Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Linear Algebra and its Applications 391 (2004) 203–221

LINEAR ALGEBRA AND ITS APPLICATIONS

www.elsevier.com/locate/laa

Implementation of the regularized structured total least squares algorithms for blind image deblurring

N. Mastronardi [a,*,1], P. Lemmerling [b,2], A. Kalsi [c], D.P. O'Leary [c,3], S. Van Huffel [b,4]

[a]*Istituto per le Applicazioni del Calcolo, CNR, via Amendola 122/D, Bari 70126, Italy*
[b]*Department of Electrical Engineering, ESAT-SCD (SISTA), Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium*
[c]*Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA*

Received 1 September 2003; accepted 26 July 2004

Submitted by P.C. Hansen

Abstract

The structured total least squares (STLS) problem has been introduced to handle problems involving structured matrices corrupted by noise. Often the problem is ill-posed. Recently, regularization has been proposed in the STLS framework to solve ill-posed blind

* Corresponding author.
E-mail address: n.mastronardi@area.ba.cnr.it (N. Mastronardi).

[1] This work was partially supported by MIUR, grant number 2002014121.

[2] Philippe Lemmerling is supported by a post-doctoral K. U. Leuven fellowship.

[3] This work was supported by the National Science Foundation under Grant CCR 0204084.

[4] Research supported by Research Council KUL: GOA-Mefisto 666, IDO /99/003 and /02/009 (Predictive computer models for medical classification problems using patient data and expert knowledge), several PhD/postdoc & fellow grants; Flemish Government: FWO: PhD/postdoc grants, projects, G.0078.01 (structured matrices), G.0407.02 (support vector machines), G.0269.02 (magnetic resonance spectroscopic imaging), G.0270.02 (nonlinear Lp approximation), research communities (ICCoS, ANMMM); IWT: PhD Grants, Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006): Dynamical Systems and Control: Computation, Identification & Modelling)); EU: PDT-COIL, BIOPATTERN, ETU-MOUR.

0024-3795/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2004.07.006

deconvolution problems encountered in image deblurring when both the image and the blurring function have uncertainty. The kernel of the regularized STLS (RSTLS) problem is a least squares problem involving Block–Toeplitz–Toeplitz–Block matrices.

In this paper an algorithm is described to solve this problem, based on a particular implementation of the generalized Schur Algorithm (GSA). It is shown that this new implementation improves the computational efficiency of the straightforward implementation of GSA from $O(N^{2.5})$ to $O(N^2)$, where N is the number of pixels in the image.

Keywords: Displacement rank; Block Toeplitz matrix; Structured total least squares; Generalized Schur algorithm; Image deblurring; Tikhonov regularization

1. Introduction

Image restoration is the process of reconstructing the true image from a degraded one. The mathematical model of the two-dimensional (2-D) blurred image is a 2-D first-kind Fredholm integral equation

$$\int_{\Omega} \mathscr{A}(s,t)\mathbf{x}(t)\mathrm{d}t = \mathbf{b}(s) + \nu(s) = \mathbf{b}^*(s), \tag{1}$$

where the spatial coordinates $s \in \mathbb{R}^2, t \in \mathbb{R}^2$ and Ω is a closed region containing the domain of the image. The blurring of the unknown true image $\mathbf{x} : \mathbb{R}^2 \to \mathbb{R}$ is modelled with the *point spread function* (PSF) $\mathscr{A} : \mathbb{R}^4 \to \mathbb{R}$ plus the additional noise $\nu : \mathbb{R}^2 \to \mathbb{R}$. The function $\mathbf{b} : \mathbb{R}^2 \to \mathbb{R}$ is the measured image, while $\mathbf{b}^*$ is the exact blurred image. The model is discretized into a matrix equation

$$Ax = b + \beta, \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$ is the discretized counterpart of $\mathscr{A}$ and x, b and β are also the discrete version of $\mathbf{x}$, $\mathbf{b}$ and ν. Often, the point spread function $\mathscr{A}$ is assumed spatially invariant, i.e., $\mathscr{A}(s,t) = \mathscr{A}(s-t)$. In this case, the integral equation (1) is a convolution, and the corresponding matrix A is a block Toeplitz matrix with Toeplitz blocks (BTTB). If the cause of the blur, and hence $\mathscr{A}$, is not known exactly, then the matrix A is also uncertain, and the problem is known as blind deconvolution. In this case (2) should be replaced by the following problem,

$$(A+E)x = b + \beta, \tag{3}$$

with E and β unknown. Since in image deblurring problems the matrix A is structured, it is natural to require that the matrix E has the same structure as A, too. The latter constraint leads to the *Structured Total Least Squares* (STLS) formulation [21,14]

$$\begin{aligned}&\min_{E,\beta,x} \| \left[E|\beta\right] \|_F^2\\ &\text{such that } (A+E)x = b + \beta,\\ &\text{and } E \text{ same structure as } A.\end{aligned}$$

