CANADIAN APPLIED
MATHEMATICS QUARTERLY
Volume 12, Number 3, Fall 2004

A CLASSIC TRENCH-TYPE ALGORITHM FOR SKEW-SYMMETRIC TOEPLITZ MATRICES

IYAD T. ABU-JEIB

ABSTRACT. We present an efficient classic Trench-type algorithm for the inversion of real skew-symmetric Toeplitz matrices of even order.

1 Introduction We extend Trench's algorithm (see [**11**]) for symmetric Toeplitz matrices to skew-symmetric Toeplitz matrices. In our algorithm, we present an $O(n^2)$ method to find the inverse of a finite even-order real skew-symmetric Toeplitz matrix (with some restrictions). Our algorithm is simple and it uses very similar techniques to those used by Trench. It is easy to derive and easy to implement. In addition, we do not require the matrix to be positive definite. We recall here that well-known researchers thought that the classic Durbin's, Levinson's (see [**3, 6, 7**]), and Trench's algorithms can not be generalized to skew-symmetric Toeplitz matrices. But, we managed to generalize them. Our algorithm was tested on skew-symmetric Toeplitz matrices that appear in Sinc methods. It was tested on matrix S_n of $I_n^{(-1)}$, where

$$I_n^{(-1)} = [\eta_{ij}]_{i,j=1}^{n}$$

where $\eta_{ij} = e_{i-j}$, $e_k = 1/2 + s_k$, and $s_k = \int_0^k \operatorname{sinc}(x)\,dx$, where

$$\operatorname{sinc}(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x} & \text{for } x \neq 0 \\ 1 & \text{for } x = 0. \end{cases}$$

Thus, $I_n^{(-1)}$ can be expressed in the form

$$I_n^{(-1)} = \left[\frac{1}{2}\right] + S_n,$$

AMS subject classification: 65F05, 15A57.
Keywords: skew-symmetric Toeplitz matrix, Trench algorithm, inverse, skew-centrosymmetric.

where $[1/2]$ is the $n \times n$ matrix whose elements are all equal to $1/2$. We tested the algorithm also on matrix $I_n^{(1)}$ of Sinc methods. $I_n^{(1)}$ is an $n \times n$ skew-symmetric Toeplitz matrix defined as follows

$$I_n^{(1)} = \begin{bmatrix} 0 & -1 & \frac{1}{2} & \dots & \frac{(-1)^{n-1}}{n-1} \\ 1 & 0 & -1 & \dots & \frac{(-1)^{n-2}}{n-2} \\ -\frac{1}{2} & 1 & 0 & \dots & \frac{(-1)^{n-3}}{n-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \frac{(-1)^{n}}{n-1} & \frac{(-1)^{n-1}}{n-2} & \frac{(-1)^{n-2}}{n-3} & \dots & 0 \end{bmatrix}.$$

For more about the matrices of Sinc methods and about Sinc methods, see [**2, 4, 5, 8, 9, 10**].

2 Preliminaries We employ the following notation. We denote the transpose of a matrix A by A^T. As usual, I_k denotes the $k \times k$ identity matrix. When counting flops, we treat addition/subtraction the same as multiplication/division. By the *main counterdiagonal* (or simply *counterdiagonal*) of a square matrix we mean the positions which proceed diagonally from the last entry in the first row to the first entry in the last row.

Definition 2.1. The *counteridentity* matrix, denoted J, is the square matrix whose elements are all equal to zero except those on the counterdiagonal, which are all equal to 1.

We note that multiplying a matrix A by J from the left results in reversing the rows of A and multiplying A by J from the right results in reversing the columns of A. Throughout this paper, we will denote the $k \times k$ counteridentity matrix by J_k. Note that multiplying a matrix or a vector by J does not contribute to the running time.

Definition 2.2. A matrix A is *skew-centrosymmetric* if $JAJ = -A$, *persymmetric* if $JAJ = A^T$, and *Toeplitz* if the elements along each diagonal are equal.

