.1029e-14$	$3.4554e-14$	$1.0917e-13$
	$\|x-x^*\|$	$5.0120e-15$	$1.1966e-14$	$3.6421e-14$	$1.1471e-13$
	CPU(s)	0.0026	0.017	0.16	1.65
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$2.4751e-15$	$7.1534e-15$	$2.2246e-14$	*Failure*
	$\|x-x^*\|$	$3.4255e-15$	$1.2474e-14$	$3.9506e-14$	*Failure*
	CPU(s)	0.0021	0.027	7.74	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$2.2093e-15$	$3.9968e-15$	$1.1265e-14$	*Failure*
	$\|x-x^*\|$	$3.1225e-15$	$5.6512e-15$	$1.5931e-14$	*Failure*
	CPU(s)	0.0046	0.049	1.07	*Failure*

Table 4: Numerical results of Example 3

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$1.4789e-14$	$2.1224e-14$	$2.1224e-14$	$2.1224e-14$
	$\|x-x^*\|$	$1.4937e-15$	$2.1384e-15$	$2.1384e-15$	$2.1384e-15$
	CPU(s)	0.0013	0.0088	0.091	0.99
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$1.6829e-14$	$2.2399e-14$	$2.2399e-14$	$2.2399e-14$
	$\|x-x^*\|$	$1.5662e-15$	$2.1614e-15$	$2.1614e-15$	$2.1614e-15$
	CPU(s)	0.0028	0.017	0.30	1.98
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$2.2227e-14$	$5.8135e-14$	$1.7827e-13$	*Failure*
	$\|x-x^*\|$	$2.2343e-15$	$1.0427e-14$	$3.4901e-14$	*Failure*
	CPU(s)	0.0022	0.029	6.94	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$1.6998e-14$	$4.9182e-14$	$1.5401e-13$	*Failure*
	$\|x-x^*\|$	$1.7659e-15$	$5.0304e-15$	$1.5722e-14$	*Failure*
	CPU(s)	0.0058	0.054	1.02	*Failure*

Table 5: Numerical results of Example 4

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$1.1783e-14$	$1.1783e-14$	$1.1783e-14$	$1.1783e-14$
	$\|x-x^*\|$	$7.7716e-16$	$7.7716e-16$	$7.7716e-16$	$7.7716e-16$
	CPU(s)	0.0016	0.0091	0.092	0.90
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$1.2561e-14$	$1.2561e-14$	$1.2561e-14$	$1.2561e-14$
	$\|x-x^*\|$	$8.1584e-16$	$8.1584e-16$	$8.1584e-16$	$8.1584e-16$
	CPU(s)	0.0038	0.020	0.21	1.72
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$1.0358e-14$	$1.0358e-14$	$1.0358e-14$	*Failure*
	$\|x-x^*\|$	$1.1697e-15$	$2.6296e-15$	$7.8982e-15$	*Failure*
	CPU(s)	0.0043	0.024	6.03	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$5.0243e-15$	$5.0243e-15$	$5.0243e-15$	*Failure*
	$\|x-x^*\|$	$7.8439e-16$	$7.8439e-16$	$7.8439e-16$	*Failure*
	CPU(s)	0.0056	0.054	0.98	*Failure*

Table 6: Numerical results of Example 5

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$1.2829e-14$	$1.8539e-14$	$4.6029e-14$	$1.4095e-13$
	$\|x-x^*\|$	$7.0497e-15$	$9.7099e-15$	$2.3195e-14$	$7.0536e-14$
	CPU(s)	0.0015	0.0095	0.097	0.86
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$1.5414e-14$	$2.0423e-14$	$4.6819e-14$	$1.4121e-13$
	$\|x-x^*\|$	$7.1702e-15$	$9.7914e-15$	$2.3229e-14$	$7.0547e-14$
	CPU(s)	0.0033	0.021	0.18	1.75
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$1.0975e-14$	$2.8832e-14$	$8.9056e-14$	*Failure*
	$\|x-x^*\|$	$2.9395e-15$	$2.0180e-14$	$6.6339e-14$	*Failure*
	CPU(s)	0.0030	0.024	7.23	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$7.1158e-15$	$2.1213e-14$	$6.6660e-14$	*Failure*
	$\|x-x^*\|$	$3.3510e-15$	$1.3738e-14$	$4.4313e-14$	*Failure*
	CPU(s)	0.0052	0.049	0.94	*Failure*

For all numerical tests, we can see that our algorithm takes less CPU time and is about twice as fast as other algorithms.

3.2. Experiment 2

In this experiment, we considered the exact solution $x^* = [-3,-3,...,-3]^T$, and T a pentadiagonal CUPL-Toeplitz matrix. Then, our pentadiagonal CUPL-Toeplitz linear system is written as follows:

$$\begin{bmatrix} 9 & -1 & 2 & & & & \\ 1 & 10 & -1 & 2 & & & \\ 1 & 2 & 10 & -1 & 2 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 2 \\ & & & 1 & 2 & 10 & -1 \\ & & & & 1 & 2 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \\ -3 \\ \vdots \\ -3 \\ -3 \\ -3 \end{bmatrix}$$

The compared results, absolute errors $\|x-x^*\|$, residuals $\|Ax-f\|$, and CPU time between our algorithm and algorithme [12], algorithme [14] and MATLAB Solver are presented in Table 7.

