Int. J. Appl. Math. Comput. Sci., 2013, Vol. 23, No. 4, 809–821
DOI: 10.2478/amcs-2013-0061

amcs

APPLICATION OF THE PARTITIONING METHOD TO SPECIFIC TOEPLITZ MATRICES

PREDRAG STANIMIROVIĆ [*], MARKO MILADINOVIĆ [*], IGOR STOJANOVIĆ [**], SLADJANA MILJKOVIĆ [*]

[*] Faculty of Sciences and Mathematics
University of Niš, Višegradska 33, 18000 Niš, Serbia
e-mail: pecko@pmf.ni.ac.rs

[**]Faculty of Computer Science
Goce Delčev University, 2000 Štip, Macedonia

We propose an adaptation of the partitioning method for determination of the Moore–Penrose inverse of a matrix augmented by a block-column matrix. A simplified implementation of the partitioning method on specific Toeplitz matrices is obtained. The idea for observing this type of Toeplitz matrices lies in the fact that they appear in the linear motion blur models in which blurring matrices (representing the convolution kernels) are known in advance. The advantage of the introduced method is a significant reduction in the computational time required to calculate the Moore–Penrose inverse of specific Toeplitz matrices of an arbitrary size. The method is implemented in MATLAB, and illustrative examples are presented.

Keywords: Moore–Penrose inverse, partitioning method, Toeplitz matrices, MATLAB, image restoration.

1. Introduction

The Moore–Penrose inverse is a useful tool for solving linear systems and matrix equations (Ben-Israel and Grevile, 2003; Penrose, 1956). Also, the Moore–Penrose inverse is frequently applied in the study of numerical properties of singular matrix pencils (Röbenack and Reinschke, 2011). Górecki and Łuczak (2013) used a generalization of the Moore–Penrose pseudoinverse to resolve the problem in the Linear Discriminant Analysis (LDA) technique. More precisely, the problem appears when a sample covariance matrix is singular, and thus cannot be inverted. A lot of direct methods have been proposed to compute the Moore–Penrose generalized inverse of a matrix (see, e.g., Ben-Israel and Grevile, 2003; Shinozaki *et al*., 1972). According to Shinozaki *et al*. (1972), they can be classified as methods based on matrix decomposition, methods applicable on bordered matrices and others methods (including Greville's recursive method, methods based on the formula $A^{\dagger} = (A^*A)^{(1,3)}A^*$ and Pyle's gradient projection methods).

The method based on singular-value decomposition possesses a very high computation load (approximately $\mathcal{O}(n^3)$ operations). Courrieu (2005) proposed an algorithm for fast computation of the Moore–Penrose inverse which is based on the reverse order law property and the full-rank Cholesky factorization of possibly singular symmetric positive matrices. A fast method for computing the Moore–Penrose inverse of full rank $m \times n$ matrices and of square matrices with at least one zero row or column was introduced by Karanasios and Pappas (2006), as well as Katsikis and Pappas (2008). This method exploits a special type of the tensor product of two vectors, which is usually used in infinite dimensional Hilbert spaces. Greville (1960) proposed a recurrent rule for determining the Moore–Penrose inverse. Udwadia and Kalaba (1997) gave an alternative and a simple constructive proof of Greville's formula. Due to its ability to undertake sequential computing, Greville's partitioning method has been extensively applied in statistical inference, filtering theory, linear estimation theory, system identification, optimization as well as in analytical dynamics (Graybill, 1983; MathWorks, 2009; Kalaba and Udwadia, 1996; 1993; Rao, 1962).

Recursive computation of the Moore–Penrose inverse of a matrix to which a block is added was presented by Bhimasankaram (1971). However, the

author proposes a proof which simply verifies that the output of his algorithm satisfies the four Penrose equations. Udwadia and Kalaba (1999) provided a constructive proof for recursive determination of the Moore–Penrose inverse of a matrix to which a block of columns is added. These results were also extended to other types of generalized inverses by Udwadia and Kalaba (1999).

Our intention in the present article is an adaptation of the recursive block partitioning method of Udwadia and Kalaba (1999) as well as the partitioning method of Greville (1960) for efficient numerical computation of the Moore–Penrose inverse of specific Toeplitz matrices. An application of the proposed method in the process of removing non-uniform blurs in the image restoration is presented.

In the next section we restate some basic definitions, motivations as well as both the recursive block partitioning method and the usual partitioning method for computing the Moore–Penrose inverse. We also give an outline of the process of forming a mathematical model that reflects the removal of non-uniform blurs in images. In Section 3 we describe the adaptation of the block partitioning method of Udwadia and Kalaba (1999) and the partitioning method of Greville (1960) on characteristic sparse Toeplitz matrices. The computational complexity of the modified partitioning method is investigated and compared with the complexities of other known methods. An application of the proposed method in the image restoration is given in Section 4. Several illustrative examples and comparisons are presented in the last section.

2. Preliminaries and motivation

Let $\mathbb{R}$ be the set of real numbers, $\mathbb{R}^{m\times n}$ be the set of $m\times n$ real matrices and $\mathbb{R}_r^{m\times n}$ be the set of $m\times n$ real matrices of rank r. The notation $A_i,\ i\in\{1,\dots,n\}$ denotes the first i columns of a matrix $A\in\mathbb{R}^{m\times n}$. Particularly, a_i (resp., a^i) means the i-th column (resp., the i-th row) of A. By ${}_iA_k$, $i\in\{1,\dots,n-1\}$, $k\in\{1,\dots,n-i\}$ we denote the submatrix of A which consists of the columns $a_{i+1},\dots,a_{i+k}$. The $m\times m$ identity matrix is denoted by I_m and O_m is the zero matrix of order $m\times m$. The notation $\mathbf{0}$ stands for the zero column vector of an appropriate dimension.

For the sake of completeness, we restate the block recursive algorithm for computing the Moore–Penrose inverse of matrix $B=[A|C]$, which denotes a matrix A augmented by an appropriate matrix C.

Lemma 1. (Udwadia and Kalaba, 1999) *Let $B=[A|\ C]$ be an $m\times(r+p)$ complex matrix whose last p columns are denoted by C. Let*

$$\begin{aligned} &R=I-AA^\dagger, \quad Q=(RC)^TRC,\\ &F=I-Q^\dagger Q, \quad Z=A^\dagger CF. \end{aligned} \tag{1}$$

Then

$$B^\dagger=\begin{bmatrix} A^\dagger(I-CV)\\ V\end{bmatrix}, \tag{2}$$

where

$$V=Q^\dagger C^TR+(I+Z^TZ)^{-1}Z^TA^\dagger(I-CQ^\dagger C^TR). \tag{3}$$

We also restate Greville's single-column finite recursive algorithm (Greville, 1960).

Lemma 2. (Greville, 1960) *Let A be an $m\times n$ complex matrix and a be an $m\times 1$ constant vector. By $[A|a]$ we denote the matrix A augmented by an appropriate vector a. Then*

$$[A|a]^\dagger=\begin{bmatrix} A^\dagger-db^*\\ b^*\end{bmatrix}, \tag{4}$$

where

$$\begin{aligned} &d=A^\dagger a,\\ &b=\begin{cases} \dfrac{1}{c^*c}c, & c\neq\mathbf{0},\\ \dfrac{1}{1+d^*d}(A^\dagger)^*d, & c=\mathbf{0},\end{cases}\end{aligned} \tag{5}$$

and

$$c=(I-AA^\dagger)a. \tag{6}$$

In the sequel, we describe the mathematical model that reflects the process of removing a non-uniform linear motion blur in images.

