Symbolic algorithm for inverting cyclic pentadiagonal matrices recursively — Derivation and implementation
Moawwad El-Mikkawy, El-Desouky Rahmo *
Mathematics Department, Faculty of Science, Mansoura University, Mansoura, 35516, Egypt

Abstract
In this paper, by using parallel computing along with recursion, we describe a reliable symbolic computational algorithm for inverting cyclic pentadiagonal matrices. The algorithm is implemented in MAPLE. Two other symbolic algorithms are developed and the computational costs for all algorithms are given. An example is presented for the sake of illustration.

1. Introduction and objectives
Cyclic pentadiagonal linear systems frequently appear in science and engineering applications (see for instance, [1–7]). The $n \times n$ general cyclic pentadiagonal linear system takes the form:

$$Px = y,$$

where

$$P = \begin{bmatrix}
d_1 & a_1 & A_1 & 0 & \cdots & 0 & 0 & B_1 & b_1 \\
b_2 & d_2 & a_2 & A_2 & 0 & \cdots & 0 & 0 & B_2 \\
B_3 & b_3 & d_3 & a_3 & A_3 & 0 & \cdots & 0 & 0 \\
0 & B_4 & b_4 & d_4 & a_4 & A_4 & \ddots & \vdots & 0 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \cdots & 0 & B_{n-2} & b_{n-2} & d_{n-2} & a_{n-2} & A_{n-2} \\
A_{n-1} & 0 & \cdots & 0 & B_{n-1} & b_{n-1} & d_{n-1} & a_{n-1} & A_{n-1} \\
a_n & A_n & a_n & 0 & \cdots & 0 & B_n & b_n & d_n
\end{bmatrix},$$

$x = (x_1, x_2, \ldots, x_n)^T$, $y = (y_1, y_2, \ldots, y_n)^T$ and $n \geq 6$.

Matrix inversion and LU factorization are two principal tools of various areas of applied mathematics, physics, engineering, statistics and computer science. In many of these areas inversion of the general cyclic pentadiagonal matrix $P$ of the form (1.2) is required. The matrix $P$ in (1.2) can be stored in exactly $5n$ memory locations by using five $n$-dimensional vectors since the matrix $P$ is sparse and there is no need to store the zero elements. The computation of the inverse of banded
The elements in the matrices \( L \) and \( U \) in (2.2) and (2.3) satisfy (see also [5]):

\[
c_i = \begin{cases} 
  d_1 & \text{if } i = 1 \\
  d_2 - f_2 e_1 & \text{if } i = 2 \\
  d_i - \frac{B_i}{c_i-2} A_{i-2} - f_i e_{i-1} & \text{if } i = 3, 4, \ldots, n - 2 \\
  d_{n-1} - \sum_{l=1}^{n-1} k_l w_l & \text{if } i = n - 1 \\
  d_n - \sum_{l=1}^{n} h_l v_l & \text{if } i = n,
\end{cases}
\]

The paper is organized as follows. In Section 2, new symbolic computational algorithms, that will not break, is constructed. In Section 3, an illustrative example is given. Conclusions of the work are given in Section 4.
It should be mentioned that a generalized Doolittle LU factorization for the matrix $P$ of the form (1.2) always exists even if the matrix $P$ is singular. In fact, the generalized Doolittle LU factorization depends on at most one formal parameter which can be treated as a symbolic name whose actual value is 0 as we shall see later.

Remark 2.1. It should be mentioned that a generalized Doolittle LU factorization for the matrix $P$ of the form (1.2) always exists even if the matrix $P$ is singular. In fact, the generalized Doolittle LU factorization depends on at most one formal parameter which can be treated as a symbolic name whose actual value is 0 as we shall see later.

At this point it is convenient to formulate our first result. It is a symbolic algorithm for computing the determinant of a cyclic pentadiagonal matrix $P$ of the form (1.2) and can be considered as natural generalizations of the symbolic algorithms DETGTRI and DETGPENTA in [11] and [12], respectively.

