A fast algorithm for solving diagonally dominant symmetric quasi-pentadiagonal Toeplitz linear systems

Skander Belhaj[a,1,*], Fahd Hcini[a], Maher Moakher[a], Yulin Zhang[b]

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

Abstract

In this paper, we develop a new algorithm for solving diagonally dominant symmetric quasi-pentadiagonal Toeplitz linear systems. Numerical experiments are given in order to illustrate the validity and efficiency of our algorithm.

Keywords: Quasi-pentadiagonal Toeplitz matrix, Diagonally dominant, LU decomposition.
2019 MSC: 15A23, 35L30

1. Introduction

In this paper, we will focus on the problem of solving

$$Tx = f \tag{1}$$

where T is a quasi-pentadiagonal Toeplitz matrix.

An $n \times n$ matrix $T = (t_{ij})$ is said to be Toeplitz if $t_{i,j} = t_{i-j}$. T is said to be banded Toeplitz if there are positive integers p and q such that $p + q = k < n$ and $t_v = 0$ if $v > q$ or $v < -p$. A banded quasi-Toeplitz matrix is defined to be a banded Toeplitz matrix where there are at most p altered rows among the first p rows and at most q altered rows among the last q rows. For example, when $p = q = 1$, and only the first row and the last row of T are perturbed, then T is said to be quasi-tridiagonal Toeplitz matrix, the numerical solution of $Tx = f$ for this kind of linear equations was studied by [1], and more general, the numerical solution of block quasi-tridiagonal Toeplitz matrix was studied by [2]. Here we will study the case when $p = q = 2$, and only the first two rows and the last two rows of T are perturbed i.e., when T is a quasi-pentadiagonal Toeplitz matrix.

Pentadiagonal matrices and quasi-pentadiagonal matrices frequently arise in many application areas, such as computational physics, scientific and engineering computings [3, 4, 5, 6], as well as in the wave-function formalism [7] and density functional theory [8] in quantum chemistry. The importance of these applications motivated an extensive theoretical study of these kinds of matrices, such as determinant evaluation, eigenvalues computing and pentadiagonal linear systems solving in the last decades, see for example [9, 10] and a large literature therein.

In this work, we will present a fast algorithm for the numerical solution of an $n \times n$, nonsingular, diagonally dominant, symmetric quasi-pentadiagonal Toeplitz linear system. In other words, the cofficient matrix of (1) is

*Corresponding author
Email address: skander.belhaj@lamsin.rnu.tn (Skander Belhaj)

$$T = \begin{bmatrix} x & y & z & & & & & \\ p & q & r & s & & & & \\ c & b & a & b & c & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & c & b & a & b & c \\ & & & & t & w & k & e \\ & & & & & g & d & h \end{bmatrix}.$$

and

$$|a| > 2(|b| + |c|),\ c \neq 0. \tag{2}$$

When $x = q = k = h = a$, $s = z = t = g = c$, and $d = w = e = p = r = y = b$, the matrix T becomes a symmetric pentadiagonal Toeplitz matrix. This case was studied in [11, 12, 13]. For the general case of nonsymmetric pentadiagonal linear systems, algorithms have been introduced in [14, 15].

In the following sections, we will introduce an algorithm for solving the diagonally dominant symmetric quasi-pentadiagonal Toeplitz linear systems (1). Then present the numerical results.

2. An algorithm for solving quasi-pentadiagonal Toeplitz linear systems

In general, we can deal with the quasi-pentadiagonal Toeplitz linear systems (1) as an usual linear systems, and solve it by the LU decomposition without pivoting. Here we give an alterative choice, we factor the quasi-pentadiagonal Toeplitz into the following form

$$T = LU + SV + PQ \tag{3}$$

where

$$L = \begin{bmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ l_1 & l_2 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & l_1 & l_2 & 1 \end{bmatrix},\ U = \begin{bmatrix} u_1 & u_2 & c & & & \\ & u_1 & u_2 & c & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & c \\ & & & & u_1 & u_2 \\ & & & & & u_1 \end{bmatrix},\ l_1, l_2, u_1, u_2 \in \mathbb{R},$$

$S = \begin{bmatrix} e_1 & e_2 \end{bmatrix}$, $P = \begin{bmatrix} e_{n-1} & e_n \end{bmatrix}$ and e_i is the ith column of the identity matrix I_n.

$$V = \begin{bmatrix} x - u_1 & y - u_2 & z - c & 0 & 0 \dots 0 \\ p - l_2 u_1 & q - (l_2 u_2 + u_1) & r - (c l_2 + u_2) & s - c & 0 \dots 0 \end{bmatrix},\ \text{and}$$

$$Q = \begin{bmatrix} 0 & \cdots & 0 & t - u_1 l_1 & w - (l_1 u_2 + l_2 u_1) & k - (c l_1 + l_2 u_2 + u_1) & e - (u_2 + c l_2) \\ 0 & \cdots & 0 & 0 & g - l_1 u_1 & d - (l_1 u_2 + l_2 u_1) & h - (c l_1 + l_2 u_2 + u_1) \end{bmatrix}.$$

By (3), the system (1) becomes

$$(LU + SV + PQ)x = f. \tag{4}$$

Multiplying (4) by $(LU)^{-1}$, where it is assumed to be non-singular, we obtain the following system

$$(I + ZV + WQ)x = x' \tag{5}$$

where $Z = (LU)^{-1}S$, $W = (LU)^{-1}P$ and $x' = (LU)^{-1}f$. The matricees Z, W and x' can be calculated by Algorithm 1.