Suppose that the matrix $F\in\mathbb{R}^{r\times m}$ corresponds to the original image with picture elements $f_{i,j}, i=1,\dots,r$, $j=1,\dots,m$ and $G\in\mathbb{R}^{r\times m}$ with pixels $g_{i,j}$, $i=1,\dots,r$, $j=1,\dots,m$, is the matrix corresponding to the degraded image.

The process of non-uniform blurring assumes that the blurring of columns in the image is independent with respect to the blurring of its rows. In this case, the relation between the original and the blurred image can be expressed by the matrix equation

$$\begin{aligned} G=H_CFH_R^T, \quad &G\in\mathbb{R}^{r\times m}, \quad H_C\in\mathbb{R}^{r\times n},\\ &F\in\mathbb{R}^{n\times t}, \quad H_R\in\mathbb{R}^{m\times t}, \end{aligned} \tag{7}$$

where $n=r+l_c-1$, $t=m+l_r-1$, l_c is the length of the vertical blurring and l_r is the length of the horizontal blurring (in pixels).

To avoid the problem when the information from the original image spills over the edges of the recorded image, we supplement the original image with boundary pixels that best reflect the original scene. Without any confusion,

we use the same symbol F for the enlarged original image (matrix) with the remark that F is now of dimensions $n \times t$.

The Moore–Penrose inverse appears as a useful tool in the image restoration process (Bovik, 2005; Chountasis *et al.*, 2009a; 2010; 2009b). The approach based on the usage of the matrix pseudo-inverse in image restoration is one of the most common techniques (Bovik, 2005).

The problem of removing a uniform linear motion blur, based on the application of the algorithm of Lagrange multipliers to the model (7), was investigated by Stojanović *et al*. (2012). In order to restore the blurred image G included in the model (7), in the present paper we use the Moore–Penrose inverse approach, which leads to the solution

$$\widetilde{F} = H_C^{\dagger} G (H_R^{\dagger})^T. \tag{8}$$

In other words, the main problem we are faced with is to choose an efficient algorithm for computing the Moore–Penrose inverse $H_C^{\dagger}$ and $H_R^{\dagger}$. The algorithm used by Chountasis *et al.* (2009a; 2010) is based on the fast computational method for finding the Moore–Penrose inverse of a full rank matrix, introduced by Karanasios and Pappas (2006), as well as Katsikis and Pappas (2008).

3. Adaptation of the partitioning method

We define appropriate adaptations of both the block partitioning method and Greville's single-column partitioning method. The motivation for using these methods lies in the specific structure of convolution matrices H_C and H_R. The appropriate structure of matrices H_C and H_R reduces the computational complexity of the partitioning method in calculating pseudoinverses $H_C^{\dagger}$ and $H_R^{\dagger}$.

In the remainder of the section we investigate an adaptation of the partitioning method on computing the Moore–Penrose inverse of characteristic Toeplitz matrices, given in the form

$$H = \left[\begin{array}{c|c} H_m & {}_{m+1}H_n \end{array}\right] \in \mathbb{R}^{m \times n}, \quad n = m+l-1, \tag{9}$$

where $l \geq 1$

$$H_m = \begin{bmatrix} h_1 & h_2 & h_3 & \dots & h_l & 0 & 0 & 0 \\ 0 & h_1 & h_2 & h_3 & \dots & h_l & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \dots & \ddots & 0 \\ 0 & 0 & 0 & h_1 & h_2 & h_3 & \dots & h_l \\ 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 & \dots \\ 0 & 0 & 0 & 0 & 0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ddots & \ddots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_1 \end{bmatrix}, \tag{10}$$

$$ {}_{m+1}H_n = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ h_l & 0 & 0 & 0 \\ \dots & h_l & 0 & 0 \\ \ddots & \dots & \ddots & 0 \\ h_2 & h_3 & \dots & h_l \end{bmatrix}. \tag{11}$$

Courrieu (2005) compared the introduced method *geninv* with four usual algorithms (Greville's partitioning method, the SVD method, full rank QR and an iterative method of optimized order (Ben-Israel and Grevile, 2003)). It is claimed that the best results are achieved by the *geninv* method, while the worst results are generated by the partitioning method.

Our motivation for using the block partitioning method (Udwadia and Kalaba, 1999) and Greville's partitioning method (1960) in order to find $H^{\dagger}$ is explained as follows. The quadratic block H_m of the matrix H is clearly a nonsingular upper triangular Toeplitz matrix, so that its inverse can be computed very easily. Later, the recursive rules (1)–(3) and (4)–(6) can be significantly simplified, according to the specific structure of the block C and the vector a, respectively.

The following particular case of Lemma 1 defines simplifications of the recursive steps (1)–(3) in calculating the Moore–Penrose inverse $H^{\dagger}$.

Lemma 3. *Assume that the matrix $H \in \mathbb{R}^{m \times n}$, $n = m+l-1$, causes the blurring process in* (19). *The Moore–Penrose inverse of its first $p+k$ columns, partitioned in the block form $H_{p+k} = [H_p |\ {}_pH_k]$, $p \in \{1, \dots, n-1\}$, $k \in \{1, \dots, n-p\}$, is defined by*

$$H_{p+k}^{\dagger} = \begin{bmatrix} H_p^{\dagger}\left(I - {}_pH_k \cdot B^T\right) \\ B^T \end{bmatrix} = \begin{bmatrix} H_p^{\dagger} - DB^T \\ B^T \end{bmatrix}, \tag{12}$$

where

$$D = H_p^{\dagger} \cdot {}_pH_k, \quad B = (H_p^{\dagger})^T D (I + D^T D)^{-1}. \tag{13}$$

Proof. This follows from Lemma 1, taking into account that the degradation matrix is of full row rank and the fact that the equalities in (1) reduce to

$$\begin{aligned} &R = Q = O_m, \\ &F = I_m,\ V = B^T, \\ &Z = D = H_p^{\dagger} \cdot {}_pH_k, \end{aligned}$$

observing this particular case. ■

Also, Greville's partitioning method (4)–(6) reduces to the following computational procedure.

Lemma 4. *The Moore–Penrose inverse of the matrix H_i is equal to*

$$H_i^\dagger = \begin{bmatrix} H_{i-1}^\dagger - d_i b_i^T \\ b_i^T \end{bmatrix}, \tag{14}$$

where

$$d_i = H_{i-1}^\dagger \cdot h_i \quad b_i = \left(1 + d_i^T d_i\right)^{-1} (H_{i-1}^\dagger)^T d_i. \tag{15}$$

Since we know the inverse H_m^{-1}, which is completely determined by the vector x from (16), any pragmatical implementation of the new method uses only partitions of the form $H_{p+k} = [H_p|\ {}_pH_k], p \geq m, k \in \{1, \ldots, n-m\}$.

According to Lemma 3, we propose the following algorithm for computing the Moore–Penrose inverse of the specific Toeplitz matrix H.

Algorithm 1. Computing the Moore–Penrose inverse of the matrix H.

Input: The matrix H of dimensions $m \times (m+l-1)$ given by (9).

Step 1. Separate the block H_m of the matrix H.

Step 2. Generate $H_m^\dagger = H_m^{-1}$ using the vector x from (16).

Step 3. Take $p = m$ and choose k such that $1 \leq k \leq l-1$ as well as $(l-1)/k \in \mathbb{N}$.

Step 4. Compute $H_{p+k}^\dagger = [H_p|\ {}_pH_k]^\dagger$ according to Lemma 3.

Step 5. Set $p = p + k$.

Step 6. If $p \neq n$ then go to Step 4; otherwise, go to the next step.

Step 7. Return $H_n^\dagger$.

It is not difficult to verify that the choice $k = 1$ in all the recursive steps of Algorithm 1 produces the particular case of Greville's recursive method corresponding to Lemma 4. Also, in the case $p = m, k = l-1$, Algorithm 1 reduces to Algorithm 2.