We also have:

$$\det P = \prod_{i=1}^{n} c_i.$$  \hspace{1cm} (2.11)
**Algorithm 2.1.** To compute \( \det P \) for the cyclic pentadiagonal matrix \( P \) in (1.2), we may proceed as follows:

Step 1: Set \( c_1 = d_1 \). If \( c_1 = 0 \) then \( c_1 = t \) end if. Set \( f_2 = \frac{b_2}{c_1}, e_1 = a_1, k_1 = \frac{\Lambda_{n-3}}{c_1}, v_1 = b_1, w_1 = B_1, h_1 = \frac{a_n}{c_1}, c_2 = d_2 - f_2 e_1 \).

If \( c_2 = 0 \) then \( c_2 = t \) end if. Set \( k_2 = \frac{-k_1 c_1}{c_2}, v_2 = B_2 - f_2 v_1, w_2 = -f_2 w_1, h_2 = \frac{\Lambda_a - h_1 e_1}{c_2}, e_2 = a_2 - f_2 A_1 \).

Step 2: Compute and simplify:

For \( i \) from 3 to \( n - 3 \) do

\[
f_i = \frac{1}{c_{i-1}} \left( b_i - \frac{B_i}{c_{i-2}} e_{i-2} \right)
\]

\[
e_i = a_i - f_i A_{i-1}
\]

\[
c_i = d_i - \frac{B_i}{c_{i-2}} A_{i-2} - f_i e_{i-1}
\]

If \( c_i = 0 \) then \( c_i = t \) end if.

End do.

Set \( f_{n-2} = \frac{1}{c_{n-3}} \left( b_{n-2} - \frac{B_{n-2}}{c_{n-4}} e_{n-4} \right), c_{n-2} = d_{n-2} - \frac{B_{n-2}}{c_{n-4}} A_{n-4} - f_{n-2} e_{n-3} \).

If \( c_{n-2} = 0 \) then \( c_{n-2} = t \) end if.

Step 3: Compute and simplify:

For \( i \) from 3 to \( n - 4 \) do

\[
k_i = -\frac{k_{i-2} A_{i-2} + k_{i-1} e_{i-1}}{c_i}
\]

\[
w_i = -\frac{B_i}{c_{i-2}} w_{i-2} - f_i w_{i-1}
\]

End do.

Set

\[
k_{n-3} = \frac{B_{n-1} - k_{n-3} A_{n-5} - k_{n-4} e_{n-4}}{c_3},
\]

\[
k_{n-2} = \frac{b_{n-1} - k_{n-4} A_{n-4} - k_{n-3} e_{n-3}}{c_{n-2}},
\]

\[
w_{n-3} = A_{n-3} - \frac{B_{n-3}}{c_{n-5}} w_{n-5} - f_{n-3} w_{n-4},
\]

\[
w_{n-2} = a_{n-2} - \frac{B_{n-2}}{c_{n-4}} w_{n-4} - f_{n-2} w_{n-3},
\]

\[
c_{n-1} = d_{n-1} - \sum_{i=1}^{n-2} k_i w_i.
\]

If \( c_{n-1} = 0 \) then \( c_{n-1} = t \) end if.

Step 4: Compute and simplify:

For \( i \) from 3 to \( n - 3 \) do

\[
v_i = -\frac{B_i}{c_{i-2}} v_{i-2} - f_i v_{i-1}
\]

\[
h_i = -\frac{h_{i-2} A_{i-2} + h_{i-1} e_{i-1}}{c_i}
\]

End do.

Set

\[
v_{n-2} = A_{n-2} - \frac{B_{n-2}}{c_{n-4}} v_{n-4} - f_{n-2} v_{n-3},
\]

\[
v_{n-1} = a_{n-1} - \sum_{i=1}^{n-2} k_i v_i,
\]

\[
h_{n-2} = \frac{B_n - h_{n-4} A_{n-4} - h_{n-3} e_{n-3}}{c_{n-2}},
\]

\[
h_{n-1} = \frac{1}{c_{n-1}} \left( b_n - \sum_{i=1}^{n-2} h_i w_i \right)
\]

\[
c_n = d_n - \sum_{i=1}^{n-1} h_i v_i.
\]

If \( c_n = 0 \) then \( c_n = t \) end if.

Step 5: Compute \( \det P = (\prod_{i=1}^{n} c_i)_{i=0} \).

