International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol. 7, No. 6 (2014), pp. 17-28
http://dx.doi.org/10.14257/ijsip.2014.7.6.02

Matrix Forms of Gradient Descent Algorithms Applied to Restoration of Blurred Images

Fernando Pazos[†] and Amit Bhaya[‡]

[†] *Department of Electronics and Telecommunication Engineering, State University of Rio de Janeiro, Rua São Fransciso Xavier n° 524, Rio de Janeiro 20550-900, Brazil*

[‡] *Department of Electrical Engineering, FederalUniversity of Rio de Janeiro, PEE/COPPE/UFRJ, POBox 68504, Rio de Janeiro 21945-970, Brazil*

[†] *quini@ort.org.br*, [‡] *amit@nacad.ufrj.br*

Abstract

Iterative gradient descent algorithms to solve algebraic linear equations are well known in the literature. These algorithms can be interpreted as dynamical closed loop control systems, where the step sizes are the control variables that can be optimally calculated by the Liapunov Control Functions (CLF) and Liapunov Optimizing Control (LOC) methods. In this paper matrix versions of gradient descent algorithms are deduced, including an unpublished matrix form of the Barzilai-Borwein algorithm. The step sizes (control variables) are also calculated by CLF/LOC. The main utility of these matrix algorithms is in image deblurring in the cases where the matrix that represents the blurring process is too large to be stored in the memory of the majority of conventional computational systems.

Keywords*: Control Liapunov Functions; Liapunov Optimizing Control; iterative algorithms; image deblurring*

1. Introduction

There exist many applications in which images are used to analyze experiments, record events, among other utilities. Images recorded by a camera are rarely perfect due to environmental effects and/or technological problems, and in some cases maybe substantially distorted by blurring and noise. Image restoration refers to the process of recovering a clearer image from a degraded one.

A digital image is composed of picture elements called pixels. Each pixel represents an intensity on a gray scale or a color scale of a small rectangular segment of the scene. Thus, rectangular images can be mathematically represented by a matrix of dimensions $m \times n$, where m is the number of pixels in the columns and n is the number of pixels in the rows of the image (about different formats to represent color images, see [11, chap. 1]).

Blur in digital images refers to the fact in that the recorded intensity of a pixel is related to the intensity in a larger neighborhood of the corresponding section of the scene. Blurring can arise from many sources, such as limitations of the optical system, camera and object motion, astigmatism and environment effects [18]. The blurring process can be mathematically represented as follows. We denote the real image as $X^* \in \mathrm{R}^{m \times n}$, and $B \in \mathrm{R}^{m \times n}$ is the recorded blurred image. The function which describes how every pixel of the matrix X^* is blurred into the matrix B is called the Point Spread Function (PSF). The Point Spread Function can be

ISSN: 2005-4254 IJSIP

represented as a matrix *P* called PSF array. Knowledge of the physical process that causes the blur provides an explicit formulation of the PSF. For example, the PSF array for blurring caused by atmospheric turbulence can be represented as a two-dimensional Gaussian function, whose elements at the coordinates *(i,j)* are given by:

$$[p_{ij}] = exp\left(-\frac{1}{2}\begin{bmatrix} i-k \\ j-\ell \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & \rho^2 \\ \rho^2 & \sigma_2^2 \end{bmatrix}^{-1} \begin{bmatrix} i-k \\ j-\ell \end{bmatrix}\right) \quad (1)$$

where *p*, σ_1 and σ_2 are parameters that determine the dimension and orientation of the PSF centered at the coordinates *(k,ℓ)*. The PSF array of an astronomical telescope is often modeled by the so-called Moffat function, and the elements of this PSF array at the coordinates *(i,j)* are given by:

$$[p_{ij}] = \left(1 + \begin{bmatrix} i-k \\ j-\ell \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & \rho^2 \\ \rho^2 & \sigma_2^2 \end{bmatrix}^{-1} \begin{bmatrix} i-k \\ j-\ell \end{bmatrix}\right)^{-\beta} \quad (2)$$

where β is a positive scalar parameter (for details of these PSF arrays, see [11, p. 25]).