Algorithm 2. Computing the Moore–Penrose inverse of the matrix H in the case $k = l - 1$.

Input: The matrix H of dimensions $m \times (m + l - 1)$ defined in the blurring process (19).

Step 1. Separate the matrix H into two blocks H_m and ${}_mH_{l-1}$, that is, $H = [H_m|\ {}_mH_{l-1}]$.

Step 2. Generate $H_m^\dagger = H_m^{-1}$ using the vector x from (16).

Step 3. Compute $H^\dagger = [H_m|\ {}_mH_{l-1}]^\dagger$ according to Lemma 3.

Choosing the most efficient case with respect to the computational time, we derive an efficient method for computing the Moore–Penrose inverse of the degradation matrix H, and respectively an efficient method for image restoring processes based on Eqn. (8).

In order to invert the matrix H_m defined in (10), look at the matrix equation $H_m H_m^{-1} = I$. Since the matrix H_m is an upper triangular Toeplitz matrix, it is well known that its inverse is also an upper triangular Toeplitz matrix. Therefore, the whole matrix H_m^{-1} is determined by its last column. We denote the last column of H_m^{-1} by x. To generate the vector x, we consider the following equation:

$$H_m \cdot x = e_m, \tag{16}$$

where e_m denotes the last column of the identity matrix I_m. Looking at the methods incorporated in the programming package `MATLAB`, we decide to apply the function `linsolve()` using the option `opts.UT = true` that imposes computations adopted to upper triangular matrices. After computing the vector x, it is easy to determine the whole matrix H_m^{-1}.

3.1. Complexity of the adopted partitioning method.

In order to determine the best choice of the positive integer k in Algorithm 1, we compare the computational complexities of Algorithms 1 and 2. Let us denote by $I(n)$ the complexity of the algorithm for inverting a given $n \times n$ matrix (as in the work of Cormen *et al.* (2001)). Also, by $A(n)$ we denote the complexity of the addition/subtraction of two $n \times n$ matrices and by $M(m, n, k)$ the complexity of multiplying $m \times n$ matrix by $n \times k$ matrix. The simpler notation $M(n)$ (taken from Cormen *et al.*, 2001) is used instead of $M(n, n, n)$.

In the remainder of this section we consider the computational complexity of the two opposite choices in Algorithm 1. The choice $p = m, k = 1$ is called the *Partitioning Method* (PM). The opposite choice $p = m, k = l-1$, used in Algorithm 2, is called the *Block Partitioning Method* (BPM).

It is well known that the complexity of matrix inversion is equal to that of matrix multiplication. More precisely, the ordinary inverse of any real nonsingular $n \times n$ matrix can be computed in time $I(n) = \mathcal{O}(M(n))$ (Cormen *et al.*, 2001). The notation $\mathcal{O}(f(n))$ is described, also, by Cormen *et al.* (2001).

The complexity of Algorithm 2 is of the order

$$\begin{aligned} E_{BPM} = {} & I(m) + 3\,M(m, m, l-1) + 2\,M(m, l-1, l-1) \\ & + I(l-1) + A(l-1) + A(m). \end{aligned} \tag{17}$$

Scanning Algorithm 1 in a similar way, it is not difficult to verify that its i-th recursive step requires the complexity of the order

$$\begin{aligned} C_i = {} & M(m+i-1, m, 1) + M(1, m+i-1, 1) \\ & + M(m, m+i-1, 1) \end{aligned}$$

for each $i = 1, \ldots, l-1$. Therefore, the complexity of the complete algorithm is

$$E_{PM} = I(m) + \sum_{i=1}^{l-1} C_i. \tag{18}$$

Since $l \ll m$, we conclude that the computational complexity for $(I + D^T D)^{-1}$, equal to $I(l-1)$, is substantially smaller than the complexity of required matrix multiplications. Also, upon the adopted implementation for computing $H_m^{\dagger} = H_m^{-1}$, based on (16), we have

$$I(m) \approx \mathcal{O}\left(\frac{(m-1)m}{2}\right) = \mathcal{O}(m^2).$$

Therefore, the upper bound estimation of complexities E_{BPM} and E_{PM} does not include the computational effort of the included matrix inversions. The upper bounds for the complexity of Algorithms 2 and 1 are given respectively by

$$E_{BPM} \leq \mathcal{O}(M(m, m, l-1)),$$
$$E_{PM} \leq l \cdot \mathcal{O}(M(m, m+l, 1)).$$

If A and B are respectively $m \times n$ and $n \times k$ matrices, then the computational complexity of the product $A \cdot B$ in MATLAB is $M(m, n, k) = \mathcal{O}(nmk)$, since MATLAB does not use Strassen's method (or any other fast method) for matrix multiplication. Therefore, according to (17) and (18),

$$E_{BPM} \leq \mathcal{O}(m^2 l - m^2), \quad E_{PM} \leq \mathcal{O}(m^2 l + m l^2).$$

Consequently, the upper bound for the computational complexity of Algorithm 2 is less than the computational complexity of Algorithm 1.

According to these theoretical investigations as well as on the basis of performed numerical experiments, we conclude that Algorithm 2 is a better choice. The CPU times depend upon two parameters: computational complexity and implementation details incorporated into the programming language MATLAB.

On the other hand, according to the known result of Noda *et al*. (1997), the number of required operations for Greville's method is equal to

$$\phi(\text{Greville}) = 2m^2 n - \frac{nr^2}{2},$$

where m and n are the dimensions of the input matrix and r is its rank. In our case, the number of arithmetic operations required by the original Greville method for computing $H^{\dagger}$, $H \in \mathbb{R}_m^{m \times (m+l-1)}$, is equal to

$$E_{\text{Greville}} = 2m^2(m+l-1) - \frac{(m+l-1)m^2}{2} \approx \mathcal{O}(m^3).$$

3.2. Analysis of methods for computing $H_m^{\dagger}$. In order to confirm the efficiency of Algorithm 2, we compared the block partitioning method with three recently announced methods for computing the Moore–Penrose inverse (Chountasis *et al.*, 2009a; 2009b; Courrieu, 2005; Katsikis and Pappas, 2008). Therefore, the following algorithms for computing the Moore–Penrose inverse are compared:

1. *block partitioning method*, presented by Algorithm 2,
2. *Ginv method*, defined by the MATLAB function `ginv.m` of Katsikis and Pappas (2008),
3. *Qrginv method*, defined by the MATLAB function `qrginv.m` of Chountasis *et al*. (2009b) and Katsikis *et al*. (2011),
4. *Courrieu method* of Courrieu (2005).

A comparison of several direct algorithms for computing the Moore–Penrose inverse of full column rank matrices was presented by Smoktunowicz and Wróbel (2012). Also, the computational cost of these methods for computing the Moore–Penrose inverse of a full column rank matrix $A \in \mathbb{R}^{m \times n}$ is given in Table 1 of Smoktunowicz and Wróbel (2012). In our case, we have the situation $A = H^T \in \mathbb{R}^{m+l-1 \times m}$. According to the computational complexities presented by Smoktunowicz and Wróbel (2012), the complexity of the Courrieu method is equal to

$$E_{\text{Chol}} = 3(m+l-1)m^2 + m^3/3,$$

and the complexity of Qrginv method is

$$E_{\text{Qrpivot}} = 5(m+l-1)m^2 - 4m^3/3.$$

The Ginv method for computing $A^{\dagger}$ is based on the formula $A^{\dagger} = (AA^T)^{-1}A$ and the MATLAB implementation is based on the least squares solution of the matrix equation $(A^T A)X = A^T$. In the particular case of $A = H$, the computational complexity of the Ginv method is

$$E_{\text{Ginv}} = \mathcal{O}(M(m, m+l-1, m)) + I(m) + M(m, m, m+l-1).$$

Taking into account $l < m$, we derive the following computational complexities:

$$E_{\text{Chol}} \approx \mathcal{O}(m^3), \quad E_{\text{Qrpivot}} \approx \mathcal{O}(m^3),$$
$$E_{\text{Ginv}} \approx \mathcal{O}(m^3), \quad E_{\text{Greville}} \approx \mathcal{O}(m^3).$$

Since

$$E_{BPM} \approx \mathcal{O}(m^2 l),$$

we conclude that the block partitioning method has the smallest computational complexity.