In many deblurring problems and other discretized problems involving integral equations of the first kind, the matrix A is so ill-conditioned that noise in the observations b is magnified in solving the STLS problem and a meaningful solution cannot be obtained. In this case, regularization methods [7,8] must be considered in order to stabilize the STLS problem and to obtain better results.

An iterative method to solve the STLS problem was proposed in [21]. If A is a Toeplitz matrix, the bulk of each iteration of the iterative method is the solution of a least squares problem involving a particular Toeplitz block matrix. A fast algorithm for solving this least squares problem, based on a particular implementation of the Generalized Schur Algorithm (GSA) [9], has been proposed in [14]. Regularization in the STLS (RSTLS) framework has been considered in [18,10,15]. Moreover, a fast algorithm for solving the least squares problem, the kernel of the iterative method for solving the RSTLS, when A is a Toeplitz matrix, is considered in [15]. The latter algorithm is an extension of the fast implementation of GSA for STLS problems [14]. In [18,10] the RSTLS algorithm is applied to the image deblurring problem. At each iteration of the algorithm a highly structured least squares problem needs to be solved. In this paper we propose a fast implementation of the GSA for solving this least squares problem. The newly proposed algorithm improves the computational efficiency of the algorithms in [18,10] from $\mathrm{O}(N^{2.5})$ to $\mathrm{O}(N^2)$, where N is the number of pixels in the image. Other approaches for solving RSTLS problems have been considered in [17,3].

The paper is organized as follows. We review the RSTLS problem for image deblurring problems, and an algorithm for solving it, in Section 2. Then we describe an implementation of GSA for solving it in Section 3. The accuracy of the RSTLS estimator for image deblurring problems is compared to the STLS estimator in Section 4 and then we conclude in Section 5.

2. The RSTLS problem for image deblurring

In order to make STLS more robust in the presence of noise, Tikhonov regularization has been introduced in the STLS framework [18,15,10]. The regularized STLS (RSTLS) problem is formulated in the following way:

$$\begin{aligned}&\min_{E,x} \alpha^{\mathrm{T}}\alpha + \beta^{\mathrm{T}}\beta + \lambda^2\|Cx\|_2^2\\ &\text{such that } (A + E(\alpha))x = b + \beta,\\ &E \text{ same structure as } A,\end{aligned} \tag{4}$$

with $A, E \in \mathbb{R}^{m\times n}$, E the correction applied to A, $\beta \in \mathbb{R}^{m\times 1}$ the correction applied to $b \in \mathbb{R}^{m\times 1}$ the output, $x \in \mathbb{R}^{n\times 1}$ the impulse response, C a regularization operator, and λ the regularization parameter. The operator C is commonly chosen to be the identity matrix or a difference operator [7]. For the sake of simplicity, in this paper C is chosen equal to the identity matrix. Fast RSTLS algorithms can also be developed in case C is chosen to be a difference operator [15]. Moreover, α is a vector denoting the *minimal parametrization* of the structured matrix E. For instance, if E is the Toeplitz matrix

$$E = \begin{bmatrix} \alpha_n & \alpha_{n-1} & \cdots & \alpha_1 \\ \alpha_{n+1} & \alpha_n & \cdots & \alpha_2 \\ \vdots & \ddots & & \vdots \\ \alpha_{m+n-1} & \alpha_{m+n-2} & \cdots & \alpha_m \end{bmatrix} \tag{5}$$

then the minimal parametrization α of E is the vector

$$\alpha = [\alpha_1, \alpha_2, \ldots, a_{m+n-1}]^{\mathrm{T}}.$$

Using the zeroth and first order terms of the Taylor series expansion of $\beta = (A + E(\alpha))x - b$ (where we use the notation $E(\alpha)$ to denote the dependence of E on α) around $[\alpha^{\mathrm{T}} x^{\mathrm{T}}]^{\mathrm{T}}$, we obtain the Gauss–Newton method for solving (4) (for a proof, see [21]). The RSTLS algorithm, as outlined in [18,15], is then as follows:

