A fast and reliable algorithm for evaluating $n$th order pentadiagonal determinants

Moawwad E.A. El-Mikkawy

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

Abstract

In the current article we present a fast and reliable algorithm for evaluating $n$th order pentadiagonal determinants in linear time. It is a natural generalization of the DETGTRI algorithm [M. El-Mikkawy, A fast algorithm for evaluating $n$th order tri-diagonal determinants, J. Comput. Appl. Math. 166 (2004) 581–584]. The algorithm is suited for implementation using computer algebra systems (CAS) such as MACSYMA and MAPLE. Some illustrative examples are given.

Keywords: Pentadiagonal matrix; Determinants; Algorithm; Computer algebra systems (CAS)

1. Introduction and preliminaries

We begin this section by considering the general tri-diagonal determinant $T$ of order $n$ of the form:

$$T = \begin{vmatrix} d_1 & a_1 & 0 & \cdots & \cdots & 0 \\ b_2 & d_2 & a_2 & \ddots & \vdots \\ 0 & b_3 & d_3 & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \vdots & 0 & \cdots & \cdots & a_{n-1} \\ 0 & \cdots & 0 & b_n & d_n \end{vmatrix} \quad (1.1)$$

These types of determinants and their corresponding matrices play a fundamental role in many areas of science and engineering. In [1], the author derived an efficient and reliable algorithm called DETGTRI to compute $n$th order tri-diagonal determinants of the form (1.1) in linear time. Based on the DETGTRI algorithm we see that the determinant $T$ in (1.1) is also given by
provided that \( p_{i+1}q_i = b_{i+1}a_i \), \( i = 1, 2, \ldots, n - 1 \). As a direct consequence, we see that, by choosing \( p_{i+1} = 1 \) and \( q_i = b_{i+1}a_i \), \( i = 1, 2, \ldots, n - 1 \), then any \( n \)th order general tri-diagonal determinant of the form (1.1) can be stored in \( 2n - 1 \) memory locations using only two vectors. These vectors are \( q = (q_1, q_2, \ldots, q_{n-1}) \) and \( d = (d_1, d_2, \ldots, d_n) \). This is always a good habit in computation in order to save memory space. In programming, the vector \( d = (d_1, d_2, \ldots, d_n) \) can be reused in order to store the components of the vector \( c = (c_1, c_2, \ldots, c_n) \) which are now given by (see [1])

\[
c_i = \begin{cases} 
  d_1, & \text{if } i = 1, \\
  d_i - \frac{q_{i-1}}{p_{i-1}}, & \text{if } i = 2, 3, \ldots, n.
\end{cases}
\]  

(1.3)

The form (1.2) is very useful in many situations. For example, this form enables us to represent the Fibonacci numbers \( F_n \) in the forms:

\[
F_n = \begin{bmatrix} 
1 & -1 & 0 & \cdots & 0 \\
1 & 1 & -1 & \ddots & \vdots \\
0 & 1 & 1 & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & -1 \\
0 & \cdots & 0 & 1 & 1
\end{bmatrix} = \begin{bmatrix} 
1 & 1 & 0 & \cdots & 0 \\
1 & 1 & 1 & \ddots & \vdots \\
0 & 1 & 1 & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & 1 & 1 \\
0 & \cdots & 0 & -1 & 1
\end{bmatrix}, \quad n = 1, 2, \ldots
\]

(1.4)

instead of the form

\[
F_n = \begin{bmatrix} 
1 & 2 & 0 & \cdots & 0 \\
-\frac{1}{2} & 1 & 3 & \ddots & \vdots \\
0 & -\frac{1}{3} & 1 & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & n \\
0 & \cdots & 0 & -\frac{1}{n} & 1
\end{bmatrix}, \quad n = 1, 2, \ldots
\]

(1.5)

which is given in [7].