4. Application in image restoration

Images are aimed to memorize useful information, but unfortunately, the presence of blurs is unavoidable. A motion blur is an effect caused by relative motion between the camera and the scene during image exposure time. Restoration of motion-blurred images has been a fundamental problem in digital imaging for a long time. The recovery of an original image from degraded observations is of crucial importance and finds application in several scientific areas including medical imaging and diagnosis, military surveillance, satellite and astronomical imaging, remote sensing, etc. Expectably, the field of image restoration has been of great interest in the scientific literature (Banham and Katsaggelos, 1997; Chantas *et al.*, 2007; Chountasis *et al.*, 2009a; 2010; Hillebrand and Muller, 2007; Schafer *et al*, 1981). Also, edge preserving regularization methods, in the context of image restoration and denoising, are presented by Prasath (2011).

It is known that an arbitrary linear blurring process can be represented by (7), where the matrices H_C and H_R are characteristic Toeplitz matrices of the form (9)–(11) with the sum of the elements $h_1, \ldots, h_l$ equal to 1 (see, e.g., Hansen *et al.*, 2006).

The parameter l represent the length of the horizontal/vertical blur (in pixels).

In order to see how boundary conditions can be incorporated in the model, for the sake of simplicity, let us retain the horizontal blurring model ($H_C = I, H_R = H$). An arbitrary i-th row g^i of the blurred image can be expressed using the i-th row f^i of the original image extended by incorporating the boundary conditions as

$$\left(g^i\right)^T = H\ \left(f^i\right)^T \Longleftrightarrow \begin{bmatrix} g_{i,1} \\ \vdots \\ g_{i,m} \end{bmatrix} = H \begin{bmatrix} f_{i,1} \\ \vdots \\ f_{i,n} \end{bmatrix}, \quad (19)$$

$i = 1, 2, \ldots, r$, where $l-1$ elements of the vector f^i are not actually the pixels from the original scene; rather, they are *boundary pixels*. How many boundary pixels are placed at the top of the vector f^i depends of the nature and direction of the movement (the cause of the blur). However, the rest of them, i.e., $l-1$ minus the number of pixels placed on top of the vector f^i, would present the boundary pixels right of the horizontal line, and should be placed at the bottom of the vector f^i (Hansen *et al.*, 2006).

We consider the problem of removing a non-uniform blur, which corresponds to an integral number of pixels, in images. A real-life linearly blurred image (denoted by the image array G) can be modeled as a linear convolution of the original image (represented by the image array F) with a PSF, also known as the blurring kernel (represented by the matrix H).

We pay special attention to a Gaussian blur. Blurring that is caused by atmospheric turbulence, out-of-focus and motion of the camera, can be modeled by the Gaussian blurring function (Hufnagel and Stanley, 1964). In the Gaussian blur model the vector $h = [h_1, h_2, \ldots, h_l]$ is equal to

$$h = [\gamma(-p), \ldots, \gamma(0), \ldots, \gamma(k)],$$

where $\gamma(i) = e^{-i^2/(2s^2)}$, $p = \lfloor l/2 \rfloor$, $k = \lceil l/2 \rceil$. The parameter s represents the width of the blurring function. The vector h is normalized by dividing each element of h by the sum of its elements. This vector represents the so-called one-dimensional Gaussian function. The non-uniform Gaussian blurring model $G = H_C F H_R^T$ corresponds to the model where the blurring matrix is obtained by convolving the original matrix F with the PSF function which is equal to the two-dimensional Gaussian matrix $PSF = [p_{i,j}]$ with entries $p_{i,j} = e^{-i^2/(2s^2)-j^2/(2s^2)}$.

4.1. Blur and noise restoration. In this section, attention is paid to the model images degraded by a sequence of mutually independent operations. First, noise is imposed on the image and after that the noisy image is blurred by the non-uniform Gaussian function. In this case the mathematical model of the non-uniform blurring process presented by (7) becomes

$$G_N = H_C(F + N)H_R^T = H_C F_N H_R^T, \quad (20)$$

where G_N is a blurred noisy image and N is an additive noise. Two steps are used to restore the original image:

1. Calculate the restored matrix $\widetilde{F}_N = H_C^\dagger G_N (H_R^\dagger)^T$ of F_N.
2. Generate the image $\widetilde{F}$ by applying the filtering process to the image $\widetilde{F}_N$. Depending on the type of noise, we use a rotationally symmetric Gaussian low-pass filter or a two dimensional median filter.

5. Numerical results

5.1. Experiments on a randomly generated matrix H. In this subsection we compare the CPU time required for computation of the Moore–Penrose inverse of a randomly generated Toeplitz matrix H of the form (9)–(11). Experiments are done using `MATLAB` on an Intel(R) Core(TM) i5 CPU M430 @ 2.27 GHz 64/32-bit system with 4 GB RAM memory. Since the algorithms we compare with are implemented in `MATLAB`, we also chose `MATLAB` as a framework for the implementation of the proposed algorithms.

In Figs. 1 and 2 we present the results which refer to the computational time t (sec) needed to compute the Moore–Penrose inverse $H^\dagger$ as a function of the length of the blurring process $l \leq 90$ (pixels). The values

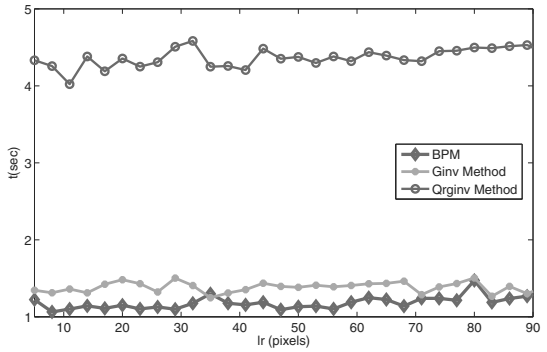


Fig. 1. CPU time for computing the MP inverse of the random matrix H versus l_r ($l_c = 30$).

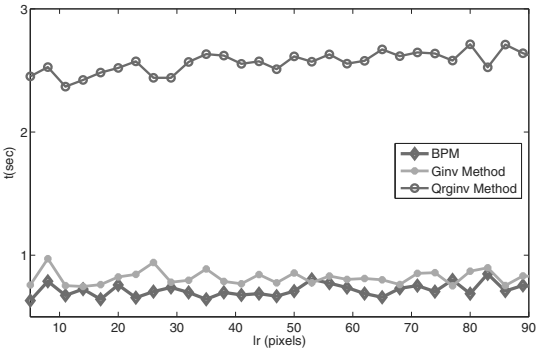


Fig. 3. CPU time for removing a blur caused by the Gaussian function and noise versus l_r ($l_c = 25$).

incorporated in these figures are obtained for a randomly generated matrix of dimensions 1000×1200. CPU times illustrated in Figs. 1 and 2 confirm that the proposed BPM for computing $H^{\dagger}$ is faster than the other methods considered.

It is easy to observe that the block partitioning method overcomes the Ginv and Qrginv methods. On the other hand, Katsikis and Pappas (2008) concluded that the Ginv method is faster with respect to the Courrieu method. Thus, after the modifications described before, the partitioning method becomes the fastest compared with the three examined methods for computing the Moore–Penrose inverse.