RSTLS Algorithm
Input: extended data matrix $[A\ b] \in \mathbb{R}^{m\times(n+1)} (m > n)$ of full rank $n + 1$, and λ.
Output: correction vector α and parameter vector x s.t. $\alpha^{\mathrm{T}}\alpha + \beta^{\mathrm{T}}\beta + \lambda^2\|x\|_2^2$ is as small as possible and $\beta = (A + E(\alpha))x - b$.

```
Step 1: $\alpha \leftarrow 0$
        $x \leftarrow A\backslash b$
Step 2: while stop criterion not satisfied
          Step 2.1: $\min_{\Delta x, \Delta\alpha} \|Mz + v\|_2$,
                    with $M = \begin{bmatrix} A+E & X \\ \lambda I_n & 0 \\ 0 & I_{m+n-1} \end{bmatrix}$, $z = \begin{bmatrix} \Delta x \\ \Delta\alpha \end{bmatrix}$, $v = \begin{bmatrix} \beta \\ \lambda x \\ \alpha \end{bmatrix}$,
          Step 2.2: $x \leftarrow x + \Delta x$
                    $\alpha \leftarrow \alpha + \Delta\alpha$ and construct $E$ from $\alpha$
                    $\beta \leftarrow (A + E)x - b$
        end
```

The matrix X is defined such that $X\alpha = Ex$. For instance, if E is the Toeplitz matrix defined in (5), X becomes

$$X = \begin{bmatrix} x_n & x_{n-1} & \cdots & x_1 & & \\ & \ddots & \ddots & \ldots & \ddots & \\ & & x_n & x_{n-1} & \cdots & x_1 \end{bmatrix}_{m\times(m+n-1)}.$$

Note that $A\backslash b$ in Step 1 is a shorthand notation for the LS solution of the over-determined system of equations $Ax \approx b$. The latter LS problem can be solved in a fast way by means of GSA [9] if A is a structured matrix. We apply this algorithm to deblurring images whose point-spread function is spatially invariant. In this case, we have measured a set of pixel values

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1m} \\ y_{21} & y_{22} & \cdots & y_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mm} \end{bmatrix}.$$

The aim is to reconstruct the true image

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}.$$

Let us order the pixels by rows to create a one-dimensional vector of unknowns:

$$x = [x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{n1}, x_{n2}, \ldots, x_{nn}]^{\mathrm{T}},$$

and, similarly, we create a vector of observations,

$$b = [y_{11}, y_{12}, \ldots, y_{1m}, \ldots, y_{m1}, y_{m2}, \ldots, y_{mm}]^{\mathrm{T}}.$$

For definiteness we will assume that the blurring function averages the p^2 nearest neighbours of each pixel, with $p \ll n$ and that $m = n + p - 1$. In this case the matrix A has p block diagonals, each with p diagonals:

$$A = \begin{bmatrix} T_1 & & & & \\ T_2 & T_1 & & & \\ \vdots & \ddots & \ddots & & \\ T_p & \ddots & \ddots & \ddots & \\ & \ddots & \ddots & T_2 & T_1 \\ & & \ddots & \ddots & T_2 \\ & & & \ddots & \vdots \\ & & & & T_p \end{bmatrix}, \quad T_j = \begin{bmatrix} t_{j1} & & & \\ t_{j2} & t_{j1} & & \\ \vdots & \ddots & \ddots & \\ t_{jp} & \ddots & \ddots & t_{j1} \\ & \ddots & \ddots & t_{j2} \\ & & t_{jp} & \vdots \\ & & & t_{jp} \end{bmatrix}, \quad j = 1, \ldots, p.$$

The dimension of A is $m^2 \times n^2$, and the dimension of T_j is $m \times n$. The matrix E has the same structure as A, with entries α_{ji}. The relation $X\alpha = Ex$ holds if we define

$$X = \begin{bmatrix} X_1 & & & \\ X_2 & X_1 & & \\ \vdots & X_2 & \ddots & \\ X_p & \ddots & \ddots & X_1 \\ \vdots & & & \vdots \\ X_n & X_{n-1} & \vdots & X_{n-p+1} \\ & X_n & \ddots & \vdots \\ & & \ddots & X_{n-1} \\ & & & X_n \end{bmatrix},$$

$$X_j = \begin{bmatrix} x_{j1} & & & \\ x_{j2} & x_{j1} & & \\ \vdots & x_{j2} & \ddots & \\ x_{jp} & \ddots & \ddots & x_{j1} \\ \vdots & & & \vdots \\ x_{jn} & \ddots & \ddots & x_{jn-p+1} \\ & \ddots & x_{j_{n-1}} & \vdots \\ & & x_{jn} & x_{jn-1} \\ & & & x_{jn} \end{bmatrix}, \quad j = 1, \ldots, n.$$

