British Journal of Mathematics & Computer Science
4(9): 1228-1239, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

SCIENCEDOMAIN

New Algorithms for Solving k-Tridiagonal Linear Systems

Moawwad El-Mikkawy[*1] **and Faiz Atlan**[1]

[1] *Mathematics Department, Faculty of Science, Mansoura University, Mansoura 35516, Egypt*

Original Research Article

Received: 22 December 2013
Accepted: 25 January 2014
Published: 27 February 2014

Abstract

The current paper focus on solving k-tridiagonal linear systems of equations via transformation. It investigates numeric and symbolic algorithms for solving such systems. Based on the symbolic algorithm, a MAPLE program is written. The computational cost of the algorithms is given. Some illustrative examples are introduced.

Keywords: Matrices; Algorithm; LU factorization; MAPLE procedure.
2010 Mathematics Subject Classification: 65F05, 65Fxx, 68W30.

1 Introduction

In many scientific and engineering applications, different special linear systems of equations arise. For such systems the coefficient matrix has special structure. Sparse matrices which contain a majority of zeros occur are often encountered. It is usually more efficient to solve these systems using tailor-made algorithms, much faster and with less storage than a full matrix. This can be achieved by taking advantage of the special structure of the coefficient matrix, see for instance, ([1], [2], [3]). Important examples are band matrices, and the most common cases are the matrices of tridiagonal type. A tridiagonal matrix is one with nonzero entries along the main diagonal and one diagonal above and below the main one. Consequently tridiagonal matrix takes the form:

$$T = \begin{bmatrix} d_1 & a_1 & 0 & \dots & 0 \\ b_1 & d_2 & a_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & b_{n-2} & d_{n-1} & a_{n-1} \\ 0 & \dots & 0 & b_{n-1} & d_n \end{bmatrix}, \tag{1.1}$$

The matrix in (1.1) is frequently appears in many applications. For example in parallel computing, telecommunication system analysis, solving differential equations using finite differences, heat conduction

*Corresponding author: E-mail: m_elmikkawy@yahoo.com and fluid flow problems. The interested reader may refer to ([4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]) and the references therein.

Definition 1.1. For positive integers $n = 3, 4, ...$ and $k = 1, 2, ..., n-1$, a matrix $T_n^{(k)} = (t_{ij})_{i,j=1}^{n}$ of order n is called k-tridiagonal if

$$t_{ij} = 0, \quad \text{for} \quad |i-j| \neq 0 \quad \text{and} \quad k. \tag{1.2}$$

For example,

$$T_5^{(3)} = \begin{bmatrix} d_1 & 0 & 0 & a_1 & 0 \\ 0 & d_2 & 0 & 0 & a_2 \\ 0 & 0 & d_3 & 0 & 0 \\ b_1 & 0 & 0 & d_4 & 0 \\ 0 & b_2 & 0 & 0 & d_5 \end{bmatrix}.$$

For $k \geq n$, the matrix $T_n^{(k)}$ is a diagonal matrix and the case $k = 1$ gives the ordinary tridiagonal matrix in (1.1). As pointed out in [15], the matrix $T_n^{(k)}$ plays an important role in describing generalized k-Fibonacci numbers. Furthermore, the matrix $T_n^{(k)}$ has recently received attention by some authors ([16], [17], [18]). The nonzero elements of the matrix in (1.2) can be stored in $3n-2k$ memory locations by using the three vectors ***a*** $= [a_1, a_2, \ldots, a_{n-k}]$, ***b*** $= [b_1, b_2, \ldots, b_{n-k}]$, and ***d*** $= [d_1, d_2, \ldots, d_n]$.

