Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9065438, 16 pages
http://dx.doi.org/10.1155/2016/9065438

Research Article

Computing the Pseudoinverse of Specific Toeplitz Matrices Using Rank-One Updates

Predrag S. Stanimirović,[1] Vasilios N. Katsikis,[2] and Igor Stojanović[3]

[1]*University of Niš, Faculty of Sciences and Mathematics, Višegradska 33, 18000 Niš, Serbia*
[2]*Department of Economics, Division of Mathematics and Informatics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece*
[3]*Faculty of Computer Science, Goce Delčev University, Goce Delčev 89, 2000 Štip, Macedonia*

Correspondence should be addressed to Predrag S. Stanimirović; pecko@pmf.ni.ac.rs

Received 29 February 2016; Accepted 3 July 2016

Academic Editor: Masoud Hajarian

Application of the pure rank-one update algorithm as well as a combination of rank-one updates and the Sherman-Morrison formula in computing the Moore-Penrose inverse of the particular Toeplitz matrix is investigated in the present paper. Such Toeplitz matrices appear in the image restoration process and in many scientific areas that use the convolution. Four different approaches are developed, implemented, and tested on a number of numerical experiments.

1. Introduction

Let $\mathbb{C}^{m\times n}$ and $\mathbb{C}_r^{m\times n}$ denote the set of all complex $m \times n$ matrices and the set of all complex $m \times n$ matrices of rank r, respectively. The identity matrix of an appropriate order is denoted by I. The conjugate transpose, the range, the rank, and the null space of $A \in \mathbb{C}^{m\times n}$ are denoted by A^*, $\mathscr{R}(A)$, rank(A), and $\mathscr{N}(A)$, respectively.

Representation and computation of various generalized inverses are closely related to the following Penrose equations:

$$\begin{aligned} AXA &= A, \\ XAX &= X, \\ (AX)^* &= AX, \\ (XA)^* &= XA. \end{aligned} \tag{1}$$

The set of all matrices obeying the conditions contained in a subset $\mathcal{S} \subseteq \{1, 2, 3, 4\}$ is denoted by $A\{\mathcal{S}\}$. Any matrix from $A\{\mathcal{S}\}$ is called $\mathcal{S}$-inverse of A and is denoted by $A^{(\mathcal{S})}$. By $A\{\mathcal{S}\}_s$ we denote the set of all $\mathcal{S}$-inverses of A with rank s. For any matrix A there exists a single element in the set $A\{1, 2, 3, 4\}$, called the Moore-Penrose inverse of A and denoted by $A^\dagger$.

A rank-one modification of a matrix $A \in \mathbb{C}^{m\times n}$ is the matrix $M = A + bc^*$, which is created from A and two vectors $b \in \mathbb{C}^m$ and $c \in \mathbb{C}^n$. The Sherman-Morrison formula (S-M shortly) gives the basic relationship between the inverses M^{-1} and A^{-1} (for more details see, e.g., [1]):

$$M^{-1} = A^{-1} - \frac{1}{1 + c^* A^{-1} b} A^{-1} bc^* A^{-1}. \tag{2}$$

The identity (2) provides a numerically efficient way to compute the inverse M^{-1} of the rank-one update M. The S-M formula is important in many different fields of numerical computation; see for example [2–7].

On the other hand, Toeplitz matrices arise in a number of various theoretical investigations and applications. A number of iterative processes for finding generalized inverses of an arbitrary Toeplitz matrix by modifying Newton's method have been developed so far. The main results were stated in

[8–13]. Adaptations of the iterative processes to the Toeplitz structure are based on the usage of the displacement operator as well as the concept of displacement representation and ε-displacement rank of matrices.

A variety of methods for computing the Moore-Penrose inverse of a rank-one modified matrix have been developed so far. Main results were derived in [14–18]. Relationships between various generalized inverses of an arbitrary matrix and corresponding generalized inverses of its rank-one modifications were investigated in [19]. The leading idea in [15, 16] was successive computation of the symmetric rank-one (SR1) updates $(A_{i-1} + a_i a_i^*)^\dagger$ of a given matrix A, where a_i^* denotes the ith row of A and $A_{i-1} = \sum_{j=1}^{i-1} a_i a_i^*$. The authors of the paper [15] introduced a computational procedure for the Moore-Penrose inverse of a symmetric rank-one perturbed matrix. Using this method, the authors of [16] proposed a finite method for computing the minimum-norm least-squares solution of the linear system $Ax = b$.

The results derived in [20] reveal that both the SR1 updates techniques and the S-M recursive rule are useful tools in the computation of various matrix products involving the Moore-Penrose inverse of certain symmetric matrices. Particularly, the algorithms introduced in [20] are numerically efficient in computation of $\{2, 4\}$ and $\{2, 3\}$ inverses.

In the present paper, we investigate the possibilities to apply the SR1 update procedure and the S-M formula in the computation of the Moore-Penrose inverse of specific Toeplitz matrices that appear in the image restoration process. Our main motivation arises from the convenience to apply the SR1 update and the S-M procedure in removing the blur which is always present in digital images. Firstly, both the SR1 update and the S-M formula are based on the usage of columns (or rows) of the input matrix. On the other hand, the matrices which appear in the mathematical model of blur in computer-generated images possess very specific structure which can be used to accelerate SR1 and S-M procedures. Namely, entries in Toeplitz matrices are constant along main diagonal parallels and, moreover, possess a significant proportion of zero elements.

The paper is organized as follows. Some basic notations and necessary facts are restated in Section 2. Also, some additional motivation is presented in the same section. Usage of the pure SR1 update algorithm, proposed in [15], in the computation of the Moore-Penrose inverse of a kind of Toeplitz matrices is considered in Section 3. A hybrid combination of the SR1 and the S-M recursive rules is defined in Section 4. An improvement of the SR1 procedure, which is derived on the basis of the specific structure of the underlying Toeplitz matrix, is presented in the same section. An application of introduced methods in image restoration is presented in Section 5.

2. Preliminaries and Motivation

Toeplitz matrices or diagonally constant matrices are matrices having constant diagonal entries. Toeplitz matrices which are applicable in the image restoration process contain ℓ nonzero main diagonal parallels above the main diagonal, where ℓ defines the blurring process. In what follows, let us consider the Toeplitz matrix of such form:

$$H = \left[\begin{array}{cccccccc|cccc} t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & t_3 & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots & t_\ell \end{array}\right]. \quad (3)$$

The assumption $t_1 \neq 0$ is active.

To clarify notation, Toeplitz matrices of the general form (3) will be denoted shortly by

$$\mathrm{Tpl}\left(\{t_1\}, \{t_1, \ldots, t_\ell\}\right). \quad (4)$$

We investigate the use of the SR1 update method, as described in [15, Algorithm 2], during the numerical computation of the Moore-Penrose inverse of Toeplitz matrices satisfying the $\mathrm{Tpl}\left(\{t_1\}, \{t_1, \ldots, t_\ell\}\right)$ pattern. Also, we examine different improvements of the original method. The improvements are based on appropriate adaptations of the SR1 method and the S-M formula to the characteristic structure of underlying matrices of type $\mathrm{Tpl}\left(\{t_1\}, \{t_1, \ldots, t_\ell\}\right)$. The method of SR1 updates is based on the expression which computes the Moore-Penrose inverse of the first k columns of the initial matrix A using the Moore-Penrose inverse of its first $k-1$ columns. In detail, the SR1 method from [15] starts from the well-known representation $A^\dagger = (A^*A)^\dagger A^*$ of the Moore-Penrose. If the ith row of A is denoted by a_i^*, then

$$A^*A = \sum_{i=1}^{m} a_i a_i^*. \quad (5)$$

Chen and Ji in [15] defined the matrix sequences A_k and X_k, as

$$\begin{aligned} A_0 &= \mathbb{O} \in \mathbb{C}^{n\times n}, \\ A_k &= \sum_{i=1}^{k} a_i a_i^*, \quad k = 1, \ldots, m, \\ X_k &= A_k^\dagger A^*, \quad k = 1, \ldots, m. \end{aligned} \quad (6)$$

Clearly, $A_k = A_{k-1} + a_i a_i^*$ is the rank-one modification of A_{k-1} and

$$X_m = A_m^\dagger A^* = \left(\sum_{i=1}^{m} a_i a_i^*\right)^\dagger A^* = A^\dagger. \quad (7)$$

Recall that the Moore-Penrose inverse of a general rank-one modified matrix $M = A + bc^*$, where A is an arbitrary

Table 1: Comparison of the rank-one updates method and the BP method.