If we solve Step 2.1 of the RSTLS algorithm using normal equations, we would solve the linear system of equations

$$\hat{M}z = M^{\mathrm{T}}v, \tag{6}$$

where $\hat{M} = M^{\mathrm{T}}M$. Forming $\hat{M}$ and computing its Cholesky factorization $\hat{M} = LL^{\mathrm{T}}$ would cost $\mathrm{O}((n^2 + m^2)^3)$ operations. In the next section we show how the generalized Schur algorithm can be used to compute the Cholesky factorization in $\mathrm{O}((n^2 + m^2)^2)$ operations by exploiting the structure of the matrix.

3. Generalized Schur algorithm for image deblurring

The generalized Schur algorithm allows fast computation of a variety of decompositions (i.e., QR, $LDL^{\mathrm{T}}, \ldots$) of structured matrices. To understand its use, we introduce the basic concepts of *displacement theory* for symmetric positive definite (SPD) matrices (since $\hat{M}$ is a SPD matrix). An extensive treatment of this topic can be found in [9].

Let R be a SPD matrix and Φ be a strictly lower triangular matrix of order $\hat{N}$, respectively. Let κ be the number of subdiagonals below and including the main diagonal of Φ equal to zero, i.e.,

$$\Phi = \begin{array}{r@{}l} \kappa\left\{\vphantom{\begin{array}{c}0\\ \vdots\\ 0\end{array}}\right. & \\ \vphantom{\begin{array}{c}\times\\ \vdots\\ \times\end{array}} & \end{array}\left[\begin{array}{cccccc} 0 & 0 & \cdots & \cdots & \cdots & 0\\ \vdots & 0 & \ddots & \cdots & \cdots & \vdots\\ 0 & \ddots & \ddots & \ddots & \cdots & \vdots\\ \times & 0 & \ddots & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots\\ \times & \cdots & \times & 0 & \cdots & 0 \end{array}\right].$$

The displacement of R, with respect to Φ, is defined as

$$\nabla_{\Phi} R \equiv R - \Phi R \Phi^{\mathrm{T}}. \tag{7}$$

If $\nabla_{\Phi} R$ has low rank, *independent* of $\hat{N}$, then R is said to be *structured* with respect to Φ, and the rank δ of ∇_{Φ} is called the *displacement rank* of R. We observe that the eigenvalues of $\nabla_{\Phi} R$ are real, and we define η to be the number of strictly positive eigenvalues of $\nabla_{\Phi} R$ and γ to be the number of negative ones. Therefore, the displacement rank is $\delta = \eta + \gamma$ and we can write

$$\nabla_{\Phi} R = R - \Phi R \Phi^{\mathrm{T}} = G^{\mathrm{T}} J G, \quad \text{where } J = J^{\mathrm{T}} = (I_n \oplus -I_{\gamma}) \tag{8}$$

is a signature matrix, and $G \in \mathbb{R}^{\delta \times \hat{N}}$. Without loss of generality, we suppose $\kappa \leqslant \eta$. The pair $\{G, J\}$ is called a ∇_{Φ}*-generator* of R and G is called a ∇_{Φ}-generator matrix, or, if there is no ambiguity, simply a generator matrix. This representation is clearly not unique; for example, $\{\Theta G, J\}$ is also a generator for any J-orthogonal matrix Θ (i.e., for any Θ such that $\Theta^{\mathrm{T}} J \Theta = J$), since

$$G^{\mathrm{T}} \underbrace{\Theta^{\mathrm{T}} J \Theta}_{J} G = G^{\mathrm{T}} J G.$$