Note that skew-symmetric Toeplitz matrices are skew-symmetric skew-centrosymmetric and if an $n \times n$ matrix A is persymmetric, then $A(i, j) =$

$A(n-j+1, n-i+1)$. Note also that if T_n is an $n \times n$ skew-symmetric Toeplitz matrix, then T_n has the following form

$$T_n = \begin{bmatrix} 0 & \sigma_1 & \sigma_2 & \dots & \sigma_{n-1} \\ -\sigma_1 & 0 & \sigma_1 & \dots & \sigma_{n-2} \\ -\sigma_2 & -\sigma_1 & 0 & \dots & \sigma_{n-3} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ -\sigma_{n-1} & -\sigma_{n-2} & -\sigma_{n-3} & \dots & 0 \end{bmatrix}.$$

The above form is the form we will refer to in the next section. Note that the first row (excluding the first element) *generates* (determines) T_n, i.e., the vector $h_n = [\sigma_1, \sigma_2, \cdots, \sigma_{n-1}]^T$ is a *generator* of T_n. For example, in matrix S_n of $I_n^{(-1)}$ described in the introduction, $\sigma_k = \int_0^{-k} \operatorname{sinc}(x)dx$, and in matrix $I_n^{(1)}$, $\sigma_k = (-1)^k/k$.

Definition 2.3. Let A be an $n \times n$ matrix. The *leading principal matrix* of A of order k is the matrix formed from A by deleting the last $n-k$ columns and the last $n-k$ rows of A.

3 The algorithm Throughout the rest of the paper, let k be even, and let T_k be a $k \times k$ real skew-symmetric Toeplitz matrix and assume T_i, $\forall i \in \{2, 3, \cdots, k\} \cap 2\mathbb{Z}$, is nonsingular (i.e., all leading principal matrices of T_k of even order are nonsingular). We recall that Trench has similar restrictions in his algorithm. We note also that it happens sometimes that all even-order matrices of a family of skew-symmetric Toeplitz matrices are non-singular as it is the case with matrix S_n of $I_n^{(-1)}$ and matrix $I_n^{(1)}$. For the proofs, see [**2, 4**]. Note that odd-order skew-symmetric (and odd-order skew-centrosymmetric matrices) are singular, and hence, it is essential to have a two-step algorithm that skips the odd-order matrices. Thus, our Durbin-type algorithm is a two-step algorithm because it moves from order k to order $k+2$ instead of moving from order k to order $k+1$. Now we repeat some of the steps mentioned in our paper [**1**] that presents Durbin-type and Levinson-type algorithms for skew-symmetric Toeplitz matrices. First, note that T_{k+2} can be written as

$$T_{k+2} = \begin{bmatrix} T_k & J_k R_k \\ -R_k^T J_k & T_2 \end{bmatrix},$$

where

$$R_k = \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \\ \vdots & \vdots \\ \sigma_k & \sigma_{k+1} \end{bmatrix}.$$

Once again, in each step we will move from T_k to T_{k+2} instead of T_{k+1}. We start with T_2. Now, we extend Durbin's algorithm. If we know the solution of $T_k Y = R_k$, where Y is $k \times 2$, then we can know the solution of

$$\begin{bmatrix} T_k & J_k R_k \\ -R_k^T J_k & T_2 \end{bmatrix} \begin{bmatrix} Z \\ W \end{bmatrix} = \begin{bmatrix} R_k \\ S_k \end{bmatrix},$$

where Z is $k \times 2$, W is 2×2, and $S_k = \left[\begin{smallmatrix} \sigma_{k+1} & \sigma_{k+2} \\ \sigma_{k+2} & \sigma_{k+3} \end{smallmatrix}\right]$. Note that T_k and J_k are $k \times k$, R_k and Z are $k \times 2$, and T_2 and S_k are 2×2. Now note that

