An analytical approach: Explicit inverses of periodic tridiagonal matrices

Tim Hopkins[a], Emrah Kılıç[b,*],

[a] *Computing Laboratory University of Kent, Canterbury, Kent CT2 7NF, UK*
[b] *TOBB University of Economics and Technology, Mathematics Department, 06560 Ankara, Turkey*

Abstract

We derive an explicit formula for the inverse of a general, periodic, tridiagonal matrix. Our approach is to derive its LU factorization using backward continued fractions (BCF) which are an essential tool in number theory. We then use these formulae to construct an algorithm for inverting a general, periodic, tridiagonal matrix which we implement in Maple[1]. Finally, we present the results of testing the efficiency of our new algorithm against another published implementation and against the library procedures available within Maple to invert a general matrix and to compute its determinant.

Keywords: Matrix inversion, LU-Factorization, Inverse, Backward continued fraction

1. Introduction

Tridiagonal matrices have been frequently used in various application areas ranging from engineering to economics (see [1, 2, 3, 4, 5, 6, 7, 8]) as well as in the computation of special functions, PDEs and number theory (see [9, 10, 11, 12, 13, 14]). Various features of tridiagonal matrices are used to solve the systems of linear equations that arise from these applications (see, for example, [10, 12]) and many authors (for example, [15, 16, 17, 18, 5, 19, 20, 21, 22]) have studied various tridiagonal matrices and their properties such as LU decompositions, determinants and inverses.

Define the $n \times n$ periodic tridiagonal matrix $G(\delta, \mu)$, or briefly G, with $G_{kk} = x_k$ for $1 \le k \le n$, $G_{k+1,k} = z_k$ and $G_{k,k+1} = y_k$ for $1 \le k \le n-1$, $G_{1n} = \delta$, $G_{n1} = \mu$ and 0 otherwise. This has the form

$$G = \begin{bmatrix} x_1 & y_1 & & & \delta \\ z_1 & x_2 & y_2 & & \\ & z_2 & x_3 & \ddots & \\ & & \ddots & \ddots & y_{n-1} \\ \mu & & & z_{n-1} & x_n \end{bmatrix}. \tag{1.1}$$

Many methods have been reported for solving systems of linear equations whose coefficient matrix is periodic tridiagonal; see, for example, [23, 24, 25, 26, 27]. The efficacy of a particular algorithm is often dependent both on the type of data used to define G (floating point numbers, rational values or algebraic expressions) and the required result (inverse matrix, solution of a linear system, determinant, etc.).

For some particular forms of (1.1) simple closed form inverses are known; see, for example, [28, 17, 29, 13, 22]. However, in general, such closed forms are not available and obtaining exact inverses to (1.1) efficiently, when the elements are rational numbers or algebraic quantities, requires a symbolic algebra (SA) system like Maple [30] or Mathematica [31]. Experience with computing the inverse of (1.1) when the elements are floating point values suggests that we should be able to construct SA algorithms which are far more efficient that a general matrix inversion procedure by taking account of the special structure.

When $\delta = \mu = 0$, the periodic, tridiagonal matrix G is reduced to the general tridiagonal matrix T. Mallik [29] gives an explicit formula based on linear difference equations for the elements of T^{-1}.

[1] Maple is a trademark of Waterloo Maple Inc.
[*] Corresponding author
Email addresses: `T.R.Hopkins@kent.ac.uk` (Tim Hopkins), `ekilic@etu.edu.tr` (Emrah Kılıç)

Two kinds of LU decompositions for the matrix T may be found in [15]. Another derivation of the LU decomposition of T where the entries of L and U are expressed via backward continued fractions (BCF) is given in [32] along with explicit forms for the determinant and inverse of T also in terms of BCFs.

Recently the authors of [25] show that the elements of the inverse matrix $G^{-1} = (f_{ij})$, for $1 \leq i, j \leq n$, are given by:

$$f_{ij} = \frac{1}{\Delta} \begin{cases} (-1)^{i+j} \left[S(j,i) \prod_{k=i}^{j-1} y_k - \delta \Delta_{i+1}^{j-1} \prod_{k=j}^{n-1} z_k \prod_{k=1}^{i-1} z_k \right] & \text{if } i \leq j, \\ S(i,j) \prod_{k=j}^{i-1} z_k + (-1)^{i+j+n} \mu \Delta_{j+1}^{i-1} \prod_{k=1}^{j-1} y_k \prod_{k=i}^{n-1} y_k & \text{if } j \leq i, \end{cases}$$

where

$$S(i,j) = \Delta_1^{j-1} \Delta_{i+1}^{n} - \delta\mu \Delta_2^{j-1} \Delta_{i+1}^{n-1}$$

and

$$\Delta = x_n \Delta_1^{n-1} - \delta\mu \Delta_2^{n-1} + z_{n-1} y_{n-1} \Delta_1^{n-2} - (-1)^n \mu \prod_{k=1}^{n-1} y_k + \delta \prod_{k=1}^{n-1} z_k$$

with $\prod_{k+1}^{k} (\cdot) = 1$. The sequence of determinants $\left\{ \Delta_i^j \right\}$ is computed using recurrence relations of order two, which are given in [25, Equations (3.4)–(3.6)].

In Sections 3 and 4 we consider the *LU* decomposition of (1.1) and find explicit formulae for the elements of L and U as well as their inverses, L^{-1} and U^{-1}, in terms of BCFs. From these we are then able to generate explicit formulae for the elements of G^{-1}. A simple illustrative example of how to use these formulae to compute the inverse of an order 4 matrix with integer elements is given in Section 5. In Section 6 we look in detail at the computational implementation of both our proposed algorithm and the one presented in [25] and we show how to improve the execution speed of the latter. We also provide comparisons, using a number of test examples, of the efficiency of the two algorithms and the Maple library functions for computing the inverse and the determinant of a general matrix on a number of test problems. Finally, in Section 7, we present our conclusions.

First, we recall some basic definitions and results related to the backward continued fractions (for more details, see [32]).

2. Backward Continued Fractions – Basic Definitions and Results

A continued fraction representation of a real x is

$$x = a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{a_3 + \cfrac{}{\ddots + \cfrac{b_n}{a_{n+1}}}}}, \tag{2.1}$$

where the a_i and b_i are integers. This representations may also be written in condensed form as

$$x = a_1 + \frac{b_1}{a_2+} \frac{b_2}{a_3+} \cdots \frac{b_n}{a_{n+1}}.$$

Definition 1. The fraction

$$C_k = \left[a_1 + \frac{b_1}{a_2+} \frac{b_2}{a_3+} \cdots \frac{b_k}{a_{k+1}} \right]$$

with $k = 1, 2, \ldots,$ where $k \leq n$, is called the k^{th} convergent of (2.1).

Convergents are denoted by $C_n = P_n/Q_n$ with $C_1 = a_1$, for $n > 1$, where the recurrences $\{P_n\}$ and $\{Q_n\}$ are given by [33]:

$$\begin{aligned} P_n &= a_n P_{n-1} + b_{n-1} P_{n-2}, \quad & P_0 = 1,\ P_1 = a_1 \\ Q_n &= a_n Q_{n-1} + b_{n-1} Q_{n-2}, \quad & Q_0 = 0,\ Q_1 = 1. \end{aligned}$$

The concept of *backward continued fraction* was first used in [34] for vector valued continued fractions but in a different context.

Let the a_i and b_i be real numbers. We recall the definition of a general backward continued fraction.

Definition 2. The general backward continued fraction has the form

$$a_{n+1}+\cfrac{b_n}{a_n+\cfrac{b_{n-1}}{a_{n-1}+\cfrac{}{\ddots+\cfrac{b_1}{a_1}}}}$$

with the compressed form

$$C^B=\left[a_1+\frac{b_1}{a_2+}\frac{b_2}{a_3+}\cdots\frac{b_n}{a_{n+1}}\right]_B$$

for compactness. The k^{th} convergent is denoted by

$$C_k^B=\left[a_1+\frac{b_1}{a_2+}\frac{b_2}{a_3+}\cdots\frac{b_k}{a_{k+1}}\right]_B.$$

Convergents of a backward continued fraction are given by the following Theorem:

Theorem 1 ([32]). *Given a backward continued fraction* $A=\left[a_1+\frac{b_1}{a_2+}\frac{b_2}{a_3+}\cdots\frac{b_n}{a_{n+1}}\right]_B$. *If* $1\le k\le n$ *and* C_k^B *is the* k^{th} *backward convergent to* A, *then*

$$C_k^B=\frac{P_k}{P_{k-1}},$$

where P_k *is the* k^{th} *term of the sequence* $\{P_n\}$.

Corollary 2 ([32]). *Given a usual continued fraction* $\left[a_1+\frac{b_1}{a_2+}\frac{b_2}{a_3+}\cdots\frac{b_n}{a_{n+1}}\right]$ *and a backward continued fraction* $\left[a_1+\frac{b_1}{a_2+}\frac{b_2}{a_3+}\cdots\frac{b_n}{a_{n+1}}\right]_B$. *Then the numerators of these fractions are the same,* P_n.

3. LU Factorization of a periodic tridiagonal matrix

We give the LU-decomposition of the matrix G using backward continued fractions. Suppose we have the entries of G, that is, x_i for $1\le i\le n$, y_i and z_i for $1\le i\le n-1$, μ and δ.

Define the backward continued fraction C_n^B via elements of the matrix G as

$$C_n^B=\left[x_1+\frac{-y_1z_1}{x_2+}\frac{-y_2z_2}{x_3+}\cdots\frac{-y_{n-1}z_{n-1}}{x_n}\right]_B. \tag{3.1}$$

The first few convergents of C_n^B are

$$C_1^B=[x_1]_B=\frac{P_1}{P_0}=x_1,\quad C_2^B=\left[x_1+\frac{-y_1z_1}{x_2}\right]_B=\frac{P_2}{P_1}=\frac{x_1x_2-y_1z_1}{x_1},$$
$$C_3^B=\left[x_1+\frac{-y_1z_1}{x_2+}\frac{-y_2z_2}{x_3}\right]_B=\frac{P_3}{P_2}=\frac{x_3x_2x_1-x_3y_1z_1-x_1y_2z_2}{x_1x_2-y_1z_1},$$

where the three-term recurrence $\{P_n\}$ is defined, for $n>0$, by the rule

$$P_n=x_nP_{n-1}-y_{n-1}z_{n-1}P_{n-2}, \tag{3.2}$$

where $P_0=1$ and $P_1=x_1$.