The motivation of the current paper is to develop new algorithms for solving k-tridiagonal linear systems of the form:

$$T_n^{(k)}\, \boldsymbol{x} = \boldsymbol{f}, \tag{1.3}$$

with

$$T_n^{(k)} = \begin{bmatrix} d_1 & 0 & \ldots & 0 & a_1 & 0 & \ldots & 0 \\ 0 & d_2 & 0 & \ldots & 0 & a_2 & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ldots & \ddots & \ddots & 0 \\ 0 & \ldots & \ddots & \ddots & \ddots & \ldots & \ddots & a_{n-k} \\ b_1 & 0 & \ldots & \ddots & \ddots & \ddots & \ldots & 0 \\ 0 & b_2 & \ddots & \ldots & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ldots & 0 & d_{n-1} & 0 \\ 0 & \ldots & 0 & b_{n-k} & 0 & \ldots & 0 & d_n \end{bmatrix}, \tag{1.4}$$

where $1 \leq k < n$, ***x*** $= [x_1, x_2, ..., x_n]^T$ and ***f*** $= [f_1, f_2, ..., f_n]^T$.

Throughout this paper, $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Also, the word 'simplify' means simplify the expression under consideration to its simplest rational form.
The organization of the paper is as follows. The main results are given in Section 2. In Section 3, a MAPLE procedure is introduced. Some illustrative examples are given in Section 4.

2 Main Results

In this section, we are going to consider the construction of new algorithms for solving linear systems of equations of k-tridiagonal type. For this purpose it is convenient to introduce the vector ***e*** $= [e_1, e_2, ..., e_n]$ with components given by:

$$e_i = \begin{cases} d_i, & \text{for } i = n, n-1, ..., n-k+1 \\ d_i - \frac{a_i\, b_i}{e_{i+k}}, & \text{for } i = n-k, n-k-1, ..., 1. \end{cases} \tag{2.1}$$

The system (1.3) can be described by the augmented matrix, G given by:

$$G = \left[\begin{array}{cccccccc|c} d_1 & 0 & \dots & 0 & a_1 & 0 & \dots & 0 & f_1 \\ 0 & d_2 & 0 & \dots & 0 & a_2 & \ddots & \vdots & f_2 \\ \vdots & 0 & \ddots & \ddots & \dots & \ddots & \ddots & 0 & f_3 \\ 0 & \dots & \ddots & \ddots & \ddots & \dots & \ddots & a_{n-k} & \vdots \\ b_1 & 0 & \dots & \ddots & \ddots & \ddots & \dots & 0 & \vdots \\ 0 & b_2 & \ddots & \dots & \ddots & \ddots & 0 & \vdots & f_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \dots & 0 & d_{n-1} & 0 & f_{n-1} \\ 0 & \dots & 0 & b_{n-k} & 0 & \dots & 0 & d_n & f_n \end{array}\right], \tag{2.2}$$

Let R$_i$ denotes the i-th row of the matrix G. Performing the following row operations on G, in the same order:

```
For $i = n, n-1, ..., n-k+1$ do
        R$_i \longleftarrow$ R$_i/e_i$
End do.
For $i = n-k, n-k-1, ..., 1$ do
        R$_i \longleftarrow$ R$_i - a_i$R$_{i+k}$
        R$_i \longleftarrow$ R$_i/e_i$
End do.
```

Then we have the equivalent transformed linear system of the form:

$$\left[\begin{array}{cccccccc|c} 1 & 0 & \dots & & & & \dots & 0 & Z_1 \\ 0 & 1 & & & & & & \vdots & Z_2 \\ \vdots & 0 & \ddots & & & & & & Z_3 \\ 0 & \vdots & \ddots & \ddots & & & & & \vdots \\ Y_1 & \ddots & \vdots & \ddots & \ddots & & & & \vdots \\ 0 & Y_2 & \ddots & \dots & \ddots & \ddots & & \vdots & Z_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \dots & 0 & \ddots & 0 & Z_{n-1} \\ 0 & \dots & 0 & Y_{n-k} & 0 & \dots & 0 & 1 & Z_n \end{array}\right], \tag{2.3}$$

where

$$Y_{i-k} = \frac{b_{i-k}}{e_i}, \quad i = n, n-1, ..., k+1,$$

and

$$Z_i = \begin{cases} \frac{f_i}{e_i}, & \text{for } i = n, n-1, ..., n-k+1 \\ \frac{1}{e_i}(f_i - a_i\, Z_{i+k}), & \text{for } i = n-k, n-k-1, ..., 1. \end{cases} \tag{2.4}$$

The transformed system (2.3) is easy to solve by using forward substitution. Therefore the linear system (1.3) can be solved using the following algorithm:

Algorithm 2.1. Numeric algorithm for solving k-tridiagonal linear system.

