American Journal of Computational Mathematics, 2015, 5, 258-266
Published Online September 2015 in SciRes. http://www.scirp.org/journal/ajcm
http://dx.doi.org/10.4236/ajcm.2015.53023

A New Symbolic Algorithm for Solving General Opposite-Bordered Tridiagonal Linear Systems

Faiz Atlan*, Moawwad El-Mikkawy

Mathematics Department, Faculty of Science, Mansoura University, Mansoura, Egypt
Email: *faizatlan11@yahoo.com, m_elmikkawy@yahoo.com

Received 16 June 2015; accepted 28 August 2015; published 31 August 2015

Abstract

In the current article we propose a new efficient, reliable and breakdown-free algorithm for solving general opposite-bordered tridiagonal linear systems. An explicit formula for computing the determinant of an opposite-bordered tridiagonal matrix is investigated. Some illustrative examples are given.

Keywords

Opposite-Bordered Tridiagonal Matrix, Algorithm, Linear System of Equations, Schur Complement, MATLAB

1. Introduction

The $n \times n$ general tridiagonal matrix T_n takes the form:

$$T_n = \begin{bmatrix} d_1 & a_1 & 0 & \cdots & 0 \\ b_1 & d_2 & a_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & b_{n-2} & d_{n-1} & a_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & d_n \end{bmatrix}. \quad (1)$$

The matrix in (1) frequently appears in many applications, for example, in parallel computing, telecommunication system analysis, solving differential equations using finite differences, heat conduction and fluid flow

*Corresponding author.

How to cite this paper: Atlan, F. and El-Mikkawy, M. (2015) A New Symbolic Algorithm for Solving General Opposite-Bordered Tridiagonal Linear Systems. *American Journal of Computational Mathematics*, **5**, 258-266.
http://dx.doi.org/10.4236/ajcm.2015.53023

problems. The interested reader may refer to [1]-[12] and the references therein.

Inverting tridiagonal matrices in (1) have been considered by many authors. See for instance, [13]-[22]. To study matrices of the form (1) it is advantageous to introduce an n-dimensional vector $\boldsymbol{e}$ in the following way [23]:

$$\boldsymbol{e} = [e_1, e_2, \cdots, e_n], \tag{2}$$

whose components $e_1, e_2, \cdots, e_n$ are given by:

$$e_i = \begin{cases} d_i, & \text{if } i = 1, \\ d_i - \dfrac{a_{i-1}b_{i-1}}{e_{i-1}}, & \text{if } i = 2, 3, \cdots, n. \end{cases} \tag{3}$$

The symbolic algorithm **DETGTRI** [23] is based on (2) and (3). By using the LU factorization of T_n, it is known that [23]

$$\det(T_n) = \prod_{r=1}^{n} e_r. \tag{4}$$

There are great interests in solving general opposite-bordered tridiagonal linear system, and hereafter it will be referred to as OBTLS, of the form:

$$A\boldsymbol{x} = \boldsymbol{f}, \tag{5}$$

in which the coefficient matrix A is given by:

$$A = \begin{bmatrix} d_0 & a_0 & 0 & \cdots & \cdots & 0 & p_0 \\ b_0 & d_1 & a_1 & \ddots & & \vdots & p_1 \\ q_1 & b_1 & d_2 & a_2 & \ddots & \vdots & p_2 \\ q_2 & 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & b_{n-4} & d_{n-3} & a_{n-3} & p_{n-3} \\ q_{n-3} & \vdots & & \ddots & b_{n-3} & d_{n-2} & a_{n-2} \\ q_{n-2} & 0 & \cdots & \cdots & 0 & b_{n-2} & d_{n-1} \end{bmatrix}, \tag{6}$$

$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^t$ and $\boldsymbol{f} = [f_0, f_1, \cdots, f_{n-1}]^t$.

This system frequently occurs in engineering computation and science, e.g. in the numerical solution of an ablation and heat transfer problem as referred in [24]-[28]. The matrix A in (6) can be stored in $5n-6$ memory locations only.

In [28], the author presented a numeric algorithm for solving the linear system (5) with $p_0 = q_{n-2} = 0$. The algorithm is based on the elementary column operations (ECO's). It is noted that the numerical algorithm in [28] fails to solve some OBTLS of the form (5). Therefore, the main objective of the present paper is to construct a new symbolic and breakdown-free algorithm for solving the OBTLS in (5).

