Inversion of general tridiagonal matrices

Moawwad El-Mikkawy*, Abdelrahman Karawia

Department of Mathematics, Faculty of Science, Mansoura University, Mansoura, 35516, Egypt

Received 10 September 2005; received in revised form 26 October 2005; accepted 4 November 2005

Abstract


© 2005 Elsevier Ltd. All rights reserved.

Keywords: Tridiagonal matrix; Inverse matrix; Determinants; Computer algebra systems (CAS)

1. Introduction

The general tridiagonal matrix $T = (t_{ij})_{1 \leq i, j \leq n}$ takes the form

$$
T = \begin{bmatrix}
    d_1 & a_1 & 0 & \ldots & \ldots & 0 \\
    b_2 & d_2 & a_2 & \ldots & \ldots & 0 \\
    0 & b_3 & d_3 & a_3 & \ldots & 0 \\
    \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & \ldots & \ldots & b_{n-1} & d_{n-1} & a_{n-1} \\
    0 & 0 & \ldots & 0 & b_n & d_n \\
\end{bmatrix}
$$

where $t_{ij} = 0$ for $|i - j| \geq 2$. This matrix appears frequently in many areas of science and engineering, for example in parallel computing, telecommunication system analysis and in solving differential equations using finite differences. Finding the inverse of such a matrix is usually required in these fields. This problem has been investigated by many authors (see for instance, [1,2], [4–8]). Computational approaches for finding the inverse of a general tridiagonal matrix $T$ of form (1.1) are given in [1,2,7,8]. In [2,7], the authors assumed the conditions $a_1, a_2, \ldots, a_{n-1} \neq 0$ and $b_2b_3, \ldots, b_n \neq 0$. In [8] no conditions are imposed but the matrix $T$ is assumed to possess an LU decomposition [9].

* Corresponding author. Tel.: +20 2520708.

E-mail addresses: mikkawy@yahoo.com (M. El-Mikkawy), abibka@mans.edu.eg (A. Karawia).
Of course, it is advantageous to prove the nonsingularity of the matrix $T$ of form (1.1) before considering the problem of finding its inverse. A very efficient and reliable algorithm for this purpose is the DETGTRI algorithm given in [3]. To the best of our knowledge, the first complete analysis for the general tridiagonal matrix inversion problem without imposing any conditions was investigated in [1].

The present work is mainly devoted to developing a symbolic algorithm that will never break down in order to find the inverse of any nonsingular tridiagonal matrix $T$ of form (1.1). To achieve this goal we are going to modify the numeric algorithm given in [1] to remove all cases for which the numeric algorithm fails. The work is organized as follows. The main results are given in Section 2. In Section 3, an illustrative example is given.

2. Main results

Throughout this work the word ‘simplify’ indicates that the algebraic expression under consideration should be in its simplest rational form.

On the basis of the results presented in [1], we may introduce two $(n+1)$-dimensional vectors $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_{n+1})$ as follows:

$$\alpha_i = \begin{cases} 1 & \text{if } i = 0 \\ d_1 & \text{if } i = 1 \\ d_i \alpha_{i-1} - b_i a_{i-1} \alpha_{i-2} & \text{if } i = 2, 3, \ldots, n \end{cases}$$ \hspace{1cm} (2.1)

and

$$\beta_i = \begin{cases} 1 & \text{if } i = n + 1 \\ d_n & \text{if } i = n \\ d_i \beta_{i+1} - b_{i+1} a_i \beta_{i+2} & \text{if } i = n - 1, n - 2, \ldots, 1. \end{cases}$$ \hspace{1cm} (2.2)

The elements of the vectors $\alpha$ and $\beta$ in (2.1) and (2.2) are precisely the determinants of specific submatrices of the tridiagonal matrix $T$ in (1.1). For this matrix, we obtained the following results (see also [3,10,11]) whose proof will be omitted.

Theorem 2.1. For the matrix $T$ in (1.1) the following are valid.

(i) $\det T = \alpha_n = \beta_1$.

(ii) $T$ is nonsingular if and only if $\alpha_n \neq 0$.

(iii) $T$ is nonsingular if and only if $\beta_1 \neq 0$.

(iv) $T$ possesses an LU decomposition if and only if $\alpha_k \neq 0$, $k = 1, 2, \ldots, n - 1$.