For any sequence $\{a_n\}$, we assume that

$$a_n! = a_n a_{n-1} \dots a_1 \tag{3.3}$$

with $a_0! = 1$.

Define the $n \times n$ upper triangular matrix U as

$$U_{ij} = \begin{cases} C_i^B & i = j; 1 \le i \le n-1 \\ U_{nn} & i = j = n \\ y_i & j = i+1; 1 \le i \le n-2 \\ \delta (-1)^{i-1} z_{i-1}!/C_{i-1}^B! & j = n; 1 \le i \le n-2 \\ y_{n-1} + (-1)^n z_{n-2}!\delta/C_{n-2}^B! & i = n-1, j = n. \\ 0 & \text{otherwise} \end{cases}$$

where

$$U_{nn} = x_n - \frac{y_{n-1}z_{n-1}}{C_{n-1}^B} - \frac{(-1)^n y_{n-1}!\mu}{C_{n-1}^B!} - \frac{\delta (-1)^n z_{n-1}!}{C_{n-1}^B!} - \delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!},$$

which, with some rearrangement and (3.2), becomes

$$\begin{aligned} U_{nn} &= \frac{x_n C_{n-1}^B - y_{n-1}z_{n-1}}{C_{n-1}^B} - (-1)^n \frac{\mu y_{n-1}! + \delta z_{n-1}!}{C_{n-1}^B!} - \delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!} \\ &= C_n^B - (-1)^n \frac{y_{n-1}!\mu + \delta z_{n-1}!}{C_{n-1}^B!} - \delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!}, \end{aligned}$$

where C_i^B is the i^{th} convergent of the BCF given in (3.1) for $1 \le i \le n$.

Using (3.3) and noting that $P_0 = 1$ we have

$$C_n^B! = \frac{P_1}{P_0} \cdots \frac{P_{n-1}}{P_{n-2}} \frac{P_n}{P_{n-1}} = \frac{P_n}{P_0} = P_n. \tag{3.4}$$

Clearly

$$U = \begin{bmatrix} C_1^B & y_1 & & & & z_0!\delta/C_0^B! \\ & C_2^B & y_2 & & & -z_1!\delta/C_1^B! \\ & & C_3^B & \ddots & & \vdots \\ & & & \ddots & y_{n-2} & -(-1)^n z_{n-3}!\delta/C_{n-3}^B! \\ & & & & C_{n-1}^B & \left(y_{n-1} - (-1)^{n-1} z_{n-2}!\delta/C_{n-2}^B!\right) \\ 0 & & & & & U_{nn} \end{bmatrix}, \tag{3.5}$$

where U_{nn} is defined as before.

Define the $n \times n$ unit lower triangular matrix L as

$$L_{ij} = \begin{cases} 1 & i = j; 1 \le i \le n \\ z_{i-1}/C_{i-1}^B & j = i-1; 2 \le i \le n-1 \\ (-1)^{j+1} y_{j-1}!\mu/C_j^B! & i = n; 1 \le j \le n-2 \\ z_{n-1}/C_{n-1}^B + (-1)^n \mu y_{n-2}!/C_{n-1}^B! & i = n, j = n-1 \\ 0 & \text{otherwise} \end{cases}$$

Clearly

$$L = \begin{bmatrix} 1 & & & & & 0 \\ \frac{z_1}{C_1^B} & 1 & & & & \\ & \frac{z_2}{C_2^B} & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \frac{z_{n-2}}{C_{n-2}^B} & 1 & \\ \frac{\mu y_0!}{C_1^B!} & -\frac{\mu y_1!}{C_2^B!} & \cdots & \left\{(-1)^{n-1} \frac{\mu y_{n-3}!}{C_{n-2}^B!}\right\} & \left\{\frac{z_{n-1}}{C_{n-1}^B} + (-1)^n \frac{\mu y_{n-2}!}{C_{n-1}^B!}\right\} & 1 \end{bmatrix}, \tag{3.6}$$

where C_i^B is the i^{th} convergent of the BCF given by (3.1) for $1 \leq i \leq n$.

We can now explicitly define the LU decomposition of G with the following Theorem.

Theorem 3. *The LU decomposition of the matrix G is given by*

$$G = L \cdot U,$$

where L and U given by (3.6) and (3.5), respectively.

Proof. We consider three main cases. First, the case $i = j$ for $1 \leq i \leq n-1$. By a matrix multiplication, we have for $1 \leq i \leq n-1$

$$G_{ii} = \sum_{k=1}^{n} L_{ik}U_{ki} = L_{i,i-1}U_{i-1,i} + L_{ii}U_{ii} = y_{i-1}\frac{z_{i-1}}{C_{i-1}^B} + C_i^B,$$

which, from Theorem 1 and (3.2), gives

$$G_{ii} = x_i,$$

as claimed. For $i = j = n$, we write

$$G_{nn} = \sum_{k=1}^{n} L_{ik}U_{ki} = L_{n1}U_{1n} + L_{n2}U_{2n} + \ldots + L_{nn}U_{nn},$$

which, by the definitions of L and U, gives

$$\begin{aligned}
&L_{n1}U_{1n} + L_{n2}U_{2n} + \cdots + L_{n,n-1}U_{n-1,n} + U_{nn} \\
&= \mu\delta\frac{y_0!z_0!}{C_1^B!C_0^B!} + \cdots + \mu\delta\frac{y_{n-3}!z_{n-3}!}{C_{n-2}^B!C_{n-3}^B!} \\
&\quad + \left(\frac{z_{n-1}}{C_{n-1}^B} + (-1)^n\mu\frac{y_{n-2}!}{C_{n-1}^B!}\right)\left(y_{n-1} - (-1)^{n-1}\delta\frac{z_{n-2}!}{C_{n-2}^B!}\right) + U_{nn} \\
&= \delta\mu\sum_{k=1}^{n-2}\frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!} + \frac{y_{n-1}z_{n-1}}{C_{n-1}^B} + \mu\frac{(-1)^n y_{n-1}!}{C_{n-1}^B!} \\
&\quad - \delta\frac{(-1)^{n-1} z_{n-1}!}{C_{n-1}^B!} + \frac{\delta\mu z_{n-2}!y_{n-2}!}{C_{n-1}^B!C_{n-2}^B!} + U_{nn},
\end{aligned}$$

and hence, by the definition of U_{nn}, we have

$$G_{nn} = x_n,$$

as claimed. Consider the case $i + 1 = j$. For $1 \leq i \leq n-1$, we can write

$$G_{i,i+1} = \sum_{k=1}^{n} L_{ik}U_{k,i+1} = L_{ii}U_{i,i+1} = U_{i,i+1} = y_i.$$

Finally, we consider the case $i = j + 1$. For $1 \leq i \leq n-1$, we can write

$$G_{i+1,i} = \sum_{k=1}^{n} L_{i+1,k}U_{ki} = L_{i+1,i}U_{ii} = \frac{z_i}{C_i^B}C_i^B = z_i.$$

Also note that

$$G_{1n} = \sum_{k=1}^{n} L_{1k}U_{kn} = U_{1n} = \frac{z_0!\delta}{C_0^B!} = \delta \text{ and } G_{n1} = \sum_{k=1}^{n} L_{nk}U_{k1} = L_{n1}U_{11} = \frac{\mu y_0!}{C_1^B!}C_1^B = \mu.$$

Thus the proof is complete for all the cases.

We also have the following Corollary.

Corollary 4. *For $n > 1$,*

$$\det G = C_n^B! - (-1)^n \left(\mu y_{n-1}! + \delta z_{n-1}!\right) - C_{n-1}^B!\delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!}.$$

Proof. From Theorem 3, we have $G = LU$. Thus $\det G = \det(LU)$. Since L is the unit triangular matrix, we write

$$\begin{aligned}
\det G &= \prod_{i=1}^{n} U_{ii} = U_{nn} \prod_{i=1}^{n-1} U_{ii} = U_{nn} C_{n-1}^B! \\
&= C_{n-1}^B! \left(C_n^B - (-1)^n \frac{\mu y_{n-1}! + \delta z_{n-1}!}{C_{n-1}^B!} - \delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!} \right) \\
&= C_n^B! - (-1)^n \left(\mu y_{n-1}! + \delta z_{n-1}!\right) - C_{n-1}^B!\delta\mu \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{C_k^B!C_{k-1}^B!},
\end{aligned}$$

which completes the proof.

We can also express $\det G$ in terms of $\{P_n\}, \{y_n\}$ and $\{z_n\}$:

Corollary 5. *For $n > 1$*

$$\det G = P_n - (-1)^n \left(\mu y_{n-1}! + \delta z_{n-1}!\right) - \delta\mu P_{n-1} \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{P_{k-1}P_{k-2}},$$

where $\{P_n\}$ is given by (3.2).

Proof. From Theorem 3 and Corollary 4, we have $\det G = U_{nn} C_{n-1}^B!$. Using (3.4) then gives

$$\begin{aligned}
\det G_n &= U_{nn} P_{n-1} && (3.7) \\
&= P_n - (-1)^n \left(\mu y_{n-1}! + \delta z_{n-1}!\right) - \delta\mu P_{n-1} \sum_{k=1}^{n-1} \frac{z_{k-1}!y_{k-1}!}{P_{k-1}P_{k-2}}, && (3.8)
\end{aligned}$$

as claimed.

4. Inverse of the tridiagonal matrix G

We may now use of backward continued fractions to compute the inverse matrix, G^{-1}. First we provide some results for the inverses of L and U starting with following Lemma.

Lemma 6. *Denote the inverse of L by Q. Then*

$$Q_{ij} = \frac{(-1)^j}{z_{j-1}!C_{i-1}^B!} \begin{cases} (-1)^i z_{i-1}!C_{j-1}^B! & \text{if } 1 \le j \le i \le n-1, \\ \mu C_{j-1}^B!T(n,j)\,C_{n-1}^B! + (-1)^n z_{n-1}!C_{j-1}^B! & \text{if } i = n, \\ 0 & \text{if } i < j, \end{cases}$$

where

$$T(n,m) = \sum_{t=m}^{n-1} \frac{y_{t-1}!z_{t-1}!}{C_t^B!C_{t-1}^B!} \qquad (4.1)$$

with $\sum_{i=n}^{m} f(i) = 0$ for $n > m$.

For odd n, the matrix Q takes the form

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -d_1!/d_0! & 1 & 0 & 0 & 0 & 0 \\ d_2!/d_0! & -d_2!/d_1! & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ -d_{n-2}!/d_0! & d_{n-2}!/d_1! & \cdots & \cdots & 1 & 0 \\ Q_{n1} & Q_{n2} & Q_{n3} & \cdots & Q_{n,n-1} & 1 \end{bmatrix},$$

where Q_{ni} is defined as before for $1 \le i \le n-1$ and $d_k = \dfrac{z_k}{C_k^B}$.