Throughout this paper, the word "simplify" means simplifying the algebraic expression under consideration to its simplest rational form. Also, λ is a formal parameter which can be treated as a symbolic name whose actual value is 0 as we will see later.

The organization of the paper is as follows. The main results are given in the next section. Some illustrative examples are given in Section 3. In Section 4, we present some concluding remarks.

2. Main Results

In this section, we are going to formulate a new algorithm for solving OBTLS of the form (5). We begin by considering the singly bordered tridiagonal linear systems of the form (7)-(8) below.

2.1. A Symbolic Algorithm for Solving Singly Bordered Tridiagonal Linear Systems

The $k \times k$ singly bordered tridiagonal linear system takes the form:

$$S_k \boldsymbol{y} = \tilde{\boldsymbol{f}}, \tag{7}$$

where

$$S_k = \begin{bmatrix} d_1 & a_1 & 0 & \cdots & \cdots & 0 & p_1 \\ b_1 & d_2 & a_2 & \ddots & & \vdots & p_2 \\ 0 & b_2 & d_3 & a_3 & \ddots & \vdots & p_3 \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & b_{k-3} & d_{k-2} & a_{k-2} & p_{k-2} \\ \vdots & & & \ddots & b_{k-2} & d_{k-1} & a_{k-1} \\ 0 & \cdots & \cdots & \cdots & 0 & b_{k-1} & d_k \end{bmatrix}, \tag{8}$$

$\boldsymbol{y} = [y_1, y_2, \cdots, y_k]^t$ and $\tilde{\boldsymbol{f}} = [f_1, f_2, \cdots, f_k]^t$.

The Doolittle LU factorization of S_k is given by [1]:

$$S_k = LU \tag{9}$$

where

$$L = \begin{bmatrix} 1 & 0 & \cdots & & \cdots & 0 \\ \gamma_1 & 1 & \ddots & & & \vdots \\ \vdots & \gamma_2 & 1 & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \gamma_{k-2} & 1 & 0 \\ 0 & \cdots & \cdots & 0 & \gamma_{k-1} & 1 \end{bmatrix}, \tag{10}$$

and

$$U = \begin{bmatrix} c_1 & a_1 & 0 & \cdots & 0 & \beta_1 \\ 0 & c_2 & a_2 & \ddots & \vdots & \beta_2 \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & \ddots & c_{k-2} & a_{k-2} & \beta_{k-2} \\ \vdots & & & \ddots & c_{k-1} & \beta_{k-1} \\ 0 & \cdots & \cdots & \cdots & 0 & c_k \end{bmatrix}, \tag{11}$$

where the quantities $c_i, i = 1, 2, \cdots, k$, γ_i and $\beta_i, i = 1, 2, \cdots, k-1$ are given, respectively, by:

$$c_i = \begin{cases} d_1, & \text{for } i = 1, \\ d_i - \gamma_{i-1} a_{i-1}, & \text{for } i = 2, 3, \cdots, k-1, \\ d_k - \gamma_{k-1} \beta_{k-1}, & \text{for } i = k, \end{cases} \tag{12}$$

$$\gamma_i = \frac{b_i}{c_i}, \quad \text{for } i = 1, 2, \cdots, k-1 \tag{13}$$

and

$$\beta_i = \begin{cases} p_1, & \text{for } i = 1, \\ p_i - \gamma_{i-1} \beta_{i-1}, & \text{for } i = 2, 3, \cdots, k-2, \\ a_{k-1} - \gamma_{k-2} \beta_{k-2}, & \text{for } i = k-1. \end{cases} \tag{14}$$

It follows from (9)-(11) that

$$\det(S_k) = \prod_{i=1}^{k} c_i. \tag{15}$$

At this point, it should be mentioned that the above LU factorization is always possible even if the matrix S_k is singular.

The solution of the system in (7) reduces to solving the two standard linear systems:

$$L\boldsymbol{z} = \tilde{\boldsymbol{f}}, \tag{16}$$

and

$$U\boldsymbol{y} = \boldsymbol{z}. \tag{17}$$

We are now ready to formulate the following algorithm for solving the linear system (7).