To solve the linear system of the form (1.3), we may proceed as follows:

```
INPUT: Order of the coefficient matrix $n$, value of $k$ and the values, $a_i,\ b_i,\ i = 1, 2, \ldots, n-k$,
       $d_i,\ f_i,\ i = 1, 2, \ldots, n$.
OUTPUT: The solution vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$.
Step 1: If $k <= \lfloor \frac{n}{2} \rfloor$ then
            For $i = n, n-1, ..., n-k+1$ do
                Set:
                $e_i = d_i$.
                $Y_{i-k} = \frac{b_{i-k}}{e_i}$
                $Z_i = \frac{f_i}{e_i}$.
            End do.
            For $i = n-k, n-k-1, ..., k+1$ do
                Compute and simplify:
                $e_i = d_i - a_i\, Y_i$.
                $Y_{i-k} = \frac{b_{i-k}}{e_i}$,
                $Z_i = \frac{1}{e_i}(f_i - a_i\, Z_{i+k})$.
            End do.
            For $i = k, k-1, ..., 1$ do
                $e_i = d_i - a_i\, Y_i$.
                $Z_i = \frac{1}{e_i}(f_i - a_i\, Z_{i+k})$.
            End do.
        Else
            For $i = n, n-1, ..., n-k+1$ do
                Set:
                $e_i = d_i$.
                $Z_i = \frac{f_i}{e_i}$.
            End do.
            For $i = n-k, n-k-1, ..., 1$ do
                Compute and simplify:
                $Y_i = \frac{b_i}{e_{i+k}}$,
                $e_i = d_i - a_i\, Y_i$.
                $Z_i = \frac{1}{e_i}(f_i - a_i\, Z_{i+k})$.
            End do.
        End if.
Step 2: Use the k-DETGTRI algorithm [16] to check the non-singularity of the coefficient
        matrix of the system (1.3).
Step 3: If $\det(T_n^{(k)}) = 0$, then Exiterror('No solutions') end if.
Step 4: Compute the solution vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ using
        For $i = 1, 2, ..., k$ do
            $x_i = Z_i$,
        End do.
        For $i = k+1, k+2, ..., n$ do
            $x_i = Z_i - Y_{i-k}\, x_{i-k}$
        End do.
```

The Algorithm 2.1, will be referred to as **k-TRANSTRI-I** algorithm. The cost of this algorithm is $(5n - 4k)$ multiplications/divisions and $(3n - 3k)$ addtions/subtractions, provided that the coefficient matrix (1.4) is nonsingular.

The following symbolic version algorithm is developed in order to remove the cases where the numeric algorithm **k-TRANSTRI-I** fails. The parameter 't' in the algorithm is just a symbolic name. It is a dummy argument and its actual value is zero.

Algorithm 2.2. Symbolic version algorithm for solving k-tridiagonal linear system.