The transformed determinant of the form (1.2) can also be used to prove some facts directly. For example, it can be used to prove that the following \( n \)th order determinant:
does not depend on $x$ (see also [8, p. 105]).
Perhaps more interesting is the fact that by using the form (1.2), we may now formulate the following modified DETGTRI algorithm which is more economical than the DETGTRI.

- The Modified DETGTRI Algorithm

To compute the $n$th order tri-diagonal determinant of the form (1.1), we first transform it to the form (1.2) and then proceed as follows:

**Step 1:** Use the recurrence relation

$$
c_k = \begin{cases} 
  d_1, & \text{if } k = 1, \\
  d_k - \frac{q_{k-1}}{c_{k-1}}, & \text{if } k = 2, 3, \ldots, n 
\end{cases} \quad (1.7)
$$

to compute the simplest rational forms of the $n$ components of the vector $c = (c_1, c_2, \ldots, c_n)$. If $c_k = 0$ for any $k \leq n$, set $c_k = \mu$ ($\mu$ is just a symbolic name) and continue to compute $c_{k+1}, c_{k+2}, \ldots, c_n$ in terms of $\mu$ by using (1.7).

**Step 2:** The simplest rational form of the product $\prod_{r=1}^{n} c_r$ (this product is a polynomial in $\mu$) evaluated at $\mu = 0$ is equal to the determinant $T$ in (1.1).

On the other hand, pentadiagonal linear systems of the form

$$
\begin{bmatrix}
d_1 & a_1^* & 0 & 0 & \cdots & 0 \\
b_2 & d_2 & a_2^* & 0 & \cdots & 0 \\
b_3^* & b_3 & d_3 & a_3^* & \cdots & \vdots \\
0 & b_4^* & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & b_{n-1} & b_{n-1}^* & d_{n-1} & a_{n-1} \\
0 & \cdots & 0 & b_n^* & b_n & d_n
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_{n-1} \\
x_n
\end{bmatrix}
= 
\begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
\vdots \\
r_{n-1} \\
r_n
\end{bmatrix}
\quad (1.8)
$$

frequently appear in the solution of fourth order boundary value problems and elsewhere. For such systems it is generally recommended to check the non-singularity of the pentadiagonal coefficient matrix before we solve the system. For example, to make sure that a solution of the system (1.8) exists and is unique. Therefore, the motivation of the current paper is indeed devoted to develop a fast computational algorithm for computing the $n$th order pentadiagonal determinants. Some authors actually developed fast numerical algorithms for this purpose. For example, Sweet [6], Evans [2] and the most recent work by Sogabe [4,5]. All these algorithms compute pentadiagonal determinants in linear time. The interested reader may refer to Sogabe [4,5] for more details.

In the next section we are going to derive a new computational symbolic algorithm to compute $n$th order pentadiagonal determinants also in linear time. However, an important advantage of the new algorithm is that the new algorithm will never fail. An algorithm for computing special pentadiagonal Toeplitz determinants is given in Section 3. In Section 4, two examples are given for the sake of illustration.

2. A new computational algorithm

Throughout this section and the next sections, the parameter $\lambda$ is just a symbolic name and $\det P$ is the determinant of the pentadiagonal coefficient matrix of the system (1.8).
It is now time to formulate our first result:

Algorithm 2.1. To compute $\det P$ we may proceed as follows:

1. Set $c_1 = d_1$. If $c_1 = 0$ then $c_1 = \lambda$ end if.
2. Compute and simplify $c_2 = d_2 - \frac{b_2}{c_1}a_1$. If $c_2 = 0$ then $c_2 = \lambda$ end if.
3. Set $e_1 = a_1$ and $f_2 = \frac{b_2}{c_1}$ then compute and simplify.
   For $k$ from 3 to $n$ do:
   $$g_k = \frac{b_k^2}{c_{k-2}}$$
   $$e_{k-1} = a_{k-1} - f_{k-1}a_{k-2}$$
   $$f_k = \frac{(b_k - g_ke_{k-2})}{c_{k-1}}$$
   $$c_k = d_k - f_ke_{k-1} - g_ka_{k-2}$$
   If $c_k = 0$ then $c_k = \lambda$ end if.
   End do.
4. Compute $\det P = (\prod_{r=1}^n c_r)|_{\lambda=0}$.