Method	Sizes n, ℓ, s	CPU time	r_1	r_2	r_3	r_4
BP	50, 15, 10	0.0025	$2.0011e-15$	$2.0385e-14$	$3.773e-15$	$1.6463e-14$
SR1	50, 15, 10	0.0490	$2.2762e-14$	$5.5984e-13$	$1.2594e-14$	$4.0623e-13$
BP	50, 20, 10	0.0028	$6.9626e-16$	$2.8774e-14$	$3.4226e-15$	$2.7706e-14$
SR1	50, 20, 10	0.0364	$6.16e-14$	$2.0705e-12$	$1.3341e-14$	$1.3718e-12$
BP	50, 15, 500	0.0029	$8.0717e-16$	$1.6354e-14$	$3.7149e-15$	$7.0839e-15$
SR1	50, 15, 500	0.0501	$1.9666e-14$	$4.8611e-13$	$8.979e-15$	$3.5898e-13$
BP	50, 20, 500	0.0016	$6.8218e-16$	$2.1407e-14$	$3.2663e-15$	$6.0107e-15$
SR1	50, 20, 500	0.0362	$2.0844e-14$	$5.3265e-13$	$7.9594e-15$	$4.1980e-13$
BP	250, 15, 500	0.0059	$3.4724e-15$	$2.3693e-13$	$1.5467e-14$	$7.4894e-14$
SR1	250, 15, 500	23.9762	$7.81e-12$	$3.6574e-11$	$1.2707e-13$	$2.2523e-11$
BP	400, 20, 500	0.0135	$4.731e-15$	$3.7769e-13$	$2.6494e-14$	$9.909e-14$
SR1	400, 20, 500	320.0578	$3.8409e-11$	$1.4246e-10$	$2.2311e-13$	$8.0923e-11$

matrix and b, c are arbitrary vectors, is obtained in [14, Theorem 3.1.3]. The general theorem from [14] suggests six different cases that one has to follow in order to establish a relation between $M^\dagger$ and $A^\dagger$. In [15], the authors proved that the real number $\beta_l = 1 + a_l^* A_{l-1}^\dagger a_l$, corresponding to the term $1 + c^* A^{-1} b$ in (2), satisfies $\beta_l \geq 1$. Later using this result in conjunction with the fact that A_l is a positive semidefinite matrix, the six cases of Theorem 3.1.3 from [14] can be reduced to the two-case problem. This reduction simplifies the SR1 updates formulas.

Let us denote the first k columns of H by

$$H^{\{k\}} = \left[\, h^1 \,\middle|\, \cdots \,\middle|\, h^k \,\right]. \tag{8}$$

Also, the last $n-k$ columns of H are denoted by ${}^{\{k+1\}}H^{\{n\}}$. Then the matrix H is given in the block form

$$\begin{aligned} H &= \left[H^{\{m\}} \mid {}^{\{m+1\}}H^{\{n\}}\right] \in \mathbb{C}^{m\times n}, \\ n &= m + \ell - 1, \quad \ell \geq 1, \\ H^{\{m\}} &\in \mathbb{C}^{m\times m}, \\ {}^{\{m+1\}}H^{\{n\}} &\in \mathbb{C}^{m\times(n-m)}, \end{aligned} \tag{9}$$

where the square block

$$H^{\{m\}} = \begin{bmatrix} t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 & 0 \\ 0 & t_1 & t_2 & t_3 & \cdots & t_\ell & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots & t_\ell \\ 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 & \cdots \\ 0 & 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & t_1 \end{bmatrix} \tag{10}$$

is the nonsingular band Toeplitz matrix and

$${}^{\{m+1\}}H^{\{n\}} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots \\ t_\ell & \cdots & 0 & 0 \\ \cdots & t_\ell & 0 & 0 \\ t_3 & \cdots & \ddots & \vdots \\ t_2 & t_3 & \cdots & t_\ell \end{bmatrix} \tag{11}$$

collects the last $n-m$ columns of H.

An application of Greville's partitioning method from [21] and the block partitioning method (BP method, shortly) from [22] in computing the Moore-Penrose inverse of the matrix H is presented in [23]. According to Algorithms 1 and 2 and Lemma 3 from [23], it is clear that the specific structure of matrix H enables computation $H^\dagger$ simply by computing the inverse of the nonsingular block $H^{\{m\}}$.

In this paper, we investigate some alternative methods for computing $H^\dagger$ using the SR1 updates and the S-M formula. Our intention is to decrease computational complexity as much as possible using a specific structure of Toeplitz matrices of the general form $\mathrm{Tpl}(\{t_1\}, \{t_1, \ldots, t_\ell\})$.

3. Computing the Pseudoinverse of a Toeplitz Matrix by Rank-One Updates

Our first attempt consists in applying the unmodified SR1 method from [15] in order to compute the pseudoinverse of the matrix H that belongs to the class $\mathrm{Tpl}(\{t_1\}, \{t_1, \ldots, t_\ell\})$. Obtained results are compared with corresponding results derived by applying Algorithm 2 from [23] (the BP method). The results of this comparison are presented in Table 1, where n, ℓ, and s are the parameters which define the Gaussian blur modeled by the Toeplitz matrix H. In the rest of the paper, it is assumed that the matrix H is of the order $m \times n$, where $n = m + \ell - 1$ and $\ell \geq 1$ represents the width of the blurring function.

Table 2: Comparison of the combination of AlgSR1 and AlgBP methods and the block partitioning method.

Method	Sizes n, ℓ, s	CPU time	r_1	r_2	r_3	r_4
BP	50, 15, 10	0.0037	$2.0011e-15$	$2.0385e-14$	$3.773e-15$	$1.6463e-14$
HSR1 + BP	50, 15, 10	0.0592	$2.6951e-14$	$2.9305e-13$	$1.3171e-13$	$4.0769e-14$
BP	50, 20, 10	$3.4243e-04$	$6.9626e-16$	$2.8774e-14$	$3.4226e-15$	$2.7706e-14$
HSR1 + BP	50, 20, 10	0.0305	$5.3084e-14$	$4.1354e-13$	$3.1072e-13$	$9.9898e-14$
BP	50, 15, 500	0.0019	$8.0717e-16$	$1.6354e-14$	$3.7149e-15$	$7.0839e-15$
HSR1 + BP	50, 15, 500	0.0411	$2.1453e-14$	$2.2518e-13$	$7.0401e-14$	$3.2624e-14$
BP	50, 20, 500	$3.2256e-04$	$6.8218e-16$	$2.1407e-14$	$3.2663e-15$	$6.0107e-15$
HSR1 + BP	50, 20, 500	0.0288	$6.029e-15$	$1.3385e-13$	$2.6565e-14$	$1.3927e-14$
BP	250, 15, 500	0.0048	$3.4724e-15$	$2.3693e-13$	$1.5467e-14$	$7.4894e-14$
HSR1 + BP	250, 15, 500	21.9978	$1.8868e-12$	$3.4807e-10$	$1.154e-11$	$4.0825e-12$
BP	400, 20, 500	0.0132	$4.731e-15$	$3.7769e-13$	$2.6494e-14$	$9.909e-14$
HSR1 + BP	400, 20, 500	306.1949	$7.3251e-12$	$2.7048e-09$	$6.4634e-11$	$2.0001e-11$

If an approximation of $H^{\dagger}$ is denoted by X, then the residual norms are denoted by

$$\begin{aligned} r_1 &= \|HXH - H\|_2, \\ r_2 &= \|XHX - X\|_2, \\ r_3 &= \left\|(HX)^* - HX\right\|_2, \\ r_4 &= \left\|(XH)^* - XH\right\|_2. \end{aligned} \tag{12}$$

For the sake of simplicity, let us denote Algorithm 2 from [15] by SR1 and Algorithm 2 from [23] by BP.

From Table 1, it is observable that the SR1 method is marginally efficient in terms of accuracy, but it is time consuming, especially for larger dimensions. There is a theoretical explanation for these results. The number of multiplications and divisions included in the SR1 algorithm is about $O(m^2n + mn)$ in the case $m < n$ and $O(n^2m + mn)$ in the case $n < m$. Therefore, as it is stated in [15], the rank-one updates algorithm works efficiently in cases where it holds $m \ll n$ or $m \gg n$. In our case, $n = m + \ell - 1$, where $\ell \geq 1$ represents the width of the blurring process. Since, in the general case, ℓ is a relatively small integer, the conditions $m < n$ and $m \approx n$ are satisfied. This fact causes relative inefficiency of the SR1 method.

Our second attempt is to apply Algorithm SR1 to the matrix $H^{\{m\}}$, from (10), in order to generate its inverse. Then the Moore-Penrose inverse of H comes from the block partitioning method (called BP), as it was described in [23]. This approach is shortly denoted by HSR1 + BP.

It is observable from Table 2 that the method HSR1 + BP is marginally efficient in terms of accuracy, but it is time consuming; see, for example, the case $n = 400$.