To solve the linear system of the form (1.3), we may proceed as follows:

```
INPUT: Order of the coefficient matrix $n$, value of $k$ and the values, $a_i,\ b_i,\ i = 1, 2, \ldots, n-k$,
       $d_i,\ f_i,\ i = 1, 2, \ldots, n$.
OUTPUT: The solution vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$.
Step 1: If $k <= \lfloor \frac{n}{2} \rfloor$ then
          For $i = n, n-1, ..., n-k+1$ do
            Set:
            $e_i = d_i$. If $e_i = 0$ then $e_i = t$ end if.
            $Y_{i-k} = \frac{b_{i-k}}{e_i}$
            $Z_i = \frac{f_i}{e_i}$.
          End do.
          For $i = n-k, n-k-1, ..., k+1$ do
            Compute and simplify:
            $e_i = d_i - a_i\, Y_i$. If $e_i = 0$ then $e_i = t$ end if.
            $Y_{i-k} = \dfrac{b_{i-k}}{e_i}$,
            $Z_i = \dfrac{1}{e_i}(f_i - a_i\, Z_{i+k})$.
          End do.
          For $i = k, k-1, ..., 1$ do
            $e_i = d_i - a_i\, Y_i$. If $e_i = 0$ then $e_i = t$ end if.
            $Z_i = \dfrac{1}{e_i}(f_i - a_i\, Z_{i+k})$.
          End do.
        Else
          For $i = n, n-1, ..., n-k+1$ do
            Set:
            $e_i = d_i$. If $e_i = 0$ then $e_i = t$ end if.
            $Z_i = \frac{f_i}{e_i}$.
          End do.
          For $i = n-k, n-k-1, ..., 1$ do
            Compute and simplify:
            $Y_i = \dfrac{b_i}{e_{i+k}}$,
            $e_i = d_i - a_i\, Y_i$. If $e_i = 0$ then $e_i = t$ end if.
            $Z_i = \dfrac{1}{e_i}(f_i - a_i\, Z_{i+k})$.

Step 2: Use the k-DETGTRI algorithm [16] to check the non-singularity of the coefficient
        matrix of the system (1.3).
Step 3: If $\det(T_n^{(k)}) = 0$, then Exiterror('No solutions') end if.
Step 4: Compute the solution vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ using
        For $i = 1, 2, ..., k$ do
          $x_i = Z_i$,
        End do.
        For $i = k+1, k+2, ..., n$ do
          $x_i = Z_i - Y_{i-k}\, x_{i-k}$
```

End do.

Step 5: Substitute $t = 0$ in all expressions of the solution vector $x_i, i = 1, 2, ..., n.$

The Algorithm 2.2, will be referred to as **k-TRANSTRI-II** algorithm. The solution vector ***x*** of the system (1.3) is in fact the solution vector of the linear system:

$$\begin{bmatrix} 1 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 1 & & & & & & \vdots \\ \vdots & 0 & \ddots & & & & & \\ 0 & \vdots & \ddots & \ddots & & & & \\ Y_1 & \ddots & \vdots & \ddots & \ddots & & & \\ 0 & Y_2 & \ddots & \cdots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & Y_{n-k} & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ \vdots \\ \vdots \\ Z_{n-2} \\ Z_{n-1} \\ Z_n \end{bmatrix}, \tag{2.5}$$

where the values $Z_i,\ i = 1, 2, ..., n$ in (2.4) are the solution of the linear system:

$$\begin{bmatrix} e_1 & 0 & \cdots & 0 & a_1 & 0 & \cdots & 0 \\ 0 & e_2 & 0 & \cdots & 0 & a_2 & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \cdots & \ddots & \ddots & 0 \\ & & \ddots & \ddots & \ddots & \cdots & \ddots & a_{n-k} \\ \vdots & & & \ddots & \ddots & \ddots & \cdots & 0 \\ & & & & \ddots & \ddots & 0 & \vdots \\ \vdots & & & & & \ddots & e_{n-1} & 0 \\ 0 & \cdots & & & & \cdots & 0 & e_n \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ \vdots \\ \vdots \\ Z_{n-2} \\ Z_{n-1} \\ Z_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix}. \tag{2.6}$$