In addition, we compare the accuracy of the results of our method with the other three methods. We use the *ginvtest* function of Katsikis and Pappas (2008) and the accuracy was examined with the matrix 2-norm in error matrices corresponding to the four Moore–Penrose equations. In Table 1 we present average errors for different values of the parameter s, regarding the four Penrose equations. The presented average values were generated by varying the parameter l from 5 to 90 with Step 1. Based on the results shown in Table 1, the following conclusions are imposed. The greatest norms and thus the worst results are generated by the Courrieu method. The BPM produces the best results for the Penrose equations (1)–(3). The Ginv and Qrginv methods give slightly better results regarding the matrix equation (4) with respect to the BPM.

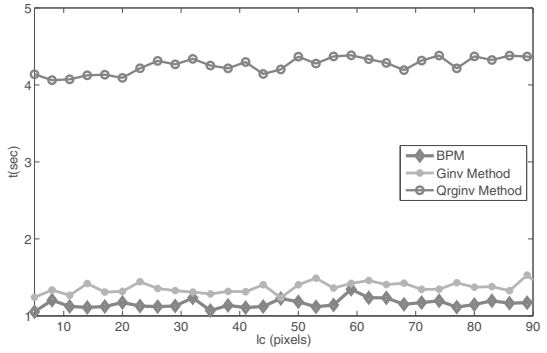


Fig. 2. CPU time for computing the MP inverse of the random matrix H versus l_c ($l_r = 25$).

5.2. Experiments on the matrix H resulting from a linear system convolution kernel. In this subsection we compare the CPU time required for computation of the Moore–Penrose inverse of Toeplitz matrix H resulting from a linear system convolution kernel. In this way, we compare the computational time of the image restoration methods which are based on the Moore–Penrose inverse approach. Thus, we compare that required by our method with the CPU time required by the previously mentioned methods. The computational time needed to restore the degraded X-ray image by means of the methods which use the Moore–Penrose inverse approach is shown in Figs. 3 and 4. For a given image, the varying parameter is the parameter l_r (l_c) that takes values between 5 and 90.

As expected, the proposed method shows better performances with respect to the other tested methods.

The proposed method is not only restricted to restoration of blurred X-ray images. As a confirmation, we compare the speed of our method with the speed of other methods on a standard `MATLAB` image Lena. Results for the Lena image (under the periodic boundary condition), degraded by the Gaussian white noise of mean 0 and variance 0.01 and blurred by the non-uniform Gaussian function (20), are presented in Figs. 5 and 6. A rotationally symmetric Gaussian low pass filter of size 3 with standard deviation 45 is used for filtering.

Table 1. Average error results regarding the four Moore–Penrose equations for $5 <= l <= 90$.

s	2-norm in error matrices	BPM	Ginv	Qrginv	Courrieu
100	$\|TT^{\dagger}T - T\|_2$	1.4630×10^{-14}	1.9417×10^{-14}	1.1482×10^{-14}	1.0614×10^{-10}
	$\|T^{\dagger}TT^{\dagger} - T^{\dagger}\|_2$	3.0370×10^{-12}	7.8286×10^{-9}	1.6753×10^{-11}	6.3123×10^{-8}
	$\|TT^{\dagger} - (TT^{\dagger})^*\|_2$	9.8858×10^{-14}	1.1352×10^{-11}	9.9422×10^{-13}	5.7959×10^{-8}
	$\|T^{\dagger}T - (T^{\dagger}T)^*\|_2$	3.0602×10^{-13}	1.8363×10^{-14}	1.0475×10^{-13}	5.4743×10^{-11}
200	$\|TT^{\dagger}T - T\|_2$	1.4206×10^{-14}	1.9703×10^{-14}	1.1243×10^{-14}	1.0124×10^{-10}
	$\|T^{\dagger}TT^{\dagger} - T^{\dagger}\|_2$	2.9713×10^{-12}	7.9161×10^{-9}	1.7050×10^{-11}	5.9056×10^{-8}
	$\|TT^{\dagger} - (TT^{\dagger})^*\|_2$	9.6917×10^{-14}	1.1066×10^{-11}	9.9735×10^{-13}	5.4223×10^{-8}
	$\|T^{\dagger}T - (T^{\dagger}T)^*\|_2$	2.8939×10^{-13}	1.8014×10^{-14}	1.0298×10^{-13}	5.2727×10^{-11}
300	$\|TT^{\dagger}T - T\|_2$	1.3947×10^{-14}	1.9369×10^{-14}	1.1166×10^{-14}	1.0056×10^{-10}
	$\|T^{\dagger}TT^{\dagger} - T^{\dagger}\|_2$	2.9527×10^{-12}	7.7109×10^{-9}	1.6426×10^{-11}	5.8595×10^{-8}
	$\|TT^{\dagger} - (TT^{\dagger})^*\|_2$	9.5872×10^{-14}	1.0968×10^{-11}	9.5566×10^{-13}	5.3950×10^{-8}
	$\|T^{\dagger}T - (T^{\dagger}T)^*\|_2$	2.8583×10^{-13}	1.7938×10^{-14}	1.0321×10^{-13}	5.2229×10^{-11}
400	$\|TT^{\dagger}T - T\|_2$	1.4104×10^{-14}	1.9736×10^{-14}	1.1157×10^{-14}	1.0066×10^{-10}
	$\|T^{\dagger}TT^{\dagger} - T^{\dagger}\|_2$	2.9505×10^{-12}	7.8669×10^{-9}	1.7028×10^{-11}	5.8529×10^{-8}
	$\|TT^{\dagger} - (TT^{\dagger})^*\|_2$	9.6935×10^{-14}	1.0957×10^{-11}	9.8089×10^{-13}	5.3809×10^{-8}
	$\|T^{\dagger}T - (T^{\dagger}T)^*\|_2$	2.9034×10^{-13}	1.7916×10^{-14}	1.0235×10^{-13}	5.2246×10^{-11}
500	$\|TT^{\dagger}T - T\|_2$	1.4107×10^{-14}	1.9210×10^{-14}	1.1180×10^{-14}	9.9086×10^{-11}
	$\|T^{\dagger}TT^{\dagger} - T^{\dagger}\|_2$	2.8297×10^{-12}	7.2584×10^{-9}	1.6637×10^{-11}	5.7496×10^{-8}
	$\|TT^{\dagger} - (TT^{\dagger})^*\|_2$	9.4099×10^{-14}	1.0957×10^{-11}	9.6248×10^{-13}	5.2747×10^{-8}
	$\|T^{\dagger}T - (T^{\dagger}T)^*\|_2$	2.9047×10^{-13}	1.7940×10^{-14}	1.0084×10^{-13}	5.2127×10^{-11}

(a) (b) (c)

Fig. 5. Removal of a blur, caused by the Gaussian model with $l_c = 40$ and $l_r = 35$, on a Lena image: original image (a), blurred noisy image (b), Moore–Penrose restored image (c).

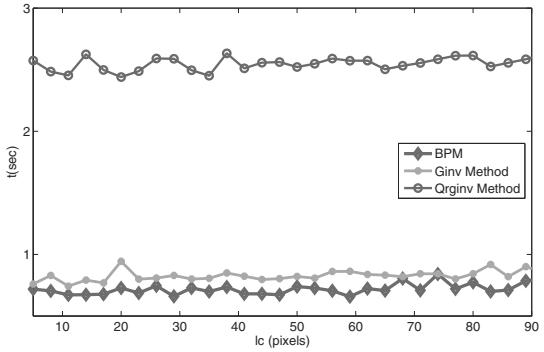


Fig. 4. CPU time for removing a blur caused by the Gaussian function and noise versus l_c ($l_r = 35$).