From the previous analysis of data included in Tables 1 and 2, we conclude that both algorithms SR1 and HSR1 + BP are not efficient approaches for computing $H^{\dagger}$ with respect to the BP method in terms of computational speed, especially for large matrices. But both approaches show marginal efficiency in terms of the accuracy.

4. A Combination of SR1 Updates and S-M Formula

Two additional algorithms are defined in the current section. The first one uses the SR1 updates to compute the matrix $H_m = H^{\{m\}}(H^{\{m\}})^*$ in the first step and the S-M formula to compute $H_m^{\dagger}$ in the subsequent step. The second algorithm replaces the usage of the SR1 updates from the first step by a direct construction of the matrix H_m. The matrix H_m can be generated efficiently in view of the specific structure of the input matrix H.

4.1. SR1 Updates in Conjunction with the S-M Formula. As usual, the ith column of H is denoted by h^i. Then, since H is of full row rank, it is advantageous to define a proper SR1 recursive computational method for generating the Moore-Penrose inverse of H on the basis of the relation

$$H^{\dagger} = H^*\left(HH^*\right)^{-1} = H^*\left(\sum_{i=1}^{n} h^i\left(h^i\right)^*\right)^{-1}. \tag{13}$$

For that purpose, it is necessary to define the matrix sequence

$$\begin{aligned} &H_0 = \mathbb{O} \in \mathbb{C}^{m\times m}, \\ &H_k = \sum_{i=1}^{k} h^i\left(h^i\right)^* = H_{k-1} + h^k\left(h^k\right)^*, \quad k = 1, \ldots, n. \end{aligned} \tag{14}$$

Lemma 1 reveals some of basic properties of the sequence H_k. Following the notation from [20], the notation $h^k \in \mathscr{L}(h^1, \ldots, h^{k-1})$ means that a column h^k is linearly dependent on the previous columns $h^1, \ldots, h^{k-1}$ and $h^k \notin \mathscr{L}(h^1, \ldots, h^{k-1})$ indicates that h^k is linearly independent of $h^1, \ldots, h^{k-1}$.

Lemma 1. *The following statements are valid:*

(i) $h^k \notin \mathscr{L}(h^1, \ldots, h^{k-1})$, *for each* $1 < k \leq n$.

(ii) $\operatorname{rank}(H_k) = \operatorname{rank}(H_{k-1}) + 1$, *for each* $1 < k \leq n$.

(iii) $\operatorname{rank}(H_k) < m$, *for each* $1 \le k < m$, *and* $\operatorname{rank}(H_k) = \operatorname{rank}(H_{k-1}) = m$, *for each* $k = m+1, \ldots, n$.

(iv) *The following concluding relation is valid:*

$$H^* H_n^{-1} = H^\dagger. \tag{15}$$

Proof. (i) This part of the proof is implied by the fact that the columns of H are linearly independent.

(ii) According to the part (i), immediately follows $\operatorname{rank}(H^{\{k\}}) = \operatorname{rank}(H^{\{k-1\}}) + 1$, for each $1 < k \le n$. Later, it is easy to verify

$$\begin{aligned} H_k &= \sum_{i=1}^{k} h^i \left(h^i\right)^* = \left[\, h^1 \,\middle|\, \cdots \,\middle|\, h^k \,\right] \begin{bmatrix} \left(h^1\right)^* \\ \vdots \\ \left(h^k\right)^* \end{bmatrix} \\ &= H^{\{k\}} \left(H^{\{k\}}\right)^*, \end{aligned} \tag{16}$$

which implies

$$\begin{aligned} \operatorname{rank}\left(H_k\right) &= \operatorname{rank}\left(H^{\{k\}}\right) = \operatorname{rank}\left(H^{\{k-1\}}\right) + 1 \\ &= \operatorname{rank}\left(H^{\{k-1\}} \left(H^{\{k-1\}}\right)^*\right) + 1 \\ &= \operatorname{rank}\left(H_{k-1}\right) + 1. \end{aligned} \tag{17}$$

(iii) This part of the proof can be verified by using (i) and (ii), together with the fact that H_k are all $m \times m$ matrices, for each $k = 1, \ldots, n$.

(iv) According to part (iii), matrices H_k, $m \le k \le n$, are invertible. Since

$$H_n = H^{\{n\}} \left(H^{\{n\}}\right)^* = HH^* \tag{18}$$

is invertible, then it holds

$$H^* H_n^{-1} = H^* \left(HH^*\right)^{-1} = H^\dagger, \tag{19}$$

which completes the proof. □

Remark 2. According to Lemma 1 the matrices H_k are of the order $m \times m$ and it holds $\operatorname{rank}(H_k) < m$, for each $1 \le k < m$, while $\operatorname{rank}(H_k) = \operatorname{rank}(H_{k-1}) = m$, for each $k = m+1, \ldots, n$. Since $\operatorname{rank}(H_n) = \operatorname{rank}(HH^*) = m$, the invertibility of H_n is guaranteed, so that we are able to apply the S-M formula which relates H_k^{-1} with H_{k-1}^{-1}, for each $k = m+1, \ldots, n$.

According to Lemma 1, matrices H_k, $m \le k \le n$, are invertible. Therefore, it is possible to define the following matrix sequence:

$$X_k = H^* H_k^{-1}, \quad k = m, \ldots, n. \tag{20}$$

Identity (15) immediately implies $X_n = H^\dagger$.

Lemma 3. *It holds* $1 + (h^k)^* H_{k-1}^{-1} h^k \ge 1$, *for each* $k = m+1, \ldots, n$.

Proof. The proof of lemma follows from the fact that H_{k-1} is regular, for each $k = m+1, \ldots, n$. □

Now, we turn our attention to (20). Since the matrices H_{k-1}, $k = m+1, \ldots, n$ are regular, it is possible to find a relation between $H_k^{-1} = (H_{k-1} + h^k (h^k)^*)^{-1}$ and H_{k-1}^{-1} by applying the Sherman-Morrison formula (2). This relation is given in Proposition 4.

Proposition 4. *Let the matrix* H *be defined as in (3). Assume that the matrix sequence* H_k *is defined in (14). The inverse of* H_k *is equal to*

$$\begin{aligned} H_k^{-1} &= \left(H_{k-1} + h^k \left(h^k\right)^*\right)^{-1} \\ &= H_{k-1}^{-1} \\ &\quad - \left(1 + \left(h^k\right)^* H_{k-1}^{-1} h^k\right)^{-1} H_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1}, \end{aligned} \tag{21}$$

for each $k = m+1, \ldots, n$.

In order to present the finite recursive algorithm for computing the Moore-Penrose inverse of H with the help of the actual SR1 update formulas (20) and (21), we first define $H_0 = \mathbb{O} \in \mathbb{C}^{m \times m}$. For each $k = 1, \ldots, m$, it is necessary to compute $H_k = H_{k-1} + h^k (h^k)^*$. This requires calculated values of the vectors $y_{m,t} = H_m^{-1} h^t \in \mathbb{C}^m$, $t = 1, \ldots, n$ and the matrix $X_m = H^* H_m^{-1}$.

Theorem 5. *Let the matrix* H *be defined as in (3), the matrix sequence* H_k *defined in (14), and the matrix sequence* X_k *defined in (20). Then the following recursive relations are true:*

$$\begin{aligned} y_{k,t} &= y_{k-1,t} - \frac{y_{k-1,k} \left(h^k\right)^* y_{k-1,t}}{1 + \left(h^k\right)^* y_{k-1,k}} \\ &= \left(I_m - \frac{y_{k-1,k} \left(h^k\right)^*}{1 + \left(h^k\right)^* y_{k-1,k}}\right) \cdot y_{k-1,t}, \end{aligned} \tag{22}$$

$$X_k = X_{k-1} - \frac{H^* y_{k-1,k} \left(y_{k-1,k}\right)^*}{1 + \left(h^k\right)^* y_{k-1,k}}. \tag{23}$$

Proof. Note that from $y_{l,t} = H_l^{-1} h^t$ it follows that $(y_{l,t})^* = (H_l^{-1} h^t)^* = (h^t)^* H_l^{-1}$. Based on the use of (21), the recursive sequences $y_{k,t}$ are defined for $k = m+1, \ldots, n$ as in the following:

$$\begin{aligned} y_{k,t} = H_k^{-1} h^t &= \left(H_{k-1} + h^k \left(h^k\right)^*\right)^{-1} h^t = \left(H_{k-1}^{-1}\right. \\ &\quad \left. - \left(1 + \left(h^k\right)^* H_{k-1}^{-1} h^k\right)^{-1} H_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1}\right) h^t \\ &= H_{k-1}^{-1} h^t - \frac{H_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1} h^t}{1 + \left(h^k\right)^* H_{k-1}^{-1} h^k}, \quad t = 1, \ldots, n. \end{aligned} \tag{24}$$

Then the statement (22) can be easily verified using (24).