The symbolic Algorithm 2.1 will be referred to as \textsc{DETPENTA}. The computational cost of this algorithm is \( 36n - 102 \) operations. The new algorithm \textsc{DETPENTA} is very useful to check the nonsingularity of the matrix \( P \) when we consider, for example, the solution of the cyclic pentadiagonal linear systems of the form (1.1).
Now, we are ready to construct the recursive algorithm for inverting nonsingular cyclic pentadiagonal matrices of the form (1.2). Assume that the matrix $P$ in (1.2) is nonsingular and let

$$P^{-1} = (S_{ij})_{1 \leq i, j \leq n} = (C_1, C_2, \ldots, C_n)$$  (2.12)

where $C_i$ denotes the $r$th column of $P^{-1}$, $r = 1, 2, \ldots, n$.

Since the Doolittle LU factorization of the matrix $P$ in (1.2) is always possible then we can use parallel computations to get the elements of the last four columns $C_j = (S_{1j}, S_{2j}, \ldots, S_{nj})^T$, $j = n, n - 1, n - 2$ and $n - 3$ of $P^{-1}$ as follows:

Solving in parallel the standard linear systems whose coefficient matrix $L$ is given by (2.2)

$$L = \begin{bmatrix}
    z_1^{(n)} & z_1^{(n-1)} & z_1^{(n-2)} & z_1^{(n-3)} \\
    z_2^{(n)} & z_2^{(n-1)} & z_2^{(n-2)} & z_2^{(n-3)} \\
    \vdots & \vdots & \vdots & \vdots \\
    z_n^{(n)} & z_n^{(n-1)} & z_n^{(n-2)} & z_n^{(n-3)}
\end{bmatrix} = \begin{bmatrix}
    0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 \\
    \vdots & \vdots & \vdots & \vdots \\
    0 & 0 & 1 & 1 \\
    0 & 1 & -k_{n-2} & -f_{n-2} \\
    1 & -h_{n-1} & h_{n-1}k_{n-2} - h_{n-2} & h_{n-2} - h_{n-3} - h_{n-1}(k_{n-2}f_{n-2} - k_{n-3})
\end{bmatrix}.$$

(2.13)

we get

$$L = \begin{bmatrix}
    z_1^{(n)} & z_1^{(n-1)} & z_1^{(n-2)} & z_1^{(n-3)} \\
    z_2^{(n)} & z_2^{(n-1)} & z_2^{(n-2)} & z_2^{(n-3)} \\
    \vdots & \vdots & \vdots & \vdots \\
    z_n^{(n)} & z_n^{(n-1)} & z_n^{(n-2)} & z_n^{(n-3)}
\end{bmatrix} = \begin{bmatrix}
    0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 \\
    \vdots & \vdots & \vdots & \vdots \\
    0 & 0 & 1 & 1 \\
    0 & 1 & -k_{n-2} & -f_{n-2} \\
    1 & -h_{n-1} & h_{n-1}k_{n-2} - h_{n-2} & h_{n-2} - h_{n-3} - h_{n-1}(k_{n-2}f_{n-2} - k_{n-3})
\end{bmatrix}.$$

(2.14)

Hence, solving the following standard linear systems whose coefficient matrix $U$ is given by (2.3)

$$U = \begin{bmatrix}
    S_{1,n} & S_{1,n-1} & S_{1,n-2} & S_{1,n-3} \\
    S_{2,n} & S_{2,n-1} & S_{2,n-2} & S_{2,n-3} \\
    \vdots & \vdots & \vdots & \vdots \\
    S_{n,n} & S_{n,n-1} & S_{n,n-2} & S_{n,n-3}
\end{bmatrix} = \begin{bmatrix}
    z_1^{(n)} & z_1^{(n-1)} & z_1^{(n-2)} & z_1^{(n-3)} \\
    z_2^{(n)} & z_2^{(n-1)} & z_2^{(n-2)} & z_2^{(n-3)} \\
    \vdots & \vdots & \vdots & \vdots \\
    z_n^{(n)} & z_n^{(n-1)} & z_n^{(n-2)} & z_n^{(n-3)}
\end{bmatrix}.$$