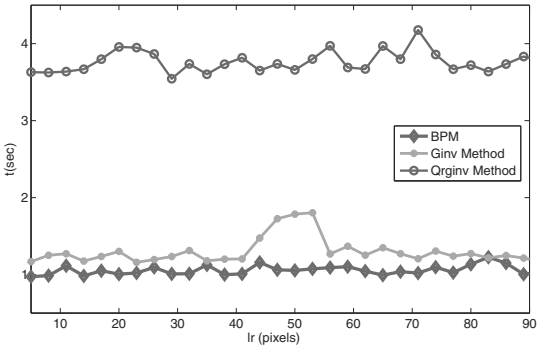


Fig. 6. Time versus l_r in the removal of a blur given by the model (20) and Gaussian noise ($l_c = 50$), for a Lena image.

5.3. Comparison with the least squares solution. For simplicity, let us again focus on a blur which is caused by a horizontal motion. In this section we compare the

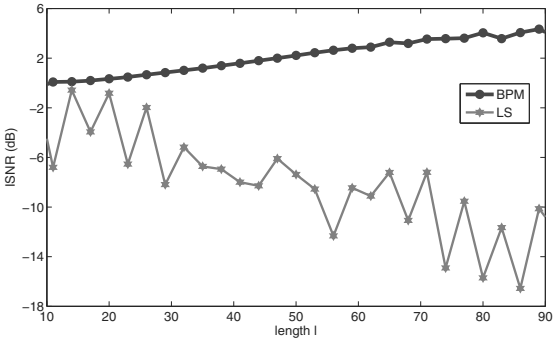


Fig. 8. ISNR values arising from the Moore–Penrose inverse approach and the LS solution.

effectiveness of the method based on the Moore–Penrose inverse with the method which uses the least squares solution of the following linear system:

$$g_{i,j} = \sum_{k=0}^{l-1} h_k f_{i,j+k}, \tag{21}$$

$i = 1, \ldots, r$ and $j = 1, \ldots, m$, arising from (9)–(19) (with $n = m + l - 1$). The corresponding solution is derived using the standard MATLAB function `mrdivide()` and it will be denoted as the LS solution in test examples. The function `mrdivide(B,A)` (or its equivalent `B/A`) performs matrix right division (forward slash) (MathWorks, 2010). Matrices B and A must have the same number of columns.

In the case of the underdetermined or overdetermined system of equations $G = FH^T$, the least squares solution $F = G/H^T$ is usually not the same as the least square solutions of the minimal norm $F = G(H^T)^\dagger$. A comparison between the Moore–Penrose approach and least square solutions for the problem (19) is illustrated in Fig. 7.

Blurring used in this example is uniform Gaussian with length $l = 50$ and Gaussian white noise with mean 0 and variance 0.05. The left picture in Fig. 7 shows the restored image obtained as the direct solution of the system (21), while the right image shows the image in which the blur is restored based on the usage of the Moore–Penrose inverse.

The difference in the quality of the restored images is in favor of the Moore–Penrose inverse approach and can be seen with the human eye. This illustration was confirmed by the corresponding values of the ISNR parameter, which are presented in Fig. 8.

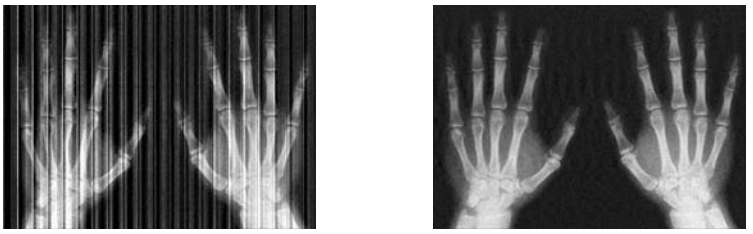
5.4. Comparison with other image restoration methods. In this subsection we compare the adopted block partitioning method with several known image restoration methods. In fact, the blurring matrices H_c and H_r are assumed to be known in these numerical experiments, whereas in practical applications of image processing they first have to be identified from the distorted image. But, eventually, one can imagine an application with a known blurring matrix, e.g., to secure the transmission and storage of original image artwork, where a synthetic image deterioration is caused by the sender and the corresponding image deblurring is done on the receiver's side.

Figure 9 demonstrates the efficiency of four different methods for image restoration: the Moore–Penrose inverse based approach, the Wiener Filter (WF), Lucy–Richardson (LR) algorithm and the Tikhonov Regularization (TR) method. The methods are tested on an image which is taken from the results obtained after the Google Image search with the keyword "X-ray image" and the location of the image is at `http://www.tetburyhospital.co.uk/admin/subroutines/blob.php?id=13&blob=ablob`.

For the implementation of the Wiener filter and the Lucy–Richardson algorithm we used incorporated built-in functions from the MATLAB package. For the Wiener filter we use a MATLAB function with two input parameters: a blurred noisy image G_N and a point-spread function PSF. For the Lucy–Richardson algorithm we an used additional parameter for the number of iterations with a constant value 10. Implementation of the Tikhonov regularization method is based on the Kronecker decomposition and the code presented by Hansen *et al*. (2006).

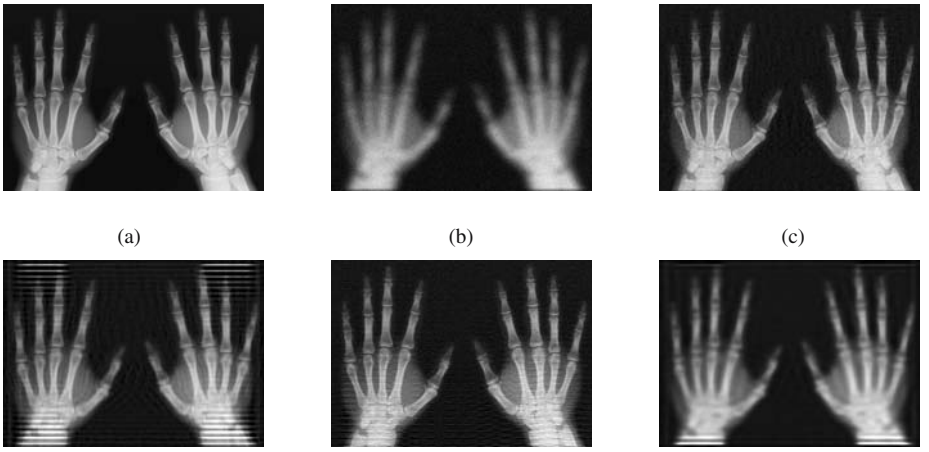
In Fig. 9 we illustrate the original, the blurred noisy and the restored images obtained by different methods. The graphics denoted as *Original image* from Fig. 9 show the original X-ray image. The image is divided into $r = 750$ rows and $m = 1050$ columns. To prevent a loss of information from the boundaries of the image, we assumed zero boundary conditions, which implies that the values of the pixels of the original image F outside the image window are zero. This choice is natural for X-ray images since the background of these images is black. The pixels of the original image are degraded by the Gaussian white noise with zero mean and variance 0.01 and later blurred by a non-uniform Gaussian function according to the model (20). For filtering we use a rotationally symmetric Gaussian low pass filter of size 3 with standard deviation 45.

The difference in quality of the restored images regarding three methods (Moore–Penrose, Wiener and Tikhonov) is insignificant and can hardly be seen by a human eye. For this reason, we use a common method for comparing restored images, i.e., we analyze the so-called Improved Signal-to-Noise Ratio (ISNR). The results for the parameters ISNR (Bovik, 2009), presented in Fig. 10, show that the restoration of the serial degraded images



(a) (b)

Fig. 7. Restoration arising from the LS solution and the Moore–Penrose inverse: LS solution (a), Moore–Penrose inverse solution (b).