$$T_k Z + J_k R_k W = R_k \quad \text{and} \quad -R_k^T J_k Z + T_2 W = S_k.$$

Thus, $Z = Y + J_k Y W$ and $W = (T_2 - {R_k}^T Y)^{-1}(S_k + {R_k}^T J_k Y)$. Note that $T_2 - {R_k}^T Y$ is nonsingular since

$$H_k^T T_{k+2} H_k = \begin{bmatrix} T_k & 0 \\ Y^T J_k T_k - R_k^T J_k & T_2 - R_k^T Y \end{bmatrix},$$

where

$$H_k = \begin{bmatrix} I_k & J_k Y \\ 0 & I_2 \end{bmatrix}.$$

(Note that $T_k J_k Y + J_k R_k = 0$ and $Y^T J_k T_k J_k Y + Y^T R_k = 0$ also.) Hence, $\det(T_{k+2}) = \det(T_k) \cdot \det(T_2 - R_k^T Y)$. Now since we are assuming T_{k+2} is nonsingular, $T_2 - R_k^T Y$ is nonsingular.

Now we reduce the cost of computing $R_k^T Y_k$. Here we will use the same notation as before (but we will replace Y by Y_k, Z by Z_k and W by W_k). Now consider

$$R_k^T Y_k = \begin{bmatrix} R_{k-2}^T & S_{k-2}^T \end{bmatrix} \begin{bmatrix} Z_{k-2} \\ W_{k-2} \end{bmatrix}.$$

But, $Z_{k-2} = Y_{k-2} + J_{k-2} Y_{k-2} W_{k-2}$. Thus,

$$R_k^T Y_k = R_{k-2}^T Y_{k-2} + (T_2 - R_{k-2}^T Y_{k-2}) W_{k-2}^2.$$

Therefore, we can use the previously computed values of W and R^TY to compute the new value of R^TY which we will call E.

Classic Durbin-type algorithm for skew-symmetric Toeplitz matrices.

Input: m (an even positive integer), $\sigma = [\sigma_1 \ \sigma_2 \ \cdots \ \sigma_{m+1}]^T$ (a generator of a real skew-symmetric Toeplitz matrix T_{m+2}).

$$Y_2 = \begin{bmatrix} -\sigma_2/\sigma_1 & -\sigma_3/\sigma_1 \\ 1 & \sigma_2/\sigma_1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0 & \sigma_1 \\ -\sigma_1 & 0 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \end{bmatrix}$$

$$E = R_2^T Y_2$$

$$W = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

for $k = 2, \cdots, m-2$, *step 2*

Let J_k be the counteridentity matrix of order k.

$$R_k = \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \\ \vdots & \vdots \\ \sigma_k & \sigma_{k+1} \end{bmatrix}$$

$$S_k = \begin{bmatrix} \sigma_{k+1} & \sigma_{k+2} \\ \sigma_{k+2} & \sigma_{k+3} \end{bmatrix}$$

$$E = E + (T_2 - E)W^2$$

$$P_k = (T_2 - E)^{-1}$$

$$W = P_k(S_k + R_k^T J_k Y_k)$$

$$Z_k = Y_k + J_k Y_k W$$

$$Y_{k+2} = \begin{bmatrix} Z_k \\ W \end{bmatrix}$$

end for

Output: Y_m (the solution of $T_m Y_m = R_m$, where T_m is the skew-symmetric Toeplitz matrix whose generator is $[\sigma_1 \ \sigma_2 \ \cdots \ \sigma_{m-1}]^T$, and R_m is as defined above).

It is clear that the number of flops of the previous algorithm is $4m^2 + O(m)$. We remind the reader that we define a flop to be one addition *or* one multiplication, while some people define it to be one addition *and* one multiplication. If we define the flop to be one addition and one multiplication, then the running time of the previous algorithm will be almost half of the running time we have above.