(2.15)

gives the four columns $C_j, j = n, n - 1, n - 2$ and $n - 3$ in the forms:

$$S_{n,n} = \frac{1}{c_n},$$

$$S_{n-1,n} = -\frac{v_{n-1}}{c_{n-1}} S_{n,n},$$

$$S_{n,n-1} = -\frac{h_{n-1}}{c_n},$$

(2.16)  (2.17)  (2.18)
\[
S_{n-1,n-1} = \frac{1}{c_{n-1}} (1 - v_{n-1} S_{n,n-1}) ,
\]
(2.19)
\[
S_{n,n-2} = \frac{1}{c_n} (h_{n-1} k_{n-2} - h_{n-2}) ,
\]
(2.20)
\[
S_{n-1,n-2} = -\frac{1}{c_{n-1}} (k_{n-2} + v_{n-1} S_{n,n-2}) ,
\]
(2.21)
\[
S_{n-2,n-2} = \frac{1}{c_{n-2}} (1 - w_{n-2} S_{n-1,n-2} - v_{n-2} S_{n,n-2}) ,
\]
(2.22)
\[
S_{n,n-3} = \frac{1}{c_n} (h_{n-2} f_{n-2} - h_{n-3} - h_{n-1} (k_{n-2} f_{n-2} - k_{n-3})) ,
\]
(2.23)
\[
S_{n-1,n-3} = \frac{1}{c_{n-1}} (k_{n-2} f_{n-2} - k_{n-3} - v_{n-1} S_{n,n-3}) ,
\]
(2.24)
\[
S_{n-2,n-3} = -\frac{1}{c_{n-2}} (f_{n-2} + w_{n-2} S_{n-1,n-3} + v_{n-2} S_{n,n-3}) ,
\]
(2.25)
\[
S_{n-3,n-3} = \frac{1}{c_{n-3}} (1 - e_{n-3} S_{n-2,n-3} - w_{n-3} S_{n-1,n-3} - v_{n-3} S_{n,n-3}) ,
\]
(2.26)
\[
S_{n-2,j} = -\frac{1}{c_{n-2}} (w_{n-2} S_{n-1,j} + v_{n-2} S_{n,j}) , \quad j = n, n-1
\]
(2.27)
\[
S_{n-3,j} = -\frac{1}{c_{n-3}} (e_{n-3} S_{n-2,j} + w_{n-3} S_{n-1,j} + v_{n-3} S_{n,j}) , \quad j = n, n-1, n-2
\]
(2.28)
\[
S_{ij} = -\frac{1}{c_i} (e_i S_{i+1,j} + a_i S_{i+2,j} + w_i S_{n-1,j} + v_i S_{n,j}) , \quad j = n, n-1, n-2, n-3, i = n-4, n-5, \ldots, 1
\]
(2.29)

Using (2.16)–(2.29) together with the fact that \( P^{-1} P = I_n \), where \( I_n \) is the \( n \times n \) identity matrix, elements in the remaining \((n - 4)\) columns of \( P^{-1} \) may be obtained recursively using:
\[
C_j = \frac{1}{A_j} (E_{i+2} - a_{j+1} C_{j+1} - d_{j+2} C_{j+2} - b_{j+3} C_{j+3} - B_{j+4} C_{j+4}) , \quad j = n-4, n-5, \ldots, 1
\]
(2.30)

**Remark 2.2.** Eqs. (2.30) suggest an additional assumption \( \prod_{i=1}^{n-4} A_i \neq 0 \), which is only formal and can be omitted by introducing auxiliary parameter \( t \) in Algorithm 2.2 given below.

Now we formulate a second result. It is a symbolic computational algorithm to compute the inverse of a general cyclic pentadiagonal matrix of the form (1.2) when it exists.

**Algorithm 2.2.** To find the \( n \times n \) inverse matrix of the general cyclic pentadiagonal matrix \( P \) in (1.2) by using the relations (2.16)–(2.30).