Also, matrices X_k, $k = m+1, \ldots, n$ are defined as

$$\begin{aligned} X_k &= H^* H_k^{-1} = H^* \left(H_{k-1}^{-1} \right. \\ &\quad \left. - \left(1 + \left(h^k\right)^* H_{k-1}^{-1} h_k\right)^{-1} H_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1} \right) \\ &= H^* H_{k-1}^{-1} \\ &\quad - H^* \left(\left(1 + \left(h^k\right)^* H_{k-1}^{-1} h^k\right)^{-1} G_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1} \right) \\ &= X_{k-1} - \frac{H^* H_{k-1}^{-1} h^k \left(h^k\right)^* H_{k-1}^{-1}}{1 + \left(h^k\right)^* H_{k-1}^{-1} h^k}. \end{aligned} \tag{25}$$

So, (23) holds. □

According to Theorem 5, we present Algorithm 1 for computing the Moore-Penrose inverse of the Toeplitz matrix H. The algorithm is based on the SR1 modifications to compute H_m and subsequently the S-M formula to compute $H^\dagger$. For this purpose, we use the abbreviation HSR1 + S-M to denote Algorithm 1.

A *Matlab* implementation of Algorithm 1 is placed in Appendix.

4.2. An Improvement of the Hybrid Combination in Algorithm 1. By taking into account the fact that the Toeplitz matrix H has a specific structure, the matrix H_m can be generated in an efficient way. In this way, an improvement of step (1) of Algorithm 1 is introduced. The effectiveness of this method will be justified in the numerical experiments presented in Section 4.3.

According to the proper structure of the matrix H and step (1) of Algorithm 1, the general structure of the matrix H_m is defined as

$$H_m = \left[\begin{array}{cccccc|cccccc} \sum_{i=1}^{\ell} t_i^2 & \sum_{i=1}^{\ell-1} t_i t_{i+1} & \sum_{i=1}^{\ell-2} t_i t_{i+2} & \cdots & \cdots & t_1 t_l & 0 & 0 & \cdots & \cdots & 0 \\ \sum_{i=1}^{\ell-1} t_i t_{i+1} & \sum_{i=1}^{\ell} t_i^2 & \sum_{i=1}^{\ell-1} t_i t_{i+1} & \ddots & \ddots & \ddots & t_1 t_l & 0 & \ddots & \ddots & 0 \\ \sum_{i=1}^{\ell-2} t_i t_{i+2} & \sum_{i=1}^{\ell-1} t_i t_{i+1} & \sum_{i=1}^{\ell} t_i^2 & \sum_{i=1}^{\ell-1} t_i t_{i+1} & \ddots & \ddots & \ddots & t_1 t_l & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \sum_{i=1}^{\ell} t_i^2 & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \sum_{i=1}^{\ell} t_i^2 & \ddots & \ddots & \ddots & \ddots & t_1 t_\ell \\ \hline \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \sum_{i=1}^{\ell-1} t_i^2 & \ddots & \ddots & \ddots & t_1 t_{\ell-1} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \sum_{i=1}^{2} t_i t_{i+1} & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \sum_{i=1}^{2} t_i t_{i+1} & \sum_{i=1}^{2} t_i^2 & t_1 t_2 \\ 0 & \cdots & \cdots & \cdots & 0 & t_1 t_\ell & \cdots & \cdots & \cdots & t_1 t_2 & t_1^2 \end{array}\right] \begin{array}{l} \} m-\ell \\ \} \ell \end{array} \tag{26}$$

$$\underbrace{\qquad\qquad}_{m-\ell} \qquad \underbrace{\qquad\qquad}_{\ell}$$

Let $w = (w_1, \ldots, w_n)$ be one-dimensional vector; then the symmetric Toeplitz matrix generated by w is denoted by Toeplitz(w):

$$\text{Toeplitz}\,(w) = \begin{bmatrix} w_1 & w_2 & w_3 & \cdots & w_n \\ w_2 & w_1 & w_2 & \cdots & w_{n-1} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ w_{n-1} & \cdots & \cdots & w_1 & w_2 \\ w_n & \cdots & \cdots & w_2 & w_1 \end{bmatrix}. \tag{27}$$

Also, for two appropriate vectors u, v, we denote by $F = \text{Hankel}(u, v)$ the (symmetric) Hankel matrix whose first column is u and whose last row is v.

Theorem 6. *The matrix H_m can be expressed as the difference of a specific banded symmetric Toeplitz matrix and a partitioned matrix of the form*

$$\left[\begin{array}{c|c} \mathbb{O}_{m-\ell+1} & \mathbb{O}_{(m-\ell+1)\times(\ell-1)} \\ \hline \mathbb{O}_{(\ell-1)\times(m-\ell+1)} & F_{\ell-1} \end{array}\right], \tag{28}$$

where the block $F_{\ell-1}$ is defined by $F_{\ell-1} = F^2$, for an appropriate Hankel matrix F.

Require: Start with $H \in \mathbb{C}^{m\times n}$ and $n > m$. Let h^i be the ith column of H, for $i = 1, \ldots, n$, $H_0 = \mathbb{O} \in \mathbb{C}^{m\times m}$.
(1) (SR1 Step) For $k = 1, \ldots, m$, compute $H_k = H_{k-1} + h^k (h^k)^*$.
(2) Set $y_{m,t} = H_m^{-1} h^t \in \mathbb{C}^m$ for all $t = 1, \ldots, n$ and $X_m = H^* H_m^{-1}$.
(3) (S-M Step) Compute $y_{l,t}$ and X_l by using (22) and (23), respectively for all $l = m+1, \ldots, n$.
(4) Output $H^\dagger = X_n$.

Algorithm 1: Computing the Moore-Penrose inverse of the Toeplitz matrix H.

Proof. Let u and v be vectors defined by

$$u = \begin{pmatrix} t_\ell & t_{\ell-1} & \cdots & t_2 \end{pmatrix}, \tag{29}$$

$$v = \begin{pmatrix} \underbrace{0 \ \cdots \ 0}_{\ell-2} & t_\ell \end{pmatrix}, \tag{30}$$

respectively. Then we define the following matrices:

$$\begin{aligned} &E \\ &= \text{Toeplitz}\left(\left(\sum_{i=1}^{\ell} t_i^2 \ \ \sum_{i=1}^{\ell-1} t_i t_{i+1} \ \ \sum_{i=1}^{\ell-2} t_i t_{i+2} \ \cdots \ t_1 t_\ell \ \ \underbrace{0 \ \cdots \ 0}_{m-\ell}\right)\right), \\ &F = \text{Hankel}\,(v, u), \\ &L = \left[\begin{array}{c|c} \mathbb{O}_{m-\ell+1} & \mathbb{O}_{(m-\ell+1)\times(\ell-1)} \\ \hline \mathbb{O}_{(\ell-1)\times(m-\ell+1)} & F_{\ell-1} \end{array}\right], \quad F_{\ell-1} = F^2. \end{aligned} \tag{31}$$

A simple verification shows that $H_m = E - L$. □

Next we present an example, in low dimensions, in order to illustrate Theorem 6.

Example 7. Consider the Toeplitz matrix $H \in \mathbb{C}^{m\times n}$, defined in (3) for $m = 5$, $\ell = 3$, and $n = m + \ell - 1 = 7$:

$$H = \left[H^{\{5\}} \mid {}^{\{6\}}H^{\{7\}}\right] = \left[\begin{array}{ccccc|cc} t_1 & t_2 & t_3 & 0 & 0 & 0 & 0 \\ 0 & t_1 & t_2 & t_3 & 0 & 0 & 0 \\ 0 & 0 & t_1 & t_2 & t_3 & 0 & 0 \\ 0 & 0 & 0 & t_1 & t_2 & t_3 & 0 \\ 0 & 0 & 0 & 0 & t_1 & t_2 & t_3 \end{array}\right]. \tag{32}$$

According to (14),

$$\begin{aligned} H_m = H_5 &= H^{\{5\}} \left(H^{\{5\}}\right)^* \\ &= \begin{bmatrix} t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 & 0 & 0 \\ t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 & 0 \\ t_1 t_3 & t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 \\ 0 & t_1 t_3 & t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 & t_1 t_2 \\ 0 & 0 & t_1 t_3 & t_1 t_2 & t_1^2 \end{bmatrix}. \end{aligned} \tag{33}$$