Now we derive a classic Trench-type algorithm. Assume we have the solution (from the previous algorithm) of $T_{n-2}Y = R_{n-2}$, where Y is $(n-2) \times 2$. Now, we want to find the inverse of T_n. Recall that T_n can be written as

$$T_n = \begin{bmatrix} T_{n-2} & J_{n-2}R_{n-2} \\ -R_{n-2}^T J_{n-2} & T_2 \end{bmatrix},$$

where

$$R_{n-2} = \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \\ \vdots & \vdots \\ \sigma_{n-2} & \sigma_{n-1} \end{bmatrix}.$$

Now we can write

$$T_n^{-1} = \begin{bmatrix} A & B \\ -B^T & C \end{bmatrix},$$

where A is $(n-2) \times (n-2)$, B is $(n-2) \times 2$, and C is 2×2. Thus, we have to find A, B and C. But,

$$T_n \begin{bmatrix} B \\ C \end{bmatrix} = \begin{bmatrix} 0 \\ I_2 \end{bmatrix}.$$

Hence, $B = J_{n-2}YC$ and $C = (T_2 - R_{n-2}^T Y)^{-1}$. Note that C is 2×2 and note also that we proved earlier that $T_2 - R_{n-2}^T Y$ is invertible. Also, we have $T_{n-2}A - J_{n-2}R_{n-2}B^T = I_{n-2}$. Thus, $A = T_{n-2}^{-1} - J_{n-2}YB^T$. Let $M = J_{n-2}YB^T$. Then,

$$A(i,j) = T_{n-2}^{-1}(i,j) - M(i,j).$$

Now since A and T_{n-2}^{-1} are persymmetric,

$$A(i,j) = T_{n-2}^{-1}(n-j-1, n-i-1) - M(i,j),$$

and

$$A(i,j) = A(n-j-1, n-i-1) + M(n-j-1, n-i-1) - M(i,j).$$

But, T_n^{-1} is also persymmetric. Hence, $A(i,j) = A(n-j+1, n-i+1)$, which implies $A(i+2, j+2) = A(n-j-1, n-i-1)$. Therefore,

$$A(i+2, j+2) = A(i,j) - M(n-j-1, n-i-1) + M(i,j),$$

or equivalently,

$$A(i,j) = A(i-2, j-2) - M(n-j+1, n-i+1) + M(i-2, j-2).$$

Now note that B and C determine the last two columns of T_n^{-1} and also note that the first row of T_n^{-1} is the transpose of the reverse of the last column of T_n^{-1} and the second row of T_n^{-1} is the transpose of the reverse of the column of T_n^{-1} that proceeds the last column. Thus, the first two rows of A can be determined as follows:

$A(1,1) = A(2,2) = 0$
$A(1,2) = C(1,2)$
$A(2,1) = C(2,1)$
for $j = 3, \cdots, n-2$
$\quad A(1,j) = B(n-j+1, 2)$
$\quad A(2,j) = B(n-j+1, 1)$
end for

Now, we can determine the remaining elements of A, and hence, of T_n^{-1} from the first two rows. Note that it suffices to determine the quarter of T_n^{-1} that is above the main diagonal and which lies between the main diagonal and the main counterdiagonal. The remaining part of T_n^{-1} that lies above the main diagonal can be determined from the persymmetry property of T_n^{-1} and the half of T_n^{-1} that lies below the main diagonal can be determined from the skew-symmetry property of T_n^{-1}. Finally, note that the main diagonal of T_n^{-1} consists only of zeros. Thus, after assigning the elements of the first two rows of A, it suffices to execute the following:

for $i = 3, \cdots, \lfloor \frac{n-1}{2} \rfloor + 1$
$\quad$ for $j = i+1, \cdots, n-i+1$
$\quad\quad A(i,j) = A(i-2, j-2) - M(n-j+1, n-i+1)$
$\quad\quad\quad\quad + M(i-2, j-2).$
$\quad$ *end for*

end for

Thus, our $O(n^2)$ Trench-type algorithm is the following:

Classic Trench-type algorithm for skew-symmetric Toeplitz matrices.