Algorithm 1: A Symbolic algorithm for solving singly bordered tridiagonal linear system.

To solve the linear system of the form (7), we may proceed as follows:
Input: Order of the matrix n and the components, a_i, d_i, b_i, p_i, f_i.
Output: The solution vector $\mathbf{y} = [y_1, y_2, \ldots, y_k]^t$.
Step 1: Set $z_1 = f_1$, $c_1 = d_1$, $\gamma_1 = \frac{b_1}{c_1}$
If $c_1 = 0$ **then** $c_1 = \lambda$. **end**
Step 2:
for $i = 2, 3, ..., k-1$ **do**
  Compute and simplify:
  $c_i = d_i - a_{i-1}\,\gamma_{i-1}$.
  If $c_i = 0$ **then** $c_i = \lambda$ **end**
  $z_i = f_i - \gamma_{i-1}\,z_{i-1}$.
  $\gamma_i = \frac{b_i}{c_i}$
end
Step 3:
for $i = 2, 3, ..., k-2$ **do**
  Compute and simplify:
  $\beta_i = p_i - \beta_{i-1}\,\gamma_{i-1}$.
end
$\beta_{k-1} = a_{k-1} - \beta_{k-2}\,\gamma_{k-2}$.
$z_k = f_k - z_{k-1}\,\gamma_{k-1}$.
$c_k = d_k - \beta_{k-1}\,\gamma_{k-1}$.
If $c_k = 0$ **then** $c_k = \lambda$ **end**
Step 4: $y_k = \frac{z_k}{c_k}$
$y_{k-1} = \frac{z_{k-1} - \beta_{k-1} y_k}{c_{k-1}}$
for $i = k-2, k-3, ..., 1$ **do**
  Compute and simplify:
  $y_i = (z_i - a_i y_{i+1} - \beta_i\, y_k)/c_i$.
end
Step 5: Substitute $\lambda = 0$ in all expressions of the solution vector $y_i, i = 1, 2, ..., k$.

2.2. A Symbolic Algorithm for Solving General OBTLS

In order to solve the general OBTLS (5) it is convenient to introduce the following notations:

$$\boldsymbol{g}^t = [a_0, 0, \cdots, p_0] \in \mathbb{R}^{1\times(n-1)}, \tag{18}$$

$$\boldsymbol{q} = [b_0, q_1, q_2, \cdots, q_{n-3}, q_{n-2}]^t \in \mathbb{R}^{(n-1)\times 1}, \tag{19}$$

$$S_{n-1} = \begin{bmatrix} d_1 & a_1 & 0 & \cdots & 0 & p_1 \\ b_1 & d_2 & a_2 & \ddots & \vdots & p_2 \\ 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & b_{n-4} & d_{n-3} & a_{n-3} & p_{n-3} \\ \vdots & & \ddots & b_{n-3} & d_{n-2} & a_{n-2} \\ 0 & \cdots & \cdots & 0 & b_{n-2} & d_{n-1} \end{bmatrix}, \tag{20}$$

$$\hat{\boldsymbol{x}} = x_1, \quad \tilde{\boldsymbol{x}} = [x_2, x_3, \cdots, x_n]^t,$$

and

$$\hat{\boldsymbol{f}} = f_0, \quad \tilde{\boldsymbol{f}} = [f_1, f_2, \cdots, f_{n-1}]^t.$$

Based on the above notations, the linear system in (5) can be written in the partitioned form:

$$A\begin{bmatrix}\hat{\boldsymbol{x}}\\ \tilde{\boldsymbol{x}}\end{bmatrix} = \left[\begin{array}{c|c} d_0 & \boldsymbol{g}^t \\ \hline \boldsymbol{q} & S_{n-1}\end{array}\right]\begin{bmatrix}\hat{\boldsymbol{x}}\\ \tilde{\boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}\hat{\boldsymbol{f}}\\ \tilde{\boldsymbol{f}}\end{bmatrix}. \tag{21}$$