From (5), the final solution of (1) is given by

$$x = (I + ZV + WQ)^{-1}x'. \tag{6}$$

Next step, we will use Sherman-Morrison-Woodbury inversion formula to give the inverse of $(I + ZV + WQ)$.

Let $G = \begin{bmatrix} Z & W \end{bmatrix}$ and $H = \begin{bmatrix} V \\ Q \end{bmatrix}$, then $ZV + WQ = GH$, now apply Sherman-Morrison-Woodbury inversion formula directly to $(I + GH)^{-1}$, we have that

$$(I + GH)^{-1} = I - G(I + HG)^{-1}H = I - G(I + \begin{bmatrix} V \\ Q \end{bmatrix} [Z, W])^{-1}H = I - [Z, W]N^{-1} \begin{bmatrix} V \\ Q \end{bmatrix}$$

where

$$N = \begin{bmatrix} I + VZ & VW \\ QZ & I + QW \end{bmatrix}$$

is a matrix of the order 4×4, which is assumed to be non-singular and its inverse is very easy to get.

Finally we can obtain the solution x of (1) as

$$x = x' - [Z, W]N^{-1} \begin{bmatrix} V \\ Q \end{bmatrix} x' \tag{7}$$

Now all we need is to determine u_1, u_2, l_1 and l_2, once these values are determined, we may go to Algorithm 2 to solve our equation.

2.1. Determination of parameters u_1, u_2, l_1 and l_2

In this section we discuss how to determine the parameters u_1, u_2, l_2 and l_2.

By (3) we have the four equations

$$l_1 u_1 = c \tag{8}$$
$$cl_2 + u_2 = b \tag{9}$$
$$l_1 u_2 + l_2 u_1 = b \tag{10}$$
$$cl_1 + l_2 u_2 + u_1 = a. \tag{11}$$

By (9), $u_1 = \frac{c}{l_1}$, and by (10,) $u_2 = b - cl_2$. Replacing u_1 and u_2 into equations (11) and (12), respectively, we obtain two quadratic equations

$$(b - l_2 c)l_1^2 - bl_1 + cl_2 = 0 \tag{12}$$

$$cl_1 l_2^2 - bl_1 l_2 - (c - al_1 + cl_1^2) = 0. \tag{13}$$

By solving equation (12) we have

$$l_1 = 1 \text{ or } l_1 = \frac{cl_2}{b - cl_2}.$$

Case 1: When $l_1 = 1$.

In this case we have that $u_1 = c$,

$$l_2 = \frac{b \pm \sqrt{b^2 - 4c(a - 2c)}}{2c},$$

and

$$u_2 = \frac{b \pm \sqrt{b^2 - 4c(a - 2c)}}{2}.$$

These solutions are not stable in digital tests.

Case 2: When $l_1 = \frac{cl_2}{b-cl_2}$. (Here we assume that $b \neq 0$, if not $l_1 = 1$.)

First we assume that $b - cl_2 \neq 0$. Replacing $l_1 = \frac{cl_2}{b-cl_2}$ in (13), we obtain the following quadratic equation

$$l_2^4 + m_1 l_2^3 + m_2 l_2^2 + m_3 l_2 + m_4 = 0, \tag{14}$$

where $m_1 = -\frac{2b}{c}$, $m_2 = \frac{b^2+ac+2c^2}{c^2}$, $m_3 = -\frac{2bc+ab}{c^2}$, $m_4 = (\frac{b}{c})^2$.
Let $l_2 = \gamma - \frac{m_1}{4}$, then (14) becomes

$$\gamma^4 + \xi\gamma^2 + \eta = 0, \tag{15}$$

with

$$\xi = m_2 - \frac{3m_1^2}{8} = \frac{1}{2c^2}(4c^2 + 2ac - b^2),$$

$$\eta = m_4 - \frac{3m_1^4}{256} + \frac{m_1^2 m_2}{16} - \frac{m_1 m_3}{4} = \frac{1}{16c^2}(b^4 + 8b^2c^2 - 4acb^2).$$

After a simple calculation, we get the solutions of (15). Furthermore, we get the four roots of (14), they are

$$l_2^{(1)} = \frac{1}{2c}\left(b - \sqrt{-2c\sqrt{4ac + a^2 - 4b^2 + 4c^2} - 2ac + b^2 - 4c^2}\right)$$

$$l_2^{(2)} = \frac{1}{2c}\left(b - \sqrt{2c\sqrt{4ac + a^2 - 4b^2 + 4c^2} - 2ac + b^2 - 4c^2}\right)$$

$$l_2^{(3)} = \frac{1}{2c}\left(b + \sqrt{-2c\sqrt{4ac + a^2 - 4b^2 + 4c^2} - 2ac + b^2 - 4c^2}\right)$$

$$l_2^{(4)} = \frac{1}{2c}\left(b + \sqrt{2c\sqrt{4ac + a^2 - 4b^2 + 4c^2} - 2ac + b^2 - 4c^2}\right).$$

Knowing l_2, we may calculate l_1, u_1 and u_2 easily.
When $b - cl_2 = 0$, that is $u_2 = 0$, we may get $u_1 = c$, $l_1 = 1$, and $l_2 = \frac{b}{c}$.