We exploit the matrix

$$E = \text{Toeplitz}\left(\left(\sum_{i=1}^{3} t_i^2 \ \ \sum_{i=1}^{2} t_i t_{i+1} \ \ t_1 t_3 \ \ 0 \ \ 0\right)\right) = \begin{bmatrix} t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 & 0 & 0 \\ t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 & 0 \\ t_1 t_3 & t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 & t_1 t_3 \\ 0 & t_1 t_3 & t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 & t_1 t_2 + t_2 t_3 \\ 0 & 0 & t_1 t_3 & t_1 t_2 + t_2 t_3 & t_1^2 + t_2^2 + t_3^2 \end{bmatrix}. \tag{34}$$

Let $u = \begin{pmatrix} t_3 & t_2 \end{pmatrix}$ and $v = \begin{pmatrix} 0 & t_3 \end{pmatrix}$; then

$$\begin{aligned} F &= \text{Hankel}\,(v, u) = \begin{bmatrix} 0 & t_3 \\ t_3 & t_2 \end{bmatrix}, \\ F^2 &= \begin{bmatrix} t_3^2 & t_2 t_3 \\ t_2 t_3 & t_2^2 + t_3^2 \end{bmatrix}. \end{aligned} \tag{35}$$

Finally, it is necessary to construct the matrix

$$L = \left[\begin{array}{c|c} \mathbb{O}_3 & \mathbb{O}_{3\times 2} \\ \hline \mathbb{O}_{2\times 3} & F^2 \end{array}\right] = \left[\begin{array}{ccc|cc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & t_3^2 & t_2 t_3 \\ 0 & 0 & 0 & t_2 t_3 & t_2^2 + t_3^2 \end{array}\right]. \tag{36}$$

It is easy to see that $H_5 = E - L$.

According to Theorem 6 we present the following improved version of Algorithm 1, called Algorithm 2. A *Matlab* implementation of Algorithm 2 is placed in Appendix. It is appropriate to use the term IHSR1 to denote the improvement of the HSR1 step of Algorithm 1 and IHSR1 + S-M to denote Algorithm 2.

Note that proposed Algorithms 1 and 2 are not using directly, at any step, the Sherman-Morrison formula to compute the inverse of some matrix. In Section 4.4, we provide more information regarding this situation.

4.3. Numerical Experiments. In this subsection we analyze numerical data arising during the computation of the Moore-Penrose inverse of the Toeplitz matrix H by applying a *Matlab* implementation of Algorithms 1 and 2. In order to test the time efficiency as well as the accuracy of considered methods, we enrich our collection of matrices used in Tables

Require: Start with $H \in \mathbb{C}^{m\times n}$, $n > m$. Let h^i be the ith column of H, for $i = 1, \ldots, n$.
(1) (Improved HSR1 Step) Define E and L as in Theorem 6. Then, set $H_m = E - L$.
(2) Set $y_{m,t} = H_m^{-1} h^t \in \mathbb{C}^m$ for all $t = 1, \ldots, n$ and $X_m = H^* H_m^{-1}$.
(3) (S-M Step) Compute $y_{l,t}$ and X_l by using (22) and (23), respectively for all $l = m + 1, \ldots, n$.
(4) Output $H^\dagger = X_n$.

Algorithm 2: Computing the Moore-Penrose inverse of the Toeplitz matrix H.

Table 3: Comparison of Algorithm 1 (HSR1 + S-M), Algorithm 2 (IHSR1 + S-M), and BP.

Method	Sizes n, ℓ, s	CPU time	r_1	r_2	r_3	r_4
HSR1 + S-M	50, 15, 10	0.0054	1.6449$e-14$	2.5845$e-12$	3.5344$e-13$	1.9362$e-14$
IHSR1 + S-M	50, 15, 10	0.0035	2.0486$e-14$	2.7318$e-12$	5.4426$e-13$	2.3222$e-14$
BP	50, 15, 10	0.0018	2.0011$e-15$	2.0385$e-14$	3.773$e-15$	1.6463$e-14$
HSR1 + S-M	50, 20, 10	0.0040	1.8748$e-14$	8.8674$e-13$	2.2185$e-13$	1.5379$e-14$
IHSR1 + S-M	50, 20, 10	0.0032	9.4733$e-15$	8.9539$e-13$	1.9076$e-13$	1.4775$e-14$
BP	50, 20, 10	0.0027	9.2684$e-16$	1.6646$e-14$	3.2679$e-15$	5.8648$e-15$
HSR1 + S-M	50, 20, 500	0.0053	1.2862$e-14$	9.6327$e-13$	1.2288$e-13$	1.4447$e-14$
IHSR1 + S-M	50, 20, 500	0.0044	8.6971$e-15$	1.5068$e-12$	2.2331$e-13$	1.5216$e-14$
BP	50, 20, 500	0.0022	6.8218$e-16$	2.1407$e-14$	3.2663$e-15$	6.0107$e-15$
HSR1 + S-M	250, 15, 500	0.0482	2.5535$e-13$	3.7054$e-11$	4.7471$e-12$	2.5608$e-13$
IHSR1 + S-M	250, 15, 500	0.0195	2.209$e-13$	3.7065$e-11$	5.5401$e-12$	2.4239$e-13$
BP	250, 15, 500	0.0032	3.4724$e-15$	2.3693$e-13$	1.5467$e-14$	7.4894$e-14$
HSR1 + S-M	400, 20, 500	0.3234	4.5138$e-13$	6.0769$e-10$	1.4669$e-11$	5.5243$e-13$
IHSR1 + S-M	400, 20, 500	0.1466	5.881$e-13$	6.0878$e-10$	1.5866$e-11$	5.3143$e-13$
BP	400, 20, 500	0.0116	6.5483$e-15$	6.052$e-13$	3.0382$e-14$	1.3117$e-13$
HSR1 + S-M	1200, 20, 500	11.0062	3.0653$e-12$	1.7107$e-08$	1.4049$e-10$	3.2612$e-12$
IHSR1 + S-M	1200, 20, 500	2.3018	3.5064$e-12$	1.711$e-08$	1.3136$e-10$	3.1795$e-12$
BP	1200, 20, 500	0.1049	1.2805$e-14$	3.179$e-12$	8.9833$e-14$	2.3321$e-13$
HSR1 + S-M	1500, 20, 500	21.7539	4.3812$e-12$	3.2457$e-08$	2.2848$e-10$	4.2712$e-12$
IHSR1 + S-M	1500, 20, 500	3.7358	4.2345$e-12$	3.2456$e-08$	2.0156$e-10$	4.2919$e-12$
BP	1500, 20, 500	0.1886	1.3786$e-14$	4.0422$e-12$	1.1081$e-13$	3.0133$e-13$
HSR1 + S-M	2000, 50, 500	74.5230	5.5113$e-12$	4.6497$e-08$	5.1185$e-10$	6.7183$e-12$
IHSR1 + S-M	2000, 50, 500	30.8622	5.3682$e-12$	4.6504$e-08$	5.4329$e-10$	6.6102$e-12$
BP	2000, 50, 500	0.4038	1.8118$e-14$	6.5462$e-12$	1.408$e-13$	3.6715$e-13$

1 and 2 with larger matrices. In order to complete our numerical study, we also compare the results derived by applying Algorithm 2 from [23] (Algorithm BP). A comparison of Algorithms 1 and 2 and BP is presented in Table 3.

From Table 3, it is observable that the level of improvement in CPU time that can be achieved by using Algorithm 2 instead of Algorithm 1 is significant. Also, it is clear that the accuracy of both algorithms is almost the same. Clearly, the BP method requires minimal CPU time.

4.4. Rounding Error Analysis. The starting motivation of this subsection is the following basic problem: approximation error grows with repeated use of the S-M formula. As a consequence, computations which involve the S-M rule become unstable. For this purpose, it is interesting to investigate the error curves corresponding to the four Penrose equations during the iterations included in step (3) of Algorithm 2. Nevertheless, it is important to note that the proposed Algorithm 2 does not use in each step the S-M rule. Of course, Theorem 5 is in the heart of Algorithm 2 but also it is clear that the poor stability of the S-M formula has no serious influence on the stability of proposed algorithms.

So, in this subsection, we shall present error estimations regarding Algorithm 2. That is, we record and investigate the residual matrix norms r_1, r_2, r_3, and r_4, corresponding to the four Penrose equations during the execution of step (3) of Algorithm 2. The results are presented in Figures 1, 2, and 3 for the cases: (a) $n = 1200$, $l = 20$, and $s = 500$, (b) $n = 1500$, $l = 20$, and $s = 500$, and (c) $n = 2000$, $l = 20$, and$s = 500$, respectively. The choice of these matrices from Table 3 is random.