Input: n (an even positive integer), $\sigma = [\sigma_1\ \sigma_2\ \cdots\ \sigma_{n-1}]^T$ (a generator of a real skew-symmetric Toeplitz matrix T_n).

$$T_2 = \begin{bmatrix} 0 & \sigma_1 \\ -\sigma_1 & 0 \end{bmatrix}$$

$$m = n - 2$$

$$R = \begin{bmatrix} \sigma_1 & \sigma_2 \\ \sigma_2 & \sigma_3 \\ \vdots & \vdots \\ \sigma_{n-2} & \sigma_{n-1} \end{bmatrix}$$

$$Y = Durbin(m, \sigma)$$

Let J be the counteridentity matrix of order m

$$C = (T_2 - R^T\ Y)^{-1}$$

$$B = J\ Y\ C$$

$$M = J\ Y\ B^T$$

Now declare A to be an $m \times m$ two-dimensional array

$A(1,1) = A(2,2) = 0$

$A(1,2) = C(1,2)$

$A(2,1) = C(2,1)$

$v = n + 1$

for $j = 3, \cdots, m$

$k = v - j$

$A(1,j) = B(k,2)$

$A(2,j) = B(k,1)$

end for

$\gamma = \lfloor \frac{n-1}{2} \rfloor + 1$

for $i = 3, \cdots, \gamma$

$k = v - i$

$p = i - 2$
$w = i + 1$
for $j = w, \cdots, k$
$q = j - 2$
$A(i,j) = A(p,q) - M(v - j, k) + M(p,q)$
end for
end for

It is easy to see that the running time of the previous algorithm is about $5n^2 + O(n)$. Note that the remaining elements of T_n^{-1} are known. If the reader is interested in determining them, here is how:

(1) Let M be the matrix whose first $n-2$ rows are the rows of B and whose last two rows are the rows of C. Then, the last two columns of T_n^{-1} are the rows of M (in order).
(2) Let $T_n^{-1}(i,j) = A(i,j)$ for every element of A we determined above.
(3) The first $n-2$ elements of each of the last two rows of T_n^{-1} are the rows (in order) of $-B^T$.
(4) The remaining elements of the main diagonal are zeros.
(5) Execute the code:

for $i = 3, \cdots, m$
$k = n - i + 1$
for $j = k, \cdots, m$
$T_n^{-1}(i,j) = T_n^{-1}(n - j + 1, k)$
end for
end for

(6) Execute the code:

for $i = 2, \cdots, m$
$k = i - 1$
for $j = 1, \cdots, k$
$T_n^{-1}(i,j) = -T_n^{-1}(j,i)$
end for
end for

4 An example and an Octave program

4.1 Example In the following example we find (calculations are done by Octave which is a math-oriented programming language similar to MATLAB) the inverse of $I_8^{(1)}$ (see the introduction) whose generator σ

is

$$\sigma = \begin{bmatrix} -1 \\ 1/2 \\ -1/3 \\ 1/4 \\ -1/5 \\ 1/6 \\ -1/7 \end{bmatrix}.$$

Note that the inputs to the algorithm are 8 and σ.

$$Y = \begin{bmatrix} 0.55361 & -0.39618 \\ 0.92117 & -0.47462 \\ 0.57966 & -0.34580 \\ 0.92117 & -0.50067 \\ 0.55361 & -0.31735 \\ 1.00000 & -0.55361 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} -1.00000 & 0.50000 \\ 0.50000 & -0.33333 \\ -0.33333 & 0.25000 \\ 0.25000 & -0.20000 \\ -0.20000 & 0.16667 \\ 0.16667 & -0.14286 \end{bmatrix}$$