The blurring process can be mathematically represented by a two-dimensional discrete convolution operation between the real image and the PSF array; this convolution is performed with suitable boundary conditions. Roughly speaking, boundary conditions are determined by the assumptions we make on the behavior of the scene outside the boundary of a given image. Thus, the blurred image $B = P * X^*$. The discrete convolution is essentially a linear operation, so the blurring process can be also represented as a system of equations

$$A\mathbf{x}^* = \mathbf{b} \quad (3)$$

where $\mathbf{x} = \mathbf{vec}(X^*)$ is the column-wise stacked image X^*, i.e. the vector $\mathbf{vec}(X^*)=[\mathbf{x}_1{}^{*T}\mathbf{x}_2{}^{*T} \cdots \mathbf{x}_n{}^{*T}]$ and $\mathbf{x}_i^*$, $i \in \{1,...,n\}$, is the i^{th} column vector of the matrix X^*, $\mathbf{b} = \mathbf{vec}(B)$ is the column-wise stacked image *B*, and the matrix $A \in \mathrm{R}^{nm \times nm}$ is a blurring matrix that depends on the PSF array *P* and on the boundary conditions. The matrix *A* is, in general, unsymmetric and presents a specific structure involving Toeplitz, circular, and Hankel matrices [11, chap. 4] [16-18].

The goal of a deblurring process is to recover an image $X \in \mathrm{R}^{m \times n}$ as faithful as possible to the real image X^*, from the recorded blurred image *B*. The blurring matrix *A* usually has large dimensions. For example, for an image of 512 by 512 pixels, $A \in \mathrm{R}^{262144 \times 262144}$. These dimensions make the solution of (3) via Gaussian elimination or explicit inversion of the matrix *A* very expensive in computational terms. The alternative is to use iterative algorithms that start at an arbitrary initial guess $\mathbf{x}_0$ and calculate, at the k^{th} iteration, a point $\mathbf{x}_k$, until for large enough *k* a point arbitrarily close to the solution point $\mathbf{x}^*$ is reached. The iterative algorithms *steepest descent* (SD), *orthomin* (OM), *Barzilai-Borwein* (BB) and *conjugate gradient* (CG) are well known in the literature and they can be interpreted as dynamical closed loop control systems [3]. However, as the dimensions of *A* are usually too large to store it in the memory of the majority of the conventional computational systems, the algorithms just mentioned cannot be applied to solve (3).

We present here matrix versions of these iterative algorithms that do not require the explicit formulation of the matrix *A*, thus avoiding its storage in memory. Thus, these matrix algorithms are suitable to deblur images using conventional computers or embedded systems. A matrix version of the CG algorithm was first presented, without deduction, in [13]. We

deduce the equations of the CG algorithm as well as extend this deduction to matrix forms of the other iterative algorithms mentioned. In addition, computational experiments show that the matrix version of the BB algorithm may converge faster and execute less floating point operations than the matrix version of the CG algorithm.

2. Gradient Descent Algorithms Interpreted as Dynamical Closed Loop Control Systems

Many problems of science and engineering can be expressed as linear algebraic equations (LAE).This system of equations can be expressed as $A\mathbf{x} = \mathbf{b}$, where $A \in R^{n\times n}$ is symmetric positive definite, $\mathbf{x} \in R^n$ is the vector of unknowns, $\mathbf{b} \in R^n$ is the vector of known data. As A usually has a large dimension, one approach to solve LAE is to minimize instead the scalar convex quadratic function [15]

$$\varphi(\mathbf{x}) = \tfrac{1}{2}\,\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{4}$$

since its unique optimal point is given by $\mathbf{x} = A^{-1}\mathbf{b}$. Several algorithms are based on the standard idea of generating a sequence of points, starting from an arbitrary initial guess $\mathbf{x}_0 \in R^n$, and proceeding in the descent direction (negative gradient of $\varphi(\mathbf{x})$), with an adequate choice of the step size. In mathematical terms:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{r}k\,, \quad k = 0, 1, 2 \cdots \tag{5}$$

where $\mathbf{r}_k := \nabla\varphi(\mathbf{x}_k) = A\mathbf{x}_k - \mathbf{b}$ is called the residue and $\alpha_k \in R$ is the step size.

Algorithm (5) can be interpreted as a dynamical closed loop control system (see details in [4] and [3]), where the step size is the control variable.

The update vector of the state variable $\mathbf{x}_k$ can be chosen by the Control Liapunov Function (CLF) method and optimized by the Liapunov Optimizing Control (LOC) method. Roughly speaking, the CLF method is as follows: given a positive definite scalar Liapunov function candidate $V(\mathbf{x})$: $R^n \rightarrow R_+$, choose *any* update vector of the state variable $\mathbf{x}$ that makes the time derivative of the Liapunov function negative definite, thus ensuring the asymptotic convergence of the trajectories to the solution point $\mathbf{x}^*$ by the Liapunov direct method. The LOC method is to choose the update vector that minimize $V(\mathbf{x})$, thus ensuring the fastest convergence of the trajectories to the solution point (see a complete description of the methods in [3]). In the context of gradient descent iterative algorithms (5), the CLF/LOC method is essentially this: given a scalar discrete Liapunov function candidate $V(\mathbf{x}_k)$, choose the step size that minimizes the decrement of the Liapunov function between two consecutive iterations.