From (2.5) and (2.6), we get

$$\begin{bmatrix} e_1 & 0 & \cdots & 0 & a_1 & 0 & \cdots & 0 \\ 0 & e_2 & 0 & \cdots & 0 & a_2 & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \cdots & \ddots & \ddots & 0 \\ & & \ddots & \ddots & \ddots & \cdots & \ddots & a_{n-k} \\ \vdots & & & \ddots & \ddots & \ddots & \cdots & 0 \\ & & & & \ddots & \ddots & 0 & \vdots \\ \vdots & & & & & \ddots & e_{n-1} & 0 \\ 0 & \cdots & & & & \cdots & 0 & e_n \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & & & & \cdots & 0 \\ 0 & 1 & & & & & & \vdots \\ \vdots & 0 & \ddots & & & & & \\ 0 & \vdots & \ddots & \ddots & & & & \\ Y_1 & \ddots & \vdots & \ddots & \ddots & & & \\ 0 & Y_2 & \ddots & \cdots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \cdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & Y_{n-k} & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix}, \tag{2.7}$$

From (1.3) and (2.7), we obtain the Doolittle UL factorization [19] of $T_n^{(k)}$ in the form:

$$
\begin{aligned}
T_n^{(k)} &= \begin{bmatrix}
d_1 & 0 & \dots & 0 & a_1 & 0 & \dots & 0 \\
0 & d_2 & 0 & \dots & 0 & a_2 & \ddots & \vdots \\
\vdots & 0 & \ddots & \ddots & \dots & \ddots & \ddots & 0 \\
0 & \dots & \ddots & d_{n-k} & \ddots & \dots & \ddots & a_{n-k} \\
b_1 & 0 & \dots & \ddots & \ddots & \ddots & \dots & 0 \\
0 & b_2 & \ddots & \dots & \ddots & \ddots & 0 & \vdots \\
\vdots & \ddots & \ddots & \ddots & \dots & 0 & d_{n-1} & 0 \\
0 & \dots & 0 & b_{n-k} & 0 & \dots & 0 & d_n
\end{bmatrix} \\
&= \begin{bmatrix}
e_1 & 0 & \dots & 0 & a_1 & 0 & \dots & 0 \\
0 & e_2 & 0 & \dots & 0 & a_2 & \ddots & \vdots \\
\vdots & 0 & \ddots & \ddots & \dots & \ddots & \ddots & 0 \\
 & & \ddots & \ddots & \ddots & \dots & \ddots & a_{n-k} \\
\vdots & & & \ddots & \ddots & \ddots & \dots & 0 \\
 & & & & \ddots & \ddots & 0 & \vdots \\
\vdots & & & & & \ddots & e_{n-1} & 0 \\
0 & \dots & & & & \dots & 0 & e_n
\end{bmatrix}
\begin{bmatrix}
1 & 0 & \dots & & & & \dots & 0 \\
0 & 1 & & & & & & \vdots \\
\vdots & 0 & \ddots & & & & & \\
0 & \vdots & \ddots & \ddots & & & & \\
Y_1 & \ddots & \vdots & \ddots & \ddots & & & \\
0 & Y_2 & \ddots & \dots & \ddots & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \ddots & \dots & 0 & \ddots & 0 \\
0 & \dots & 0 & Y_{n-k} & 0 & \dots & 0 & 1
\end{bmatrix} \\
&= U_n^{(k)} L_n^{(k)}.
\end{aligned}
\tag{2.8}
$$

Using (2.8), we get the Crout UL factorization [19] of $T_n^{(k)}$ in the form:

$$
T_n^{(k)} = \begin{bmatrix}
1 & 0 & \dots & 0 & \frac{a_1}{e_{k+1}} & 0 & \dots & 0 \\
0 & 1 & 0 & \dots & 0 & \frac{a_2}{e_{k+2}} & \ddots & \vdots \\
\vdots & 0 & \ddots & \ddots & \dots & \ddots & \ddots & 0 \\
 & & \ddots & \ddots & \ddots & \dots & \ddots & \frac{a_{n-k}}{e_n} \\
\vdots & & & \ddots & \ddots & \ddots & \dots & 0 \\
 & & & & \ddots & \ddots & 0 & \vdots \\
\vdots & & & & & \ddots & 1 & 0 \\
0 & \dots & & & & \dots & 0 & 1
\end{bmatrix}
\begin{bmatrix}
e_1 & 0 & \dots & & & & \dots & 0 \\
0 & e_2 & & & & & & \vdots \\
\vdots & 0 & \ddots & & & & & \\
0 & \vdots & \ddots & \ddots & & & & \\
b_1 & \ddots & \vdots & \ddots & \ddots & & & \\
0 & b_2 & \ddots & \dots & \ddots & \ddots & & \vdots \\
\vdots & \ddots & \ddots & \ddots & \dots & 0 & \ddots & 0 \\
0 & \dots & 0 & b_{n-k} & 0 & \dots & 0 & e_n
\end{bmatrix}.
\tag{2.9}
$$

Corollary 2.1. *Let $\hat{T}_n^{(k)}$ be the backward matrix of the k-tridiagonal matrix $T_n^{(k)}$ in (1.4), and given by:*

$$\hat{T}_n^{(k)} = \begin{bmatrix} 0 & \dots & \dots & 0 & a_1 & 0 & \dots & 0 & d_1 \\ \vdots & & 0 & a_2 & \iddots & \dots & \iddots & d_2 & 0 \\ \vdots & \iddots & \iddots & \iddots & \dots & \iddots & d_3 & \iddots & \vdots \\ 0 & \iddots & \iddots & \dots & \iddots & \iddots & \iddots & \iddots & 0 \\ a_{n-k} & \iddots & \dots & \iddots & \iddots & \iddots & \iddots & \iddots & b_1 \\ 0 & \dots & \iddots & \iddots & \iddots & \iddots & \iddots & b_2 & 0 \\ \vdots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots \\ 0 & d_{n-1} & \iddots & \iddots & \iddots & \iddots & \iddots & & \vdots \\ d_n & 0 & \dots & 0 & b_{n-k} & 0 & \dots & \dots & 0 \end{bmatrix}. \tag{2.10}$$

Then the backward tridiagonal linear system

$$\hat{T}_n^{(k)}\,[u_1, u_2, ..., u_n]^T = \mathbf{f} \tag{2.11}$$

has the solution: $u_i = x_{n+1-i}$, $i = 1, 2, ..., \lfloor \frac{n}{2} \rfloor$, where $[x_1, x_2, \dots, x_n]^T$ is the solution vector of the linear system (1.3).

***Proof**. Consider the $n \times n$ permutation matrix P defined by:*

$$P = \begin{bmatrix} 0 & \dots & \dots & 0 & 1 \\ \vdots & & \iddots & 1 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & 1 & \iddots & & \vdots \\ 1 & 0 & \dots & \dots & 0 \end{bmatrix}. \tag{2.12}$$

For this matrix, we have:

$$P^{-1} = P^T = P. \tag{2.13}$$

Since

$$\hat{T}_n^{(k)} = T_n^{(k)}\,P \tag{2.14}$$

Then using (2.13) and (2.14), the result follows.

Corollary 2.2. *The determinants of the coefficient matrices $T_n^{(k)}$ and $\hat{T}_n^{(k)}$ in (1.4) and (2.10) are given respectively by:*

$$\det(T_n^{(k)}) = \prod_{r=1}^{n} e_r \tag{2.15}$$

and

$$\det(\hat{T}_n^{(k)}) = (-1)^{\frac{n(n-1)}{2}} (\prod_{r=1}^{n} e_r), \tag{2.16}$$

where $e_1, e_2, ..., e_n$ as in (2.1).