$$C = \begin{bmatrix} 6.8672e-17 & 8.9273e-01 \\ -8.9273e-01 & 8.8480e-17 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.49422 & 0.89273 \\ 0.28331 & 0.49422 \\ 0.44697 & 0.82235 \\ 0.30870 & 0.51747 \\ 0.42371 & 0.82235 \\ 0.35368 & 0.49422 \end{bmatrix}$$

$$M = \begin{bmatrix} M_1 & M_2 \end{bmatrix}$$

where

$$M_1 = \begin{bmatrix} -1.6507e-17 & 9.7034e-03 & -8.2948e-03 \\ -9.7034e-03 & -7.4945e-18 & -1.3531e-02 \\ 8.2948e-03 & 1.3531e-02 & 3.3041e-17 \\ -2.2227e-02 & -6.6803e-03 & -2.5283e-02 \\ 3.1551e-02 & 2.6405e-02 & 2.1423e-02 \\ -8.0075e-02 & -3.8958e-02 & -7.8355e-02 \end{bmatrix},$$

$$M_2 = \begin{bmatrix} 2.2227e-02 & -3.1551e-02 & 8.0075e-02 \\ 6.6803e-03 & -2.6405e-02 & 3.8958e-02 \\ 2.5283e-02 & -2.1423e-02 & 7.8355e-02 \\ 7.3726e-18 & -3.8764e-02 & 3.4111e-02 \\ 3.8764e-02 & 4.9060e-17 & 9.1230e-02 \\ -3.4111e-02 & -9.1230e-02 & 7.9147e-18 \end{bmatrix}.$$

The inverse before using persymmetry and skew-symmetry (note that the default elements are zeros) is:

$$\begin{bmatrix} N_1 & N_2 \end{bmatrix}$$

where

$$N_1 = \begin{bmatrix} 0.00000 & 0.89273 & 0.49422 & 0.82235 \\ -0.89273 & 0.00000 & 0.35368 & 0.42371 \\ 0.00000 & 0.00000 & 0.00000 & 0.81120 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ -0.49422 & -0.28331 & -0.44697 & -0.30870 \\ -0.89273 & -0.49422 & -0.82235 & -0.51747 \end{bmatrix},$$

$$N_2 = \begin{bmatrix} 0.51747 & 0.82235 & 0.49422 & 0.89273 \\ 0.30870 & 0.44697 & 0.28331 & 0.49422 \\ 0.45181 & 0.76623 & 0.44697 & 0.82235 \\ 0.37891 & 0.00000 & 0.30870 & 0.51747 \\ 0.00000 & 0.00000 & 0.42371 & 0.82235 \\ 0.00000 & 0.00000 & 0.35368 & 0.49422 \\ -0.42371 & -0.35368 & 0.00000 & 0.89273 \\ -0.82235 & -0.49422 & -0.89273 & 0.00000 \end{bmatrix}.$$

The inverse after using persymmetry and before using skew-symmetry is

$$\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$$

where

$$Q_1 = \begin{bmatrix} 0.00000 & 0.89273 & 0.49422 & 0.82235 \\ -0.89273 & 0.00000 & 0.35368 & 0.42371 \\ 0.00000 & 0.00000 & 0.00000 & 0.81120 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 0.00000 & 0.00000 \\ -0.49422 & -0.28331 & -0.44697 & -0.30870 \\ -0.89273 & -0.49422 & -0.82235 & -0.51747 \end{bmatrix},$$

$$Q_2 = \begin{bmatrix} 0.51747 & 0.82235 & 0.49422 & 0.89273 \\ 0.30870 & 0.44697 & 0.28331 & 0.49422 \\ 0.45181 & 0.76623 & 0.44697 & 0.82235 \\ 0.37891 & 0.45181 & 0.30870 & 0.51747 \\ 0.00000 & 0.81120 & 0.42371 & 0.82235 \\ 0.00000 & 0.00000 & 0.35368 & 0.49422 \\ -0.42371 & -0.35368 & 0.00000 & 0.89273 \\ -0.82235 & -0.49422 & -0.89273 & 0.00000 \end{bmatrix}.$$