2.2. Selection of l_2

In this section, we will discuss the choice of l_2. There are four l_2s, the selected l_2 must gaurantee the inverse of L and U exist. Let's look at the structure of L^{-1} and U^{-1}. (The inverse can be calculated by $A^{-1} = adj(A)/detA$).

Let

$$L = \begin{bmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ l_1 & l_2 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & l_1 & l_2 & 1 \end{bmatrix}, \quad \text{then} \quad L^{-1} = \begin{bmatrix} 1 & & & & & \\ \pi_1 & 1 & & & & \\ \pi_2 & \pi_1 & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ \pi_n & \dots & \dots & \pi_2 & \pi_1 & 1 \end{bmatrix}$$

where $\pi_1 = -l_2$, $\pi_2 = \begin{vmatrix} l_2 & 1 \\ l_1 & l_2 \end{vmatrix}$, $\cdots$, $\pi_n = (-1)^{n-1} \begin{vmatrix} l_2 & 1 & \dots & 0 \\ l_1 & l_2 & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & \dots & l_1 & l_2 \end{vmatrix}$.

From here, we can see that, if $|l_2| > 1$, with the increase of n, the down-left corner of L^{-1} will become larger and larger, and at the end, tends infinity. So $|l_2| < 1$ is a sufficient condition to guarantee our process going on. On the orther hand,

$$U = \begin{bmatrix} u_1 & u_2 & c & & & & \\ & u_1 & u_2 & c & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & u_1 & u_2 & c \\ & & & & & u_1 & u_2 \\ & & & & & & u_1 \end{bmatrix}, \text{ then } U^{-1} = \begin{bmatrix} \phi_1 & \phi_2 & \dots & \dots & \dots & \dots & \phi_n \\ & \phi_2 & \phi_2 & \ddots & & & \vdots \\ & & \ddots & \ddots & \ddots & & \vdots \\ & & & \ddots & \ddots & \ddots & \vdots \\ & & & & \phi_1 & \phi_2 & \vdots \\ & & & & & \phi_1 & \phi_2 \\ & & & & & & \phi_1 \end{bmatrix}.$$

where $\phi_1 = \frac{1}{u_1}$, $\phi_2 = -\frac{u_2}{u_1^2}$, $\cdots$, $\phi_n = \dfrac{(-1)^{n-1}\begin{vmatrix} u_2 & c & \dots & 0 \\ u_1 & u_2 & c & \\ & \ddots & \ddots & \ddots \\ 0 & \dots & u_1 & u_2 \end{vmatrix}}{u_1^n}$.

$$\begin{aligned} \phi_n =& (-1)^{n-1}\frac{* + \cdots + (n-1)cu_1u_2^{n-3} + u_2^{n-1}}{u_1^n} \\ =& (-1)^{n-1}(* + \cdots + (n-1)c(\frac{u_2}{u_1})^{n-3}\frac{1}{u_1^2} + (\frac{u_2}{u_1})^{n-1}\frac{1}{u_1}), \end{aligned}$$

so if $|\frac{u_2}{u_1}| < 1$, ϕ_n is convergent.
From $u_2 = b - cl_2$ and $l_1 = \frac{cl_2}{b-cl_2} = \frac{cl_2}{u2}$, then we have that

$$u_1 = \frac{c}{l_1} = \frac{u_2}{l_2},$$

which gives

$$l_2 = \frac{u_2}{u_1},$$

so

$$|\frac{u_2}{u_1}| < 1 \text{ is equivalent to } |l_2| < 1.$$

Therefore $|l_2| < 1$ is sufficiently to guarantee that L^{-1} and U^{-1} converge.
In the next, we take a, b, c all positive as an example to show that there exists an l_2 which satisfies the required conditions. For the other cases, the arguments are similar.

We take $l_2^{(2)}$ as an example to prove that $l_2 \in \mathbb{R}$.

Theorem 1. *Suppose that a, b and c are all positive, and let $l_2 = l_2^{(2)}$, i.e.,*

$$l_2 = \frac{1}{2c}(b - \sqrt{2c\sqrt{4ac + a^2 - 4b^2 + 4c^2} - 2ac + b^2 - 4c^2}).$$

Then l_2 is real.