Now we give the proof of Lemma 6.

Proof. Denote LQ by R. If $1 \le j \le i \le n$, then

$$R_{ii} = \sum_{k=1}^{n} L_{ik} Q_{ki} = L_{ii} Q_{ii} = 1.$$

If $i+1 > j \ge 1$, then by the definitions of L and Q, for $1 < i < n$

$$\begin{aligned}
R_{ij} &= \sum_{k=1}^{n-1} L_{ik} Q_{kj} = L_{i,i-1} Q_{i-1,j} + L_{ii} Q_{ij} = \frac{z_{i-1}}{C_{i-1}^B} v_{i-1,j} + v_{ij} \\
&= (-1)^{i+j-1} \frac{z_{i-1} C_{j-1}^B ! z_{i-2}!}{C_{i-1}^B z_{j-1}! C_{i-2}^B !} + (-1)^{i+j} \frac{z_{i-1}! C_{j-1}^B !}{C_{i-1}^B ! z_{j-1}!} \\
&= (-1)^{i+j-1} \frac{z_{i-1}! C_{j-1}^B !}{C_{i-1}^B ! z_{j-1}!} + (-1)^{i+j} \frac{z_{i-1}! C_{j-1}^B !}{C_{i-1}^B ! z_{j-1}!} = 0,
\end{aligned}$$

as claimed. If $i > j \ge 1$, then for $i = n$, we have

$$\begin{aligned}
R_{nj} &= \sum_{k=1}^{n} L_{nk} Q_{kj} = \sum_{k=1}^{n-2} L_{nk} Q_{kj} + L_{n,n-1} Q_{n-1,j} + Q_{nj} \\
&= \sum_{k=j}^{n-2} (-1)^{k+1} \frac{y_{k-1}! \mu}{C_k^B !} (-1)^{k+j} \frac{z_{k-1}! C_{j-1}^B !}{z_{j-1}! C_{k-1}^B !} \\
&+ \left(\frac{z_{n-1}}{C_{n-1}^B} + (-1)^n \mu \frac{y_{n-2}!}{C_{n-1}^B !} \right) (-1)^{n-1+j} \frac{z_{n-2}! C_{j-1}^B !}{z_{j-1}! C_{n-2}^B !} \\
&+ \mu (-1)^j \frac{C_{j-1}^B !}{z_{j-1}!} \sum_{t=j}^{n-1} \frac{y_{t-1}! z_{t-1}!}{C_t^B ! C_{t-1}^B !} + (-1)^{n+j} \frac{z_{n-1}! C_{j-1}^B !}{z_{j-1}! C_{n-1}^B !} \\
&= \mu (-1)^{j+1} \frac{C_{j-1}^B !}{z_{j-1}!} \sum_{t=j}^{n-2} \frac{y_{t-1}! z_{t-1}!}{C_t^B ! C_{t-1}^B !} + \mu (-1)^j \frac{C_{j-1}^B !}{z_{j-1}!} \sum_{t=j}^{n-1} \frac{y_{t-1}! z_{t-1}!}{C_t^B ! C_{t-1}^B !} \\
&+ \left(\frac{z_{n-1}}{C_{n-1}^B} + (-1)^n \frac{\mu y_{n-2}!}{C_{n-1}^B !} \right) (-1)^{n-1+j} \frac{z_{n-2}! C_{j-1}^B !}{z_{j-1}! C_{n-2}^B !} + (-1)^{n+j} \frac{z_{n-1}! C_{j-1}^B !}{z_{j-1}! C_{n-1}^B !} \\
&= (-1)^{n-1+j} \frac{z_{n-1}}{C_{n-1}^B} \frac{z_{n-2}! C_{j-1}^B !}{z_{j-1}! C_{n-2}^B !} + (-1)^n \mu \frac{y_{n-2}!}{C_{n-1}^B !} (-1)^{n-1+j} \frac{z_{n-2}! C_{j-1}^B !}{z_{j-1}! C_{n-2}^B !} \\
&+ (-1)^{n+j} \frac{z_{n-1}! C_{j-1}^B !}{z_{j-1}! C_{n-1}^B !} + (-1)^j \frac{\mu y_{n-2}! z_{n-2}! C_{j-1}^B !}{C_{n-1}^B ! z_{j-1}! C_{n-2}^B !} = 0,
\end{aligned}$$

as claimed. If $i+1 = j \ge 1$, then we immediately obtain $R_{i,i+1} = 0$. It is also clear that $r_{ij} = 0$ when $i < j$, whence, we obtain $R = I_n$ where I_n is the $n \times n$ unit matrix and the proof is complete.

Lemma 7. *Denote the inverse of U by H. Then*

$$H_{ij} = \begin{cases} (-1)^i \dfrac{C_{i-1}! q_{n,i}}{C_{n-1}! U_{nn}} & \text{if } j = n, \\ (-1)^{i+j} \dfrac{y_{j-1}! C_{i-1}!}{y_{i-1}! C_j!} & \text{if } i \le j \le n-1, \\ 0 & \text{if } i > j, \end{cases}$$

where

$$q_{n,i} = \frac{\delta C_{n-1}^B ! T(n,i) + (-1)^n y_{n-1}!}{y_{i-1}!} \tag{4.2}$$

and $T(n,m)$ is defined in Lemma 6.

Before the proof of Lemma 7, by the definitions of H and $q_{n,n}$, we have that $q_{n,n} = (-1)^n$ and so

$$H_{nn} = (-1)^n \frac{C_{n-1}!q_{n,n}}{C_{n-1}!U_{nn}} = (-1)^n \frac{q_{n,n}}{U_{nn}} = \frac{1}{U_{nn}}.$$

Clearly the matrix H takes the form

$$H = \begin{bmatrix} \frac{y_0!C_0^B!}{y_0!C_1^B!} & -\frac{y_1!C_0^B!}{y_0!C_2^B!} & \frac{y_2!C_0^B!}{y_0!C_3^B!} & \cdots & \frac{(-1)^n y_{n-2}!C_0^B!}{y_0!C_{n-1}!} & -\frac{C_0^B!q_{n,1}}{C_{n-1}^B!U_{nn}} \\ 0 & \frac{y_1!C_1^B!}{y_1!C_2^B!} & -\frac{y_2!C_1^B!}{y_1!C_3^B!} & \cdots & \frac{(-1)^{n-1} y_{n-2}!C_1^B!}{y_1!C_{n-1}^B!} & \frac{C_1^B!q_{n,2}}{C_{n-1}^B!U_{nn}} \\ 0 & 0 & \frac{y_2!C_2^B!}{y_2!C_3^B!} & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & -\frac{y_{n-2}!C_{n-3}^B!}{y_{n-3}!C_{n-1}^B!} & \frac{(-1)^n C_{n-3}^B!q_{n,n-2}}{C_{n-1}^B!U_{nn}} \\ 0 & 0 & 0 & 0 & \frac{y_{n-2}!C_{n-2}^B!}{y_{n-2}!C_{n-1}^B!} & \frac{(-1)^{n-1} C_{n-2}^B!q_{n,n-1}}{C_{n-1}^B!U_{nn}} \\ 0 & 0 & 0 & 0 & 0 & \frac{(-1)^n C_{n-1}^B!q_{n,n}}{C_{n-1}^B!U_{nn}} \end{bmatrix}.$$

Proof. Denote UH by E. If $i = j \leq n-1$, then we obtain

$$E_{ii} = \sum_{k=1}^{n} U_{ik}H_{kj} = U_{ii}H_{ii} = C_i^B (-1)^{i+i} \frac{y_{i-1}!C_{i-1}!}{y_{i-1}!C_i!} = \frac{y_{i-1}!C_{i-1}!}{y_{i-1}!C_{i-1}!} = 1.$$

For the case $i = j = n$, by the definition of H_{nn} we may write

$$E_{nn} = U_{nn}H_{nn} = U_{nn}\frac{1}{U_{nn}} = 1.$$

If $1 \leq i \leq n-2$, then

$$\begin{aligned} E_{ij} &= \sum_{k=1}^{n} U_{ik}H_{kj} = U_{ii}H_{ij} + U_{i,i+1}H_{i+1,j} \\ &= C_i^B (-1)^{i+j} \frac{y_{j-1}!C_{i-1}^B!}{y_{i-1}!C_{j-1}^B!} + y_i (-1)^{i+1+j} \frac{y_{j-1}!C_i^B!}{y_i!C_{j-1}^B!} \\ &= (-1)^{i+j} \frac{y_{j-1}!C_i^B!}{y_{i-1}!C_{j-1}^B!} + (-1)^{i+1+j} \frac{y_{j-1}!C_i^B!}{y_{i-1}!C_{j-1}^B!} = 0, \end{aligned}$$

as claimed. If $1 \leq i \leq n-2$ and $j = n$, then

$$\begin{aligned} E_{in} &= \sum_{k=1}^{n} U_{ik}H_{kj} = U_{ii}H_{in} + U_{i,i+1}H_{i+1,n} + U_{in}H_{nn} \\ &= C_i^B (-1)^i \frac{C_{i-1}^B!q_{n,i}}{C_{n-1}^B!U_{nn}} + y_i (-1)^{i+1} \frac{C_i^B!q_{n,i+1}}{C_{n-1}^B!U_{nn}} + \delta\frac{(-1)^{i-1} z_{i-1}!}{C_{i-1}^B!U_{nn}} \\ &= (-1)^i \frac{C_i^B!q_{n,i}}{C_{n-1}^B!U_{nn}} + y_i (-1)^{i+1} \frac{C_i^B!q_{n,i+1}}{C_{n-1}^B!U_{nn}} + \delta\frac{(-1)^{i-1} z_{i-1}!}{C_{i-1}^B!U_{nn}} \\ &= \frac{(-1)^i C_i^B!}{C_{n-1}^B!U_{nn}} \left[q_{n,i} - y_i q_{n,i+1} - \frac{\delta z_{i-1}!C_{n-1}^B!}{C_{i-1}^B!C_i^B!} \right]. \end{aligned}$$