The solution of the linear system (21), may be obtained by solving the two linear systems:

$$d_0\hat{\boldsymbol{x}} + \boldsymbol{g}^t\tilde{\boldsymbol{x}} = \hat{\boldsymbol{f}}, \tag{22}$$

$$\boldsymbol{q}\hat{\boldsymbol{x}} + S_{n-1}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{f}}, \tag{23}$$

It is not difficult to prove that:

$$\tilde{\boldsymbol{x}} = S_{n-1}^{-1}\tilde{\boldsymbol{f}} - S_{n-1}^{-1}\boldsymbol{q}\hat{\boldsymbol{x}}. \tag{24}$$

As can be seen from (24), we need to compute $\boldsymbol{v} = S_{n-1}^{-1}\tilde{\boldsymbol{f}}$ and $\boldsymbol{w} = S_{n-1}^{-1}\boldsymbol{q}$. By solving the following singly bordered systems with two right-hand sides we obtain the solution vectors $\boldsymbol{v}$ and $\boldsymbol{w}$:

$$S_{n-1}\left[\boldsymbol{v}\middle|\boldsymbol{w}\right] = \left[\tilde{\boldsymbol{f}}\middle|\boldsymbol{q}\right]. \tag{25}$$

Consequently, we have from (24)

$$x_2 = v_1 - x_1 w_1, \tag{26}$$

and

$$x_n = v_{n-1} - x_1 w_{n-1}. \tag{27}$$

By substituting (26) and (27) into (22), it follows that

$$x_1 = \frac{f_0 - a_0 v_1 - p_0 v_{n-1}}{d_0 - a_0 w_1 - p_0 w_{n-1}}. \tag{28}$$

Therefore, we get

$$\tilde{\boldsymbol{x}} = \boldsymbol{v} - x_1 \cdot \boldsymbol{w}. \tag{29}$$

Hence, the solution vector of the OBTLS (5) is $\boldsymbol{x} = [\hat{\boldsymbol{x}}, \tilde{\boldsymbol{x}}]^t$.

The proofs of the following result may be found in [29].

Theorem 1. (*Schur determinant identity*) *Let* M_1, M_2, M_3 *and* M_4 *are* $m\times m$, $m\times(n-m)$, $(n-m)\times m$ *and* $(n-m)\times(n-m)$ *matrices, respectively. Let* M *be a* 2×2 *block matrix given by*

$$M = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}. \tag{30}$$

Then

$$\det(M) = \begin{cases} \det(M_1)\det\left(M_4 - M_3 M_1^{-1} M_2\right) & \text{if } M_1 \text{ is nonsingular,} \\ \det(M_4)\det\left(M_1 - M_2 M_4^{-1} M_3\right) & \text{if } M_4 \text{ is nonsingular.} \end{cases}$$

By noticing (21), we see that the matrix A in (6) can be written in the partitioned form:

$$A = \left[\begin{array}{c|c} d_0 & \boldsymbol{g}^t \\ \hline \boldsymbol{q} & S_{n-1}\end{array}\right]. \tag{31}$$

Hence, by applying Theorem 1 on this matrix, we get the following result:

Corollary 1. *Let* A *be the* $n\times n$ *matrix given in* (31), *then the determinant of* A *is given by*:

$$\det(A) = \left(\left(\prod_{i=1}^{n-1} c_i \right) \left(d_0 - a_0 w_1 - p_0 w_{n-1} \right) \right) \Big|_{\lambda=0}, \tag{32}$$

provided S_{n-1} *is a nonsingular matrix.*

The main result of the present paper may now be formulated as follows:

Algorithm 2: A Symbolic algorithm for solving general nonsingular opposite-bordered tridiagonal linear system.

To solve the linear system of the form (7), we may proceed as follows:
Input: Order of the matrix n and a_i, d_i, b_i, p_i, q_i, f_i.
Output: The solution vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^t$.
Step 1: Solve the linear systems in $S_{n-1}[\mathbf{v}|\mathbf{w}] = [\tilde{\mathbf{f}}|\mathbf{q}]$ by using Algorithm 1.
Step 2: Compute x_1 by using (29).
Step 3: Compute $\tilde{\mathbf{x}} = \mathbf{v} - x_1.\mathbf{w}$.
Step 4: The solution vector is $\mathbf{x} = [\hat{\mathbf{x}}, \tilde{\mathbf{x}}]^t|_{\lambda=0}$.