Proof. We first show $4ac + a^2 - 4b^2 + 4c^2 > 0$. By our hypothsis, the matrix T is diagonally dominant i.e., $|a| > 2(|b| + |c|)$, or $|a| - 2|c| > |b|$.
So

$$4ac + a^2 - 4b^2 + 4c^2 = (a + 2c)^2 - (2b)^2 > (2b)^2 - (2b)^2 = 0.$$

Next, we will show

$$2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2\geq 0.$$

When $-2ac+b^2-4c^2\geq 0$, the inequality holds true. We consider only

$$2ac-b^2+4c^2>0.$$

In fact,

$$\begin{aligned} & 2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2\geq 0 \\ \Leftrightarrow\quad & 4c^2(4ac+a^2-4b^2+4c^2)\geq (2ac-b^2+4c^2)^2 \\ \Leftrightarrow\quad & 4ac-8c^2-b^2\geq 0 \end{aligned}$$

By $a>2(b+c)$, we have

$$4ac-8c^2-b^2>4(2(b+c))c-8c^2-b^2=8bc-b^2.$$

When $c>b$, then

$$8bc-b^2>8b^2-b^2>0,$$

the inequality holds true.

When $c<b$, we consider in two cases.

(i) $c\leq b\leq 2c$. In this case, $a>2(b+c)\geq 4c$.
Then we have $b^2\leq 4c^2$, $b^2+8c^2\leq 4c^2+8c^2=12c^2$ and $4ac>4(4c)c=16c^2$.

So that

$$4ac-8c^2-b^2>16c^2-(8c^2+b^2)>16c^2-12c^2>0$$

(ii) $b>2c$. In this case, $a>2(b+c)\geq 6c$ and $2ac>2(6c)c=12c^2$. Since we consider only $2ac>b^2-4c^2$, so we have $b^2+8c^2=b^2-4c^2+12c^2<2ac+12c^2$
Then

$$4ac-8c^2-b^2>4ac-(2ac+12c^2)=2ac-12c^2>0.$$

So l_2 is real, and we end the proof.

Theorem 2. *Under the assumption of Theorem 1, we have that* $|l_2|<1$.

Proof. We first prove that $|l_2|<1$, i.e., $-1<l_2<1$. We begin by proving the left side, that is $-1<l_2$.

$$\begin{aligned} -2c<\ & b-\sqrt{2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2} \\ \Longrightarrow\quad & (2c+b)^2\ >\ \sqrt{2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2}^2 \\ \Longrightarrow\quad & 2ac+4bc+8c^2>2c\sqrt{4ac+a^2-4b^2+4c^2} \\ \Longrightarrow\quad & 2a+4b+8c>2\sqrt{4ac+a^2-4b^2+4c^2} \\ \Longrightarrow\quad & (2a+4b+8c)^2>(2\sqrt{4ac+a^2-4b^2+4c^2})^2 \\ \Longrightarrow\quad & 4a^2+16ab+32ac+16b^2+64bc+64c^2>4a^2+16ac-16b^2+16c^2 \end{aligned}$$

$$\implies 4a^2+16ab+32ac+16b^2+64bc+64c^2-(4a^2+16ac-16b^2+16c^2)>0$$

$$\implies 32b^2+64bc+16ab+48c^2+16ac>0$$

Since a, b, c are positive, so the left side holds true.

Now we prove the right side, that is $l_2 < 1$.

$$b-\sqrt{2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2}\ <2c$$

gives

$$b-2c<\sqrt{2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2}.$$

If $b-2c<0$, then the inequality holds true. In the following, we suppose that $b>2c$.

$$(b-2c)^2\ <\ (\sqrt{2c\sqrt{4ac+a^2-4b^2+4c^2}-2ac+b^2-4c^2})^2$$

$$\implies -4bc+4c^2<2c\sqrt{a^2+4ac-4b^2+4c^2}-2ac\ -4c^2$$

$$\implies (2\sqrt{a^2+4ac-4b^2+4c^2})^2>\ (-4b+8c+2a\ )^2$$

$$\implies 4a^2+16ac-16b^2+16c^2>4a^2-16ab+32ac+16b^2-64bc+64c^2$$

$$\implies -2b^2+4bc+ab-3c^2-ac>0$$

Since $b>2c$ and $a>2(b+c)$, so

$-2b^2+4bc+ab-3c^2-ac=-2b^2+4bc-3c^2+ab-ac=-2b^2+4bc-3c^2+a(b-c)>$
$-2b^2+4bc-3c^2+2(b+c)(b-c)=4bc-5c^2>4(2c)c-5c^2>0.$

Therefore $|l_2|<1$ is true. So the proof of this theorem is concluded.

According to the signs of a, b, c, we give the following table for the selection of l_2.

a	b	c	l_2
+	+	+	$l_2^{(2)}$
−	+	+	$l_2^{(1)}$
+	−	+	$l_2^{(2)}$
+	+	−	$l_2^{(2)}$
−	−	+	$l_2^{(3)}$
+	−	−	$l_2^{(4)}$
−	+	−	$l_2^{(1)}$
−	−	−	$l_2^{(3)}$

2.3. Case $b=0$

In the previous section, we assume that $b\neq 0$. Here we study what happens when $b=0$. By solving equations (8)-(11), we get 6 solutions of the system.