If (4) is chosen as Liapunov function candidate, the choice of step size by the CLF/LOC method yields the well-known *steepest descent* (SD) algorithm, presented in [6]. The related algorithm *orthomin* (OM) [1] minimizes the decrement of $\|\mathbf{r}_k\|^2$ between two consecutive iterations. Both algorithms are characterized by slow convergence to the solution $\mathbf{x}^*$ when the condition number of A is large [7]. Barzilai and Borwein [2] presented a gradient descent algorithm that uses the step size of the SD delayed by one iteration. This simple modification speeds up the convergence of the algorithm surprisingly, and its remarkable improvement over the SD and OM algorithms has been much studied. However, a complete explanation of why this simple modification of the SD algorithm improves its performance considerably has

not yet been found. Table 1 shows the step sizes used by these algorithms. Other step sizes for (5), also designed using control-theoretic ideas, are presented in [5].

Table 1. Step Sizes of the Methods: Steepest Descent (SD), Orthomin (OM), Barzilai–Borwein (BB)

Method	step size α_k	convergence
SD	$\frac{\mathbf{r}_k^\dagger \mathbf{r}_k}{\mathbf{r}_k^\dagger A \mathbf{r}_k}$	monotonic in $\phi(\mathbf{x})$
OM	$\frac{\mathbf{r}_k^\dagger A \mathbf{r}_k}{\mathbf{r}_k^\dagger A^2 \mathbf{r}_k}$	monotonic in $\|\mathbf{r}\|_2$ and $\phi(\mathbf{x})$
BB	$\frac{\mathbf{r}_{k-1}^\dagger \mathbf{r}_{k-1}}{\mathbf{r}_{k-1}^\dagger A \mathbf{r}_{k-1}}$	non monotonic

The conjugate gradient algorithm (CG) developed by Hestenes and Stiefel [12] is the most popular conjugate direction method. At each iteration the algorithm minimizes the scalar function (4) along conjugate directions also searched at each iteration; as conjugate directions are linearly independent, the convergence of the sequence of points $\mathbf{x}_k$ to the solution point $\mathbf{x}^*$ is produced after at most n iterations in exact arithmetic (see [8] for a proof). In the CG algorithm, the residual vector $\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b}$ and a direction vector $\mathbf{d}_k$ must be calculated at each iteration. The CG algorithm can be described as follows. Starting at any $\mathbf{x}_0 \in R^n$, and choosing $\mathbf{d}_0 = \mathbf{r}_0 = A\mathbf{x}_0 - \mathbf{b}$, at the k^{th} iteration calculate

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \alpha_k \mathbf{d}_k \\ \mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k A\mathbf{d}_k \\ \mathbf{d}_{k+1} &= \mathbf{r}_{k+1} - \beta_k \mathbf{d}_k \end{aligned} \qquad (6)$$

The step size α_k is calculated in order to minimize the decrement of the scalar convex function (4) at each iteration; the step size β_k is calculated to guarantee that $\mathbf{d}_{k+1}$ is conjugate with respect to the directions calculated at the former iterations, i.e. $\mathbf{d}_{k+1}{}^T A\mathbf{d}_j = 0$ for all $j \leq k$, according to the Gram-Schmidt orthogonalization method. These step sizes can be also deduced by the CLF/LOC method: choosing $V_k = \varphi(\mathbf{x}_k)$ and $W_k = \mathbf{d}_k{}^T A\mathbf{d}_k$ respectively as discrete Liapunov functions candidates, the step sizes that minimize the decrement of these functions are given by:

$$\alpha_k = \frac{r_k^T d_k}{d_k^T A d_k}, \qquad \beta_k = \frac{r_{k+1}^T A d_k}{d_k^T A d_k} \qquad (7)$$

To deblur an image B (which can be also represented by its column-wise stacked vector $\mathbf{b}$) it is necessary to solve the linear algebraic equation (3). As the blurring matrix A is often unsymmetric, iterative gradient descent algorithms are applied to solve instead the normal equation

$$A^T A\mathbf{x} = A^T\mathbf{b}$$

the solution of which is the least squares solution of (3), i.e.: $\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$.