**INPUT:** Order of the matrix \( n \) and the components, \( B_i, b_i, d_i, a_i, A_i, i = 1, 2, \ldots, n. \)

**OUTPUT:** Inverse matrix, \( P^{-1} = (S_{ij})_{1 \leq i, j \leq n}. \)

**Step 1:** If \( A_i = 0 \) for any \( i = 1, 2, \ldots, n-4 \), set \( A_i = t \) (\( t \) is just a symbolic name).

**Step 2:** If \( B_i = 0 \) for any \( i = 5, \ldots, n \), set \( B_i = t \).

**Step 3:** Use the DETPCPENTA algorithm to check the nonsingularity of the matrix \( P. \) If the matrix \( P \) is singular then OUTPUT (“The matrix \( P \) is singular”); Stop.

**Step 4:** For \( i = 1, 2, \ldots, n \), compute and simplify the components \( S_{i,n}, S_{i,n-1}, S_{i,n-2}, S_{i,n-3} \) of the columns \( C_j, j = n, n-1, n-2 \) and \( n-3 \), respectively, by using (2.16)–(2.29).

**Step 5:** For \( j = n-4, n-5, \ldots, 1 \), do

| For \( i = 1, 2, \ldots, n \), do |
| Compute and simplify the components \( S_{ij} \) by using (2.30). |
| End do |
| End do |

**Step 6:** Substitute the actual value \( t = 0 \) in all expressions to obtain the elements, \( S_{ij}, i, j = 1, 2, \ldots, n. \)

The symbolic Algorithm 2.2 will be referred to as CPINV algorithm. The computational cost of CPINV algorithm is \( 8n^2 + 37n - 235 \) operations. The Algorithms 2.3 and 2.2 in [10] and [13], respectively, are now special cases of the CPINV algorithm.
Concerning the positive definiteness of cyclic pentadiagonal matrices of the form \((1.2)\), we state the following result without proof (see [14]).

**Theorem 2.1.** Denote by:

\[
\begin{align*}
\begin{array}{cccccccc}
 d_1 & a_1 & A_1 & 0 & \cdots & \cdots & \cdots & 0 \\
 b_2 & d_2 & a_2 & A_2 & 0 & \cdots & \cdots & 0 \\
 B_3 & b_3 & d_3 & a_3 & A_3 & 0 & \cdots & \cdots \\
 & & & & & \cdots & \cdots & \\
 0 & & & & & & \cdots & 0 \\
 & & & & & & \cdots & \cdots \\
 u_i = & & & & & \cdots & 0 & B_{i-2} & b_{i-2} & d_{i-2} & a_{i-2} & A_{i-2} \\
 & & & & & & \cdots & 0 & B_{i-1} & b_{i-1} & d_{i-1} & a_{i-1} \\
 & & & & & 0 & & & & & & \\
 & & & & & & & & & & & \end{array}
\end{align*}
\]

for \(i = 2, 3, \ldots, n - 2\).

\[
\begin{align*}
\begin{array}{cccccccc}
 d_1 & a_1 & A_1 & 0 & \cdots & \cdots & \cdots & 0 \\
 b_2 & d_2 & a_2 & A_2 & 0 & \cdots & \cdots & 0 \\
 B_3 & b_3 & d_3 & a_3 & A_3 & 0 & \cdots & \cdots \\
 & & & & & \cdots & \cdots & \\
 0 & & & & & & \cdots & 0 \\
 & & & & & & \cdots & \cdots \\
 u_{n-1} = & & & & & \cdots & 0 & B_{n-3} & b_{n-3} & d_{n-3} & a_{n-3} & A_{n-3} \\
 & & & & & & \cdots & 0 & B_{n-2} & b_{n-2} & d_{n-2} & a_{n-2} \\
 A_{n-1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
 & & & & & & & & \end{array}
\end{align*}
\]

and

\[
\begin{align*}
\begin{array}{cccccccc}
 d_1 & a_1 & A_1 & 0 & \cdots & \cdots & \cdots & 0 \\
 b_2 & d_2 & a_2 & A_2 & 0 & \cdots & \cdots & 0 \\
 B_3 & b_3 & d_3 & a_3 & A_3 & 0 & \cdots & \cdots \\
 & & & & & \cdots & \cdots & \\
 0 & & & & & & \cdots & 0 \\
 & & & & & & \cdots & \cdots \\
 u_n = \det P = & & & & & \cdots & 0 & B_{n-2} & b_{n-2} & d_{n-2} & a_{n-2} & A_{n-2} \\
 & & & & & & \cdots & 0 & B_{n-1} & b_{n-1} & d_{n-1} & a_{n-1} \\
 A_{n-1} & A_n & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\
 & & & & & & & & \end{array}
\end{align*}
\]