(a) (b) (c)

(d) (e) (f)

Fig. 9. Removal of a blur, caused by a Gaussian model with $l_c = 25$ and $l_r = 45$, on an X-ray image: original image (a), blurred noisy (b), Moore–Penrose restoration (c), Wiener filter restoration (d), Tikhonov regularization (e), Lucy–Richardson restoration (f).

with the Moore–Penrose inverse is more reliable and accurate than restoration with other mentioned methods. In Fig. 10 we used $s = l_r/2$ and $s = l_c/2$ for the non-uniform blurring process. The graph marked as *MP inverse* illustrates the numerical values generated by an arbitrary method for computing the Moore–Penrose inverse.

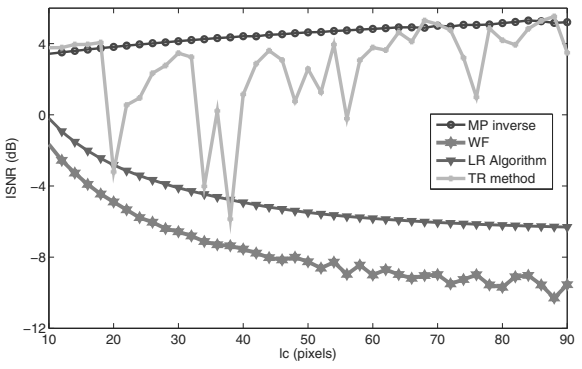


Fig. 10. ISNR versus l_c for the removal of a blur given by the model (20) ($l_r = 35$).

Similar results are generated for other values of the parameter s. To illustrate this fact, in Fig. 11 are presented results for the ISNR corresponding to the choice $s = 30$.

To confirm our results, in the next two figures we present results for the Dice Coefficient (DC) as a similarity measure between sets. The rank of the DC is from 0 to 1, where 0 indicates the sets are disjoint and 1 indicates the sets are identical (Dice, 1945; Craddock *et al.*, 2012). The parameters used in Figs. 12 and 13 are the same as in Figs. 10 and 11.

6. Conclusion

The presented method is based on appropriate adaptations of well-known computational methods introduced by Udwadia and Kalaba (1999), as well as Greville (1960). It is suitable for application to specific (sparse) Toeplitz matrices. Using the specific structure of these matrices (denoted by H) as well as the fact that the inverse of its

square part ($H_m^\dagger$) can be computed easily, we adjust the partitioning methods in order to obtain the most efficient adaptation.

We compare the obtained method with respect to two methods for fast computing the Moore–Penrose inverse introduced by Karanasios and Pappas (2006), as well as Katsikis and Pappas (2008), and used by Chountasis *et al.* (2009a; 2010; 2009b) along with the Courrieu method (Courrieu, 2005).

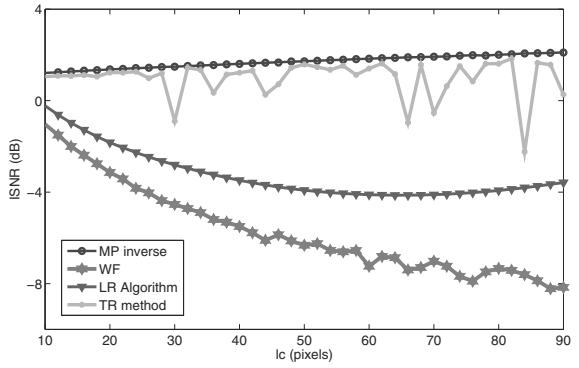


Fig. 11. ISNR versus l_c for the removal of a blur given by the model (20) ($l_r = 35$ and $s = 30$).

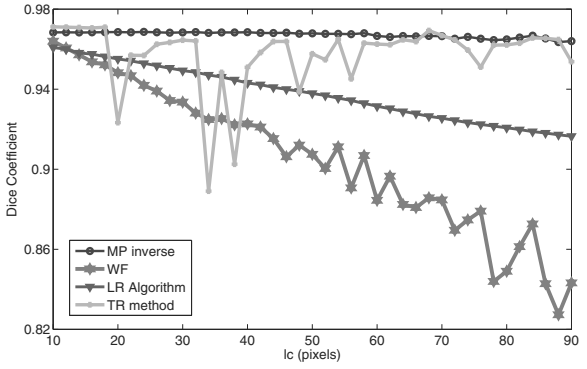


Fig. 12. DC versus l_c for the removal of a blur given by the model (20) ($l_r = 35$).

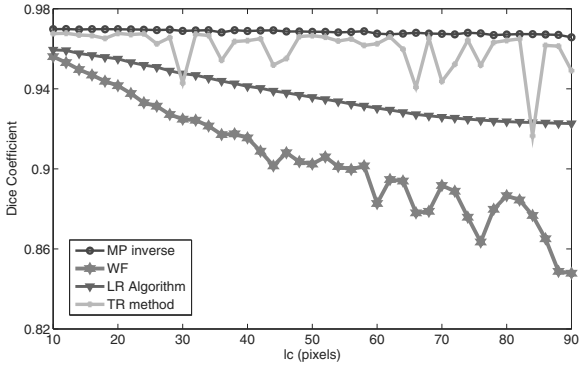


Fig. 13. DC versus l_c for the removal of a blur given by the model (20) ($l_r = 35$ and $s = 30$).

The advantage of the proposed method is a significant reduction of the CPU time required to obtain the Moore–Penrose inverse of the observed class of matrices. This fact coincides with the given theoretical results concerning the analysis of computational complexities of different algorithms.

The idea of observing this type of matrices comes from the fact that they appear in image restoration theory. Also, the computation of the Moore–Penrose inverse is required in the image deblurring process in the case when the blurring kernel is known in advance.

Acknowledgment

The authors gratefully acknowledge the support within the research project 174013 of the Serbian Ministry of Science.

References

Banham, M.R. and Katsaggelos, A.K. (1997). Digital image restoration, *IEEE Signal Processing Magazine* **14**(2): 24–41.

Ben-Israel, A. and Grevile, T.N.E. (2003). *Generalized Inverses, Theory and Applications, Second Edition*, Canadian Mathematical Society/Springer, New York, NY.

Bhimasankaram, P. (1971). On generalized inverses of partitioned matrices, *Sankhya: The Indian Journal of Statistics, Series A* **33**(3): 311–314.

Bovik, A. (2005). *Handbook of Image and Video Processing*, Elsevier Academic Press, Burlington.

Bovik, A. (2009). *The Essential Guide to the Image Processing*, Elsevier Academic Press, Burlington.

Chantas, G.K., Galatsanos, N.P. and Woods, N.A. (2007). Super-resolution based on fast registration and maximum a posteriori reconstruction, *IEEE Transactions on Image Processing* **16**(7): 1821–1830.

Chountasis, S., Katsikis, V.N. and Pappas, D. (2009a). Applications of the Moore–Penrose inverse in digital image restoration, *Mathematical Problems in Engineering* **2009**, Article ID: 170724, DOI: 10.1155/2010/750352.

Chountasis, S., Katsikis, V.N. and Pappas, D. (2009b). Image restoration via fast computing of the Moore–Penrose inverse matrix, *16th International Conference on Systems, Signals and Image Processing, IWSSIP 2009,Chalkida, Greece,* Article number: 5367731.

Chountasis, S., Katsikis, V.N. and Pappas, D. (2010). Digital image reconstruction in the spectral domain utilizing the Moore–Penrose inverse, *Mathematical Problems in Engineering* **2010**, Article ID: 750352, DOI: 10.1155/2010/750352.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C. (2001). *Introduction to Algorithms, Second Edition*, MIT Press, Cambridge, MA.