3. Separable PSF and Matrix Forms

When the blurring process in the columns of the image is independent of the blurring process in the rows, the PSF array has rank 1 and can be represented as

$$P = \mathbf{c}\mathbf{r}^{T} \tag{8}$$

where $\mathbf{c} \in R^{m}$ represents the vertical components of the blur and $\mathbf{r} \in R^{n}$ represents the horizontal components [11]. This is the case of the PSF array with Gaussian distribution with equal variance in both axes. In these cases the matrix A can be represented as [11, chap. 4]

$$A = A_r \otimes A_c \tag{9}$$

where $A_r \in R^{nxn}$ and $A_c \in R^{mxm}$ are matrices that depend on the vectors $\mathbf{r}$ and $\mathbf{c}$, respectively, and on the boundary conditions. This Kronecker decomposition has the advantage that the original image X^* can be restored without the necessity of explicit formulation of the matrix A, as can be demonstrated from the following properties of the Kronecker product:

$$\begin{aligned} &A_r \otimes A_c \,\mathbf{vec}(X) = \mathbf{vec}(A_c X A_r^{T}) \\ &(A_r \otimes A_c)^{T} = A_r^{T} \otimes A_c^{T} \\ &(A_r \otimes A_c)^{-1} = A_r^{-1} \otimes A_c^{-1} \end{aligned} \tag{10}$$

Substituting (9) in (3) and applying the properties (10):

$$A_c X^* A_r^{T} = B \tag{11}$$

so that

$$X^* = A_c^{-1} B A_r^{-T} \tag{12}$$

in exact arithmetic, where A_r^{-T} is the transpose of the inverse matrix.

Therefore, given the blurred image B, the restoration process to recover an image X as similar as possible to the original image X^* can be carried out without the explicit formulation of the matrix A.

Evidently, although the matrices A_r and A_c have dimensions similar to that of the original image X^*, their explicit inversions can also be expensive in computational terms. In the following section matrix forms of iterative algorithms will be presented in order to obtain a restored image X without the formulation of the matrix A.

4. Matrix Forms of Iterative Algorithms

Consider the linear system (3), for which the normal system is defined as

$$A^{T} A \,\mathbf{vec}(X^*) = A^{T} \mathbf{vec}(B) \tag{13}$$

We define the inner product between two matrices $X, Y \in R^{m \times n}$ as:

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij} = \boldsymbol{trace}(\boldsymbol{X}^{T}\boldsymbol{Y}) = \boldsymbol{vec}(\boldsymbol{X})^{T} \mathbf{vec}(\boldsymbol{Y}) \tag{14}$$

Note that this definition observes all the conditions required by an inner product. Note also that $(X,X) = \|X\|_F^2$.

The matrix solution of (13) is also the unique minimum of the scalar quadratic convex function defined as:

$$\begin{aligned}\varphi(X) &:= \tfrac{1}{2}\,\mathbf{x}^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{x} - (A^{\mathrm{T}}\mathbf{b})^{\mathrm{T}}\mathbf{x}\\ &= \tfrac{1}{2}\,\mathbf{vec}(X)^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{vec}(X) - (A^{\mathrm{T}}\mathbf{vec}(B))^{\mathrm{T}}\mathbf{vec}(X)\\ &= \tfrac{1}{2}\,\mathbf{vec}(A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}})\,\mathbf{vec}(A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}) - \mathbf{vec}(X)^{\mathrm{T}}\mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}})\\ &= \tfrac{1}{2}\,\mathrm{trace}(A_{\mathrm{r}}X^{\mathrm{T}}A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}) - \mathrm{trace}(X^{\mathrm{T}}A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}})\end{aligned} \tag{15}$$

where the properties presented in (10) were applied. The gradient vector of (15) is defined as the residue:

$$\begin{aligned}\mathbf{r} = \nabla\varphi(X) = \mathbf{vec}(R) &:= A^{\mathrm{T}}A\,\mathbf{x} - A^{\mathrm{T}}\mathbf{b}\\ &= A^{\mathrm{T}}\,\mathbf{vec}(A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}) - \mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}})\\ &= \mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}A_{\mathrm{r}} - A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}})\end{aligned} \tag{16}$$

The error vector is defined as:

$$\mathbf{e} = \mathbf{vec}(\mathrm{E}) := \mathbf{x} - \mathbf{x}^* = \mathbf{vec}(X - X^*) \tag{17}$$

The relation between the residue and the error is given by:

$$\begin{aligned}A^{\mathrm{T}}A\mathbf{e} = A^{\mathrm{T}}A\,\mathbf{vec}(E) &= \mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}E\,A_{\mathrm{r}}^{\mathrm{T}}A_{\mathrm{r}}) = \mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}A_{\mathrm{r}} - A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}X^*A_{\mathrm{r}}^{\mathrm{T}}A_{\mathrm{r}})\\ &= \mathbf{vec}(A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}XA_{\mathrm{r}}^{\mathrm{T}}A_{\mathrm{r}} - A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}}) = \mathbf{vec}(R) = \mathbf{r}\end{aligned}$$