Then, we have the two-term recurrence

\[ u_i = c_i u_{i-1} \text{ for } i = 1, 3, \ldots, n \text{ with the } c_i \text{ as defined in } (2.4). \]

As a direct consequence of Theorem 2.1, we see that if the cyclic pentadiagonal matrix \(P\) in \((1.2)\) is symmetric then it is positive definite if and only if \(c_i > 0, i = 1, 2, \ldots, n\).

We conclude this section by formulating a symbolic version of the numerical algorithm \texttt{NPENTA} in [5] to remove all cases where the numerical algorithm \texttt{NPENTA} fails (see the illustrative example in Section 3).

**Algorithm 2.3.** To solve the linear system \((1.1)\) with coefficient matrix given by \((1.2)\), we may proceed as follows:

Step 1: Use the symbolic algorithm \texttt{DETCPENTA} to compute \(\det P\). If \(\det P \neq 0\) then go to step 2 else Stop end if.

Step 2: Repeat steps 8–11 of the algorithm \texttt{NPENTA} in [5].

Step 3: Substitute \(t = 0\) in the expressions of the elements \(x_i, i = 1, 2, \ldots, n\).

Step 4: OUTPUT the solution \(x_i, i = 1, 2, \ldots, n\).

The Algorithm 2.3 will be referred to as \texttt{SYMBNPENTA} algorithm. The \texttt{SYMBNPENTA} algorithm will not break. It is a natural generalization of the \texttt{KPENTA} algorithm in [2]. For the implementation of the Algorithm \texttt{CPINV} in MAPLE, see the Appendix.
3. An illustrative example

In this section we give an example for the sake of illustration.

**Example 3.1.** Consider the $7 \times 7$ cyclic pentadiagonal linear system

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 2 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 2 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 2 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 2 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 2 & 2 \\
2 & 1 & 0 & 0 & 1 & 2 & 2
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
x_7
\end{bmatrix}
= \begin{bmatrix}
6 \\
4 \\
6 \\
5 \\
6 \\
7 \\
8
\end{bmatrix}.
\] (3.1)

By considering the coefficient matrix of the system (3.1) and applying the CPINV algorithm, we obtain

- $A_2 = (\text{Step } 1)$.
- $(f_2, f_3, f_4, f_5) = (e_1, e_2, e_3, e_4) = (1, 0, 1, 0)$,
- $(c_1, c_2, c_3, c_4, c_5) = (1, 1, 1, 1), (k_1, k_2, k_3, k_4, k_5) = \left(1, \frac{-1}{t}, -1, 3, 2\right)$,
- $(w_1, w_2, w_3, w_4, w_5) = (1, -1, 1, 2, 2), c_6 = -\frac{10t + 1}{t}$,
- $(v_1, v_2, v_3, v_4, v_5, v_6) = \left(2, -1, -2, 2, 3, -\frac{14t + 1}{t}\right)$,
- $(h_1, h_2, h_3, h_4, h_5, h_6) = \left(2, -\frac{1}{t}, -2, 3, 3, \frac{14t + 1}{10t + 1}\right), c_7 = -\frac{14t + 3}{10t + 1}$,
- $\det P = \left(\prod_{i=1}^{7} c_i\right)_{i=0} = (14t + 3)_{i=0} = 3$ (Step 3).

\[
p^{-1} = \frac{1}{14t + 3}
\begin{bmatrix}
-8t - 1 & -2 & 12t + 4 & 12t + 4 & -8t - 3 & -8t - 2 & 14t + 4 & -2t - 1 \\
4t - 2 & 8 & -6t - 7 & 4t + 6 & 4t + 2 & -14t - 7 & 8t + 4 \\
7t + 4 & -7 & 5 & -1 & -7t + 2 & -2 \\
-4t - 3 & 6 & 6t - 3 & -4t + 3 & -4t & 14t & -8t \\
-6t - 2 & 2 & 2t - 1 & -6t & 8t + 2 & -1 & 2t + 1 \\
7t + 4 & -7 & -14t + 2 & 2t & -1 & -2t - 1 & 14t + 1 \\
2t - 1 & 4 & 4t - 2 & -12t & 2t + 1 & 14t + 1 & -10t - 1
\end{bmatrix}_{i=0}
\] (Steps 4–5).