This algorithm will be referred to as the **OBS** algorithm. The computational cost for **OBS** is $21n-35$ in terms of total number of flops, where each flop represents one of the four basic arithmetic floating point operations.

A MATLAB code based on the **OBS** algorithm is available upon request from the authors.

3. Illustrative Examples

In this section we are going to consider some illustrative examples. The, symbolic computations are performed in Example 1 by using MATLAB with Symbolic Math Toolbox. Also, we compare the proposed algorithm with MATLAB back-slash and the algorithm in [28] by means of execution times and accuracy of the solutions in Example 2. Finally, we give Example 3 in order to demonstrate the validity of the **OBS** algorithm. All experiments were carried out using MATLAB 7.10.0.499 (R2010a) on a PC with Intel(R) Core(TM) i7-3770 CPU processor.

Example 1. *Solve the opposite-bordered tridiagonal linear system*:

$$\begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 5 & 0 & 0 & 0 & 0 & 7 \\ -2 & 2 & 5 & 3 & 0 & 0 & 0 & -1 \\ 1 & 0 & 2 & 1 & -1 & 0 & 0 & 2 \\ 5 & 0 & 0 & 1 & 6 & 2 & 0 & -3 \\ 3 & 0 & 0 & 0 & 1 & 1 & 3 & 4 \\ 2 & 0 & 0 & 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 3 \\ 15 \\ 7 \\ 5 \\ 11 \\ 12 \\ 8 \\ 6 \end{bmatrix}. \tag{33}$$

Solution.

Here, $n=8$, $d_0=1$, $\hat{\boldsymbol{f}}=3$, $\tilde{\boldsymbol{f}}=[15,7,5,11,12,8,6]^t$, $\boldsymbol{g}=[2,0,0,0,0,0,0]^t$, $\boldsymbol{q}=[1,-2,1,5,3,2,0]^t$, and

$$S_{n-1} = \begin{bmatrix} 2 & 5 & 0 & 0 & 0 & 0 & 7 \\ 2 & 5 & 3 & 0 & 0 & 0 & -1 \\ 0 & 2 & 1 & -1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 6 & 2 & 0 & -3 \\ 0 & 0 & 0 & 1 & 1 & 3 & 4 \\ 0 & 0 & 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 4 \end{bmatrix}$$

The numeric algorithm in [28] fails to solve the linear system (33) although $\det A = 148 \neq 0$. Applying the

OBS algorithm, we obtain:

$$\boldsymbol{v}=\left[\frac{5}{2}\frac{5\lambda-23}{11\lambda-26},\frac{15}{11\lambda-26},\frac{19\lambda-64}{11\lambda-26},\frac{4(-1+\lambda)}{11\lambda-26},\frac{3(23\lambda-58)}{11\lambda-26},-\frac{-22+7\lambda}{11\lambda-26},\frac{10(2\lambda-5)}{11\lambda-26}\right],$$

$$\boldsymbol{w}=\left[\frac{-1}{2}\frac{63+38\lambda}{11\lambda-26},\frac{41}{11\lambda-26},\frac{-2(19+3\lambda)}{11\lambda-26},-\frac{-22+3\lambda}{11\lambda-26},\frac{2(25\lambda-74)}{11\lambda-26},\frac{-2(-24+7\lambda)}{11\lambda-26},\frac{-24+7\lambda}{11\lambda-26}\right] \quad \textbf{(Step 1)}.$$

$$\hat{\boldsymbol{x}}=x_1=\frac{37+8\lambda}{49\lambda+37} \quad \textbf{(Step 2)} \text{ and}$$

$$\tilde{\boldsymbol{x}}=\left[\frac{139\lambda+74}{2(49\lambda+37)},\frac{37}{49\lambda+37},\frac{37+89\lambda}{49\lambda+37},\frac{37+20\lambda}{49\lambda+37},\frac{37+271\lambda}{49\lambda+37},\frac{37-21\lambda}{49\lambda+37},\frac{37+84\lambda}{49\lambda+37}\right] \quad \textbf{(Step 3)}.$$

The solution vector $\boldsymbol{x}=\left[\hat{\boldsymbol{x}},\tilde{\boldsymbol{x}}\right]_{\lambda=0}^{t}=\left[1,1,1,1,1,1,1\right]^{t}$ **(Step 4)**.