4.1. Steepest Descent, Orthomin and Barzilai-Borwein

The gradient descent algorithms in their matrix forms are defined as follows. From an initial guess $X_0 \in \mathrm{R}^{m\times n}$, with initial residue $R_0 = A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}X_0A_{\mathrm{r}}^{\mathrm{T}}A\mathrm{r} - A_{\mathrm{c}}^{\mathrm{T}}BA_{\mathrm{r}}$, at each iteration calculate:

$$X_{k+1} = X_k - \alpha_k R_k \qquad k = 0, 1, 2 \cdots \tag{18}$$

where α_k is the scalar step size which can be also calculated by the CLF/LOC method in order to minimize the decrement of a discrete Liapunov function candidate at each iteration. The stopping criterion can be adopted as the Frobenius norm of the residual matrix $\|R_k\|_F$ lower than a determined tolerance.

In the matrix form of the steepest descent algorithm the Liapunov function candidate is chosen as the scalar function (15). Substituting (18) in (15), the function φ(X) at the point X_{k+1} is given by:

$$\begin{aligned}\varphi(X_{k+1}) &= \varphi(X_k) - \alpha_k\,\mathbf{vec}(X_k)^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{vec}(R_k) + \alpha_k\,\mathbf{vec}(B)^{\mathrm{T}}A\mathbf{vec}(R_k)\\ &\quad + \tfrac{1}{2}\,\alpha_k^2\,\mathbf{vec}(R_k)^{\mathrm{T}}A^{\mathrm{T}}A\mathbf{vec}(R_k)\\ &= \varphi(X_k) - \alpha_k\,\mathrm{trace}(R_k^{\mathrm{T}}R_k) + \tfrac{1}{2}\,\alpha_k^2\,\mathrm{trace}(A_{\mathrm{r}}R_k^{\mathrm{T}}A_{\mathrm{c}}^{\mathrm{T}}A_{\mathrm{c}}R_kA_{\mathrm{r}}^{\mathrm{T}})\end{aligned}$$

and hence the step size that minimizes the decrement of the Liapunov function between two consecutive iterations, calculated the derivative of $\varphi(X_{k+1})-\varphi(X_k)$with respect to α_k and equating to zero, is given by:

$$\alpha_k = \frac{trace(R_k^T R_k)}{trace(A_r R_k^T A_c^T A_c R_k A_r^T)} \quad (19)$$

This step size is equal to $\alpha_k = \frac{r_k^T r_k}{r_k^T A^T A \mathbf{r_k}}$, which is the step size of the SD algorithm applied to the system (13). However, the use of (18)-(19) allows the calculation of a restored image X arbitrarily close to the original image X^* without explicitly forming the matrix A.

Similarly, the step size of the *orthomin* algorithm is calculated in order to minimize the decrement of the discrete Liapunov function candidate $\|R_k\|_F^2 = \text{trace}(R_k^T R_k) = \|\mathbf{r}_k\|^2$. The application of the CLF/LOC method to the calculation of this step size yields:

$$\alpha_k = \frac{trace(A_r R_k^T A_c^T A_c R_k A_r^T)}{trace(A_r^T A_r R_k^T A_c^T A_c A_c^T A_c R_k A_r^T A_r)} \quad (20)$$

This step size is equal to $\alpha_k = \frac{r_k^T A^T A r_k}{r_k^T (A^T A)^2 \mathbf{r_k}}$, which is the step size of the OM algorithm applied to the system (13), however, here again, the use of (18)-(20) permits the restoration of an image without explicitly forming the blurring matrix A.

The matrix form of the Barzilai-Borwein algorithm consists in the algorithm (18) with the step size (19) delayed by one iteration:

$$\alpha_k = \frac{trace(R_{k-1}^T R_{k-1})}{trace(A_r R_{k-1}^T A_c^T A_c R_{k-1} A_r^T)} \quad (21)$$

Note that the use of classical gradient descent algorithms to solve the normal system (13) requires the storage of a matrix of n^2m^2 components, whereas the use of the matrix form (18) with the residual matrix defined in (16) requires the storage of two matrices of m^2 and n^2 components respectively, requiring a total storage of n^2+m^2 components.