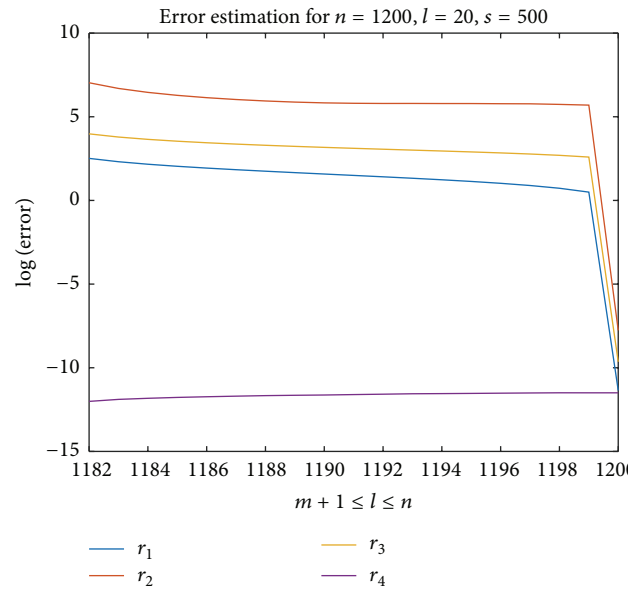


FIGURE 1: Errors recorded for $n = 1200$, $l = 20$, and $m = 500$ while the loop of step (3) is iterating.

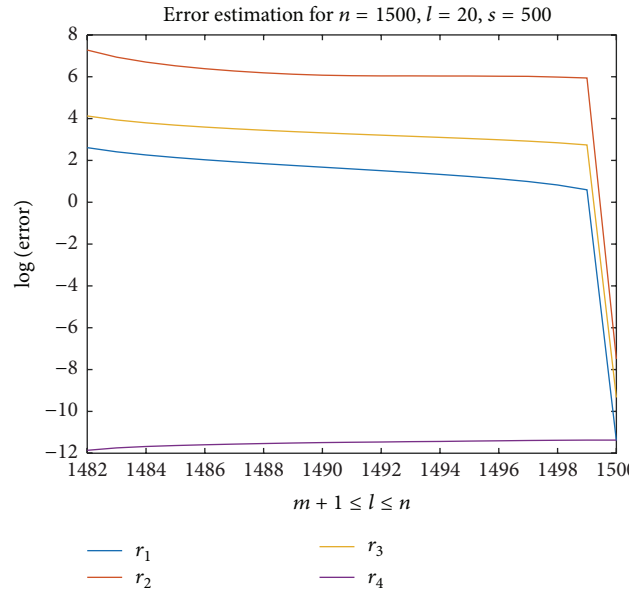


FIGURE 2: Errors recorded for $n = 1500$, $l = 20$, and $m = 500$ while the loop of step (3) is iterating.

The purple line in each of Figures 1–3 corresponds to the values of the matrix norm $\|(X_lH)^* - X_lH\|_2$, $l = m + 1, \ldots, n$. Since $X_l = H^*H_l^{-1}$ and the matrix H_l is symmetric, the matrix X_lH is symmetric. For this reason, the graph corresponding to these values is almost a constant line.

The stable convergence is observable from each of these figures. Also, it is possible to notice very fast convergence in the terminal phase of the convergence as well as a relatively slower convergence in the middle of the recurrent process. This fact is understandable if one takes into account that it is necessary to take into consideration all the columns of the input matrix in order to get the complete information on the pseudoinverse.

5. Application in Image Restoration

Several image restoration examples are presented in this subsection. Experiments are done using Matlab programming package on an Intel(R) Core(TM) i5-2540M CPU @ 2.6 GHz 64-bit system with 8 GB RAM memory.

Numerical results corresponding to the Moore-Penrose inverse are derived using Algorithm 2. Results are derived using $n = 1200$ and $s = 100$ and different values of ℓ. Suppose

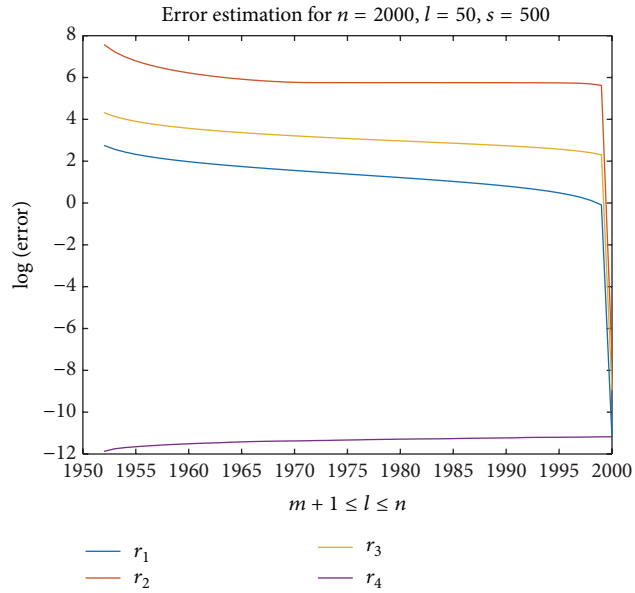


FIGURE 3: Errors recorded for $n = 2000$, $l = 50$, and $m = 500$ while the loop of step (3) is iterating.

FIGURE 4: (a) Image of Barbara. (b) Blurred noisy image of Barbara.

that the matrix $F \in \mathbb{R}^{r\times m}$ corresponds to the original image and $G \in \mathbb{R}^{r\times m}$ is the matrix corresponding to the degraded image.

The model of the nonuniform blurring is the same as in [23] and assumes that the blurring of columns in the image is independent with respect to the blurring of its rows. Therefore, the relations between the original and the blurred image are expressed by the matrix equation

$$G = H_C F H_R^T, \quad G \in \mathbb{R}^{r\times m},\ H_C \in \mathbb{R}^{r\times n},\ F \in \mathbb{R}^{n\times t},\ H_R \in \mathbb{R}^{m\times t}, \tag{37}$$

where $n = r + \ell_c - 1$, $t = m + \ell_r - 1$, ℓ_c is length of the vertical blurring, and ℓ_r is length of the horizontal blurring (in pixels). Following the approach used in [23], the Moore-Penrose inverse approach is used to restore the blurred image G, which produces the next approximation of F:

$$\widetilde{F} = H_C^{\dagger} G \left(H_R^{\dagger}\right)^T. \tag{38}$$

The algorithm for computing the Moore-Penrose inverse of the blurring matrix used in [24, 25] is based on the method for computing the Moore-Penrose inverse of a full rank matrix, introduced in [26, 27]. In our case, we use Algorithm 2.

The first approach is based on the usage of the pure SR1 update algorithm, proposed in [15]. The second approach starts with the SR1 update algorithm and, after its completion, continues with the BP method. The third approach is a hybrid combination which comprises the SR1 updates in the first

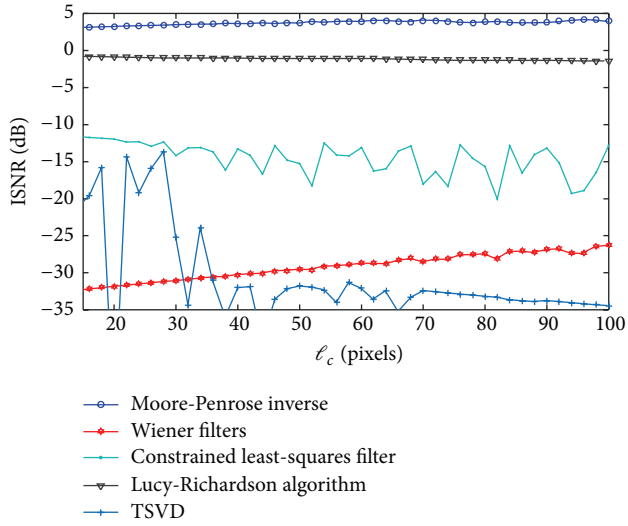


FIGURE 5: ISNR versus ℓ_c for the removal of Gaussian white noise with zero mean and variance 0.05 and Gaussian blur ($\ell_r = 35$, $s = \ell_r/2$, and $s = \ell_c/2$) for Barbara image.

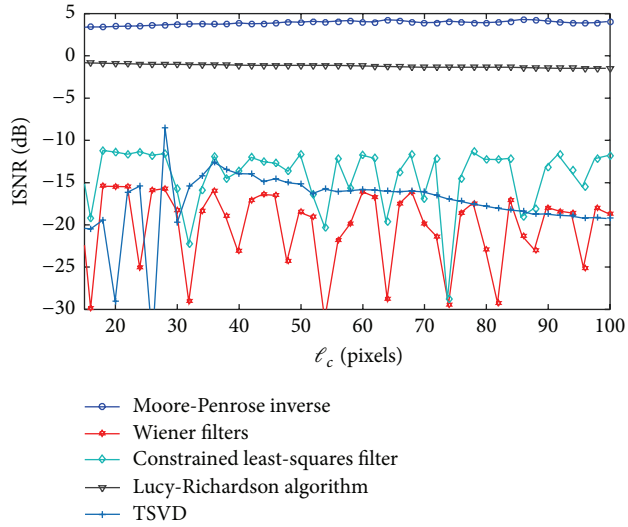


FIGURE 6: ISNR versus ℓ_c for the removal of Gaussian white noise with zero mean and variance 0.05 and Gaussian blur ($\ell_r = 35$, and $s = 100$) for Barbara image.