(1) $l_1 = \frac{1}{c}\left(\frac{1}{2}a + \frac{1}{2}\sqrt{a^2-4c^2}\right), l_2 = 0, u_1 = \frac{1}{2}a - \frac{1}{2}\sqrt{a^2-4c^2}, u_2 = 0.$

(2) $l_1 = -\frac{1}{c}\left(-\frac{1}{2}a + \frac{1}{2}\sqrt{a^2-4c^2}\right), l_2 = 0, u_1 = \frac{1}{2}a + \frac{1}{2}\sqrt{a^2-4c^2}, u_2 = 0.$

(3) $l_1 = 1, l_2 = -\frac{1}{c}\sqrt{-c(a-2c)}, u_1 = c, u_2 = \sqrt{-c(a-2c)}.$

(4) $l_1 = 1, l_2 = \frac{1}{c}\sqrt{-c(a-2c)}, u_1 = c, u_2 = -\sqrt{-c(a-2c)}.$

(5) $l_1 = -1, l_2 = -\frac{1}{c}\sqrt{c(a+2c)}, u_1 = -c, u_2 = \sqrt{-c(a+2c)}.$

(6) $l_1 = -1, l_2 = \frac{1}{c}\sqrt{c(a+2c)}, u_1 = -c, u_2 = -\sqrt{-c(a+2c)}.$

Let's look at solution (1).

$$l_1 = \frac{1}{c}\left(\frac{1}{2}a + \frac{1}{2}\sqrt{a^2-4c^2}\right),\ l_2 = 0,\ u_1 = \frac{1}{2}a - \frac{1}{2}\sqrt{a^2-4c^2},\ u_2 = 0.$$

For l_1, u_1 to be real, we need $a^2 - 4c^2 \geq 0$, since our cofficient matrix is diagonally dominant, so this condition is guaranteed.

When $l_2 = 0$ and $u_2 = 0$,

$$L^{-1} = \begin{bmatrix} 1 & & & & & & & \\ 0 & 1 & & & & & & \\ -l_1 & 0 & \ddots & & & & & \\ 0 & -l_1 & \ddots & \ddots & & & & \\ l_1^2 & 0 & -l_1 & \ddots & \ddots & & & \\ 0 & l_1^2 & 0 & -l_1 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ (-1)^n l_1^{(\frac{n-1}{2})} & \dots & 0 & l_1^2 & 0 & -l_1 & 0 & 1 \end{bmatrix}$$

and

$$U^{-1} = \begin{bmatrix} \frac{1}{u_1} & 0 & -\frac{c}{u_1^2} & 0 & \dots & \dots & (-1)^n \frac{c^{(\frac{n-1}{2})}}{u_1^{n-1}} \\ & \frac{1}{u_1} & 0 & -\frac{c}{u_1^2} & 0 & & \vdots \\ & & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \ddots & \ddots & 0 \\ & & & & \ddots & \ddots & -\frac{c}{u_1^2} \\ & & & & & \ddots & 0 \\ & & & & & & \frac{1}{u_1} \end{bmatrix},$$

we need $|l_1| \leq 1$ and $|u_1| \geq 1$ to guarantee the convergence of L^{-1} and U^{-1}. We consider first $|l_1| \leq 1$, that is equivalent to

$$|a + \sqrt{a^2-4c^2}| \leq 2|c|.$$

When $c > 0$,

$$|a + \sqrt{a^2-4c^2}| \leq 2|c| \iff -2c \leq a + \sqrt{a^2-4c^2} \leq 2c.$$

When $c < 0$,

$$|a + \sqrt{a^2 - 4c^2}| \leq 2|c| \iff 2c \leq a + \sqrt{a^2 - 4c^2} \leq -2c.$$

By a straightforward calculation, we get the solution for $|l_1| \leq 1$, which is $a < 0$.

Now we consider $|u_1| \geq 1$. Again, by a straightforward calculation, we get the solution for $|u_1| \geq 1$, which is $a \leq -2$. So when $a \leq -2$, we have that $|l_1| \leq 1$ and $|u_1| \geq 1$.

By analogous arguments, we get that when $a \geq 2$, $|l_1| \leq 1$ and $|u_1| \geq 1$ are guaranteed by solution (2), i.e., .

$$(2) \quad l_1 = -\frac{1}{c}\left(-\frac{1}{2}a + \frac{1}{2}\sqrt{a^2 - 4c^2}\right), \quad l_2 = 0, \quad u_1 = \frac{1}{2}a + \frac{1}{2}\sqrt{a^2 - 4c^2}, \quad u_2 = 0.$$

So when $a \leq -2$, we choose solution (1) and when $a \geq 2$ we choose solution (2). And when $a \in (-2, 2)$, we may simply solve the system $\frac{2}{a}Tx = \frac{2}{a}f$.

Remark 1. The other four solutions are not suitable for the case $b = 0$.

2.4. The algorithm

In this subsection we give an algorithm for solving (1). We first give the Algorithm 1 to solve $LUy = f$, then Algorithm 2 to solve the equation (1), i.e., $Tx = f$.