Here note that

$$\begin{aligned}
&q_{n,i} - y_i q_{n,i+1} \\
&= \frac{\delta C^B_{n-1}!}{y_{i-1}!} \sum_{k=0}^{n-i-1} \frac{z_{k+i-1}! y_{k-1+i}!}{C^B_{k+i-1}! C^B_{k+i}!} + (-1)^n \frac{y_{n-1}!}{y_{i-1}!} \\
&- \frac{\delta y_i C^B_{n-1}!}{y_i!} \sum_{k=0}^{n-i-2} \frac{z_{k+i}! y_{k+i}!}{C^B_{k+i}! C^B_{k+i+1}!} - (-1)^n y_i \frac{y_{n-1}!}{y_i!} \\
&= C^B_{n-1}! \left(\frac{\delta z_{n-2}! y_{n-2}!}{C^B_{n-2}! y_{i-1}! C^B_{n-1}!} + \frac{\delta}{y_{i-1}!} \sum_{k=0}^{n-i-2} \frac{z_{k+i-1}! y_{k-1+i}!}{C^B_{k+i-1}! C^B_{k+i}!} \right) \\
&- \frac{\delta y_i C^B_{n-1}!}{y_i!} \sum_{k=0}^{n-i-2} \frac{z_{k+i}! y_{k+i}!}{C^B_{k+i}! C^B_{k+i+1}!} \\
&= C^B_{n-1}! \left(\frac{\delta z_{n-2}! y_{n-2}!}{C^B_{n-2}! y_{i-1}! C^B_{n-1}!} + \frac{\delta}{y_{i-1}!} \sum_{k=0}^{n-i-2} \frac{z_{k+i-1}! y_{k-1+i}!}{C^B_{k+i-1}! C^B_{k+i}!} \right) \\
&- \frac{\delta y_i C^B_{n-1}!}{y_i!} \sum_{k=0}^{n-i-2} \frac{z_{k+i}! y_{k+i}!}{C^B_{k+i}! C^B_{k+i+1}!} \\
&= \frac{\delta z_{n-2}! y_{n-2}!}{C_{n-2}! y_{i-1}!} + \frac{\delta C^B_{n-1}!}{y_{i-1}!} \sum_{k=0}^{n-i-2} \frac{z_{k+i-1}! y_{k-1+i}!}{C^B_{k+i-1}! C^B_{k+i}!} - \frac{y_i C^B_{n-1}!}{y_i!} \sum_{k=0}^{n-i-2} \frac{\delta z_{k+i}! y_{k+i}!}{C^B_{k+i}! C^B_{k+i+1}!} \\
&= \frac{\delta z_{n-2}! y_{n-2}!}{C_{n-2}! y_{i-1}!} + \frac{\delta C^B_{n-1}!}{y_{i-1}!} \sum_{k=0}^{n-i-2} \left(\frac{z_{k+i-1}! y_{k-1+i}!}{C^B_{k+i-1}! C^B_{k+i}!} - \frac{z_{k+i}! y_{k+i}!}{C^B_{k+i}! C^B_{k+i+1}!} \right),
\end{aligned}$$

which, by creative telescoping, equals

$$\frac{\delta z_{n-2}!}{C_{n-2}!} \frac{y_{n-2}!}{y_{i-1}!} + \frac{\delta C^B_{n-1}!}{y_{i-1}!} \left(\frac{z_{i-1}! y_{i-1}!}{C^B_{i-1}! C^B_i!} - \frac{z_{n-2}! y_{n-2}!}{C^B_{n-2}! C^B_{n-1}!} \right) = \frac{\delta C^B_{n-1}! z_{i-1}!}{C^B_{i-1}! C^B_i!}.$$

Whence

$$\begin{aligned}
E_{in} &= \frac{(-1)^i C^B_i!}{C^B_{n-1}! U_{nn}} \left[q_{n,i} - y_i q_{n,i+1} - \frac{\delta z_{i-1}! C^B_{n-1}!}{C^B_{i-1}! C^B_i!} \right] \\
&= \frac{(-1)^i C^B_i!}{C^B_{n-1}! U_{nn}} \left[\frac{\delta C^B_{n-1}! z_{i-1}!}{C^B_{i-1}! C^B_i!} - \frac{\delta z_{i-1}! C^B_{n-1}!}{C^B_{i-1}! C^B_i!} \right] = 0,
\end{aligned}$$

as claimed.

Now consider the case $i = n-1$. In that case, we see that $n-1 \le j \le n$ and

$$E_{n-1,j} = \sum_{k=1}^{n} U_{n-1,k} H_{kj} = U_{n-1,n-1} H_{n-1,j} + U_{n-1,n} H_{nj}.$$

For $j = n-1$, we obtain

$$E_{n-1,n-1} = U_{n-1,n-1} H_{n-1,n-1} = C^B_{n-1} \frac{y_{n-2}! C^B_{n-2}!}{y_{n-2}! C^B_{n-1}!} = 1,$$

as claimed. For $j = n$, we have

$$\begin{aligned}
E_{n-1,n} &= U_{n-1,n-1} H_{n-1,n} + U_{n-1,n} H_{nn} \\
&= C^B_{n-1} (-1)^{n-1} \frac{C^B_{n-2}! q_{n,n-1}}{C^B_{n-1}! U_{nn}} + \left(y_{n-1} + (-1)^{n-2} \frac{z_{n-2}! \delta}{C^B_{n-2}!} \right) \frac{1}{U_{nn}} \\
&= (-1)^{n-1} \frac{q_{n,n-1}}{U_{nn}} + \frac{y_{n-1}}{U_{nn}} - (-1)^{n-1} \frac{z_{n-2}! \delta}{C^B_{n-2}! U_{nn}} \\
&= \frac{1}{U_{nn}} \left[(-1)^{n-1} q_{n,n-1} + y_{n-1} - (-1)^{n-1} \frac{z_{n-2}! \delta}{C^B_{n-2}!} \right],
\end{aligned}$$

which, by the definition of $q_{n,n-1}$, equals

$$\frac{(-1)^{n-1}}{U_{nn}}\left[\left(\frac{\delta z_{n-2}!y_{n-2}!C_{n-1}^B!}{C_{n-2}!y_{n-2}!C_{n-1}^B!}+(-1)^n\frac{y_{n-1}!}{y_{n-2}!}\right)+y_{n-1}-\frac{z_{n-2}!\delta}{C_{n-2}^B!}\right]$$
$$=\frac{(-1)^{n-1}}{U_{nn}}\left[\frac{\delta z_{n-2}!}{C_{n-2}^B!}-y_{n-1}!+y_{n-1}-\frac{z_{n-2}!\delta}{C_{n-2}^B!}\right]=\frac{(-1)^{n-1}}{U_{nn}}\left[\frac{\delta z_{n-2}!}{C_{n-2}^B!}-\frac{z_{n-2}!\delta}{C_{n-2}^B!}\right]=0,$$

as claimed. It is clear that $E_{ij}=0$ for the case $i>j$. Thus $E=I_n$ and we are finished.

Suppose that $y_iz_i\neq 0$ for $1\leq i\leq n-1$. Then one of our main results is given by the following Theorem.

Theorem 8. *Denote the inverse of matrix G of order n by W. Then*

$$W_{ij}=\frac{(-1)^{i+j}C_{i-1}^B!C_{j-1}^B!}{z_{j-1}!C_{n-1}^B!U_{nn}}\begin{cases}D_n(i,i;i) & \text{if } i=j,\\ D_n(j,j;i) & \text{if } i<j,\\ D_n(i,j;i) & \text{if } i>j,\end{cases}$$

where

$$D_n(a,b;i)=\frac{C_{n-1}^B!U_{nn}}{y_{i-1}!}T(n,a)+\mu q_{n,i}T(n,b)+\frac{(-1)^n q_{n,i}z_{n-1}!}{C_{n-1}^B!} \tag{4.3}$$

and the $q_{n,i}$ are defined by (4.2).

Here note that

$$W_{in}=(-1)^i\frac{C_{i-1}^B!q_{n,i}}{C_{n-1}^B!U_{nn}}\text{ and, specifically, }W_{nn}=\frac{1}{U_{nn}},$$

and

$$W_{nj}=\frac{(-1)^{n+j}C_{j-1}^B!\left(C_{n-1}^B!+\mu y_{n-1}!\right)T(n,j)}{z_{j-1}!y_{n-1}!}+\frac{(-1)^j z_{n-1}!C_{j-1}^B!}{z_{j-1}!C_{n-1}^B!}.$$

Proof. There are three subcases. First, consider $i=j$. Since $H_{ij}=0$ for $i>j$ and $W=(LU)^{-1}=HQ$, we have

$$W_{ii}=\sum_{k=1}^n H_{ik}Q_{ki}=\sum_{k=i}^n H_{ik}Q_{ki}=\sum_{k=i}^{n-1}H_{ik}Q_{ki}+H_{in}Q_{ni}$$
$$=\sum_{k=i}^{n-1}(-1)^{i+k}\frac{y_{k-1}!C_{i-1}^B!}{y_{i-1}!C_k^B!}(-1)^{k+i}\frac{z_{k-1}!C_{i-1}^B!}{z_{i-1}!C_{k-1}^B!}+(-1)^i\frac{C_{i-1}^B!q_{n,i}}{C_{n-1}^B!U_{nn}}$$
$$\times\left(\frac{\mu(-1)^iC_{i-1}^B!}{z_{i-1}!}\sum_{t=i}^{n-1}\frac{y_{t-1}!z_{t-1}!}{C_t^B!C_{t-1}^B!}+(-1)^{n+i}\frac{z_{n-1}!C_{i-1}^B!}{z_{i-1}!C_{n-1}^B!}\right)$$
$$=\left(\frac{\left(C_{i-1}^B!\right)^2}{y_{i-1}!z_{i-1}!}+\frac{\mu q_{n,i}\left(C_{i-1}^B!\right)^2}{z_{i-1}!C_{n-1}^B!U_{nn}}\right)T(n,i)+(-1)^n\left(\frac{C_{i-1}^B!}{C_{n-1}^B!}\right)^2\frac{q_{n,i}z_{n-1}!}{U_{nn}z_{i-1}!}.$$

For $i=n$, we examine the formula:

$$W_{nn}=\frac{\left(C_{n-1}^B!\right)^2}{z_{n-1}!}\left(\frac{1}{y_{n-1}!}+\frac{\mu q_{n,n}}{C_{n-1}^B!U_{nn}}\right)\sum_{k=n}^{n-1}\frac{y_{k-1}!z_{k-1}!}{C_k^B!C_{k-1}^B!}+(-1)^n\left(\frac{C_{n-1}^B!}{C_{n-1}^B!}\right)^2\frac{q_{n,n}z_{n-1}!}{U_{nn}z_{n-1}!},$$

which, since the empty sum is 0 and by the definition of $q_{n,n}$, leads to

$$W_{nn}=(-1)^n\frac{q_{n,n}}{U_{nn}}=\frac{(-1)^n}{U_{nn}}\left(\sum_{k=0}^{-1}\frac{\delta z_{k+n-1}!}{C_{k+n-1}^B!}\frac{y_{k-1+n}!}{y_{i-1}!}\frac{C_{n-1}^B!}{C_{k+i}^B!}+(-1)^n\frac{y_{n-1}!}{y_{n-1}!}\right)=\frac{1}{U_{nn}}.$$