3 COMPUTER PROGRAM

In this section, we are going to introduce a MAPLE procedure for solving linear system of k-tridiagonal type (1.3). The procedure is based on the algorithm **k-TRANSTRI-II**. The procedure alters the contents of the vectors **d**, **b** and ***f***. Eventually, the contents of the vectors **e**, **Y** and **Z** are stored in **d**, **b** and ***f***, respectively.

```
> restart:
 ktritrans:= proc(n::posint, k::posint,d::vector, a::vector, b::vector,f::vector)
     local i:
     global x,T:
     x:= vector(n):
     if $k <= $ floor($\frac{n}{2}$) then
        for i from n by -1 to n-k+1 do
           if d[i] = 0 then d[i]:=t fi:
           b[i-k]:=simplify(b[i-k]/d[i]): f[i]:=simplify(f[i]/d[i]):
        od:
        for i from n-k by -1 to k+1 do
           d[i] := simplify(d[i]-a[i]*b[i]);
           if d[i] = 0 then d[i] := t; fi:
           b[i-k] := simplify(b[i-k]/d[i]);
           f[i] := simplify((f[i]-f[i+k]*a[i])/d[i]);
        od;
        for i from k by -1 to 1 do
           d[i] := simplify(d[i]-a[i]*b[i]);
           if d[i] = 0 then d[i] := t; fi:
           f[i] := simplify((f[i]-f[i+k]*a[i])/d[i]);
        od;
      else
        for i from n by -1 to n-k+1 do
           if d[i] = 0 then d[i]:=t fi:
           f[i]:=simplify(f[i]/d[i]):
        od:
        for i from n-k by -1 to 1 do
           b[i]:=simplify(b[i]/d[i+k]):
           d[i] := simplify(d[i]-a[i]*b[i]);
           if d[i] = 0 then d[i] := t; fi:
           f[i] := simplify((f[i]-f[i+k]*a[i])/d[i]);  od;
     fi:
     # To compute the determinant of the k-tridiagonal matrix#
     T:= subs(t =0,simplify(product(d[r],r= 1..n))):
      if T = 0 then
        error("Singular Matrix")
      else
        # To compute the Solution X, of the system. #
        for i to k do
           x[i]:=f[i];
        od:
        for i from k+1 to n do
           x[i]:=simplify((f[i]-b[i-k]*x[i-k]));
        od;
         eval(x);
```

```
    fi:
end proc :
```

4 Illustrative Examples

Example 4.1. *Solve the k-tridiagonal linear system*

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 3 \\ 4 \\ 0 \\ -6 \\ 7 \\ 4 \\ 1 \\ 3 \end{bmatrix}$$

Solution: *Here, we have*
$n = 10$, $k = 6$, **a** $= [1, -1, 2, 4]$, ***d*** $= [2, 1, -1, 3, 4, -2, 5, 3, -1, 3]$,
b $= [2, -1, 3, 2]$, *and* **f** $= [3, 0, 3, 4, 0, -6, 7, 4, 1, 3]$.
*By applying the **k-TRANSTRI-I** algorithm, we get*

- **e**=$[\frac{8}{5}, \frac{2}{3}, 5, \frac{1}{3}, 4, -2, 5, 3, -1, 3]$.
- $\det(T_{10}^{(6)}) = \prod_{i=1}^{10} e_i = 640.$
- *The solution vector is given by:* **x** = $[1, 2, 1, 0, 0, 3, 1, 2, 2, 1]^T$.

*By using the algorithm **k-TRANSTRI-II**, we obtain the same solution vector.*

Example 4.2. *Solve the k-tridiagonal linear system*

$$\begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 4 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 3 \\ 0 & 0 & 3 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 0 \\ 13 \\ 6 \\ 5 \\ 0 \\ 9 \\ 0 \\ 6 \end{bmatrix} \tag{4.1}$$

Solution: *Here, we have*
$n = 10$, $k = 4$, **a** $= [1, -1, 2, 4, 1, 3]$, **d** $= [2, 1, -1, 3, 1, 3, 5, 3, -1, 3]$,
b $= [2, -1, 3, 2, 1, 3]$, *and* **f** $= [4, 2, 0, 13, 6, 5, 0, 9, 0, 6]$.