Thus we can choose G for computational convenience, and we will exploit this fact. The first η and the last γ rows of G are called *positive* and *negative* ∇_{Φ}*-generators*, respectively, or, if there is no ambiguity, simply *positive* and *negative generators*. A generator matrix is in *proper form* if

$$G = \begin{array}{r@{}l} \kappa\left\{\vphantom{\begin{array}{c}\times\\ \vdots\\ \times\end{array}}\right. & \\ \vphantom{\begin{array}{c}0\\ \vdots\\ 0\end{array}} & \\ & \end{array}\begin{array}{l}\left[\begin{array}{cccccc} \times & \cdots & \times & \times & \cdots & \times\\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\ \times & \cdots & \times & \times & \cdots & \vdots\\ 0 & \cdots & 0 & \times & \cdots & \vdots\\ \vdots & \cdots & \vdots & \vdots & \cdots & \vdots\\ 0 & \cdots & 0 & \times & \cdots & \times \end{array}\right]. \\ \ \underbrace{\hphantom{0 \;\cdots\; 0}}_{\kappa} \end{array}$$

The computation of the LDL^{T} factorization of R, in a standard way, where L is a block lower triangular matrix and D is a nonsingular block diagonal matrix, with the diagonal blocks of L and D of order κ, is accomplished in $\upsilon = \lceil \hat{N}/\kappa \rceil$ steps (for simplicity, we assume $\upsilon = \hat{N}/\kappa$). Since R is a SPD matrix, L and D can be chosen equal to a lower triangular matrix and the identity matrix, respectively. In this case, the computed factor L is the Cholesky factor of R.

At the first step, one computes L_1, the first κ columns of L, and D_1, the first block diagonal of order κ of D, from the first κ columns (rows) of R. Let $R_1 = R$. At the second step, L_2, the next κ columns of L, and D_2, the next block diagonal of order κ of D, are computed from the first κ columns (rows) of the matrix R_2, the Schur complement of R_1 with respect to $R_1(1:\kappa, 1:\kappa)$,

$$\left[\begin{array}{c|c} 0 & \\ \hline & R_2 \end{array}\right] = I_{\kappa,\hat{N}}(R_1 - L_1 D_1^{-1} L_1^{\mathrm{T}}) I_{\kappa,\hat{N}}, \quad \text{where } I_{\kappa,\hat{N}} = \left[\begin{array}{c|c} 0_\kappa & \\ \hline & I_{\hat{N}-\kappa} \end{array}\right].$$

The LDL^{T} factorization of R is then accomplished applying the same machinery at each step of the algorithm.

We now show that the same computations are made by GSA for computing the LDL^{T} factorization of R.

Using induction and the fact that Φ is nilpotent, one can derive from (8) that

$$R = \sum_{i=0}^{\upsilon-1} \Phi^i G^{\mathrm{T}} J G \Phi^{i^{\mathrm{T}}}. \tag{9}$$

Suppose we have computed G. (We discuss how to do this in Section 3.1) Let

$G_1 \equiv G$, and let Θ_1 be a J-orthogonal matrix such that

$$\hat{G}_1 = \Theta_1 G_1 = \begin{bmatrix} \dot{G}_1 \\ \ddot{G}_1 \end{bmatrix} \tag{10}$$

is in proper form and $\dot{G}_1$ contains the first κ rows of $\hat{G}_1$, with $\det(\dot{G}_1(1:\kappa, 1:\kappa)) \neq 0$. It turns out that

$$L_1 D_1^{-1} L_1^{\mathrm{T}} = \dot{G}_1^{\mathrm{T}} \dot{G}_1, \tag{11}$$

hence, D_1 and L_1 can be directly computed from (11). Taking (9) into account, R_2 can be written in the following way,

$$\begin{aligned} \left[\begin{array}{c|c} 0 & \\ \hline & R_2 \end{array}\right] &= I_{\kappa,\hat{N}} \left(R - L_1 D_1^{-1} L_1^{\mathrm{T}} \right) I_{\kappa,\hat{N}} \\ &= I_{\kappa,\hat{N}} \left(\sum_{i=0}^{\upsilon-1} \Phi^i \hat{G}_1^{\mathrm{T}} J \hat{G}_1 \Phi^{i^{\mathrm{T}}} \right) I_{\kappa,\hat{N}} - I_{\kappa,\hat{N}} L_1 D_1^{-1} L_1^{\mathrm{T}} I_{\kappa,\hat{N}} \\ &= I_{\kappa,\hat{N}} \left[\hat{G}_1^{\mathrm{T}} J \hat{G}_1 \right] I_{\kappa,\hat{N}} + \sum_{i=1}^{\upsilon-1} \Phi^i \hat{G}_1^{\mathrm{T}} J \hat{G}_1 \Phi^{i^{\mathrm{T}}} - I_{\kappa,\hat{N}} L_1 D_1^{-1} L_1^{\mathrm{T}} I_{\kappa,\hat{N}}. \end{aligned}$$