Example 2. *Consider the opposite-bordered tridiagonal linear system*:

$$\begin{bmatrix} 4 & 1.2 & 0 & \cdots & \cdots & 0 & 0 \\ 2.3 & 4 & 1.2 & \ddots & & \vdots & 1.5 \\ 2.5 & 2.3 & 4 & 1.2 & \ddots & \vdots & 1.5 \\ 2.5 & 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & 2.3 & 4 & 1.2 & 1.5 \\ 2.5 & \vdots & & \ddots & 2.3 & 4 & 1.2 \\ 0 & 0 & \cdots & \cdots & 0 & 2.3 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix}=\begin{bmatrix} 5.2 \\ 9 \\ 11.5 \\ \vdots \\ 11.5 \\ 10 \\ 6.3 \end{bmatrix}. \tag{34}$$

The exact solution of this system is $\boldsymbol{x}^{*}=\left[\underbrace{1,1,\cdots,1}_{n}\right]^{t}$. **Table 1** shows the CPU times (after 100 tests) obtained from the **OBS** algorithm, the algorithm in [28] and MATLAB back-slash operator for n = 1000, 2000, $\cdots$, 10,000. The absolute errors $\left\|\boldsymbol{x}-\boldsymbol{x}^{*}\right\|$ are shown in **Figure 1**. Here, $\|.\|$ is the Euclidean vector norm.

Example 3. *Consider the opposite-bordered tridiagonal linear system*:

$$\begin{bmatrix} 4 & 2 & 0 & \cdots & \cdots & 0 & 1 \\ 1 & 4 & 2 & \ddots & & \vdots & 1 \\ 2 & 1 & 4 & 2 & \ddots & \vdots & \vdots \\ 2 & 0 & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & 1 & 4 & 2 & 1 \\ 2 & \vdots & & \ddots & 1 & 4 & 2 \\ 2 & 0 & \cdots & \cdots & 0 & 1 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix}=\begin{bmatrix} 7 \\ 8 \\ 10 \\ \vdots \\ 10 \\ 9 \\ 7 \end{bmatrix}. \tag{35}$$

Table 2 gives the absolute errors and CPU times (after 100 tests) obtained from the **OBS** algorithm for n = 1000, 5000, 10,000, 20,000, 30,000, 40,000, 50,000.

Table 1. Mean value of the CPU times after 100 tests.

n	1000	2000	3000	4000	5000	6000	7000	8000	9000	10,000
OBS Algorithm	2.360e−4	2.503e−4	3.290e−4	4.531e−4	5.536e−4	6.607e−4	7.613e−4	8.7950e−4	9.870e−4	0.0011
Algorithm in [28]	2.372e−4	2.56e−4	3.420e−4	4.578e−4	5.849e−4	6.974e−4	8.129e−4	9.267e−4	0.0011	0.0012
MATLAB	0.0013	0.0027	0.0043	0.0056	0.0069	0.0084	0.0100	0.0114	0.0125	0.0145

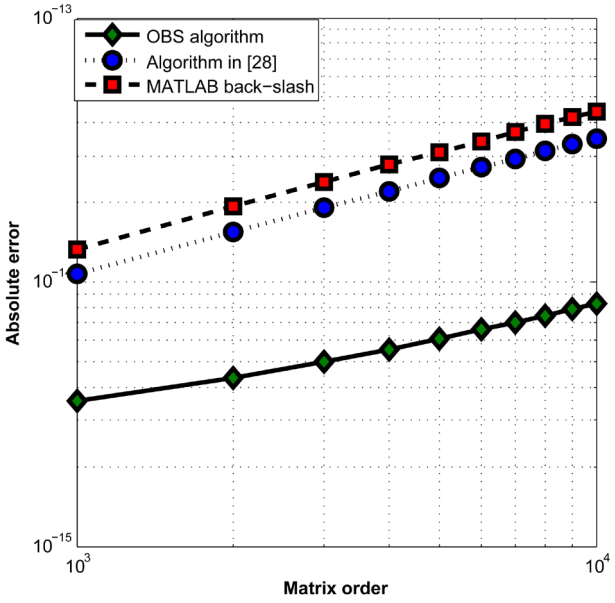

Figure 1. Absolute errors for Example 2.