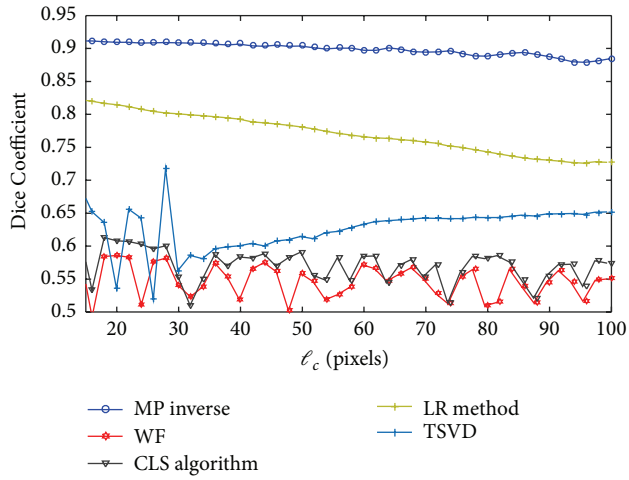
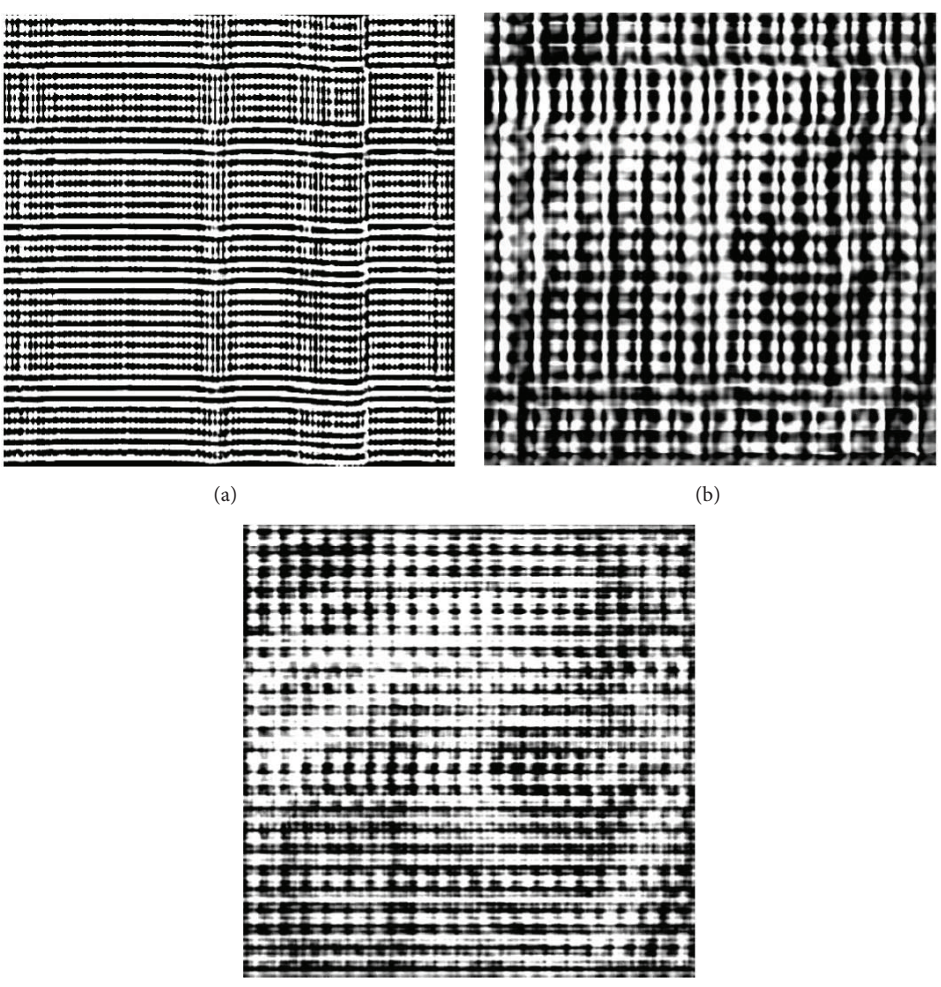


FIGURE 7: DC versus ℓ_c for the removal of Gaussian white noise with zero mean and variance 0.05 and Gaussian blur ($\ell_r = 35$, and $s = 100$) for Barbara image.

(a) (b)

Figure 8: (a) Moore-Penrose restored image. (b) Lucy-Richardson restored image.



(a) (b)

(c)

Figure 9: (a) Wiener filter restored image, (b) constrained least-squares filter restored image, and (c) truncated singular value decomposition restored image.

```
function alg41 = alg41(A)
%**************************%
% General Information.       %
%**************************%
% Synopsis:
% alg41 = alg41(A)
% Input:
% A = the initial matrix of interest.
% Output:
% alg41 = mp-inverse by using rank-one updates and the Sherman_Morrison formula.
% This function follows Algorithm 1.
% The correct performance of alg41b requires the presence of yltXl function.

Astar = A';
[m,n] = size(A);
G0 = zeros(m,m);
if m > 1
 for i = 1:m-1
    Gkout = G0+A(:,i)*Astar(i,:);
    G0 = Gkout;
 end

 Gm = Gkout+A(:,m)*Astar(m,:);
 else
    Gm = G0;
end

Gkout = Gm;
Y02 = Gkout\A;
X02 = Y02';

for l = m+1:n-1
[yltout,Xlout] = yltXl(Y02,X02,Astar,l,m,n);
Y02 = yltout;
X02 = Xlout;
end
alg41 = X02 - Astar*Y02(:,n)*Y02(:,n)'/(1+Astar(n,:)*Y02(:,n));
```

Algorithm 3

step and the S-M formula in the second phase (Algorithm 1). The fourth approach is based on the improvement of the SR1 update step method of Algorithm 1, which leads to Algorithm 2 with the best performances. The improvement is achieved using the specific structure of the underlying Toeplitz matrix.

The most popular algorithm for image restoration based on the matrix pseudoinverse is developed using the singular value decomposition (SVD) [28]. But, the SVD is sensitive to singular values very close to zero. For this purpose, it is common approach to leave out small singular values, corresponding to high-frequency components. The resulting method is known as the truncated SVD (or TSVD shortly) [28]. On the other hand, the methods proposed in the present paper are based on different approach and do not require any kind of the regularization. In order to verify the applicability of the proposed method in the image restoration, we present a comparison of results generated by the TSVD and the rank-one updates.

Parameters of Barbara image degradation are $\ell_r = 39$, $\ell_c = 25$, $s = \ell_r/2$, and $s = \ell_c/2$ and salt pepper noise with the noise density of 0.05 is assumed. The original and damaged image are presented in Figure 4.

Comparison with the state-of-the-art methods in image processing is presented. Many of them are constrained least-squares filter (CLS); Wiener filter (WF); Lucy-Richardson algorithm (LR); and truncated singular value decomposition (TSVD) [28]. Improvements in signal-to-noise ratio (ISNR) and Dice Coefficient (DC) are considered.

Results corresponding to ISNR values are presented in Figures 5 and 6.

Values corresponding to Dice Coefficient (DC) [29, 30] are presented in Figure 7.

Restored images are presented in Figures 8 and 9.

```
function alg42 = alg42(A,ll)
%***************************%
% General Information.      %
%***************************%
% Synopsis:
% alg42 = alg42(A)
% Input:
% A = the mxn initial matrix of interest.
% ll=n-m+1
% Output:
% alg42 = improvement of Algorithm 1 according to Theorem 6.
% This function follows Algorithm 2.
% The correct performance of alg42 requires the presence of yltXl function.

Astar = A';
[m,n] = size(A);

f = A(1,1:ll);
E0 = zeros(1,m);
product = f'*f;

for i = 1:ll
    E0(1,i) = sum(diag(product,i-1));
end
E1=E0;
E = toeplitz(E1);

G0=zeros(ll-1);
Gnew = hankel([zeros(1,ll-2) f(end)], flip(f(1,2:end)));
for i = 1:ll-1
    Gkoutnew = G0+Gnew(:,i)*Gnew(i,:);
    G0=Gkoutnew;
end

Gmnew = G0;
fullmatrix = [zeros(m-ll+1) zeros(m-ll+1,ll-1);zeros(ll-1,m-ll+1) Gmnew];
Gkout = E-fullmatrix;
Y02 = Gkout\A;
X02 = Y02';

for l = m+1:n-1
[yltout,Xlout] = yltXl(Y02,X02,Astar,l,m,n);
Y02 = yltout;
X02 = Xlout;
end

alg42 = X02 - Astar*Y02(:,n)*Y02(:,n)'/(1+Astar(n,:)*Y02(:,n));
```

Algorithm 4

6. Conclusion

The present paper investigates the application of the SR1 updates procedure and S-M formula in computation of the Moore-Penrose inverse of specific Toeplitz matrices that appear in the image restoration process.