4.2. Conjugate Gradient

The matrix form of the conjugate gradient algorithm can be expressed as follows. Starting at an initial guess $X_0 \in \mathrm{R}^{m \times n}$, with initial residue and direction $R_0 = D_0 := A_c^T A_c X_0 A_r^T A_r - A_c^T B A_r$, at each iteration calculate:

$$\begin{aligned} X_{k+1} &= X_k - \alpha_k D_k \\ R_{k+1} &= A_c^T A_c X_{k+1} A_r^T A_r - A_c^T B A_r \\ &= R_k - \alpha_k A_c^T A_c D_k A_r^T A_r \\ D_{k+1} &= R_{k+1} - \beta_k D_k \end{aligned} \quad (22)$$

where α_k and β_k are the step sizes, and D_i, i = {0,...,k+1} are conjugate matrices with respect to A^TA, i.e.:

$$\mathbf{vec}(D_i)^T A^T A \, \mathbf{vec}(D_j) = 0 \quad \forall \, i \neq j$$

which implies that $\text{trace}(A_r D_i^T A_c^T A_c D_j A_r^T) = 0$ forall $i \neq j$.

Note that the application of the **vec** operator in the equations (22) yields in the classical vectorial form of the conjugate gradient algorithm (6).

The step size α_k is calculated in order to minimize the decrement of the function $\varphi(X_k)$ at every iteration in the direction vector given by $\mathbf{d}_k = \mathbf{vec}(D_k)$. The application of the CLF/LOC method to calculate this step size yields:

$$\alpha_k = \frac{trace(D_k^T R_k)}{trace(A_r D_k^T A_c^T A_c D_k A_r^T)} \quad (23)$$

The step size β_k is calculated in order to observe the $A^T A$-orthogonality of the vectors $\mathbf{vec}(D_k)$, according to the Gram-Schmidt orthogonalization method, i.e.:

$$\beta_k = \frac{trace(A_r R_{k+1}^T A_c^T A_c D_k A_r^T)}{trace(A_r D_k^T A_c^T A_c D_k A_r^T)} \quad (24)$$

These step sizes are equal to those defined in (7) applied to the normal equation (13). However, differently from what happens with the classical SD algorithm, the application of (22)-(23)-(24) permits the recovery of a matrix X without explicitly forming the matrix A. The algorithm (22) with its step sizes (23) and (24) were presented, without proofs, in [13, p. 304].

In the case that the blurred image B is also corrupted by noise and the blurring matrix A is ill-posed (as it usually is), some kind of regularization method must be applied in order to filter the noise effect. The Tikhonov regularization method [20] is perhaps the most frequently used. A matrix form of the Tikhonov method can be defined as:

$$A_c^T A_c X A_r^T A_r + \lambda X = A_c^T B A_r \quad (25)$$

where $\lambda \in R$ is the Tikhonov regularization parameter chosen in order to minimize the regularization error plus the error introduced by the noise. Iterative methods to solve regularized linear algebraic equations that aim to restore noise-contaminated blurred images were presented in [9-10, 14, 16]. The calculation of the regularization parameter λ without the use of either the matrix A or its singular value decomposition will be investigated in future work.

5. Computational Experiments

The goal of the experiments presented in this section is to restore a noise-free blurred image. The original image X^* (Figure 1a) is 256 pixels by 256 pixels and has 256 gray levels.

The original image was blurred by a PSF array $P \in R^{31x31}$ with Gaussian distribution with mean 0 and variance 8 in both axes and reflexive boundary conditions (Figure 1b).

The initial guess of the iterative algorithms was chosen as $X_0 \in R^{256x256}$ such that $x_{0ij} = 128$, $\forall i,j \in \{1,..., 256\}$. The stopping criterion was chosen as $||R_k||_F < 10^{-3}$.

The *Barzilai-Borwein* algorithm required 1947 iterations to reach the stopping condition, and the final error norm achieved was $||E_k||_F := ||X_k|| - ||X^*||_F = 3.4783e+03$. Figure 1c shows the image recovered by this algorithm.

The *conjugate gradient* algorithm required 3700 iterations to reach the stopping condition, and the final error attained was $||X_k|| - ||X^*||_F = 2.6609e+03$. Figure 1d shows the image recovered by this algorithm.

The results of the *steepest descent* and the *orthomin* algorithms are not presented because they did not reach the stopping condition in 21000 iterations.

The experiment was repeated with other dimensions of the PSF array of Gaussian distribution with mean zero, and with different values of the variance, maintaining the boundary conditions, the initial guess and the stopping criterion. Table 2 presents the results of these experiments.

The same experiments were also carried out with the image presented in Figure 2a. This image is 256 pixels by 256 pixels and has 256 gray levels. In the first series of experiments, the original image was blurred with a PSF array $P \in R^{31x31}$ with Gaussian distribution with mean 0 and variance 8 in both of the axes and reflexive boundary conditions (Figure 2b). The initial matrix was also chosen as $X_0 = 128I$ and the stopping criterion as $\|R_k\|_F < 10^{-3}$.