\[
= \frac{1}{3}
\begin{bmatrix}
-1 & -2 & 4 & -3 & -2 & 4 & -1 \\
-2 & 8 & -7 & 6 & 2 & -7 & 4 \\
4 & -7 & 5 & -3 & -1 & 2 & -2 \\
-3 & 6 & -3 & 3 & 0 & 0 & 0 \\
-2 & 2 & -1 & 0 & 2 & -1 & 1 \\
4 & -7 & 2 & 0 & -1 & -1 & 1 \\
-1 & 4 & -2 & 0 & 1 & 1 & -1
\end{bmatrix}
\] (Step 6).

The numerical algorithm NPENTA in [5] fails to solve the linear system (3.1) above although $\det P = 3 \neq 0$. Applying the SYMBNPPENTA algorithm gives:

\[
x = \left(3 \frac{2t + 1}{14t + 3}, -10t + 3, 2t + 3, \frac{2t + 3}{14t + 3}, 10t + 3, \frac{28t + 3}{14t + 3}, \frac{2t + 1}{14t + 3}\right)^T_{i=0} = (1, 1, 1, 1, 1, 1)^T.
\]

Finally, by using Theorem 2.1, we see that the symmetric matrix $P$ is not positive definite.

4. Conclusions

In this work new recursive computational algorithms have been developed for computing the determinant and inverse of general cyclic pentadiagonal matrices and for solving linear systems of cyclic pentadiagonal type. The algorithms are reliable, computationally efficient and will not fail. The algorithms are natural generalizations of some algorithms in current use.
Acknowledgements

The authors wish to thank the anonymous referees and Prof. Dr. Ervin Rodin, Editor-in-Chief, for their useful comments and suggestions.

Appendix. Implementation of the Algorithm CPINV in MAPLE

The following is a MAPLE procedure based on the CPINV algorithm
> # Cyclic Pentadiagonal matrix Inversion , n >= 6.
restart:
cp_inv := proc(n::posint,B::vector,b::vector,d::vector,a::vector,A::vector)
local i,j;
global c,Detval,detval,f,e,h,k,v,w,Q,M;

c:=array(1..n):f:=array(2..n-2):e:=array(1..n-3):h:=array(1..n-1):
k:=array(1..n-2):v:=array(1..n-1):w:=array(1..n-2):
Q:=array(1..n,1..n): M:= array(1..n,1..n):

# Step 1.
# Set A[i]= t whenever A[i]=0 , i=1,2,...,n-4#
for i to n-4 do
  if A[i]=0 then A[i]:= t fi:
od:
# Set B[i]= t whenever B[i]=0 , i=5,...,n#
i:='i':
for i from 5 to n do
  if B[i]=0 then B[i]:= t fi:
od:
# Step 2.
c[1]:= d[1]:
if c[1] = 0 then c[1]:=t fi:
if c[2]=0 then c[2]:=t fi:

# Step 3.
for i from 3 to n-3 do
  f[i] := simplify((b[i]-B[i]/c[i-2]*e[i-2])/c[i-1]):
e[i] := simplify(a[i]-f[i]*A[i-1]):
c[i] := simplify(d[i]-B[i]/c[i-2]*A[i-2]-f[i]*e[i-1]):
  if c[i]=0 then c[i]:=t fi:
od:
# Step 4.
f[n-2] := simplify((b[n-2]-B[n-2]/c[n-4]*e[n-4])/c[n-3]):
if c[n-2]=0 then c[n-2]:=t fi:
# Step 5.
i:='i':
for i from 3 to n-4 do
  k[i] := simplify((k[i-2]*A[i-2]+k[i-1]*e[i-1])/c[i]):
w[i] := simplify(B[i]/c[i-2]*w[i-2]+f[i]*w[i-1]):
od:
# Step 6.
k[n-3] := simplify((B[n-1]-k[n-1]*A[n-5]-k[n-4]*e[n-4])/c[n-3]):
k[n-2] := simplify((b[n-1]-k[n-4]*A[n-4]-k[n-3]*e[n-3])/c[n-2]):
w[n-3] := simplify(A[n-3]-B[n-3]/c[n-5]*w[n-5]-f[n-3]*w[n-4]):
w[n-2] := simplify(a[n-2]-B[n-2]/c[n-4]*w[n-4]-f[n-2]*w[n-3]):
i:='i':
c[n-1] := simplify(d[n-1]-sum(k[i]*w[i],i=1..n-2)):
if c[n-1]=0 then c[n-1]:=t fi:
# Step 7.
i:='i':
for i from 3 to n-3 do
    v[i] := -simplify(B[i]/c[i-2]*v[i-2]+f[i]*v[i-1]):
    h[i] := -simplify((h[i-2]*A[i-2]+h[i-1]*e[i-1])/c[i]):
od:
# Step 8.
v[n-1] := simplify(a[n-1]-sum(k[i]*w[i],i=1..n-2)):
h[n-2] := simplify(B[n-1]*h[n-4]*A[n-4]-h[n-3]*e[n-3])/c[n-2]):
h[n-1] := simplify((b[n-1]-sum(h[m]*w[m],m=1..n-2))/c[n-1]):
i:='i':
c[n] := simplify(d[n]-sum(h[i]*v[i],i=1..n-1)):
if c[n]=0 then c[n]:=t fi:
# Step 9.
# To compute the determinant of P.
i:='i':
Detval := simplify(product(c[i],i=1..n)):
detval := coeff (Detval,t,0):
if detval = 0 then
    ERROR (' Singular Matrix !!!!' ):
else
    # Step 10.
    # Components of the columns Cj , j= n,n-1,n-2, n-3. #
i:='i':
    Q[n,n]:= 1/c[n]:
    Q[n-1,n]:= -simplify(v[n-1]*Q[n,n]/c[n-1]):
    Q[n-2,n]:= -simplify((w[n-2]*Q[n-1,n]+v[n-2]*Q[n,n])/c[n-2]):
    Q[n,n-1] := -h[n-1]/c[n]:
    Q[n-1,n-1] := simplify((1-v[n-1]*Q[n,n-1])/c[n-1]):
    Q[n-2,n-1] := simplify((w[n-2]*Q[n-1,n-1]+v[n-2]*Q[n,n-1])/c[n-2]):
    Q[n,n-2] := simplify((-h[n-2]+h[n-1]*k[n-2])/c[n]):
    Q[n-1,n-2] := simplify((k[n-2]+v[n-2]*Q[n,n-2])/c[n-1]):
    Q[n,n-3] := simplify((-h[n-3]+h[n-2]*f[n-2]+h[n-1]*k[n-3]-
h[n-1]*k[n-2]*f[n-2])/c[n]):
    Q[n-1,n-3] := simplify((-k[n-3]+k[n-2]*f[n-2]-
v[n-1]*Q[n,n-3])/c[n-1]):
    j:='j':
for j from n by -1 to n-2 do
    Q[n-3,j] := simplify((e[n-3]*Q[n-2,j]+w[n-3]*Q[n-1,j])+
    v[n-3]*Q[n,j])/c[n-3]):
od:
Q[n-3,n-3] := simplify((1-e[n-3]*Q[n-2,n-3]-w[n-3]*Q[n-1,n-3]-
v[n-3]*Q[n,n-3])/c[n-3]):
i:='i': j:='j':
for j from n by -1 to n-3 do
    for i from n-4 by -1 to 1 do
        Q[i,j] := simplify((e[i]*Q[i+1,j]+A[i]*Q[i+2,j]+w[i]*Q[n-1,j]+v[i]*Q[n,j])/c[i]):
    od:
    od:
# Step 11.
# Components of the remaining columns, Cj for j=1,...,n-4#
i:='i':
for j from n-4 by -1 to 1 do
for i to n do
    if i = j+2 then
        Q[i,j] := simplify(1 / A[j] + Q[i,j])
    fi:
od:
fi:
# Step 12.
M := evalm(Q):
eval(evalm(M), t=0):
end proc:

References