For $i<j$, we may write

$$W_{ij}=\sum_{k=1}^n H_{ik}Q_{kj}=\sum_{k=j}^n H_{ik}Q_{kj}=\sum_{k=j}^{n-1}H_{ik}Q_{kj}+H_{in}Q_{nj},$$

which, after some rearrangement, gives

$$\begin{aligned}
W_{ij} &= (-1)^{i+j}\,\frac{C^B_{i-1}!C^B_{j-1}!}{z_{j-1}!}\left(\frac{1}{y_{i-1}!}+\frac{\mu q_{n,i}}{C^B_{n-1}!U_{nn}}\right)T(n,j)+(-1)^{n+i+j}\,\frac{z_{n-1}!C^B_{j-1}!C^B_{i-1}!q_{n,i}}{z_{j-1}!C^B_{n-1}!C^B_{n-1}!U_{nn}}\\
&= \frac{(-1)^{i+j}\,C^B_{i-1}!C^B_{j-1}!}{z_{j-1}!C^B_{n-1}!U_{nn}}\left[\left(\frac{C^B_{n-1}!U_{nn}}{y_{i-1}!}+\mu q_{n,i}\right)T(n,j)+(-1)^n\,\frac{z_{n-1}!q_{n,i}}{C^B_{n-1}!}\right].
\end{aligned}$$

For $j=n$, we have $T(n,n)=0$ and hence

$$W_{in}=(-1)^i\,\frac{C^B_{i-1}!q_{n,i}}{C^B_{n-1}!U_{nn}}.$$

We consider the last case $i>j$. By the definition of H, we see that

$$\begin{aligned}
W_{ij} &= \sum_{k=1}^{n}H_{ik}Q_{kj}=\sum_{k=i}^{n}H_{ik}Q_{kj}=\sum_{k=i}^{n-1}H_{ik}Q_{kj}+H_{in}Q_{nj}\\
&= (-1)^{i+j}\,\frac{C^B_{i-1}!C^B_{j-1}!}{y_{i-1}!z_{j-1}!}\sum_{k=i}^{n-1}\frac{y_{k-1}!z_{k-1}!}{C^B_k!C^B_{k-1}!}+\mu(-1)^{i+j}\,\frac{C^B_{i-1}!C^B_{j-1}!q_{n,i}}{C^B_{n-1}!z_{j-1}!U_{nn}}\\
&\quad\times\sum_{t=j}^{n-1}\frac{y_{t-1}!z_{t-1}!}{C^B_t!C^B_{t-1}!}+(-1)^{n+i+j}\,\frac{q_{n,i}z_{n-1}!C^B_{i-1}!C^B_{j-1}!}{z_{j-1}!U_{nn}\left(C^B_{n-1}!\right)^2}\\
&= (-1)^{i+j}\,\frac{C^B_{i-1}!C^B_{j-1}!}{z_{j-1}!}\\
&\quad\times\left[\frac{1}{y_{i-1}!}\sum_{k=i}^{n-1}\frac{y_{k-1}!z_{k-1}!}{C^B_k!C^B_{k-1}!}+\frac{\mu q_{n,i}}{C^B_{n-1}!U_{nn}}\sum_{t=j}^{n-1}\frac{y_{t-1}!z_{t-1}!}{C^B_t!C^B_{t-1}!}+\frac{(-1)^n\,q_{n,i}z_{n-1}!}{U_{nn}\left(C^B_{n-1}!\right)^2}\right],
\end{aligned}$$

as claimed and we have the required result for all three cases.

Using (3.4), the following Corollary may be used to calculate the inverse matrix, $W=G^{-1}$, more easily.

Corollary 9. *Let W denote the inverse of G. Then*

$$W_{ij}=\frac{(-1)^{i+j}\,P_{i-1}P_{j-1}}{z_{j-1}!P_{n-1}U_{nn}}\begin{cases}D_n(i,i;i) & \text{if } i=j,\\ D_n(j,j;i) & \text{if } i<j,\\ D_n(i,j;i) & \text{if } i>j,\end{cases}\tag{4.4}$$

where $D(a,b;i)$ is defined as before.

Consequently we see that the $D(a,b;i)$ are key to computing the entries of W.

5. An illustrative example

As a simple example, we consider the matrix

$$G=\begin{bmatrix}2&1&0&-1\\3&3&1&0\\0&2&4&1\\5&0&1&1\end{bmatrix}$$

and show in detail the steps required to compute the determinant of G and the inverse matrix, $W=G^{-1}$.

Here $\delta=-1,\ \mu=5,\ y_1=y_2=y_3=1;\ z_1=3,z_2=2,z_3=1;$ and, $x_1=2,x_2=3,x_3=x_4=1$.

1. Compute $y_i!$ and $z_i!$ for $1\le i\le 3$ as

$$y_1!=y_2!=y_3!=1;\ z_1!=3,\ z_2!=z_3!=6.$$

2. Compute P_i for $0 \leq i \leq 4$ using (3.2)

$$P_0 = 1,\ P_1 = 2,\ P_2 = 3,\ P_3 = 8 \text{ and } P_4 = 5.$$

3. Compute the $T\,(4, m)$ values for $1 \leq m \leq 3$ using (4.1) and noting that $C_t^B = P_t$, etc., whence

$$T\,(4,1) = \frac{5}{4},\ T\,(4,2) = \frac{3}{4},\ T\,(4,3) = \frac{1}{4}.$$

4. We may now compute the determinant of G using (3.8) and (4.1)

$$\begin{aligned}\det G &= P_4 - \delta\mu P_3 T(4,1) - (-1)^4(\mu y_3! + \delta z_3!)\\ &= 56\end{aligned}$$

5. Compute the $q_{n,i}$ for $1 \leq i \leq 4$ using (4.2) and noting that $C_{n-1}^B! = P_{n-1}$ by (3.4) to give

$$q_{4,1} = -9,\ q_{4,2} = -5,\ q_{4,3} = -1,\ q_{44} = 1.$$

6. By Theorem 8, we can write the inverse matrix $W = G^{-1}$ as

$$W = \begin{bmatrix} \frac{1}{56}D_4\,(1,1;1) & -\frac{1}{84}D_4\,(2,2;1) & \frac{1}{112}D_4\,(3,3;1) & -\frac{1}{42}D_4\,(4,4;1) \\ -\frac{1}{28}D_4\,(2,1;2) & \frac{1}{42}D_4\,(2,2;2) & -\frac{1}{56}D_4\,(3,3;2) & \frac{1}{21}D_4\,(4,4;2) \\ \frac{3}{56}D_4\,(3,1;3) & -\frac{1}{28}D_4\,(3,2;3) & \frac{3}{112}D_4\,(3,3;3) & -\frac{1}{14}D_4\,(4,4;3) \\ -\frac{1}{7}D_4\,(4,1;4) & \frac{2}{21}D_4\,(4,2;4) & -\frac{1}{14}D_4\,(4,3;4) & \frac{4}{21}D_4\,(4,4;4) \end{bmatrix},$$

where we can compute the required $D(a, b; i)$ using (4.3) to give

$$D_4\,(a, b; i) = 56T\,(4, a) + 5q_{4,i}T\,(4, b) + \frac{3}{4}q_{4,i}.$$

Whence

$$\begin{bmatrix} D_4\,(1,1;1) & D_4\,(2,2;1) & D_4\,(3,3;1) & D_4\,(4,4;1) \\ D_4\,(2,1;2) & D_4\,(2,2;2) & D_4\,(3,3;2) & D_4\,(4,4;2) \\ D_4\,(3,1;3) & D_4\,(3,2;3) & D_4\,(3,3;3) & D_4\,(4,4;3) \\ D_4\,(4,1;4) & D_4\,(4,2;4) & D_4\,(4,3;4) & D_4\,(4,4;4) \end{bmatrix} = \begin{bmatrix} 7 & 3/2 & -4 & -27/4 \\ 7 & 39/2 & 4 & -15/4 \\ 7 & 19/2 & 12 & -3/4 \\ 7 & 9/2 & 2 & 3/4 \end{bmatrix}$$

and the inverse matrix follows as

$$W = \begin{bmatrix} \frac{1}{8} & -\frac{1}{56} & -\frac{1}{28} & \frac{9}{56} \\ -\frac{1}{4} & \frac{13}{28} & -\frac{1}{14} & -\frac{5}{28} \\ \frac{3}{8} & -\frac{19}{56} & \frac{9}{28} & \frac{3}{56} \\ -1 & \frac{3}{7} & -\frac{1}{7} & \frac{1}{7} \end{bmatrix}.$$

6. Computational Implementations

In this section we consider the effectiveness of the proposed algorithm (KA) as a computational method and compare its performance with both El-Shehawey's algorithm (ESA) and a pair of Maple procedures.

We look to compare the number of basic arithmetic operations required by each of the two algorithms (ESA and KA). For this exercise we assume that add/subtract and multiply/divide may be considered comparable operations. Whilst this approach often gives a good indication of how well an implementation will perform against a competing method when using floating point arithmetic, such comparisons are not generally as helpful when comparing computer algebra implementations. This is because, unlike floating point arithmetic, the time taken to perform a basic arithmetic operation on a particular platform is not constant but depends on the algebraic complexity of the two operands; note that this is even the case for operations between rational numbers as the cpu time taken to reduce results to their simplest form is not constant. However, it is still useful to perform the comparison, if only to distinguish between order of magnitude differences.

We also consider how we can easily save operations by not recomputing either useful intermediate or reusable results. Here we are trading the extra storage required for these values against the execution time required to recompute them. For floating point computations the amount of additional storage necessary may be calculated in advance, but for algebraic computation this is not the case. Indeed the amount of extra storage is dependant on the individual problem and could be large if the intermediate results are algebraically complex.

We begin by looking in detail at the ESA implementation [25, p132–133] and show how we may reduce the number of operations performed from $O(n^3)$ to $O(n^2)$. For the implementation of KA and ESA we then seek to optimize the number of arithmetic operations by storing reusable computation as far as possible.