Table 2. Absolute errors and CPU times of Example 3 for the **OBS** algorithm.

n	1000	5000	10,000	20,000	30,000	400,000	50,000
$\|x - x^*\|$	3.6333e−15	7.9060e−15	1.1142e−14	1.5729e−14	1.9252e−14	2.2224e−14	2.4843e−14
CPU time (s)	1.635e−4	7.849e−4	0.0011	0.0022	0.0035	0.0045	0.0057

4. Conclusion

In this paper, we proposed a new efficient and reliable algorithm for solving general opposite-bordered tridiagonal linear systems in linear time. An explicit formula for computing the determinant of an opposite-bordered tridiagonal matrix is obtained. Some illustrative examples are given.

Acknowledgements

The authors wish to thank anonymous referees and the editorial board of the AJCM for careful reading of the manuscript and their useful comments.

References

[1] Burden, R.L. and Faires, J.D. (2001) Numerical Analysis. 7th Edition, Books & Cole Publishing, Pacific Grove.

[2] da Fonseca, C.M. and Petronilho, J. (2001) Explicit Inverses of Some Tridiagonal Matrices. *Linear Algebra and its Applications*, **325**, 7-21. http://dx.doi.org/10.1016/S0024-3795(00)00289-5

[3] El-Mikkawy, M.E.A. (2005) A New Computational Algorithm for Solving Periodic Tri-Diagonal Linear Systems. *Applied Mathematics and Computation*, **161**, 691-696. http://dx.doi.org/10.1016/S0024-3795(00)00289-5

[4] El-Mikkawy, M.E.A. and Karawia, A. (2006) Inversion of General Tridiagonal Matrices. *Applied Mathematics Letters*, **19**, 712-720. http://dx.doi.org/10.1016/j.aml.2005.11.012

[5] El-Mikkawy, M. and Atlan, F. (2014) Algorithms for Solving Linear Systems of Equations of Tridiagonal Type via Transformations. *Applied Mathematics*, **5**, 413-422. http://dx.doi.org/10.4236/am.2014.53042

[6] El-Mikkawy, M. and Atlan, F. (2014) Algorithms for Solving Doubly Bordered Tridiagonal Linear Systems. British *Journal of Mathematics & Computer Science*, **4**, 1246-1267. http://dx.doi.org/10.9734/BJMCS/2014/8835

[7] El-Mikkawy, M., El-Shehawy, M. and Shehab, N. (2015) Solving Doubly Bordered Tridiagonal Linear Systems via Partition. *Applied Mathematics*, **6**, 967-978. http://dx.doi.org/10.4236/am.2015.66089

[8] Golub, G.H. and van Loan, C.F. (1996) Matrix Computations. 3rd Edition, Johns Hopkins University Press, Baltimore.

[9] Kavcic, A. and Moura, J.M.F. (2000) Matrices with Banded Inverses: Inversion Algorithms and Factorization of Gauss-Markov Processes. *IEEE Transactions on Information Theory*, **46**, 1495-1509. http://dx.doi.org/10.1109/18.954748

[10] Li, H.-B., Huang, T.-Z., Liu, X.-P. and Li, H. (2010) On the Inverses of General Tridiagonal Matrices. *Linear Algebra and Its Applications*, **433**, 965-983. http://dx.doi.org/10.1016/j.laa.2010.04.042

[11] Mazilu, I., Mazilu, D.A. and Williams, H.T. (2012) Applications of Tridiagonal Matrices in Non-Equilibrium Statistical Physics. *Electronic Journal of Linear Algebra*, **24**, 7-17.