Courrieu, P. (2005). Fast computation of Moore–Penrose inverse matrices, *Neural Information Processing—Letters and Reviews* **8**(2): 25–29.

Craddock, R.C., James, G.A., Holtzheimer, P.E. III, Hu, X.P. and Mayberg, H.S. (2012). A whole brain FMRI atlas generated via spatially constrained spectral clustering, *Human Brain Mapping* **33**(8): 1914–1928.

Dice, L.R. (1945). Measures of the amount of ecologic association between species, *Ecology* **26**(3): 297–302.

Górecki, T. and Łuczak, M. (2013). Linear discriminant analysis with a generalization of the Moore–Penrose pseudoinverse, *International Journal of Applied Mathematics and Computer Science* **23**(2): 463–471, DOI: 10.2478/amcs-2013-0035.

Graybill, F. (1983). *Matrices with Applications to Statistics, Second Edition*, Wadsworth, Belmont, CA.

Greville, T.N.E. (1960). Some applications of the pseudo-inverse of matrix, *SIAM Review* **3**(1): 15–22.

Hansen, P.C., Nagy, J.G. and O'Leary, D.P. (2006). *Deblurring Images: Matrices, Spectra, and Filtering*, SIAM, Philadelphia, PA.

Hillebrand, M. and Muller, C.H. (2007). Outlier robust corner-preserving methods for reconstructing noisy images, *The Annals of Statistics* **35**(1): 132–165.

Hufnagel, R.E. and Stanley, N.R. (1964). Modulation transfer function associated with image transmission through turbulence media, *Journal of the Optical Society of America* **54**(1): 52–60.

Kalaba, R.E. and Udwadia, F.E. (1993). Associative memory approach to the identification of structural and mechanical systems, *Journal of Optimization Theory and Applications* **76**(2): 207–223.

Kalaba, R.E. and Udwadia, F.E. (1996). *Analytical Dynamics: A New Approach*, Cambridge University Press, Cambridge.

Karanasios, S. and Pappas, D. (2006). Generalized inverses and special type operator algebras, *Facta Universitatis, Mathematics and Informatics Series* **21**(1): 41–48.

Katsikis, V.N., Pappas, D. and Petralias, A. (2011). An improved method for the computation of the Moore–Penrose inverse matrix, *Applied Mathematics and Computation* **217**(23): 9828–9834.

Katsikis, V. and Pappas, D. (2008). Fast computing of the Moore–Penrose inverse matrix, *Electronic Journal of Linear Algebra* **17**(1): 637–650.

MathWorks (2009). *Image Processing Toolbox User's Guide*, The Math Works, Inc., Natick, MA.

MathWorks (2010). *MATLAB 7 Mathematics*, The Math Works, Inc., Natick, MA.

Noda, M.T., Makino, I. and Saito, T. (1997). Algebraic methods for computing a generalized inverse, *ACM SIGSAM Bulletin* **31**(3): 51–52.

Penrose, R. (1956). On a best approximate solution to linear matrix equations, *Proceedings of the Cambridge Philosophical Society* **52**(1): 17–19.

Prasath, V.B.S. (2011). A well-posed multiscale regularization scheme for digital image denoising, *International Journal of Applied Mathematics and Computer Science* **21**(4): 769–777, DOI: 10.2478/v10006-011-0061-7.

Rao, C. (1962). A note on a generalized inverse of a matrix with applications to problems in mathematical statistics, *Journal of the Royal Statistical Society, Series B* **24**(1): 152–158.

Röbenack, K. and Reinschke, K. (2011). On generalized inverses of singular matrix pencils, *International Journal of Applied Mathematics and Computer Science* **21**(1): 161–172, DOI: 10.2478/v10006-011-0012-3.

Schafer, R.W., Mersereau, R.M. and Richards, M.A. (1981). Constrained iterative restoration algorithms, *Proceedings of the IEEE* **69**(4): 432–450.

Shinozaki, N., Sibuya, M. and Tanabe, K. (1972). Numerical algorithms for the Moore–Penrose inverse of a matrix: Direct methods, *Annals of the Institute of Statistical Mathematics* **24**(1): 193–203.

Smoktunowicz, A. and Wróbel, I. (2012). Numerical aspects of computing the Moore–Penrose inverse of full column rank matrices, *BIT Numerical Mathematics* **52**(2): 503–524.

Stojanović, I., Stanimirović, P. and Miladinović, M. (2012). Applying the algorithm of Lagrange multipliers in digital image restoration, *Facta Universitatis, Mathematics and Informatics Series* **27**(1): 41–50.

Udwadia, F.E. and Kalaba, R.E. (1997). An alternative proof for Greville's formula, *Journal of Optimization Theory and Applications* **94**(1): 23–28.

Udwadia, F.E. and Kalaba, R.E. (1999). General forms for the recursive determination of generalized inverses: Unified approach, *Journal of Optimization Theory and Applications* **101**(3): 509–521.

Predrag Stanimirović received his M.Sc. in 1990 and his Ph.D. in 1996 from the University of Niš. He has held a full professorial position at the same university since 2002. His research interests include linear and nonlinear programming, generalized inverses and symbolic computations.

Marko Miladinović received his B.Sc. in 2005 and his Ph.D in 2011 from the University of Niš, Faculty of Sciences and Mathematics. He works as an assistant professor at the same faculty. His research interests are in nonlinear optimization, generalized inverses, symbolic computations and image restoration.

Igor Stojanović earned an M.Sc. degree in 2002 and a Ph.D degree in 2011, both from Ss. Cyril and Methodius University in Skopje. Currently he is an assistant professor at the Faculty of Computer Science in 'Goce Delcev' University, Stip, Macedonia. His research interests are multimedia, image retrieval, image recognition, digital video and image processing.

Sladjana Miljković received her B.Sc. in 2007 at Ss. Cyril and Methodius University in Skopje and her Ph.D in 2012 from the University of Niš. She is an assistant-actuary in the Insurance Supervision Agency of Macedonia. Her research interests are in nonlinear optimization, generalized inverses and actuarial mathematics.

Appendix

In this section we present the MATLAB code for the implementation of Algorithm 2 for computing the Moore–Penrose inverse $H^{\dagger}$.The function GaussBlur generates the Toeplitz matrix H which defines the Gaussian blur. Its formal parameters are n, l and s.

```
function [PSF] = GaussBlur(n, l, s)
    m = n-l+1;
    x = -fix(l/2):ceil(l/2)-1;
    vector = exp( -(x.^2)/(2*s^2));
    vector = vector / sum(vector(:));
    r = [vector zeros(1,m-1)];
    c = zeros(m,1);
    c(1) = vector(1);
    PSF = toeplitz(c,r);
```

Implementation of Step 2 from Algorithms 1 and 2 is given in the following code:

```
function res = InvKvGauss(A)
    m = length(A);
    p = flipud(eye(m,1));
    opts.UT = true;
    vec = linsolve(A,p,opts);
    r = flipud(vec);
    c = zeros(m,1);
    c(1) = r(1);
    res = toeplitz(c,r);
```

Finally, computation of the Moore–Penrose inverse of H is made by the following function:

```
function MP = InverseOfGauss(n,l,s)
%Computes $H^\dag$ using block partitioning method
    H = GaussBlur(n,l,s);
    m = n-l+1;
    A = H(:,1:m);
    C = H(:,m+1:n);
    Inv = InvKvGauss(A);
    Z = Inv*C;
    V = (eye(n-m)+Z'*Z)\Z'*Inv;
    MP = [Inv*(eye(m)-C*V);V];
```

Received: 2 September 2012
Revised: 9 March 2013
Re-revised: 10 July 2013