The *Barzilai-Borwein* algorithm required 2408 iterations to reach the stopping condition, and the final error norm achieved was $\|E_k\|_F$:=C $\|X_k\| - \|X^*\|_F$ =4.0575e +03. Figure 2c shows the image recovered by this algorithm.

The *conjugate gradient* algorithm used 4089 iterations to reach the stopping condition, and the final error was $\|E_k\|_F := \|X_k\| - \|X^*\|_F$ =3.0278e+03. Figure 2d shows the image recovered by this algorithm.

Experiments with the original image shown in Figure 2a using other dimensions of the PSF array of Gaussian distribution, mean zero, and with different values of the variance were also performed. Table 2 presents the results of these experiments.

The Moffat point spread function (2) was not used in the tests because it is not separable, hence the PSF array cannot be written in the form (8).

Table 2 shows that in some experiments the CG algorithm reached the stopping condition in less iterations than the BB algorithm (as usually happens with their classical vectorial forms). However, the time employed to perform each iteration is proportional to the number of floating point operations executed. The BB algorithm (18)-(21) executes a number of scalar products and quotients per iteration equal to

$$N_{BB}(n,m) = nm(2m + 2n + 3) + 1$$

which means that in the experiments reported here (where $n = m = 256$) the BB algorithm executes N_{BB} = 67305473 scalar products and quotients per iteration. The CG algorithm (22)-(23)-(24) executes a number of scalar products and quotients per iteration equal to

$$N_{CG}(n,m) = nm(2m + 3n + 5) + 2$$

which means that in the experiments reported here the CG algorithm executes N_{CG} = 84213762 scalar products and quotients per iteration. The number of floating point operations executed in these experiments is also reported in Table 2. Note that in these experiments, only with a PSF array of 61 × 61 and variance 16, the CG algorithm reached the stopping condition faster than the BB algorithm. Independent of the speed of convergence, in all the experiments carried out here, the final error presented by the CG algorithm was lower than the final error presented by the BB algorithm, and the positive effect of this can be appreciated visually by comparing Figures 1(c) with 1(d) and 2(c) with 2(d), respectively.

**Figure 1. a. Original Image of 256 × 256 Pixels
b. Blurred Image with Gaussian Distribution, Mean 0, Variance 8 and Reflexive Boundary Conditions
c. Image Restored by the Barzilai-Borwein Algorithm after 1947 Iterations
d. Image Restored by the Conjugate Gradient Algorithm after 3700 Iterations**

**Figure 2. a. Original image of 256 × 256 pixels
b. Blurred image with Gaussian distribution, mean 0, variance 8 and reflexive boundary conditions
c. Image restored by the Barzilai-Borwein algorithm after 2408 iterations
d. Image restored by the conjugate gradient algorithm after 4089 iteration**

Table 2. Results of the experiments with the original images shown in Figure 1a and 2a respectively using different sizes of the PSF array of Gaussian distribution, mean zero and different values of the variance (var.). Initial guess $X_0 = 128\ I$. Stopping condition $||R_k||_F < 10^{-3}$. it.: number of iterations. flops: number of floating point operations. error=$||X_k|| - ||X^*||_F$

original image	PSF size	var.	BB			CG		
			it.	flops	error	it.	flops	error
Fig. 1a	31 × 31	8	1947	1.31e11	3478.3	3700	3.12e11	2660.9
	31 × 31	16	3577	2.40e11	2228.2	3401	2.86e11	1336.9
	61 × 61	8	2207	1.49e11	4533.2	2694	2.27e11	4349.0
	61 × 61	16	2707	1.82e11	4813.0	1562	1.32e11	3971.3
Fig. 2a	31 × 31	8	2408	1.62e11	4057.5	4089	3.44e11	3027.8
	31 × 31	16	3133	2.11e11	2474.9	3704	3.12e11	1520.3
	61 × 61	8	1785	1.20e11	5719.6	2722	2.29e11	5533.8
	61 × 61	16	3409	2.29e11	5596.2	1618	1.36e11	4762.0