The inverse after using skew-symmetry is

$$\begin{bmatrix} G_1 & G_2 \end{bmatrix}$$

where

$$G_1 = \begin{bmatrix} 0.00000 & 0.89273 & 0.49422 & 0.82235 \\ -0.89273 & 0.00000 & 0.35368 & 0.42371 \\ -0.49422 & -0.35368 & 0.00000 & 0.81120 \\ -0.82235 & -0.42371 & -0.81120 & 0.00000 \\ -0.51747 & -0.30870 & -0.45181 & -0.37891 \\ -0.82235 & -0.44697 & -0.76623 & -0.45181 \\ -0.49422 & -0.28331 & -0.44697 & -0.30870 \\ -0.89273 & -0.49422 & -0.82235 & -0.51747 \end{bmatrix},$$

$$G_2 = \begin{bmatrix} 0.51747 & 0.82235 & 0.49422 & 0.89273 \\ 0.30870 & 0.44697 & 0.28331 & 0.49422 \\ 0.45181 & 0.76623 & 0.44697 & 0.82235 \\ 0.37891 & 0.45181 & 0.30870 & 0.51747 \\ 0.00000 & 0.81120 & 0.42371 & 0.82235 \\ -0.81120 & 0.00000 & 0.35368 & 0.49422 \\ -0.42371 & -0.35368 & 0.00000 & 0.89273 \\ -0.82235 & -0.49422 & -0.89273 & 0.00000 \end{bmatrix}.$$

We note that the solution above is exactly the same as the solution obtained by using Maple. The two solutions even match for a much higher number of decimal places.

The one-norm of the difference between the inverse obtained above and the inverse obtained using Octave's inverse function is $1.8928e-15$ (note that $e-15$ in Octave means 10^{-15}).