Algorithm 1 An algorithm for solving $LUy = f$

Input: l_1, l_2, u_1,u_2, c and f

1. (solving $Lz = f$) $z_1 = f_1$, $z_2 = f_2 - l_1 z_1$, $z_i = f_i - l_1 z_{i-1} - l_2 z_{i-2}$, $i = 3$ to n
2. (solving $Uy = z$) $y_n = \frac{z_n}{u1}$, $y_{n-1} = \frac{z_{n-1} - u_2 y_n}{u_1}$, $y_i = \frac{z_i - u_2 y_{i+1} - c y_{i+2}}{u_1}$, $i = n - 2$ to 1

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

Algorithm 2 : An algorithm for solving $Tx = f$

Input: a, b, c, d, e, x, y, z, p, q, r, s, h, w, k, g, t and f;

1. Find the parameters l_i and u_i $(i = 1, 2)$;
2. Solve linear systems $LUZ = S$, $LUW = P$, and $LUx' = f$ by using Algorithm 1;

Output: Compute x by (7).

For the computational cost, when n is large this algorithm takes about $\approx 8n + O(1)$ flops.

An advantage of our algorithm is that it needs less data transmission since both the subdiagonal and superdiagonal of L and U have constant values, respectively. It only reads one vector (the right-hand side vector) and writes one vector (the solution).

The stability of Algorithm 2 depends on the step that solves the upper and lower pentadiagonal linear systems $LDU[Z, W, f] = [S, P, x']$. More precisely, two recursive iteration steps such as $z_i = f_{i-1} - l_1 z_{i-1} - l_2 z_{i-2}$ and $x_{n+1-i} = z_{n+1-1} - u_2 y_{n+2-i} - c y_{n+3-i}$ for $i = 2, \ldots, n$ corresponding to the forward and backward substitutions as in Algorithm 1 are essential for Algorithm 2. When using finite precision arithmetic, we should avoid roundoff error propagation. If all the roots of their characteristic equations $\lambda^2 + l_1\lambda + l_2 = 0$ and $\lambda^2 + u_2\lambda + c = 0$ are all less than unity in magnitude, errors of z_i and x_{n+1-i} will smaller than errors of previous values z_{i-1} and x_{n+2-i}, respectively, in which case the Algorithm 2 is stable.

3. Numerical examples

In this section, numerical results are presented to confirm the effectiveness of our algorithm. All algorithms are implemented in MATLAB R2018a and the computations are done on an Intel PENTIUM computer, (2.2 GHz), 6 GB memory. We fix the exact solution to be $x^* = [1, 1 \ldots, 1]^T$ and the right-hand side vector is set to be $f = Ax^*$.

3.1. Experiment 1: Kuramoto Sivashinsky equation

In this experiment, we will take the pentadiagonal matrices which appear in the numerical solution of Kuramoto Sivashinsky (KS) equation as an example. KS equation is a nonlinear partial differential equation first derived for the study of chemical reaction system, see [16, 17].

In paper [17], the initial vector is calculated by solving the following linear equations

$$\begin{bmatrix} 12a-3b+c & d-b & e-a & 0 & 0 & 0 & 0 \\ b-3a & c-a & d & e & 0 & 0 & 0 \\ a & b & c & d & e & 0 & 0 \\ 0 & a & b & c & d & e & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & a & b & c & d & e \\ 0 & 0 & 0 & a & b & c-e & d-3e \\ 0 & 0 & 0 & 0 & a-e & b-d & c-3d+12e \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \cdot \\ \vdots \\ \cdot \\ c_{N-1} \\ c_N \end{bmatrix} = \begin{bmatrix} u(x_0) \\ u(x_1) \\ u(x_2) \\ \vdots \\ \vdots \\ u(x_{N-2}) \\ u(x_{N-1}) \\ u(x_N) \end{bmatrix}.$$

By appling von-Neumann boundary conditions [17], the coefficient matrix becomes

$$\begin{bmatrix} 54 & 60 & 6 & 0 & 0 & 0 & 0 & 0 \\ 101/4 & 135/2 & 105/4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 26 & 66 & 26 & 1 & 0 & 0 & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 26 & 66 & 26 & 1 \\ 0 & 0 & 0 & 0 & 1 & 105/4 & 135/2 & 101/4 \\ 0 & 0 & 0 & 0 & 0 & 6 & 60 & 54 \end{bmatrix}.$$

By appling second order mixed boundary conditions [16], the coefficient matrix becomes

$$\begin{bmatrix} 121 & -2 & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 28 & 65 & 26 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & 26 & 66 & 26 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 & : & 0 \\ 0 & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & 0 \\ 0 & \cdots & 0 & 1 & 26 & 66 & 26 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & 26 & 65 & 28 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & -1 & 121 \end{bmatrix}.$$

We first compare these two examples, then we arbitrary choose some b, c, d, e, x, y, z, p, q, r, s, t, w, k, h, and g to test our algorithm.