6.1. The Implementation of El-Shehawey's Algorithm

Both a pseudocode version of the algorithm and a Maple implementation are presented in [25]. The published code contains an error in line 12 of the procedure *continuant* which should read

```
d[k+1] := -A[k+1, k]:
```

In addition, the determinant given as the true value of their example matrix is incorrect; it should read $D = a^2(a^2 + 4)$.

This algorithm requires the computation of the determinants of submatrices of the tridiagonal matrix obtained by setting the elements δ and μ in (1.1) to zero:

$$\Delta_i^j = \det(T(i:j, i:j));\ j \geq i, i = 1, \ldots, n \tag{6.1}$$

During the computation of the inverse all values of the Δ_i^j are used more than once with the exception of Δ_1^n (not used) and Δ_2^n (used only once).

In their implementation El-Shehawey et. al. compute each value of Δ_i^j from scratch as it is required by the computation; this leads to an $O(n^3)$ algorithm. The run time efficiency of their algorithm may be improved dramatically by trading improved execution time against extra storage and computing all the values of Δ_i^j for $1 \leq i \leq n$; $i \leq j \leq n$ and storing them in the upper triangular portion of a workspace array. Starting with the diagonal element, $\Delta_i^i = x_i$, each successive element in the row, $\Delta_i^j, j = i+1, \ldots, n$, may be computed in turn using just 3 multiplies and an add, giving a total operation count for all the required Δ_i^j of $3n(n+1)/2$ multiplies and $n(n+1)/2$ adds along with $n(n+1)/2$ extra storage elements. This reduces the overall complexity of the method to $O(n^2)$.

By storing an extra $4n + 4$ elements of storage and eliminating other repeated product computations we may further reduce the operation count of this algorithm to $\frac{1}{2}(21n^2 + 15n - 10)$ multiplies and $\frac{1}{2}n(5n - 1)$ additions.

The use of the recurrence relationship [25, (3.7, 3.8)]

$$\Delta_i^j = \prod_{k=i}^{j} \gamma_k \tag{6.2}$$

$$\gamma_k = \begin{cases} x_i, & k = i \\ x_k + \frac{y_{k-1} z_{k-1}}{\gamma_{k-1}}, & k = i+1, \ldots, j \end{cases}$$

results in a divide by zero if any of the $\gamma_k = 0$ for $k = i, \ldots, j-1$. This situation does not signal the non-existence of the inverse; it is purely a result of the way in which the computation is performed. In their Maple implementation, El-Shehawey et. al. 'fix' the problem by replacing the zero with an algebraic quantity, x, and continuing the computation. When the inverse has been computed the value zero is substituted for x and the final expressions simplified. This could prove to be an extremely inefficient ploy as the computation is carrying algebraic expressions which are actually zero but, until the final substitution, are just adding complexity to both intermediate results and the elements of the inverse.

The problem may be avoided by using the alternative three-term recurrence [25, (3.6)]:

$$\Delta_i^j = x_j \Delta_i^{j-1} - z_{j-1} y_{j-1} \Delta_i^{j-2} \tag{6.3}$$

with $\Delta_i^{i-1} = 1$, $\Delta_i^k = 0$ for $k < i - 2$ and $\Delta_i^i = x_i$.

Incorporating all the above improvements leads to the pseudocode algorithm given in Algorithm 1; the main procedure uses two supplementary functions that are provided in Algorithm 2.

6.2. The Implementation of the Proposed New Algorithm

A pseudocode version of KA is given in Figure 3. This version has been constructed in a similar way to the ESA method described in the previous section in that we have attempted to save reusable computations as far as is sensible. This has involved the use of twelve additional local arrays totalling $12n + 2$ elements to store reusable values compared to the $(n^2 + 11n)/2$ required by ESA. We repeat here that this does not guarantee the KA will run faster than ESA since the complexity of the operands to these basic operations is both algorithm and problem dependent and will also be affected by the degree to which intermediate computed expressions simplify.

The operation count for the implemented version of KA is $3n^2 + 21n - 4$ multiplications and $n^2 + 5n - 1$ additions which is lower than our re-implementation of ESA.

We note here that the implementation of KA imposes a number of restrictions on the structure of the matrix whose inverse is desired. Specifically,

1. $y_i,\ z_i \neq 0;\ i = 1, \ldots, n-1$, and
2. $x_1 \neq 0$, and
3. the determinants of all the principal minors of G must be non-zero; i.e., $p_i \neq 0,\ i = 1, \ldots, n-1$. Condition 2. above is a consequence of this requirement.

Checks of all these conditions may be made at the outset of the implementation.

None of the above conditions on their own signify a non-singular matrix; however, their presence will cause a divide by zero at some stage of the computation. It may be possible to lift some or all of these restrictions by rearranging the calculations but we were unable to discover how to do this.

We note here the importance of restricting the complexity of expressions generated during symbolic algebraic computations. Failure to do this will often lead to an explosive increase in the number of terms in both intermediate and final results with an associated increase in execution time. El-Shehawey et al. [25] in their code only perform a simplification step on their final result and this can (and does) cause the operation to take an excessive amount of computation time to generate a compact result. It is generally far better to simplify intermediate computations in the hope of restricting the complexity of future calculations.

6.3. Testing

The two algorithms, KA and ESA, were both implemented using Maple. A number of test matrices composed of both rational and algebraic elements were used to verify the implementation and to compare the efficiency of the two procedures. The inverses generated were compared against either known inverses or the inverses obtained by using the Maple library routines, *LinearAlgebra:-MatrixInverse* and *LinearAlgebra:-Determinant*.

We also tested a reworked version of the original ESA algorithm. This version keeps the original body of code but rewrites the procedure interface to make it consistent with the two new versions. Doing this will not effect the execution time of the procedure; it just allows us to re-use the testing and timing code.

6.3.1. Test Problems

Test 1

Our simplest test was the symmetric tridiagonal inverse of the Lehmer matrix [35] whose elements are all rational values with

$$G_{ij} = \begin{cases} \dfrac{4i^3}{4i^2 - 1} & i = j,\ 1 \leq i < n \\ \dfrac{n^2}{2n-1} & i = j = n \\ -\dfrac{i(i+1)}{2i+1} & j = i+1,\ 1 \leq i \leq n-1 \text{ and } j = i-1,\ 2 \leq i \leq n \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

Its inverse is the Lehmer matrix defined by

$$\begin{aligned} G_{ij}^{-1} &= \begin{cases} i/j, & i \leq j \\ j/i, & i > j \end{cases} \\ &= \min(i,j)/\max(i,j) \end{aligned} \tag{6.5}$$

for $1 \leq i, j \leq n$.

While this is obviously a special case (symmetric and $\delta = \mu = 0$), it did provide an easily verifiable sanity check of all the implementations.

Test 2

This is another test involving rational values which is non-symmetric and periodic; the example does not have a simple closed form inverse.

$$G_{ij} = \begin{cases} \dfrac{4i^3}{4i^2-1} & i = j,\ 1 \leq i \leq n \\ \dfrac{n^2}{2n-1} & i = j = n \\ -\dfrac{i(i+1)}{2i+1} & j = i+1,\ 1 \leq i \leq n-1 \\ \dfrac{i(i+1)}{2(2i+1)} & j = i-1,\ 2 \leq i \leq n \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

with $\delta = \dfrac{3n}{4}$ and $\mu = \dfrac{3n}{2}$. For this example the results were verified by comparing them to the inverse computed by the Maple library functions.

Test 3

This is an algebraic test matrix and represents an extension to the test case given in [25]

$$G_{ij} = \begin{cases} a & i = j,\ 1 \leq i < n \\ 1 & j = i+1,\ 1 \leq i \leq n-1 \\ -1 & j = i-1,\ 2 \leq i \leq n \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

with $\delta = -1$ and $\mu = 1$.

Here again a closed form inverse is not know and the implementations were checked by comparing their results with those generated by the Maple library functions.

Tests 4 and 5

Here we consider the tridiagonal inverse of the general KMS matrix [28, pp. E190–E191] defined by

$$G_{ij} = \begin{cases} 1/(1-\sigma\rho) & i = j = 1, n \\ (1+\sigma\rho)/(1-\sigma\rho) & i = j = 2, \ldots, n-1 \\ -\sigma/(1-\sigma\rho) & j = i-1,\ 2 \leq i \leq n \\ -\rho/(1-\sigma\rho) & j = i+1,\ 1 \leq i \leq n-1 \\ 0 & \text{otherwise} \end{cases} \tag{6.8}$$

whose inverse is given, in the same article, as

$$G_{ij}^{-1} = \begin{cases} \rho^{j-i} & i < j;\ i,j = 1, \ldots, n \\ \sigma^{i-j} & i > j;\ i,j = 1, \ldots, n \\ 1 & i = j \end{cases} \tag{6.9}$$

For Test 4 we use the symmetric version obtained by setting $\sigma = \rho$ and for Test 5 we consider the non-symmetric case as given above.

Test 6

For this test we generate a periodic, non-symmetric, tridiagonal matrix by using Test 5 above with $\delta = \sigma^2$ and $\mu = \sigma\rho$. This does not have a known simple closed form.

6.4. Performance of the Implementations

All reported timings were obtained using Maple 17 running under Ubuntu 14.04 on an 8 core Intel (R) Core i7-3840QM, 2.8GHz CPU with 16Gb of memory. The tests were run on an isolated machine (i.e., no network connection) that was as lightly loaded as possible (a single terminal window for executing the run scripts). The tests were run for a range of values of n, the order of the input matrix, and generated determinant values and inverse matrices were checked with the results returned by the Maple general matrix procedures.

To obtain execution timings Maple provides a library procedure, *CodeTools:-Usage* which may be used to calculate the cpu time required to run a procedure. In order to iron out minor deviations it is recommended that the procedure is executed multiple times, using the *iterations* parameter, to obtain an average. For larger problems the time spent in the Maple system's garbage collector (GC) may constitute a sizeable fraction of the reported total execution time. The frequency and occurrence of calls to the GC are not controllable by the user and are dependent on many factors. We have chosen to remove the time spent in the GC from the overall timings we obtained from *Usage* for each test case. We believe that this provides a clearer picture of the effectiveness of the underlying algorithms. We also think these timings provide a better indicator of which algorithm to choose as the starting point for an implementation of a periodic, tridiagonal matrix inversion code written in another SA language.

Figure 1 provides a comparison of the execution times taken by two versions of KA; the first (KASS) performs just a single simplification step on the generated inverse while the second (KA) attempts to reduce complexity by simplifying the results of all intermediate computations as they are calculated. Using timings obtained for Test 3 and 4 we clearly show the dramatic reduction in the execution times obtained from early simplification. For very small and/or simple problems this approach may be counter-productive but the gains, for a general solver, far outweigh these minor additional overheads.