[12] Olcayto, E. (1979) Recursive Formulae for Ladder Network Optimization. *Electronics Letters*, **15**, 249-250. http://dx.doi.org/10.1049/el:19790176

[13] El-Mikkawy, M.E.A. (2004) On the Inverse of a General Tridiagonal Matrix. *Applied Mathematics and Computation*, **150**, 669-679. http://dx.doi.org/10.1016/S0096-3003(03)00298-4

[14] El-Mikkawy, M.E.A. and El-Desouky, R. (2008) A New Recursive Algorithm for Inverting General Tridiagonal and Anti-Tridiagonal Matrices. *Applied Mathematics and Computation*, **204**, 368-372. http://dx.doi.org/10.1016/j.amc.2008.06.053

[15] Huang, Y. and McColl, W.F. (1997) Analytic Inversion of General Tridiagonal Matrices. *Journal of Physics A*, **30**, 7919-7933. http://dx.doi.org/10.1088/0305-4470/30/22/026

[16] Hu, G.Y. and O'Connell, R.F. (1996) Analytical Inversion of Symmetric Tridiagonal Matrices. *Journal of Physics A*, **29**, 1511-1513. http://dx.doi.org/10.1088/0305-4470/29/7/020

[17] Mallik, R.K. (2001) The Inverse of a Tridiagonal Matrix. *Linear Algebra and Its Applications*, **325**, 109-139. http://dx.doi.org/10.1016/S0024-3795(00)00262-7

[18] Marrero, J.A., Rachidi, M. and Tomeo, V. (2013) Non-Symbolic Algorithms for the Inversion of Tridiagonal Matrices. *Journal of Computational and Applied Mathematics*, **252**, 3-11. http://dx.doi.org/10.1016/j.cam.2012.05.003

[19] Ran, R.-S., Huang, T.-Z., Liu, X.-P. and Gu, T.-X. (2009) An Inversion Algorithm for General Tridiagonal Matrix. *Applied Mathematics and Mechanics*, **30**, 247-253. http://dx.doi.org/10.1007/s10483-009-0212-x

[20] Sugimoto, T. (2012) On an Inverse Formula of a Tridiagonal Matrix. *Operators and Matrices*, **6**, 465-480. http://dx.doi.org/10.7153/oam-06-30

[21] Usmani, R. (1994) Inversion of a Tridiagonal Jacobi Matrix. *Linear Algebra and Its Applications*, **212**/**213**, 413-414. http://dx.doi.org/10.1016/0024-3795(94)90414-6

[22] Yamamoto, T. and Ikebe, Y. (1979) Inversion of Band Matrices. *Linear Algebra and Its Applications*, **24**, 105-111. http://dx.doi.org/10.1016/0024-3795(79)90151-4

[23] El-Mikkawy, M.E.A. (2004) A Fast Algorithm for Evaluating *n*th Order Tri-Diagonal Determinants. *Journal of Computational and Applied Mathematics*, **166**, 581-584. http://dx.doi.org/10.1016/j.cam.2003.08.044

[24] Amar, A.J., Blackwell, B.F. and Edward, J.R. (2006) One-Dimensional Ablation Using a Full Newton's Method and Finite Control Volume Procedure. 9*th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, AIAA-2006-2910, San Francisco, 5-8 June 2006, 26. http://dx.doi.org/10.2514/6.2006-2910

[25] Amar, A.J., Blackwell, B.F. and Edward, J.R. (2007) One-Dimensional Ablation with Pyrolysis Gas Flow Using a Full Newton's Method and Finite Control Volume Procedure. 39*th AIAA Thermophysics Conference*, AIAA-2007-4535, Miami, 25-28 June 2007, 41. http://dx.doi.org/10.2514/6.2007-4535

[26] Martin, A., Reggio, M., Trepanier, J.Y. and Guo, X. (2007) Transient Ablation Regime in Circuit Breakers. *Plasma Science and Technology*, **9**, 653-656. http://dx.doi.org/10.1088/1009-0630/9/6/02

[27] Martin, A. and Boyd, I.D. (2008) Simulation of Pyrolysis Gas within a Thermal Protection System. 40*th AIAA Thermophysics Conference*, AIAA-2008-3805, Seattle, 23-26 June 2008. http://dx.doi.org/10.2514/6.2008-3805

[28] Martin, A. and Boyd, I.D. (2010) Variant of the Thomas Algorithm for Opposite-Bordered Tridiagonal Systems of Equations. *International Journal for Numerical Methods in Biomedical Engineering*, **26**, 752-759.

[29] Meyer, C.D. (2000) Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia. http://dx.doi.org/10.1137/1.9780898719512.