Table 1: Numerical resuls of **Experiment 1 (with von-Neumann boundary conditions in [17])**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. *LU*	1.7593e-14	5.5524e-14	1.7555e-13	5.5511e-13
	Our algo.	4.4402e-14	1.4043e-13	4.4409e-13	1.4043e-12
CPU(s)	Algo. *LU*	4.24e-3	6.97e-2	0.45	4.63
	Our algo.	2.18e-3	2.49e-2	0.29	2.25

Table 2: Numerical resuls of **Experiment 1 (with mixed boundary conditions in [16])**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. *LU*	1.7579e-14	5.5519e-14	1.7554e-13	5.5511e-13
	Our algo.	4.4382e-14	1.4042e-13	4.4409e-13	1.4043e-12
CPU(s)	Algo. *LU*	5.57e-3	5.48e-2	0.53	5.87
	Our algo.	3.61e-3	3.94e-2	0.39	2.69

In Figures 1, and 2 a comparison of our algorithm with *LU* method are presented.

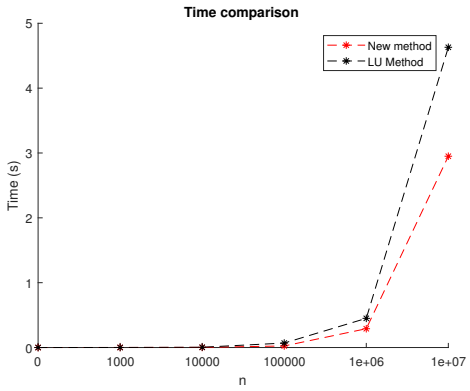

Figure 1: CPU time [s] comparison for Experiment 1 with von-Neumann boundary conditions in [17]

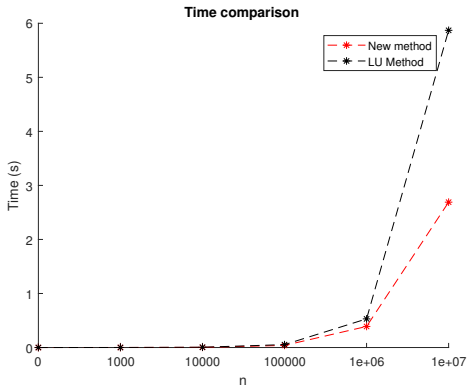

Figure 2: CPU time [s] comparison for Experiment 1 with mixed boundary conditions in [16]

It can be seen that our proposed algorithm takes less CPU time than the *LU* method.

3.2. Experiment 2

Some artificial examples were used in this experiment. The values of a, b, c, d, e, x, y, z, p, q, r, s, t, w, k, h, and g corresponding to each examples are presented in Table 3.

Table 3: a, b, c, d, e, x, y, z, p, q, r, s, t, w, k, h,and g corresponding to each examples

	a	b	c	d	e	x	y	z	p	q	r	s	t	w	k	h	g
Example 1	-62	-10	-19	-2	-5	-2.3	4	3.5	10	2	-4	3	-1	-1.7	4.2	-3.5	10
Example 2	66	10	15	1	1	8	2	-1.5	-0.7	-1	-2.3	7	2.5	1.6	-4	-3.2	4
Example 3	2.5	-0.8	0.8	2.2	1	1.3	0.4	-0.2	3	1	-4	-3	2	-1.2	1	-1	1.3
Example 4	246	30	-56	-2	1.6	0.5	-2	2.4	2.6	-7.2	2	1	-1	2.6	5	1	1
Example 5	-5	0	2	1	2.4	1	2	1	-5	5	-26	-2	0.6	-25	-6.5	0.6	2
Example 6	6.5	0	1.3	1	4.5	1.5	-3.2	-1.3	-3.2	5	-19	-7	-1	-2	-1.5	0.7	1

From tables 4 to table 9 we show the absolute accuracy $\Delta x = \|x^* - x\|_2$ of the approximate solution of (1) where $x^* = [1, 1 \ldots, 1]^T$ is the exact solution and x is the approximate solution computed by LU method, and our algorithm, respectively. We display also, in the same tables the CPU time [s] of the computed solution of our algorithm.

Table 4: Numerical resuls of **Example 1**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. LU	2.2181e-14	7.0208e-14	2.2204e-13	7.0217e-13
	Our algo.	6.0168e-15	6.0168e-15	6.0168e-15	6.0168e-15
CPU(s)	Algo. LU	3.49e-3	8.1e-2	1.11	10.02
	Our algo.	3.36e-3	3.2e-2	0.34	3.55

Table 5: Numerical resuls of **Example 2**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. LU	1.3038e-14	2.6820e-14	7.8182e-14	2.4847e-13
	Our algo.	3.9339e-14	8.3841e-14	3.8316e-13	7.8512e-13
CPU(s)	Algo. LU	4.66e-3	6.27e-2	0.87	9.80
	Our algo.	4.49e-3	2.77e-2	0.35	3.38

Table 6: Numerical resuls of **Example 3**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. LU	3.5346e-14	1.1110e-13	3.5111e-13	1.1102e-12
	Our algo.	2.3572e-14	7.0342e-14	2.2208e-13	7.0218e-13
CPU(s)	Algo. LU	7.45e-3	4.46e-2	0.98	9.84
	Our algo.	4.31e-3	2.84e-2	0.34	3.46