6. Conclusions

The matrix forms of the gradient descent algorithms are adequate for the restoration of noise-free blurred images for which the blurring matrix A has dimensions too large to be stored in the memory of conventional computer systems and the matrices produced by the Kronecker decomposition A_r and A_c are also too large to solve the linear system (3) via Gaussian elimination. In the experiments reported here, A_r, $A_c \in R^{256x256}$ and $A \in R^{65536 \times 65536}$. The matrix forms of the *steepest descent* and the *orthomin* algorithms took a large number of iterations to reach the stopping condition, as happens with their classical vectorial versions and are thus not attractive for this application. In almost all the experiments carried out here, the matrix form of the conjugate gradient algorithm employed more iterations and executed more floating point operations than the matrix form of the *Barzilai-Borwein* algorithm, differently from what usually happens with the respective vectorial versions. However, even though the norm of the residue used as stopping criterion is the same in both the algorithms, the final error achieved by the CG algorithm is lower than that achieved by the BB algorithm in all the experiments reported here; moreover, it is possible to appreciate this difference visually comparing the images restored by each algorithm, and observing the superior quality of the CG restored image.

When the original image is not only blurred but also contaminated by noise, a filtering method must be used in order to minimize the noise effects. The vectorial version of the classical CG algorithm presents an inherent capability to filter the noise [13]. This capability in the matrix version of the CG algorithm will be the object of future research, together with the use of other tools for noise filtering, such as matrix versions of regularization methods.

References

[1] U. Ascher, K. van den Doel, H. Huang, and B. Svaiter, "On fast integration to steady state and earlier times", Mathematical Modelling and Numerical Analysis, vol. 43, (**2009**), pp. 689–708.
[2] J. Barzilai and J. M. Borwein, "Two point step size gradient methods. IMA J. Numer. Anal., vol. 49, (**1988**), pp. 141–148.

[3] A. Bhaya and E. Kaszkurewicz, "Control perspectives on numerical algorithms and matrix problems", Advances in Control. SIAM, Philadelphia, (**2006**).
[4] A. Bhaya and E. Kaszkurewicz, "A control-theoretic approach to the design of zero finding numerical methods", IEEE Transactions on Automatic Control, vol. 52, no. 6, (**2007**) June, pp. 1014–1026.
[5] A. Bhaya, P.-A. Bliman and F. Pazos, "Control-theoretic design of iterative methods for symmetric linear systems of equations", In Proc. of the 48th IEEE Conference on Decision and Control, Shanghai, China, (**2009**) December.
[6] A. Cauchy, "Méthode générale pour la resolution des systèms d'équations simultanées", Comp. Rend. Sci., vo. 25, (**1847**), pp. 563–538.
[7] A. Greenbaum, "Iterative methods for solving linear systems", SIAM, Philadelphia, (**1997**).
[8] O. Güler, "Fundations of optimization", Graduate texts in mathematics, Springer, (**2010**).
[9] M. Hanke, "Iterative regularization techniques in image reconstruction", In D. Colton, H. W. Engl, A. K. Louis, J. R. McLaughlin, and W. Rundell, editors, Surveys on Solution Methods for Inverse Problems, pp. 35–52, Springer, New York, (**2000**).
[10] M. Hanke and J. G. Nagy, "Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques", Inverse Problems, vol. 12, (**1996**), pp.157–173.
[11] P. Hansen, J. G. Nagy and D. O'Leary, "Deblurring images. Matrices, spectra and filtering", Fundamentals of algorithms. SIAM, Philadelphia, (**2006**).
[12] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems", Journal of Research of the National Bureau of Standards, vol. 49, (**1952**), pp. 409–436.
[13] A Jain, "Fundamentals of digital image processing", Prentice Hall, New Jersey, (**1989**).
[14] T. K. Jensen and P. Hansen, "Iterative regularization with minimum-residual methods", BIT Numerical Mathematics, vol. 47, (**2007**), pp. 103–120.
[15] P. D. Lax, Linear algebra, John Wiley, New York, (**1997**).
[16] J. G. Nagy and K. M. Palmer, "Steepest descent, CG, and iterative regularization of ill posed problems", BIT Numerical Mathematics, vol. 43, (**2003**), pp. 1003–1017.
[17] J. G. Nagy and D. O'Leary, "Fast iterative image restoration with a spatially-varying PSF", In F. T. Luk, editor, Advanced Signal Processing: Algorithms, Architectures, and Implementations VII, vol. 3162, (**1997**) October, pp. 388–399.
[18] J. G. Nagy and D. O'Leary, "Restoring images degraded by spatially variant blur", SIAM J. Sci. Comput., vol. 19, no. 4, (**1998**), pp. 1063–1082.
[19] P. Novati and M. R. Russo, "Adaptive Arnoldi-Tikhonov regularization for image restoration", Numerical Algorithms, vol. 65, (**2014**), pp. 745–757.
[20] A. N. Tikhonov and V. Y. Arsenin, "Solutions of ill-posed problems", Winston & Sons, Washignton D.C., (**1977**).