Table 7: Numerical results of Experiment 2

	n	100	1000	10000	100000
Our Algorithm	$\|Ax-\mathbf{f}\|$	$1.7764e-14$	$1.7764e-14$	$1.7764e-14$	$1.7764e-14$
	$\|x-x^*\|$	$1.9860e-15$	$1.9860e-15$	$1.9860e-15$	$1.9860e-15$
	CPU(s)	0.0012	0.0096	0.094	0.87
Algorithm [12]	$\|Ax-\mathbf{f}\|$	$2.2748e-14$	$2.2748e-14$	$2.2748e-14$	$2.2748e-14$
	$\|x-x^*\|$	$2.4726e-15$	$2.4726e-15$	$2.4726e-15$	$2.4726e-15$
	CPU(s)	0.0030	0.020	0.19	1.75
Algorithm [14]	$\|Ax-\mathbf{f}\|$	$3.9080e-14$	$3.9080e-14$	$3.9080e-14$	*Failure*
	$\|x-x^*\|$	$5.1789e-15$	$1.0750e-14$	$3.1671e-14$	*Failure*
	CPU(s)	0.0027	0.042	8.73	*Failure*
MATLAB Solver	$\|Ax-\mathbf{f}\|$	$1.5888e-14$	$1.5888e-14$	$1.5888e-14$	*Failure*
	$\|x-x^*\|$	$3.4111e-15$	$1.0019e-14$	$3.1430e-14$	*Failure*
	CPU(s)	0.0051	0.050	1.01	*Failure*

Table 8: Time ratio of Experiment 2.

n	Time ratio		
	Our Algorithm/ Algorithm[12]	Our Algorithm/ Algorithm[14]	Our Algorithm/MATLAB Solver
100	0.4000	0.444	0.2353
1000	0.4800	0.228	0.1920
10000	0.4947	0.0107	0.0931
100000	0.4971	-	-

We can see that our proposed algorithm takes less CPU time and is about twice faster than the algorithme [12], algorithme [14] and MATLAB Solver.

Remark 1. In step 1 of the algorithm presented in [12], we used fast pentadiagonal Toeplitz solver and MATLAB Solver, is a solver for pentadiagonal linear system from MATLAB.

4. Conclusion

In this paper we have exploited the special pentadiagonal CUPL-Toeplitz structure to develop a fast algorithm for solving pentadiagonal CUPL-Toeplitz linear systems. We compare our algorithm with the algorithms [12], [14] and MATLAB Solver, the numerical results show that our algorithm is more efficient, in terms of time, almost half that of the other algorithms and in terms of error.

Funding

The last author was partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the Projects UIDB/00013/2020 and UIDP/00013/2020.

Declarations

Ethical Approval

Not Applicable.

Availability of supporting data

Not Applicable.

Competing interests

The author has no relevant financial or non-financial interests to disclose.

Authors' contributions

The authors confirm sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation

Acknowledgments

Not applicable.

References

[1] Marcel F Neuts. *Structured stochastic matrices of M/G/1 type and their applications*. CRC Press, 2021.

[2] Marcel F Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Courier Corporation, 1994.

[3] Lei Du, Tomohiro Sogabe, and Shao-Liang Zhang. A fast algorithm for solving tridiagonal quasi-toeplitz linear systems. *Applied Mathematics Letters*, 75:74–81, 2018.

[4] Skander Belhaj, Fahd Hcini, and Yulin Zhang. A fast method for solving a block tridiagonal quasi-toeplitz linear system. *Portugaliae Mathematica*, 76(3):287–299, 2020.

[5] Skander Belhaj, Fahd Hcini, Maher Moakher, and Yulin Zhang. A fast algorithm for solving diagonally dominant symmetric quasi-pentadiagonal toeplitz linear systems. *Journal of Mathematical Chemistry*, 59(3):757–774, 2021.

[6] Skander Belhaj, Fahd Hcini, and Maher Moakher. A fast method for solving quasi-pentadiagonal toeplitz linear systems and its application to the lax–wendroff scheme. *Mathematics and Computers in Simulation*, 188:77–86, 2021.

[7] Dario A Bini, Stefano Massei, and Beatrice Meini. On functions of quasi-toeplitz matrices. *Sbornik: Mathematics*, 208(11):1628, 2017.

[8] Dario Bini, Stefano Massei, and Beatrice Meini. Semi-infinite quasi-toeplitz matrices with applications to qbd stochastic processes. *Mathematics of Computation*, 87(314):2811–2830, 2018.

[9] Dario A Bini, Stefano Massei, and Leonardo Robol. Quasi-toeplitz matrix arithmetic: a matlab toolbox. *Numerical Algorithms*, 81(2):741–769, 2019.

[10] Dario A Bini and Beatrice Meini. On the exponential of semi-infinite quasi-toeplitz matrices. *Numerische Mathematik*, 141(2):319–351, 2019.

[11] Zhao-Lin Jiang, Xiao-Ting Chen, and Jian-Min Wang. The explicit inverses of cupl-toeplitz and cupl-hankel matrices. *East Asian Journal on Applied Mathematics*, 7(1):38–54, 2017.

[12] Yaru Fu, Xiaoyu Jiang, Zhaolin Jiang, and Seongtae Jhang. Fast algorithms for finding the solution of cupl-toeplitz linear system from markov chain. *Applied Mathematics and Computation*, 396:125859, 2021.

[13] William W Hager. Updating the inverse of a matrix. *SIAM review*, 31(2):221–239, 1989.

[14] Tomohiro Sogabe. New algorithms for solving periodic tridiagonal and periodic pentadiagonal linear systems. *Applied Mathematics and Computation*, 202(2):850–856, 2008.