(v) $T$ is positive definite if and only if $\alpha_k > 0$, $k = 1, 2, \ldots, n$.

(vi) $T$ is positive definite if and only if $\beta_k > 0$, $k = 1, 2, \ldots, n$.

(vii) If $T$ is positive definite then the algorithm HMGTI below will not break down.

Following [1], we see that the inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$ of the matrix $T$ in (1.1) is given by:

$$u_{11} = \left( d_1 - \frac{b_2 a_1 \beta_3}{\beta_2} \right)^{-1}$$ \hspace{1cm} (2.3)

$$u_{nn} = \left( d_n - \frac{b_n a_{n-1} \alpha_{n-2}}{\alpha_{n-1}} \right)^{-1}$$ \hspace{1cm} (2.4)

$$u_{ii} = \left( d_i - \frac{b_i a_{i-1} \alpha_{i-2}}{\alpha_{i-1}} - \frac{b_{i+1} a_i \beta_{i+2}}{\beta_{i+1}} \right)^{-1}, \quad i = 2, 3, \ldots, n - 1$$ \hspace{1cm} (2.5)

$$u_{ij} = \begin{cases} (-1)^{j-i} \left( \prod_{k=1}^{i-j} a_{j-k} \right) \frac{\alpha_{j-1}}{\alpha_{j-i}} u_{jj} & \text{if } i < j \\ (-1)^{i-j} \left( \prod_{k=1}^{i-j} b_{j+k} \right) \frac{\beta_{j+1}}{\beta_{j-i}} u_{jj} & \text{if } i > j. \end{cases}$$ \hspace{1cm} (2.6)
For the convenience of the reader we are going to restate a result presented in [1] concerning the inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$ of the matrix $T$ in (1.1) as follows:

**Algorithm 2.1.** To find the $n \times n$ inverse matrix of a general tridiagonal matrix $T$ of form (1.1).

**INPUT** order of the matrix $n$ and the components $a_i, b_i, d_i, i = 1, 2, \ldots, n$

$$(b_1 = a_n = 0).$$

**OUTPUT** inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$.

**Step 1:** Set $\alpha_0 = 1, \alpha_1 = d_1, \beta_{n+1} = 1, \beta_n = d_n$.

**Step 2:** Compute $\alpha_i, i = 2, 3, \ldots, n$, using (2.1).

**Step 3:** If $\alpha_n = 0$, then OUTPUT (‘no inverse exists’); STOP.

**Step 4:** Set $\beta_1 = \alpha_n$ and compute $\beta_i, i = n - 1, n - 2, \ldots, 2$, using (2.2).

**Step 5:** If $\alpha_i = 0$ or $\beta_{i+1} = 0$ for some $i \in \{1, 2, \ldots, n - 1\}$, then OUTPUT (‘Failure’); STOP.

**Step 6:** Compute $u_{11}, u_{nn}$ using (2.3) and (2.4) respectively. For $i = 2, 3, \ldots, n - 1$, compute $u_{ii}$ using (2.5). For $i \neq j$, compute the remaining elements of the matrix $T^{-1}$ by using (2.6).

**Step 7:** OUTPUT the inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$.

The Algorithm 2.1 will be referred to as the HMGTI algorithm.

As can be easily seen, the HMGTI algorithm breaks down unless the conditions $\alpha_i \neq 0$ and $\beta_{i+1} \neq 0$ are satisfied for all $i = 1, 2, \ldots, n - 1$. Theorem 2.5 in [1] summarizes the cases for which the HMGTI algorithm will not break down. The following symbolic algorithm is developed by these authors in order to remove the cases where the numeric algorithm HMGTI fails.

**Algorithm 2.2.** To find the $n \times n$ inverse matrix of a general tridiagonal matrix $T$ of form (1.1).

**INPUT** order of the matrix $n$ and the components $a_i, b_i, d_i, i = 1, 2, \ldots, n$

$$(b_1 = a_n = 0).$$

**OUTPUT** inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$.

**Step 1:** Set $\alpha_0 = 1, \alpha_1 = d_1, \beta_{n+1} = 1, \beta_n = d_n$. Set $\alpha_1 = x$ ($x$ is just a symbolic name) whenever $\alpha_1 = 0$ and do the same thing if $\beta_n = 0$.