The algorithm will be referred to as DETGPENTA. The cost of the this algorithm is $O(n)$ only. The algorithm is easy to implement using computer algebra systems (CAS) such as MACSYMA and MAPLE. It is worth mentioned that the DETGTRI algorithm in [1] and the DETPT algorithm in [3] are now special cases of the DETGPENTA algorithm.

3. A special case

In this section we shall be concerned with the special pentadiagonal Toeplitz matrix $H_n(x, \beta, \gamma)$ and the special tri-diagonal Toeplitz matrix $T_n(x, \beta, \gamma)$ given by (see [7]):

$$H_n(x, \beta, \gamma) = \begin{bmatrix} x & 0 & 0 & 0 & \cdots & 0 \\ 0 & x & 0 & 0 & \cdots & 0 \\ \gamma & 0 & x & 0 & \beta & \ddots \\ 0 & \gamma & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & \ddots & 0 & x & 0 \\ \vdots & \ddots & \gamma & 0 & x & 0 \\ 0 & \cdots & 0 & 0 & \gamma & 0 & x \\ \end{bmatrix}$$

and

$$T_n(x, \beta, \gamma) = \begin{bmatrix} x & \beta & 0 & 0 & \cdots & 0 \\ \gamma & x & \beta & 0 & 0 & \cdots \\ 0 & \gamma & x & \beta & \ddots & \vdots \\ 0 & 0 & \gamma & x & \beta & \ddots \\ \vdots & \ddots & \gamma & x & \beta & \ddots \\ 0 & \cdots & 0 & 0 & \gamma & x \\ \end{bmatrix}$$
We are now ready to give our second result. It is a simple and economical computational algorithm for computing the \(n\)th order pentadiagonal Toeplitz determinants \(\det H_n(x, \beta, \gamma)\) given by (3.1).

**Algorithm 3.1.** The DETGPENTA algorithm enables us to compute \(\det H_n(x, \beta, \gamma)\) as follows:

**Step 1:** Set \(c_1 = x\). If \(c_1 = 0\) then \(c_1 = \lambda\) end if.

**Step 2:** Compute and simplify

\[
c_k = x - \frac{\beta \gamma}{c_{k-1}}
\]

If \(c_k = 0\) then \(c_k = \lambda\) end if.

**End do.**

**Step 3:** Compute \(\det H_{2n}(x, \beta, \gamma) = (c_1 c_2 \ldots c_n)^2 \mid_{x=0} = [\det T_n(x, \beta, \gamma)]^2\) and

\[
\det H_{2n-1}(x, \beta, \gamma) = (c_1 c_2 \ldots c_{n-1})^2 \mid_{x=0} = [\det T_n(x, \beta, \gamma)][\det T_{n-1}(x, \beta, \gamma)].
\]

The algorithm will be referred to as DETGSPENTA.

We finish this section by considering the second order difference equation:

\[
u_k = x u_{k-1} - \beta \gamma u_{k-2}, \quad k \geq 1
\]

with the initial conditions: \(u_{-1} = 0, u_0 = 1\).

Letting \(\delta = \sqrt{x^2 - 4\beta \gamma}\) and solving the difference equation (3.3), taking into account the fact \(u_n = \det T_n(x, \beta, \gamma)\), we have the two following cases:

**Case (i):** \(\delta \neq 0\).

In this case we obtain

\[
\det H_{2n}(x, \beta, \gamma) = \frac{1}{\delta^2} \left[ (\frac{x + \delta}{2})^{n+1} - (\frac{x - \delta}{2})^{n+1} \right] \tag{3.4}
\]

and

\[
\det H_{2n-1}(x, \beta, \gamma) = \frac{1}{\delta^2} \left[ \left( \frac{x + \delta}{2} \right)^{n+1} - \left( \frac{x - \delta}{2} \right)^{n+1} \right] \left[ \left( \frac{x + \delta}{2} \right)^n - \left( \frac{x - \delta}{2} \right)^n \right]. \tag{3.5}
\]