*The numeric algorithm **k-TRANSTRI-I** fails to solve the linear system (4.1), since $e_6 = 0$. Applying the **k-TRANSTRI-II** algorithm, gives:*

- **e**=$[1, \frac{(t-1)}{t}, \frac{-11}{5}, \frac{1}{3}, 2, t, 5, 3, -1, 3]$.
- $\det(T_{10}^{(4)}) = (\prod_{i=1}^{10} e_i)_{t=0} = (66 * t - 66)_{t=0} = -66$.
- *The solution vector is given by:*
  x $= [1, 1, 0, 3, 2, -1, 0, 1, 2, 3]^T$.

5 Conclusions

In this paper, we derived new algorithms for solving k-tridiagonal linear systems. The computational cost of the algorithms is given. A MAPLE procedure based on these algorithms is presented. Finally, some examples are given for the sake of illustration.

Acknowledgment

The authors are extremely thankful to the editorial team of SDI and anonymous referees for their useful comments and suggestions that have helped in improving the readability of this paper.

Competing interests

The authors declare that no competing interests exist.

References

[1] Bau D, Trefethen L.N. Numerical Linear Algebra. Philadelphia: SIAM. 1997.

[2] Golub G.H, Charles F, Loan van. Matrix Computations. 3rd edition. Johns Hopkins University Press. 1996.

[3] Leader J.J. Numerical Analysis and Scientific Computation. Addison Wesley. 2004.

[4] Abderraman Marrero J, Rachidi M, Tomeo V. Non-symbolic algorithms for the inversion of tridiagonal matrices. J. Appl. Math. Comput. 2013;(252):3-11.

[5] Al-Hassan Q. On powers of tridiagonal matrices with nonnegative entries. Appl. Math. Sci. 2012;6(48):2357-2368.

[6] da Fonseca C.M, Petronilho J. Explicit inverses of some tridiagonal matrices. Lin. Alg. Appl. 2001;325:7-21.

[7] El-Mikkawy M.E.A, Karawia A. Inversion of general tridiagonal matrices. Appl. Math. Lett. 2006;19:712-720.

[8] Hu G.Y, O'Connell R.F. Analytical inversion of symmetric tridiagonal matrices. J. Phys. A. 1996;29:1511-1513.

[9] Huang Y, McColl W.F. Analytic inversion of general tridiagonal matrices. J. Phys. A. 1997;30:7919-7933.

[10] Mazilu I, Mazilu D.A, Williams H.T. Applications of tridiagonal matrices in non-equilibrium statistical physics. Electron. J. Linear Algebra. 2012;24:7-17.

[11] Kavcic A, Moura J.M.F. Matrices with banded inverses: Inversion algorithms and factorization of Gauss-Markov processes. IEEE Trans. Inform. Theory. 2000;46(4):1495-1509.

[12] Li H.B, Huang T.Z, Liu X.P, Li H. On the inverses of general tridiagonal matrices. Linear Algebra Appl. 2010;433:965-983.

[13] Olcayto E. Recursive formulae for ladder network optimization. Electron. Lett. 1979;15(9):249-250.

[14] Sugimoto T. On an inverse formula of a tridiagonal matrix. Operators and Matrices. 2012;6(3):465-480.

[15] El-Mikkawy M, Sogabe T. A new family of k-Fibonacci numbers. Appl. Math.Comput. 2010;215(12):4456-4461.

[16] El-Mikkawy M.E.A. A generalized symbolic Thomas algorithm. Applied Mathematics. 2012;3(4):342-345.

[17] Jia J, Sogabe T, El-Mikkawy M. Inversion of k-tridiagonal matrices with Toeplitz structure. Comput. Math. Appl. 2013;65:116-125.

[18] Sogabe T, El-Mikkawy M. Fast block diagonalization of k-tridiagonal matrices. Appl. Math.Comput. 2011;218(6):2740-2743.

[19] Burden R.L, Faires J.D. Numerical Analysis. seventh ed. Books & Cole Publishing, Pacific Grove. CA. 2001.