The first approach is based on the usage of the pure SR1 update algorithm, proposed in [15]. The second approach starts with the SR1 update algorithm and, after its completion, continues with the BP method. The third approach is a hybrid combination which comprises the SR1 updates in the first step and the S-M formula in the second phase (Algorithm 1). The fourth approach is based on the improvement of the SR1 update step method of Algorithm 1, which leads to Algorithm 2 with the best performances. The improvement is achieved using the specific structure of the underlying Toeplitz matrix.

An application of Algorithm 2 in image restoration is considered. In this case, a comparison with the modified BP method from [23] is presented.

```
function [yltout,Xlout] = yltXl(Y0,X0,G,l,m,n)
d = n-l;
Y0l = Y0(:,l);
g = 1+G(l,:)*Y0l;
g1 = Y0l*G(l,:);
multiplier = eye(m)-g1/g;
Y0lherm = Y0l*Y0l';
 Xl = X0 - G*Y0lherm/g;
   Xlout = Xl;

ylt = zeros(m,n);
  for i = 1:d
  t = l+i;
    ylt(:,t) = multiplier*Y0(:,t);
  yltout = ylt;
 end
```

Algorithm 5

Appendix

The *Matlab* implementation of Algorithm 1 is given in Algorithm 3 under the function alg41. Note that the correct performance of the function alg41 requires the presence of `yltXl` function for computing the sequences $y_{k,t}$ and X_k.

The *Matlab* implementation of Algorithm 2 is given in Algorithm 4 under the function alg42. Note that the correct performance of alg42 requires the presence of `yltXl` function for computing the sequences $y_{k,t}$ and X_k.

The `yltXl` *Function*. See Algorithm 5.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

Predrag S. Stanimirović gratefully acknowledges support from the Research Project 174013 of the Serbian Ministry of Science. Vasilios N. Katsikis gratefully acknowledges that this work was carried out at the Department of Economics, University of Athens, Greece, and supported by Special Account Research Grant 11698. Predrag S. Stanimirović and Igor Stojanović gratefully acknowledge support from the Project "Applying Direct Methods for Digital Image Restoring" of the Goce Delčev University.

References

[1] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, W.H. Freeman, San Francisco, Calif, USA, 1963.

[2] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation*, vol. 19, pp. 577–593, 1965.

[3] R. Bru, J. Cerdán, J. Marín, and J. Mas, "Preconditioning sparse nonsymmetric linear systems with the Sherman-Morrison formula," *SIAM Journal on Scientific Computing*, vol. 25, no. 2, pp. 701–715, 2003.

[4] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell Publishing, New York, NY, USA, 1964.

[5] P. Maponi, "The solution of linear systems by using the Sherman-Morrison formula," *Linear Algebra and its Applications*, vol. 420, no. 2-3, pp. 276–294, 2007.

[6] K. Moriya, L. Zhang, and T. Nodera, "An approximate matrix inversion procedure by parallelization of the Sherman-Morrison formula," *The ANZIAM Journal*, vol. 51, no. 1, pp. 1–9, 2009.

[7] E. D. Nino-Ruiz, A. Sandu, and J. Anderson, "An efficient implementation of the ensemble Kalman filter based on an iterative Sherman–Morrison formula," *Statistics and Computing*, vol. 25, no. 3, pp. 561–577, 2015.

[8] D. Bini and B. Meini, "Approximate displacement rank and applications," in *Structured Matrices in Mathematics, Computer Science and Engineering II*, V. Olshevsky, Ed., vol. 281 of *Contemporary Mathematics*, pp. 215–232, American Mathematical Society, Rhode Island, RI, USA, 2001.

[9] D. A. Bini, G. Codevico, and M. Van Barel, "Solving Toeplitz least squares problems by means of Newton's iteration," *Numerical Algorithms*, vol. 33, no. 1–4, pp. 93–103, 2003.

[10] J.-F. Cai, M. K. Ng, and Y.-M. Wei, "Modified Newton's algorithm for computing the group inverses of singular Toeplitz matrices," *Journal of Computational Mathematics*, vol. 24, no. 5, pp. 647–656, 2006.

[11] M. Miladinović, S. Miljković, and P. Stanimirović, "Modified SMS method for computing outer inverses of Toeplitz matrices," *Applied Mathematics and Computation*, vol. 218, no. 7, pp. 3131–3143, 2011.

[12] W. R. Trench, "An algorithm for the inversion of finite Toeplitz matrices," *SIAM: SIAM Journal on Applied Mathematics*, vol. 12, no. 3, pp. 515–522, 1964.

[13] Y. Wei, J. Cai, and M. K. Ng, "Computing Moore-Penrose inverses of Toeplitz matrices by Newton's iteration," *Mathematical and Computer Modelling*, vol. 40, no. 1-2, pp. 181–191, 2004.

[14] S. L. Campbell and C. D. Meyer, *Generalized Inverses of Linear Transformations*, Pitman, London, UK, 1979.

[15] X. Chen and J. Ji, "Computing the Moore-Penrose inverse of a matrix through symmetric rank-one updates," *American Journal of Computational Mathematics*, vol. 1, no. 3, pp. 147–151, 2011.

[16] X. Chen and J. Ji, "The minimum-norm least-squares solution of a linear system and symmetric rank-one updates," *Electronic Journal of Linear Algebra*, vol. 22, pp. 480–489, 2011.

[17] X. Chen, "The generalized inverses of perturbed matrices," *International Journal of Computer Mathematics*, vol. 41, no. 3-4, pp. 223–236, 1992.

[18] C. D. Meyer Jr., "Generalized inversion of modified matrices," *SIAM Journal on Applied Mathematics*, vol. 24, pp. 315–323, 1973.

[19] J. K. Baksalary and O. M. Baksalary, "Relationships between generalized inverses of a matrix and generalized inverses of its rank-one-modifications," *Linear Algebra and Its Applications*, vol. 388, pp. 31–44, 2004.

[20] P. S. Stanimirović, V. N. Katsikis, and D. Pappas, "Computing 2,4 and 2,3-inverses by using the Sherman-Morrison formula," *Applied Mathematics and Computation*, vol. 273, pp. 584–603, 2016.

[21] T. N. E. Grevile, "Some applications of the pseudo-inverse of matrix," *SIAM Review*, vol. 3, pp. 15–22, 1960.

[22] F. E. Udwadia and R. E. Kalaba, "General forms for the recursive determination of generalized inverses: unified approach," *Journal of Optimization Theory and Applications*, vol. 101, no. 3, pp. 509–521, 1999.

[23] P. S. Stanimirović, M. Miladinović, I. Stojanović, and S. Miljković, "Application of the partitioning method to specific Toeplitz matrices," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 4, pp. 809–821, 2013.

[24] S. Chountasis, V. N. Katsikis, and D. Pappas, "Applications of the Moore-Penrose inverse in digital image restoration," *Mathematical Problems in Engineering*, vol. 2009, Article ID 170724, 12 pages, 2009.

[25] S. Chountasis, V. N. Katsikis, and D. Pappas, "Digital image reconstruction in the spectral domain utilizing the Moore-Penrose inverse," *Mathematical Problems in Engineering*, vol. 2010, Article ID 750352, 14 pages, 2010.

[26] S. Karanasios and D. Pappas, "Generalized inverses and special type operator algebras," *Facta Universitatis, Series: Mathematics and Informatics*, vol. 21, pp. 41–48, 2006.

[27] V. N. Katsikis and V. D. Pappas, "Fast computing of the Moore-Penrose inverse matrix," *Electronic Journal of Linear Algebra*, vol. 17, pp. 637–650, 2008.

[28] P. C. Hansen, J. G. Nagy, and D. P. O'Leary, *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, Pa, USA, 2006.

[29] R. C. Craddock, G. A. James, P. E. Holtzheimer III, X. P. Hu, and H. S. Mayberg, "A whole brain fMRI atlas generated via spatially constrained spectral clustering," *Human Brain Mapping*, vol. 33, no. 8, pp. 1914–1928, 2012.

[30] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.