**Step 2:** Compute and simplify $\alpha_i, i = 2, 3, \ldots, n$, using (2.1) and set $\alpha_i = x$ whenever $\alpha_i = 0$ for any $i \in \{2, 3, \ldots, n\}$.

**Step 3:** If $\alpha_n = 0$, then OUTPUT (‘no inverse exists’); STOP.

**Step 4:** Set $\beta_1 = \alpha_n$, then compute and simplify $\beta_i, i = n - 1, n - 2, \ldots, 2$, using (2.2) setting $\beta_i = x$ whenever $\beta_i = 0$ for any $i \in \{2, 3, \ldots, n-1\}$.

**Step 5:** Compute and simplify $u_{11}, u_{nn}$ using (2.3) and (2.4) respectively. For $i = 2, 3, \ldots, n - 1$, compute and simplify $u_{ii}$ using (2.5). For $i \neq j$, compute and simplify the remaining elements of the matrix $T^{-1}$ using (2.6).

**Step 6:** Substitute $x = 0$ in all expressions of the elements $u_{ij}, i, j = 1, 2, \ldots, n$, to obtain the actual values of these elements.

**Step 7:** OUTPUT the inverse matrix $T^{-1} = (u_{ij})_{1 \leq i, j \leq n}$.

The symbolic Algorithm 2.2 will be referred to as the HMEKGTI algorithm.

3. An illustrative example

In this section we are going to give an illustrative example.
Example 3.1. Consider the $4 \times 4$ matrix $T$ given by:

$$
T = \begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & -1 & 0 \\
0 & -1 & 2 & p \\
0 & 0 & 0 & q
\end{bmatrix}.
$$

For this matrix we have $\det T = -q$. Thus the matrix $T$ is nonsingular as long as $q$ is not equal to zero. If this is the case, then the numeric algorithm $\text{HMGTI}$ fails since $a_2 = 0$, also the numeric algorithm presented in [2] fails since $b_4 = 0$. The symbolic algorithm $\text{HMEKGTI}$ gives:

$$
T^{-1} = \frac{1}{q} \begin{bmatrix}
-q & 2q & \frac{-q}{2x-1} & -p \\
2q & -2q & \frac{q}{2x-1} & p \\
q & -q & -\frac{q}{2x-1} & px \\
0 & 0 & 0 & 1 \\
\end{bmatrix}_{x=0}.
$$

Acknowledgement

The authors gratefully acknowledge the referee for valuable comments and suggestions.

Appendix A. A MAPLE procedure for inverting tridiagonal matrices

Notes: The procedure is based on the following results:

$$
A_{ii} = (T^{-1})_{ii} = \frac{1}{(c_i - d_i + e_i)}, \quad i = 1, 2, \ldots, n. \tag{A.1}
$$

$$
A_{ij} = (T^{-1})_{ij} = \begin{cases} 
\frac{-b_i}{c_{i-1}}(T^{-1})_{i-1,j} & \text{if } i > j \\
\frac{c_i}{a_i}(T^{-1})_{i+1,j} & \text{if } i < j
\end{cases}, \quad i, j = 1, 2, \ldots, n. \tag{A.2}
$$

where

$$
c_1 = d_1, c_i = d_i - \frac{b_i}{c_{i-1}}a_{i-1}, \quad i = 2, 3, \ldots, n. \tag{A.3}
$$

and

$$
e_n = d_n, e_i = d_i - \frac{b_{i+1}}{e_{i+1}}a_i, \quad i = 2, 3, \ldots, n. \tag{A.4}
$$

> # A MAPLE procedure.
> # Written by Prof. Dr. Moawwad E.A.El-Mikkawy.
> # Date : 23 November , 2005.
> # To compute the inverse of a tridiagonal matrix.
> # Note that : b[1] = a[n] = 0.
> restart:
> with(linalg,vector,vectdim):
> inv_tri := proc (b::vector, d ::vector,a ::vector)
local i,j,k,n ; global T,c,e,A ;
n := vectdim (d) : c := array(1..n):
e := array (1..n) : A := array(1..n,1..n):
# Components of the vector c #
c[1]:= d[1]:
if c[1] = 0 then c[1] := x ; d[1] := x; fi:
for i from 2 to n do
    c[i] := simplify(d[i]-b[i]*a[i-1]/c[i-1]);
    if c[i] = 0 then c[i] := x; fi:
od:

# To compute T = det (A) #
T := subs(x = 0, simplify(product (c[r], r = 1..n))):
if T = 0 then
    for k to 1 do
        print(' Singular Matrix !!! ');
        break
    od:
else
    # Components of the vector e #
i := 'i':
e[n] := d[n]:
    if e[n] = 0 then e[n] := x; fi:
    for i from n-1 by -1 to 1 do
        e[i] := simplify(d[i]-b[i+1]*a[i]/e[i+1]);
        if e[i] = 0 then e[i] := x; fi:
od:

    # To compute the inverse matrix. #
    # Diagonal elements #
i := 'i':
    for i to n do
        A[i,i] := simplify(1/(c[i]-d[i]+e[i]));
        if A[i,i] = 0 then A[i,i] := x; fi:
od:

    # The left triangle elements #
i := 'i':
    for i from 2 to n do
        for j to i-1 do
            A[i,j] := simplify(-b[i]/e[i]* A[i-1,j]);
            if A[i,j] = 0 then A[i,j] := x; fi:
        od:
od:

    # The right triangle elements #
i := 'i': j := 'j':
    for j from 2 to n do
        for i from j-1 by -1 to 1 do
            A[i,j] := simplify(-a[i]/c[i]* A[i+1,j]);
            if A[i,j] = 0 then A[i,j] := x; fi:
        od:
od:
    fi:
eval (A):
end :

> # Call no.1 for the procedure inv_tri.#.
>x := 'x':
>b := vector([0,1,2,1]);
    b := [0,1,2,1]
\texttt{d := vector([1,1,1,0]);
\hspace{1cm}d := [1,1,1,0]}

\texttt{a := vector([1,2,1,0]);
\hspace{1cm}a := [1,2,1,0]}

\texttt{A := inv\_tri(b,d,a);
\hspace{1cm}Singular Matrix !!!
\hspace{1cm}A := array(1..4,1..4,[])}

\texttt{T;}
\hspace{1cm}0

\texttt{op(c)};
\hspace{1cm}\begin{bmatrix} 1,x, \frac{x - 4}{x}, -\frac{x}{x - 4} \end{bmatrix}

\texttt{op(e)};
\hspace{1cm}[?1, ?2, ?3, ?4]

\texttt{# End of call no. 1 #.}
\texttt{# Call no.2 for the procedure inv\_tri.#.}
\texttt{x := 'x';}
\texttt{b := vector([0,1,2,1]);
\hspace{1cm}b := [0,1,2,1]}

\texttt{d := vector([1,1,1,2]);
\hspace{1cm}d := [1,1,1,2]}

\texttt{a := vector([1,2,1,0]);
\hspace{1cm}a := [1,2,1,0]}

\texttt{A := inv\_tri(b,d,a);}
\hspace{1cm}A := \begin{bmatrix}
\frac{7}{8} & -\frac{1}{x - 8} & \frac{4}{x - 8} & -\frac{2}{x - 8} \\
\frac{1}{8} & \frac{1}{x - 8} & -\frac{4}{x - 8} & \frac{2}{x - 8} \\
-\frac{1}{2} & -\frac{1}{x - 8} & \frac{2}{x - 8} & -\frac{x}{x - 8} \\
\frac{1}{4} & \frac{1}{x - 8} & \frac{x}{x - 8} & -\frac{x}{x - 8}
\end{bmatrix}

\texttt{T;}
\hspace{1cm}-8

\texttt{op(c)};
\hspace{1cm}\begin{bmatrix} 1,x, \frac{x - 4}{x}, \frac{x - 8}{x - 4} \end{bmatrix}
\[
\begin{align*}
\text{op(e);} & \quad \begin{bmatrix}
8 & 7 \\
\frac{1}{2} & 2
\end{bmatrix} \\
x := 0 : A := \text{map} (\text{eval}, \text{op}(A)) ; \\
A := \\
\begin{bmatrix}
\frac{7}{8} & \frac{1}{8} & -1 & 1 \\
1 & -1 & 1 & -1 \\
\frac{8}{8} & \frac{2}{8} & \frac{2}{4} \\
-1 & 1 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 1 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 2
\end{bmatrix}
\end{align*}
\]

> Check using linalg package #
> linalg[\text{inverse}](A);

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 1 & 2 & 0 \\
0 & 2 & 1 & 1 \\
0 & 0 & 1 & 2
\end{bmatrix}
\]

> End of call no. 2 #
> Call no. 3 for the procedure \text{inv\_tri}.#.