We observe that

$$I_{\kappa,\hat{N}}[\hat{G}_1^{\mathrm{T}} J \hat{G}]I_{\kappa,\hat{N}} = I_{\kappa,\hat{N}}\dot{G}_1^{\mathrm{T}}\dot{G}_1 I_{\kappa,\hat{N}} + \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}.$$

Hence taking (11) into account,

$$\left[\begin{array}{c|c} 0 & \\ \hline & R_2 \end{array}\right] = \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix} + \sum_{i=1}^{v-1} \Phi^i \hat{G}_1^{\mathrm{T}} J \hat{G}_1 \Phi^{i^{\mathrm{T}}}.$$

Furthermore, since the first κ columns of $\ddot{G}_1$ are $\mathbf{0}$,

$$\Phi^{v-1}\begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}\Phi^{(v-1)^{\mathrm{T}}} = \mathbf{0}.$$

Therefore, considering (10), we have

$$\begin{aligned}
\left[\begin{array}{c|c} 0 & \\ \hline & R_2 \end{array}\right] &= \sum_{i=0}^{v-2} \Phi^i \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}\Phi^{i^{\mathrm{T}}} + \sum_{i=1}^{v-1} \Phi^i \begin{bmatrix}\dot{G}_1\\ \mathbf{0}\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\dot{G}_1\\ \mathbf{0}\end{bmatrix}\Phi^{i^{\mathrm{T}}} \\
&= \sum_{i=0}^{v-2} \Phi^i \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\mathbf{0}\\ \ddot{G}_1\end{bmatrix}\Phi^{i^{\mathrm{T}}} + \sum_{i=0}^{v-2} \Phi^i \begin{bmatrix}\Phi\dot{G}_1\\ \mathbf{0}\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\Phi\dot{G}_1\\ \mathbf{0}\end{bmatrix}\Phi^{i^{\mathrm{T}}} \\
&= \sum_{i=0}^{v-2} \Phi^i \begin{bmatrix}\Phi\dot{G}_1\\ \ddot{G}_1\end{bmatrix}^{\mathrm{T}} J \begin{bmatrix}\Phi\dot{G}_1\\ \ddot{G}_1\end{bmatrix}\Phi^{i^{\mathrm{T}}} \\
&= \sum_{i=0}^{v-2} \Phi^i G_2^{\mathrm{T}} J G_2 \Phi^{i^{\mathrm{T}}}. \qquad (12)
\end{aligned}$$

Hence, the computation of a ∇_Φ-generator $\{G_2, J\}$ of $\left[\begin{array}{c|c} 0 & \\ \hline & R_2 \end{array}\right]$ can be summarized in the following steps,

- Step 1: reduction of G_1 into proper form.
- Step 2: computation of the product $\Phi\dot{G}_1$.

Therefore, it turns out it is essential to consider a matrix Φ such that $\nabla_\Phi R$ has low rank, and the product $\Phi\dot{G}_1$ can be computed efficiently.

Recursively proceeding, we obtain the LDL^{T} factorization of R. The procedure just described is better known under the name of generalized Schur algorithm.

There are two natural choices for the displacement operator Φ for image deblurring: either the *block shift matrix*

$$Z = \left[\begin{array}{c|c} Z_{n^2} & \\ \hline & Z_{p^2} \end{array}\right], \quad Z_{k^2} = \begin{bmatrix} 0_k & & & \\ I_k & 0_k & & \\ & \ddots & \ddots & \\ & & I_k & 0_k \end{bmatrix}_{k^2 \times k^2}$$

or the *scalar shift matrix*

$$\hat{Z} \equiv \left[\begin{array}{ccc|ccc} \hat{Z}_n & & & & & \\ & \ddots & & & & \\ & & \hat{Z}_n & & & \\ \hline & & & \hat{Z}_p & & \\ & & & & \ddots & \\ & & & & & \hat{Z}_p \end{array}\right]_{(n^2+p^2)\times(n^2+p^2)},$$

$$\hat{Z}_l = \begin{bmatrix} 0 & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}_{l\times l}.$$