4.2 An Octave program We note that we do not have to compute the counteridentity matrix, J, in the program below. We can write the program without it. We included it in the program for the sake of clarity. We note also that the program can be shortened if we use Octave's built-in functions and operators. But, we decided to write it as above to make it easy to understand for readers who do not know Octave.

```
#
1;
function J = Counter(n)
# usage: J = Counter(n)
# description: Creates the counteridentity matrix of order n.
J=zeros(n);
for i=1:n
        J(i,n-i+1)=1;
endfor;
endfunction
# ================
function Z = Durbin(sigma)
# description : Solves the system T_nY = R_n, where n is even and
# T_n is a real skew-symmetric Toeplitz matrix generated by
# [sigma(1) sigma(2) ... sigma(n-1)]^T.
# The input sigma=[sigma(1) sigma(2) ... sigma(n+1)]^T
# is the generator of T_{n+2}. It is an (n+1) by 1 column vector.
# R_n is the n by 2 matrix described in the paper.
# usage: Z = solve(sigma,D)
n = rows(sigma) - 1;
Y = [-sigma(2)/sigma(1),-sigma(3)/sigma(1);1,sigma(2)/sigma(1)];
T2 = [0,sigma(1);-sigma(1),0];
R = [sigma(1),sigma(2);sigma(2),sigma(3)];
E = R' * Y; # Note that the prime is used for transpose in Octave.
W = zeros(2,2);
for k=2:n-2
        if (rem (k, 2) != 0)
                continue;
```

```
            endif;
            R = zeros(k,2);
            for i=1:k
                    R(i,1)=sigma(i);
                    R(i,2) = sigma(i+1);
            endfor;
            S = zeros(2,2);
            S(1,1) = sigma(k+1);
            S(1,2) = sigma(k+2);
            S(2,1) = sigma(k+2);
            S(2,2) = sigma(k+3);
            J = Counter(k);
            E = E + (T2 - E) * W * W;
            P = inv(T2 - E);
            W = P * (S + R' * J * Y);
            Z = Y + J * Y * W;
            Y = [Z;W];
endfor;
Z = Y;
endfunction;
# ================
function Z = inverse(sigma)
# description : Finds the inverse of T_n where n is even, and
# T_n is a real skew-symmetric Toeplitz matrix generated by
# [sigma(1) sigma(2) ... sigma(n-1)]^T.
# The input sigma = [sigma(1) sigma(2) ... sigma(n-1)]^T
# is the generator of T_n. It is an (n-1) by 1 column vector.
# usage: Z = inverse(sigma)
Y = Durbin(sigma);
n = rows(sigma) + 1;
m = n - 2;
T2 = [0,sigma(1);-sigma(1),0];
R = zeros(m,2);
for i=1:m
            R(i,1)=sigma(i);
            R(i,2) = sigma(i+1);
endfor;
J = Counter(m);
C = inv(T2 - R' * Y); # Note that C is 2 × 2.
B = J * Y * C;
A = zeros(m,m);
```

```
M = J * Y * B';
A(1,1) = 0;
A(2,2) = 0;
A(1,2) = C(1,2);
A(2,1) = C(2,1);
v = n + 1;
for j=3:m
        k = v - j;
        A(1,j) = B(k,2);
        A(2,j) = B(k,1);
endfor;
gamma = floor((n-1)/2) + 1;
for i=3:gamma
        k = v - i;
        p = i - 2;
        w = i + 1;
        for j=w:k
                q = j - 2;
                A(i,j) = A(p,q) - M(v-j,k) + M(p,q);
        endfor;
endfor;
invT = [A,B;-B',C];
# Now determine the remaining elements of the half above
# the main diagonal by using the persymmetry property.
for i=3:m
        k = v - i;
        for j=k:m
                invT(i,j) = invT(v-j,k);
        endfor;
endfor;
# Now determine the half below the main diagonal by using the
# skew-symmetry property.
for i=2:m
        k = i - 1;
        for j=1:k
                invT(i,j) = -invT(j,i);
        endfor;
endfor;
Z = invT;
endfunction;
```

REFERENCES

1. I. T. Abu-Jeib, *Classic Two-step Durbin-type and Levinson-type algorithms for skew-symmetric Toeplitz matrices,* Can. Appl. Math. Q. **12**(3) (2004), 241–258.
2. I. T. Abu-Jeib and T. S. Shores, *On properties of matrix $I^{(-1)}$ of Sinc methods,* New Zealand J. Math. **32** (2003), 1–10.
3. D. Delsarte and Y. Genin, *The split Levinson algorithm,* IEEE Transactions on Acoustics Speech and Signal Processing ASSP **34** (1986), 470–477.
4. P. Gierke, Ph.D. thesis, University of Nebraska-Lincoln, 1999.
5. J. Lund and K. Bowers, *Sinc Methods for Quadrature and Differential Equations*, SIAM, Philadelphia, 1992.
6. A. Melman, *The even-odd split Levinson algorithm for Toeplitz systems,* SIAM J. Matrix Anal. Appl. **23** (2001), 256–270.
7. A. Melman, *A two step even-odd split Levinson algorithm for Toeplitz systems,* Linear Algebra Appl. **338** (2001), 219–237.
8. F. Stenger, *Numerical Methods Based on Sinc and Analytic Functions*, Springer-Verlag, New York, 1993.
9. F. Stenger, *Collocating convolutions*, Math. Comp. **64** (1995), 211–235.
10. F. Stenger, *Matrices of sinc methods*, J. Comput. Appl. Math. **86** (1997), 297–310.
11. W. F. Trench, *An algorithm for the inversion of finite Toeplitz matrices*, J. Soc. Indust. Appl. Math. **12** (1964), 515–522.

Department of Computer Science and Information Systems, Fenton Hall, SUNY Fredonia, Fredonia, New York 14063, USA

E-mail address: abu-jeib@cs.fredonia.edu