Test 3			Test 4		
n	$KASS$	KA	n	$KASS$	KA
10	0.037	0.008	10	0.055	0.005
15	0.741	0.028	15	1.788	0.008
20	14.451	0.033	20	34.291	0.012

Figure 1: Comparison of execution times for KA showing the effect of simplifying expressions as soon as they are generated.

Figure 2 compares the execution times for the originally published version of the ESA (with the interface changed so that it could interact with the test suite) and our reworked version which uses less operations and simplifies expressions as they are computed. The improved version clearly shows the benefits of an $O(n^2)$ implementation over an $O(n^3)$ one.

Test 2			Test 3			Test 6		
n	$ESAOP$	ESA	n	$ESAOP$	ESA	n	$ESAOP$	ESA
50	0.758	0.134	30	0.649	0.050	10	0.025	0.013
60	1.359	0.207	40	2.287	0.109	20	1.018	0.054
70	2.263	0.303	50	10.444	0.155	30	16.537	0.119

Figure 2: Comparison of execution times for ESA showing the efficiency gains obtained by storing and reusing calculations.

Figure 3 provides sample execution times for Tests 2, 3, 5 and 6 when running implementations of the new proposed algorithm (KA), the El-Shehawey et al [25] algorithm with the efficiency upgrades described in 6.1 (ESA) and for the Maple library procedures *LinearAlgebra:-MatrixInverse* and *LinearAlgebra:-Determinant* (MAPLE). Since both KA and ESA return the inverse and the determinant our Maple comparison timings record the combined execution time of the two procedures. The timings obtained for Test 4 exhibit very similar trends to those of Test 5 for both KA and MAPLE; for ESA the rapid increase in cpu time starts at a lower value of n and increases faster than for Test 4. We thus choose to present only the results for Test 5 here.

KA fares extremely well on Test 5 where, for $n = 150$ it executes 10 times faster than MAPLE and 200+ times faster than ESA. However, this is almost reversed for Test 3 where ESA is clearly the most efficient especially for larger values of n. Finally, for Test 6 MAPLE is far more efficient than ESA and, after being comparable with KA to $n = 100$, is faster than KA thereafter.

These results show very clearly how difficult it is, for SA systems, to identify an algorithm and its implementation that will perform well on general algebraic problems. As far as the basic operations counts are concerned the order of improving performance should be KA, ESA, MAPLE; however, our data shows that there is no clear cut winner. The performance is very dependent on the individual problem being solved.

Test 2				Test 3			
n	$MAPLE$	ESA	KA	n	$MAPLE$	ESA	KA
50	0.119	0.133	0.077	30	0.724	0.050	0.071
80	0.426	0.418	0.237	50	3.187	0.156	0.274
100	0.784	0.721	0.431	60	9.827	0.403	1.180
150	3.222	1.958	1.227	100	n/a	2.081	18.659
Test 5				Test 6			
n	$MAPLE$	ESA	KA	n	$MAPLE$	ESA	KA
50	0.212	0.161	0.057	50	0.292	0.568	0.328
80	0.745	0.907	0.147	80	1.836	10.261	1.416
100	1.398	7.244	0.203	100	3.787	65.097	6.253
150	4.533	94.694	0.447				

Figure 3: Comparison of execution times for MAPLE, ESA and KA on a variety of test problems

It is not surprising that the Maple procedures are generally outperformed by KA and/or ESA since they are designed to invert and find the determinant of a general $n \times n$ matrix and, therefore, cannot explicitly exploit the structure of a periodic, tridiagonal matrix. The test most closely in line with the operations counts is Test 2 where all the elements are rational number; here the execution times are less likely to be affected by an explosion in complexity that can occur when manipulating general algebraic expressions. Even so, KA is less than a factor of 2 faster than ESA; this may be due to more fortuitous cancellation in intermediate values in ESA.

Overall the proposed method (KA) could be considered as a good substitute to the Maple routine for inverting periodic, tridiagonal equations since in all tests, apart from Test 6, it was, at worst, a factor of 2 faster and more often a factor of 10+ better than the Maple library procedures. Even for Test 6 it was superior up to order 80. In addition, for larger problems, it requires substantially less storage of reusable data than ESA.

Further testing also showed that we can use both KA and ESA to invert matrices of larger orders efficiently. Figure 4 gives the order of matrix we can invert using a maximum of 10 seconds cpu time on our test platform.

Test	$MAPLE$	ESA	KA
1	240	400	650
2	210	280	310
3	50	175	100
5	190	105	560
6	155	75	100

Figure 4: Comparison of the maximum order of various test examples that can be inverted in less than 10 seconds.

Finally, it is interesting to look at the overheads of garbage collection on the total execution times (TET) of the test cases as returned by *CodeTools:-Usage*.

First we consider the ESA implementation; for Tests 1, 2, 3 and 4 with $n \leq 150$ around 10% of TET is spent in GC with the number of calls rising from a maximum of 3 for Tests 1 and 2 to around 10 for Tests 3 and 4. For Tests 5 and 6 both the frequency of calls and the time spent in GC increases rapidly with n; for example, for Test 5, $n = 150$, GC is called around 80 times per test and constitutes a massive 76% of TET (263 out of 355 seconds).

KA initiates far fewer calls to GC compared to ESA; even for very large problems the number of calls rarely exceeds 10. Although the percentage of TET can be as high as 45% (Test 1, $n = 650$), it is between 5 and 13% for $n \leq 150$ for all Tests 1-6.

Finally, we consider the Maple library procedures. For all Tests (other than Test 3), $n \leq 150$, the number of GC calls is uniformly low (less than 3 per test) accounting for a maximum of 10% of TET. In the case of Test 3, the number of GC calls steadily increases with n reaching 33 per test for $n = 70$; interestingly, the time spent in GC remains constant at about 20% of TET.

6.5. Floating Point Performance

As far as floating point calculations are concerned neither the KA or ESA are competitive with, for example, the method proposed by [27]. In addition, both KA and ESA can have problems with an uncontrollable increase in the intermediate values produced during the computation.

When attempting to generate the inverse of the matrix defined as Test 1 (Section 6.4 above) Fortran double precision versions of KA and ESA both failed through floating-point overflows for $n = 193$ and $n = 195$ respectively. For KA the overflow occurs when computing one of the temporary values used in calculating the $D(a, b; i)$ values and for ESA when computing the circulant values.

7. Conclusions and Future Work

We have shown how backward continued fractions may be used to obtain explicit formulae for the elements of the inverse of a periodic, tridiagonal matrix. From these we have constructed an efficient algorithm and an implementation in Maple.

The performance results show how difficult it is to identify an algorithm and its implementation that will perform well on general algebraic problems. As far as operation counts are concerned the proposed method should be more efficient than both the improved El-Shehawey algorithm [25] (see Section 6.1) and the Maple library procedures. However our performance results show that there is no clear 'best' procedure when comparing execution times.

That said the performance comparisons have shown that our proposed method would be a worthwhile first choice as an inversion procedure for both tridiagonal and periodic, tridiagonal matrices. Our implementation is also shown to be effective for solving problems of order up to several hundred in a relatively short time on a moderately powerful laptop.

Both our implementation and our improved version of the El-Shehawey et al code show that efficiency gains can be made over Maple library routines for computing inversions of specialized matrices.

In the future we intend to implement several other published algorithms for computing the inverse of both tridiagonal and periodic, tridiagonal matrices, and to exercise them on an extended set of test examples. We also intend to perform detailed profiling analyses of these implementations in an attempt to discover exactly where execution time is being spent and the circumstances under which excessive run times occur. These insights would then be used to identify potential improvements to the implementations and, perhaps, an optimal coding for computing the inverse of these particular matrices.

ALGORITHM 1: Pseudocode for ESA using temporary storage to save reusable computations. This implementation uses $O(n^2)$ arithmetic operations rather than $O(n^3)$ in the original presentation.

```
Procedure $ElS(x, y, z, \delta, \mu)$;
Input: $x_{1..n}$, $y_{1..n-1}$, $z_{1..n-1}$, $\delta$, $\mu$
Output: $sing$, $det$, $A^{-1}$
#
$sing = FALSE$; $c_{1..n} = [y_{1..n-1}, \mu]$;  $d_{1..n} = [\delta, z_{1..n-1}]$;
# Compute all circulant values
$\left(\Delta_i^i = x_i; \Delta_i^{i-1} = 1; \Delta_{i-1}^i = 0; i = 1..n\right)$;
$\Delta_{n+1}^n = 1$ ;
$\left(\Delta_1^j = x_j\Delta_1^{j-1} - d_j c_{j-1}\Delta_1^{j-2}, j = 2..n-1\right)$;
for $i = 2,\ (n-1)$ do
    $\left(\Delta_i^j = x_j\Delta_i^{j-1} - d_j c_{j-1}\Delta_i^{j-2}, j = i+1..n\right)$;
end
$p_{0..n} = vecProd1ton(c, 1, n); q_{0..n} = vecProd1ton(-d, 1, n)$;
# Compute determinant and deal with possible singular matrix
$det = x_n\Delta_1^{n-1} - \delta\mu\Delta_2^{n-1} - d_n c_{n-1}\Delta_1^{n-2} - (-1)^n p_n - q_n$ ;
if $det == 0$ then  return $(TRUE, det, NULL)$
$s_{1..n} = vecProdkton(-d, 1, n)$;
# Form the upper triangular portion of the inverse
for $k = 1,\ n$ do
    $[f, g, h] = \left[\Delta_1^{k-1}, \Delta_2^{k-1}, q_k\right] / det$;
    $r_{k-1..n-1} = vecProd1ton(c, k, n-1)$;
    $sgn = 1$;
    for $j = k,\ n$ do
        $A_{k,j}^{-1} = sgn \times r_{j-1}\left(f\Delta_{j+1}^n - g\delta\mu\Delta_{j+1}^{n-1}\right) + hs_j\Delta_{k+1}^{j-1}$ ;
        $sgn = -sgn$;
    end
end
# Form the lower triangular portion of the inverse
$r_{1..n} = vecProdkton(c, 1, n)$;
for $k = 2,\ n$ do
    $[f, g, h] = \left[\Delta_{k+1}^n, \Delta_{k+1}^{n-1}, r_{k-1}\right] / det$ ;
    $s_{2..k} = vecProdkton(-d, 2, k); s_1 = -d_2 s_2$;
    $sgn = 1$;
    if $n + k \mod 2 == 0$ then $sgn = -1$;
    for $j = 1,\ k-1$ do
        $A_{k,j}^{-1} = s_j\left(f\Delta_1^{j-1} - g\delta\mu\Delta_2^{j-1}\right) + sgn \times hp_{j-1}\Delta_{j+1}^{k-1}$;
        $sgn = -sgn$;
    end
end
return $(FALSE, det, A^{-1})$;
```