> x := \text{'x'}:
> b := \text{vector}([0,1,4]);
> b := [0,1,1,1,1]
> d := \text{vector}([2,5]);
> d := [2,2,2,2,2]
> a := \text{vector}([1,4,0]);
> a := [1,1,1,1,0]
> A := \text{inv\_tri} (b,d,a);

\[
A := \\
\begin{bmatrix}
5 & -2 & 1 & -1 & 1 \\
\frac{6}{3} & \frac{3}{2} & \frac{2}{3} & \frac{3}{6} \\
-2 & 4 & -1 & \frac{2}{3} & -1 \\
3 & 3 & -1 & \frac{3}{3} & \frac{3}{3} \\
1 & \frac{7}{2} & -1 & \frac{3}{2} & -1 \\
\frac{1}{2} & \frac{2}{2} & \frac{4}{2} & -2 \\
-1 & 2 & -1 & \frac{4}{3} & \frac{4}{3} \\
\frac{3}{3} & \frac{3}{3} & \frac{3}{3} & \frac{3}{3} \\
1 & -1 & 1 & \frac{3}{2} & -2 \\
\frac{1}{6} & \frac{3}{3} & \frac{2}{3} & \frac{3}{6}
\end{bmatrix}
\]

>T;

6

> op(c);
\[
\begin{bmatrix}
2, \frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5}
\end{bmatrix}
\]
\[ \begin{bmatrix} 6 & 5 & 4 & 3 \\ 5 & 4 & 3 & 2 \end{bmatrix} \]

\[ A := \begin{bmatrix} 5 & \frac{-2}{6} & \frac{1}{3} & \frac{-1}{2} & \frac{1}{6} \\ \frac{-2}{3} & \frac{4}{2} & \frac{-1}{3} & \frac{1}{2} & \frac{-1}{3} \\ \frac{3}{3} & \frac{-1}{3} & \frac{2}{1} & \frac{2}{3} & \frac{4}{3} \\ \frac{1}{2} & \frac{-3}{2} & \frac{1}{2} & \frac{2}{2} & \frac{-2}{3} \\ \frac{-1}{2} & \frac{2}{3} & \frac{1}{2} & \frac{3}{3} & \frac{5}{6} \end{bmatrix} \]

\[ \text{# Check using linalg package #} \]

\[ \text{linalg[inverse]}(A); \]

\[ A := \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \]

\[ \text{# End of call no. 3 #} \]

\[ \text{# Call no 4 for the procedure inv_tri #.} \]

\[ a := \text{vector}([-1, 1, -p, 0]); \]

\[ a := \begin{bmatrix} 1 \end{bmatrix} \]

\[ \text{A:= inv_tri } (b,d,a); \]

\[ \begin{bmatrix} -1 & \frac{-2}{2x-1} & \frac{1}{2x-1} & \frac{-1}{2x-1} & \frac{1}{2x-1} & \frac{p}{(2x-1)q} \\ 2 & \frac{1}{2x-1} & \frac{1}{2x-1} & \frac{-1}{2x-1} & \frac{1}{2x-1} & \frac{p}{(2x-1)q} \\ 1 & \frac{1}{2x-1} & \frac{-1}{2x-1} & \frac{1}{2x-1} & \frac{1}{2x-1} & \frac{-1}{(2x-1)q} \\ \frac{-1}{2x-1} & \frac{2x-1}{2} & \frac{1}{2x-1} & \frac{-1}{2x-1} & \frac{1}{2x-1} & \frac{-1}{2x-1} \end{bmatrix} \]

\[ T; \]

\[ -q \]

\[ \text{op(c)}; \]

\[ \begin{bmatrix} 1, x, \frac{2x-1}{x}, q \end{bmatrix} \]
\[
\left[ -1, \frac{1}{2}, 2, q \right]
\]

\[
A := \begin{bmatrix}
-1 & 2 & 1 & -\frac{p}{q} \\
2 & -2 & -1 & \frac{p}{q} \\
1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

> op(e);

> x := 0; A := map(eval, op(A));

> linalg[inverse](A);

> End of call no. 4 #

References