Table 7: Numerical resuls of **Example 4**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. LU	1.7668e-13	1.7668e-13	1.7668e-13	1.7668e-13
	Our algo.	8.5199e-14	9.1478e-14	1.3950e-13	3.6110e-13
CPU(s)	Algo. LU	8.0e-3	4.99e-2	0.99	9.95
	Our algo.	4.69e-3	3.03e-2	0.34	3.48

Table 8: Numerical resuls of **Example 5**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. LU	1.7624e-14	5.5537e-14	1.7555e-13	5.5511e-13
	Our algo.	2.0742e-14	3.9238e-14	1.1240e-13	3.5152e-13
CPU(s)	Algo. LU	4.71e-3	6.31e-2	1.11	9.78
	Our algo.	3.62e-3	2.79e-2	0.34	3.45

Table 9: Numerical resuls of **Example 6**

	Algorithm	$n = 10^4$	$n = 10^5$	$n = 10^6$	$n = 10^7$
$\|x^* - x\|_2$	Algo. *LU*	8.1259e-15	2.4914e-14	7.8533e-14	2.4826e-13
	Our algo.	1.3822e-15	1.3822e-15	1.3822e-15	1.3822e-15
CPU(s)	Algo. *LU*	5.36e-3	6.06e-2	1.11	10.03
	Our algo.	4.30e-3	3.48e-2	0.36	3.52

Thus, comparing with the initial *LU* method for a sparse matrix , our algorithm improves the computational cost of the numerical solution remarkably. For the accuracy, our algorithm is similar to the other well-known existing methods.

4. Conclusion

In this paper, we have proposed a new algorithm for solving diagonally dominant symmetric quasi-pentadiagonal Toeplitz linear systems. We discussed possible choices for the parametres for each situation. We implemented our method in Matlab with respect to computational costs. The numerical results show the robustness of our method. The required memory and the computational time of our algorithm are lower than those of other well-known existing methods. The effectiveness of our algorithm is confirmed by numerical experiments.

5. Acknowledgement

The authors would like to thank the supports of the Portuguese Funds through FCT–Fundacão para a Ciência e a Tecnologia, within the Project UID/MAT/00013/2013.

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

[2] 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.

[3] J M_ Sanz-Serna and I Christie. A simple adaptive technique for nonlinear wave problems. *Journal of Computational Physics*, 67(2):348–360, 1986.

[4] SS Nemani and Lawrence E Garey. An efficient method for second order boundary value problems with two point boundary conditions. *International journal of computer mathematics*, 79(9):1001–1008, 2002.

[5] Richard M Beam and Robert F Warming. The asymptotic spectra of banded toeplitz and quasi-toeplitz matrices. *SIAM Journal on Scientific Computing*, 14(4):971–1006, 1993.

[6] Miloslav Znojil. Perturbation method with triangular propagators and anharmonicities of intermediate strength. *Journal of Mathematical Chemistry*, 28(1-3):139–167, 2000.

[7] Stephen J Wright. Stable parallel algorithms for two-point boundary value problems. *SIAM Journal on Scientific and Statistical Computing*, 13(3):742–764, 1992.

[8] WR Briley and H McDonald. Solution of the multidimensional compressible navier-stokes equations by a generalized implicit method. *Journal of Computational Physics*, 24(4):372–397, 1977.

[9] ME Kanal. Parallel algorithm on inversion for adjacent pentadiagonal matrices with mpi. *The Journal of Supercomputing*, 59(2):1071–1078, 2012.

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

[11] Jeffrey Mark McNally. A fast algorithm for solving diagonally dominant symmetric pentadiagonal toeplitz systems. *Journal of computational and applied mathematics*, 234(4):995–1005, 2010.

[12] SS Nemani. A fast algorithm for solving toeplitz penta-diagonal systems. *Applied mathematics and computation*, 215(11):3830–3838, 2010.

[13] Tomohiro Sogabe. A fast numerical algorithm for the determinant of a pentadiagonal matrix. *Applied mathematics and computation*, 196(2):835–841, 2008.

[14] Zubeyir Cinkir. An elementary algorithm for computing the determinant of pentadiagonal toeplitz matrices. *Journal of Computational and Applied Mathematics*, 236(9):2298–2305, 2012.

[15] Jiteng Jia, Qiongxiang Kong, and Tomohiro Sogabe. A new algorithm for solving nearly penta-diagonal toeplitz linear systems. *Computers & Mathematics with Applications*, 63(7):1238–1243, 2012.

[16] Neeraj Dhiman and Mohammad Tamsir. Re-modified quintic b-spline collocation method for the solution of kuramoto–sivashinsky type equations. *Multidiscipline Modeling in Materials and Structures*, pages 1573–6105, 2018.

[17] RC Mittal and Geeta Arora. Quintic b-spline collocation method for numerical solution of the kuramoto–sivashinsky equation. *Communications in Nonlinear Science and Numerical Simulation*, 15(10):2798–2808, 2010.