ALGORITHM 2: Subsidiary procedures used by the efficient ESA implementation

```
**Procedure** $vecProdkton(v, i, j)$;
**Input**: $v_{i..j}, i, j$
**Output**: $r_j = 1; r_k = \prod_{m=k+1}^{j} v_m, k = i..j-1$
$r_{i-1} = 1$;
**for** $k = j-1,\ i,\ -1$ **do**
    $r_k = r_{k+1} v_{k+1}$;
**end**
**return** $r_{i..j}$;
**Procedure** $vecProd1ton(v, i, j)$;
**Input**: $v_{i..j}, i, j$
**Output**: $r_{i-1} = 1; r_k = \prod_{m=i}^{k} v_m, k = i..j$
$r_{i-1} = 1$;
**for** $k = i,\ j$ **do**
    $r_k = r_{k-1} v_k$;
**end**
**return** $r_{i-1..j}$;
```

ALGORITHM 3: Pseudocode for the KA using temporary storage to save reusable computations.

```
Procedure $KA(x, y, z, \delta, \mu)$;
Input: $x_{1..n}$, $y_{1..n-1}$, $z_{1..n-1}$, $\delta$, $\mu$
Output: $sing$, $det$, $A^{-1}$
#
$sing = FALSE; sg = -1$;
if $n \mod 2 == 0$ then $sg = 1$;
$(b_0, w_0, p_0, p_1) = (1, 1, 1, x_1)$;
# $b[i] \leftarrow y_i!$, $w[i] \leftarrow z_i!$, $p[i]$ as defined by (3.2)
for $i = 1,\ n - 1$ do
    $b_i = b_{i-1}y_i$; $w_i = w_{i-1}z_i$;
    $p_{i+1} = x_{i+1}p_i - y_i z_i p_{i-1}$;
end
# Using (4.1), $t[i] \leftarrow T(n, i)$ and, using (4.2), $q[i] \leftarrow q_{n,i}$
$(t_n, q_n) = (0, sg)$;
for $i = n - 1,\ 1,\ -1$ do
    $t_i = t_{i+1} + b_{i-1}w_{i-1}/(p_i p_{i-1})$;
    $q_i = (\delta p_{n-1}t_i + sg \times b_{n-1}/b_{i-1})$;
end
# Using (3.8) and (4.1)
$det = p_n - \delta\mu t_1 p_{n-1} - sg \times (\mu b_{n-1} + \delta w_{n-1})$;
if $det == 0$ then return $(TRUE, 0, NULL)$;
$r_{1..n} = det/b_{0..n-1}$; $f_{1..n} = \mu q_{1..n}$; $g_{1..n} = (sg \times w_{n-1}/p_{n-1})q_{1..n}$;
$h_{1..n-1} = r_{1..n-1} + f_{1..n-1}$; $r_{1..n} = f_{1..n}t_{1..n} + g_{1..n}$;
$c_{1..n} = p_{0..n-1}/det$; $d_{1..n} = p_{0..n-1}/w_{0..n-1}$;
$v_{1..n+1} = 1$;
for $i = 2,\ n + 1,\ 2$ do
    $v_i = -1$;
end
# The leading three multipliers $d_j v_{1+jval..j-1+jval} c_{1..j-1}$ in the next two
# for loops form the leading multiplier of the $D(a, b; i)$ values in (4.4)
$jval = 1$;
for $j = 2,\ n$ do
    $A^{-1}_{1..j-1,j} = d_j v_{1+jval..j-1+jval} c_{1..j-1}\,(t_j h_{1..j-1} + g_{1..j-1})$;
    $jval = 1 - jval$;
end
$jval = 0$;
for $j = 1,\ n$ do
    $A^{-1}_{j..n,j} = d_j v_{j+jval..n+jval} c_{j..n}\,(t_j f_{j..n} + r_{j..n})$;
    $jval = 1 - jval$;
end
return $(FALSE, det, A^{-1})$
```

References

[1] M. G. Beker, G. Cella, R. DeSalvo, M. Doets, H. Grote, J. Harms, E. Hennes, V. Mandic, D. S. Rabeling, J. F. J. van den Brand, C. M. van Leeuwen, Improving the sensitivity of future GW observatories in the 1–10 Hz band: Newtonian and seismic noise, General Relativity and Gravitation 43 (2) (2011) 623–656.

[2] R. Bustos-Marún, E. Coronado, H. Pastawski, Buffering plasmons in nanoparticle wave guides at the virtual-localized transition, Phys. Rev. B 82 (3) (2010) 035434.

[3] B. Choudhury, Diffusion of heat in multidimensional composite spherical body, IMA J. Appl. Math. 78 (3) (2013) 474–493.

[4] S. Dursun, A. Grigoryan, Nonlinear l_2-by-3 transform for PAPR reduction in OFDM systems, Computers & Electrical Engin. 36 (6) (2010) 1055–1065.

[5] P.-L. Giscard, S. J. Thwaite, D. Jaksch, Evaluating matrix functions by resummations on graphs: The method of path-sums, SIAM. J. Matrix Anal. & Appl. 34 (2) (2013) 445–469.

[6] N. Huang, An enhanced Hill cipher and its application in software copy protection, J. Networks 9 (10) (2014) 2582–2590.

[7] S. Ma, L. Yang, A jump-detecting procedure based on spline estimation, J. Nonparametric Stat. 23 (1) (2011) 67–81.

[8] N. Perel, U. Yechiali, The Israeli queue with a general group-joining policy, Ann. Oper. Res. (2015) 1–34doi:10.1007/s10479-015-1942-1.

[9] A. Bunse-Gerstner, R. Byers, V. Mehrmann, A chart of numerical methods for structured eigenvalue problems, SIAM J. Matrix Anal. Appl. 13 (2) (1992) 419–453.

[10] C. Fischer, R. Usmani, Properties of some tridiagonal matrices and their application to boundary value problems, SIAM J. Numer. Anal. 6 (1) (1969) 127–142.

[11] E. Kılıç, D. Tasci, Factorizations and representations of the backward second-order linear recurrences, J. Comput. Appl. Math. 201 (1) (2007) 182–197.

[12] R. Mattheij, M. Smooke, Estimates for the inverse of tridiagonal matrices arising in boundary-value problems, Linear Algebra Appl. 73 (1986) 33–57.

[13] G. Meurant, A review on the inverse of symmetric tridiagonal and block tridiagonal matrices, SIAM J. Matrix Anal. Appl. 13 (3) (1992) 707–728.

[14] S.-F. Xu, On the Jacobi matrix inverse eigenvalue problem with mixed given data, SIAM J. Matrix Anal. Appl. 17 (3) (1996) 632–639.

[15] R. Burden, J. Fairs, A. Reynolds, Numerical analysis, 2nd Edition, Weber & Schmidt, Boston, MA, 1982.

[16] C. da Fonseca, On the eigenvalues of some tridiagonal matrices, J. Comput. Appl. Math. 200 (1) (2007) 283–286.

[17] M. El-Mikkawy, A. Karawia, Inversion of general tridiagonal matrices, Appl. Math. Letters 19 (8) (2006) 712–720.

[18] D. Evans, S. Okolie, A quotient-difference algorithm for the determination of eigenvalues of periodic tridiagonal matrices, Comput. & Math. Appl. 8 (2) (1982) 157–164.

[19] W. Hager, Applied numerical linear algebra, Prentice-Hall International Editions, Englewood Cliffs, NJ, 1988.

[20] R. Hagger, The eigenvalues of tridiagonal sign matrices are dense in the spectra of periodic tridiagonal sign operators, J. Funct. Analysis 269 (5) (2015) 1563–1570.

[21] A. Yalçiner, The LU factorizations and determinants of the k-tridiagonal matrices, Asian-Eur. J. Math. 4 (1) (2011) 187–197.

[22] W.-C. Yueh, S. Cheng, Explicit eigenvalues and inverses of several Toeplitz matrices, The ANZIAM Journal 4 (1) (2006) 73–97.

[23] M. Chawla, R. Khazal, A parallel elimination method for 'periodic' tridiagonal systems, Inter. J. Computer Math. 79 (4) (2002) 473–484.

[24] M. El-Mikkawy, A new computational algorithm for solving periodic tri-diagonal linear systems, Appl. Math. Comput. 161 (2) (2005) 691–696.

[25] M. El-Shehawey, G. El-Shreef, A. Al-Henawy, Analytical inversion of general periodic tridiagonal matrices, J. Math. Anal. Appl. 345 (2008) 123–134.

[26] T. Sogabe, New algorithms for solving periodic tridiagonal and periodic pentadiagonal linear systems, Appl. Math. Comput. 202 (2) (2008) 850–856.

[27] C. Temperton, Algorithms for the solution of cyclic tridiagonal systems, Journal of Computational Physics 19 (3) (1975) 317 – 323. doi:http://dx.doi.org/10.1016/0021-9991(75)90081-9.
URL `http://www.sciencedirect.com/science/article/pii/0021999175900819`

[28] M. Dow, Explicit inverses of Toeplitz and associated matrices, ANZIAM Journal 44 (2008) E185–E215.
URL `http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/493`

[29] R. Mallik, The inverse of a tridiagonal matrix, Linear Algebra Appl. 253 (1-3) (2001) 109–139.

[30] Maplesoft, a division of Waterloo Maple Inc., Maple 2017, waterloo, Ontario, Canada.

[31] Wolfram Research, Inc., Mathematica, Version 11.2, champaign, IL, 2017.

[32] E. Kılıç, Explicit formula for the inverse of a tridiagonal matrix by backward continued fractions, Appl. Math. Comput. 197 (1) (2008) 345–357.

[33] W. Jones, W. Thron, Continued fraction, analytic theory and applications, Addison Wesley, Reading, MA, 1980.

[34] H. Zhao, G. Zhu, P. Xiao, A backward three-term recurrence relation for vector valued continued fractions and its applications, J. Comput. Appl. Math. 142 (2) (2002) 389–400.

[35] D. H. Lehmer, D. M. Smiley, M. F. Smiley, J. Williamson, Solution to problem E710, The American Mathematical Monthly 53 (9) (1946) 534–535. doi:10.2307/2305078.