Indeed, the rank of both matrices $\nabla_Z(\hat{M})$ and $\nabla_{\hat{Z}}(\hat{M})$ is $\delta = 2(n+p)$, but their sparsity patterns are much different, as shown in Fig. 1. Moreover, in both cases, the product of the shift operators times a vector is made without any computational effort. Since $\hat{M}$ has order $\tilde{N} \equiv n^2 + p^2$ the computational complexity of GSA is $\mathrm{O}(\delta \tilde{N}^2) = \mathrm{O}(n^5)$, $(n \gg p)$. We will show that, choosing the initial generators in a suitable way, and implementing the algorithm taking into account the pattern of these generators, the computational complexity of GSA can be reduced to $\mathrm{O}(n^4)$. In an analogous way, using the scalar displacement operator $\hat{Z}$, it is possible to derive an implementation of GSA with the same computational complexity. For the sake of brevity, only the implementation of GSA with respect to Z is considered in this paper.

3.1. Choice of the initial generator matrix

Choosing the generators in an appropriate way, and preserving their structure during the first n iterations of GSA, the computational complexity can be reduced. This choice is a generalization of an algorithm of [14] from the scalar to the block case.

First, we observe that n of the positive and negative generators can directly be computed from the first n rows (columns) of the displacement matrix $\nabla_Z(\hat{M})$, i.e. the first n rows (columns) of $\hat{M}$ (see Fig. 1). We do this by observing that if γ is a scalar and d is a vector, then

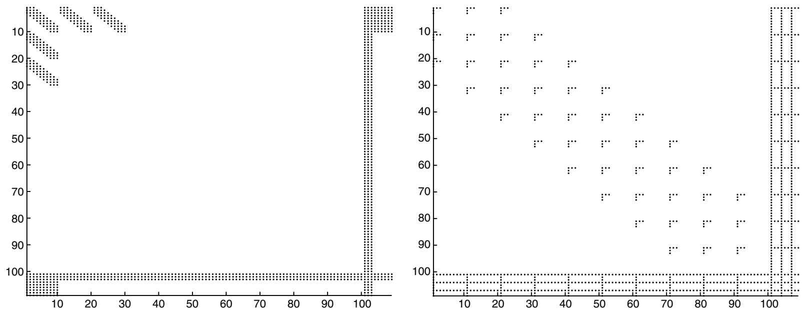

Fig. 1. Left: sparsity pattern of the displacement matrix $\nabla_Z(\hat{M})$. Right: sparsity pattern of the displacement matrix $\nabla_{\hat{Z}}(\hat{M})$.

$$\begin{bmatrix} \gamma & d^{\mathrm{T}} \\ d & 0 \end{bmatrix} = \begin{bmatrix} \sqrt{\gamma} \\ d/\sqrt{\gamma} \end{bmatrix} \begin{bmatrix} \sqrt{\gamma} & d^{\mathrm{T}}/\sqrt{\gamma} \end{bmatrix} - \begin{bmatrix} 0 \\ d/\sqrt{\gamma} \end{bmatrix} \begin{bmatrix} 0 & d^{\mathrm{T}}/\sqrt{\gamma} \end{bmatrix},$$

and sparsity of d is preserved in this representation. Since the entries from $np + 1$ up to n^2 in rows 1 through n of $\nabla_Z(\hat{M})$ are zero, the corresponding entries of the first n positive and negative generators are zero.

The remaining initial generators are computed from the rows $n^2 + k$, $k = 1, \ldots, p$, of $\hat{\nabla}_Z(\hat{M})$, the matrix obtained from $\nabla_Z(\hat{M})$ after setting to zero the first n rows and columns. This computation is accomplished in an iterative fashion. The $(n + 1)$th positive generator and the $(n + 1)$th negative generator are chosen as the generators of the rank-two matrix formed from row and column $n^2 + 1$ of $\hat{\nabla}_Z(\hat{M})$. It turns out that this pair of generators differs only in entry $n^2 + 1$.

The next generators are computed in a similar way from the rank-two matrix made by the row and column $n^2 + 2$ of $\hat{\nabla}_Z(\hat{M})$, after setting row and column $n^2 + 1$ to zero. In this case, the generators differ only in the entry $n^2 + 2$. We continue in this way to compute the remaining generators.

Summarizing, the first n pairs of positive and negative generators have entries from $np + 1$ through n^2 equal to zero. Moreover, each of the remaining $n + k$ pairs of generators, $k = 1, \ldots, p$, differs only in the entry $n^2 + k$.