**Case (ii):** \(\delta = 0\).

For this case we get

\[
\det H_{2n}(x, \beta, \gamma) = (n + 1)^2 \left( \frac{x}{2} \right)^{2n} \tag{3.6}
\]

and

\[
\det H_{2n-1}(x, \beta, \gamma) = n(n+1) \left( \frac{x}{2} \right)^{2n-1} \tag{3.7}
\]

showing that if \(\delta = 0\), then \(\det H_m(x, \beta, \gamma)\) depends only on \(m\) and \(x\) for all \(m\).

### 4. Illustrative examples

In this section we are going to give two illustrative examples.

**Example 4.1.** For the matrix \(A\) given by

\[
A = \begin{bmatrix}
3 & 1 & 1 & 0 & 0 & 0 \\
1 & 3 & 1 & 1 & 0 & 0 \\
1 & 1 & 3 & 1 & 1 & 0 \\
0 & 1 & 1 & 3 & 1 & 1 \\
0 & 0 & 1 & 1 & 3 & 1 \\
0 & 0 & 0 & 1 & 1 & 3
\end{bmatrix},
\]

we have
\[
(f_2, f_3, f_4, f_5, f_6) = \begin{pmatrix}
1 & 1 & 3 & 7 & 34 \\
\frac{3}{4} & \frac{10}{24} & \frac{115}{115}
\end{pmatrix},
\]
\[
(g_3, g_4, g_5, g_6) = \begin{pmatrix}
1 & 3 & 2 & 5 \\
\frac{3}{8} & \frac{5}{12}
\end{pmatrix},
\]
\[
(e_1, e_2, e_3, e_4, e_5) = \begin{pmatrix}
1 & 2 & 3 & 7 & 17 \\
\frac{3}{4} & \frac{10}{24}
\end{pmatrix},
\]
\[
(c_1, c_2, c_3, c_4, c_5, c_6) = \begin{pmatrix}
3 & 8 & 12 & 115 & 273 \\
\frac{3}{2} & \frac{5}{48} & \frac{115}{115}
\end{pmatrix}.
\]

Therefore, \( \det A = \left( \prod_{r=1}^{6} c_r \right) \bigg|_{r=0} = (273) \bigg|_{r=0} = 273. \)

**Example 4.2.** For the matrix \( B \) given by
\[
B = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 2 & 0 \\
1 & 1 & 3 & 2 & -1 \\
0 & 2 & -1 & 4 & -1 \\
0 & 0 & 1 & 1 & 5
\end{bmatrix},
\]
we have
\[
(f_2, f_3, f_4, f_5) = \begin{pmatrix}
1 & 0 & -\frac{1}{2} & 0 \\
0 & \frac{1}{2}
\end{pmatrix},
\]
\[
(g_3, g_4, g_5) = \begin{pmatrix}
1 & 2 & 1 \\
\frac{1}{2}
\end{pmatrix},
\]
\[
(e_1, e_2, e_3, e_4) = \begin{pmatrix}
1 & 0 & 2 & -\frac{3}{2} \\
1 & \frac{3}{2}
\end{pmatrix},
\]
\[
(c_1, c_2, c_3, c_4, c_5) = \begin{pmatrix}
1 & \lambda & 2 & \frac{5\lambda - 4}{\lambda} & \frac{11}{2} \\
\lambda & \frac{11}{2}
\end{pmatrix}.
\]

Consequently, \( \det B = \left( \prod_{r=1}^{5} c_r \right) \bigg|_{r=0} = (55\lambda - 44) \bigg|_{r=0} = -44. \)

**References**