Let us call G the generator matrix built in this way. The sparsity of the generators is crucial for considering an implementation of GSA with reduced computational complexity.

3.2. GSA for computing the Cholesky factorization

In this section the implementation of GSA for computing the Cholesky factorization of $\hat{M}$ with respect to Z is described.

Let $J_{n,P} = (I_{n+p} \oplus -I_{n+p})$. A matrix $Q \in \mathbb{R}^{2(n+p)\times(n+p)}$ is $J_{n,p}$-orthogonal if $Q^{\mathrm{T}} J_{n,p} Q = J_{n,p}$. Let $G_P \in \mathbb{R}^{(n+p)\times\tilde{N}}$ be the positive generator matrix for $\hat{M}$ with respect to Z, and let $G_N \in \mathbb{R}^{(n+p)\times\tilde{N}}$ be the negative generator matrix.

Define

$$G \equiv \begin{bmatrix} G_P \\ G_N \end{bmatrix}.$$

The generalized Schur algorithm is summarized in the following scheme. A matrix $\hat{G}_l$ is in *proper form* if

$$\hat{G}_l(i,j) = 0, \begin{cases} i > j,\ j \leqslant n, & \text{if } l \leqslant n, \\ i > j,\ j \leqslant p, & \text{if } l > n, \end{cases}$$

i.e., if $l \leqslant n$, the entries of the first n columns of the $\hat{G}_l$ below the main diagonal are zero, and if $l > n$, the entries of the first p columns of the $\hat{G}_l$ below the main diagonal are zero.

Proper form of the matrices $\hat{G}_l, l = 1, \ldots, n+p$, can be accomplished in different ways. A straightforward way is to choose Q_l as the product of n stabilized hyperbolic Householder transformations $Q_{l,k},\ k = 1, \ldots, n$ [20]. The kth hyperbolic Householder transformation puts zeros below the main diagonal in column k. Note that the first transformation ruins the sparsity of the generator matrix, since $2p$ generator rows are dense. The number of rows in the generator matrix at the iteration $k, k = 1, \ldots, n$, is $\tilde{N} - n(k-1)$. Hence the kth iteration can be accomplished in $\mathrm{O}((n+p)(\tilde{N} - n(k-1)))$ flops and, therefore, the first n iterations of GSA have a computational complexity of order $\mathrm{O}(n^5 + pn^4)$. We observe that, since $n \gg p$, the computational complexity of the last p iterations of GSA is negligible with respect to the first n ones. In order to improve the operations count below $\mathrm{O}(n^5)$, we will need to rearrange the transformations so that sparsity is preserved.

To do this, we choose in each of the first n iterations to operate on the sparse rows first using hyperbolic Householder transformations (Step l.1 in algorithm GSA below). This still leaves $2p$ rows to be reduced. Then we apply a Givens rotation to the first positive generator and the first dense positive generator in order to annihilate the first nonzero entry of the latter generator. We follow this by a hyperbolic rotation between the first positive generator and the first dense negative generator to annihilate the first nonzero entry position of the latter generator. We repeat this procedure for the first position of each of the other $p-1$ dense generator pairs, and then for the other $n-1$ columns in the first block, to complete the reduction to proper form (Step l.2). The last p^2 dense columns are easily handled in a separate loop.

We cannot afford to explicitly update any of the dense generator rows, but it turns out that each pair of generators has the effect of multiplying the entries from $np+1$ to $n^2 - p^2 - 1$ by a product of cosines, and this product is accumulated and applied only when the entries are shifted to be below a dense block in the other generators.

This procedure is summarized in algorithm GSA, with more details given following it.

Generalized Schur algorithm (GSA)

```
input : $\hat{G}$, the generator matrix of $\hat{M}$ with respect to $Z$
output : $L$, the Cholesky factor of the matrix $\hat{M}$.
$G_0 \equiv \hat{G}$
for l = 1 : n,
    Step $l.1$
        find a $J_{n,p}$ -orthogonal matrix $\tilde{Q}_l$ such that
        $\tilde{G} = \tilde{Q}_l G_{l-1}$ and $\tilde{G}_l([1:n, n+p+1:2n+p], :)$ is in proper form
    Step $l.2$
        find a $J_{n,p}$-orthogonal matrix $\hat{Q}_l$ such that
        $\hat{G}_l = \hat{Q}_l \tilde{G}_l$ is in proper form
        $L(n(l-1)+1:\tilde{N}, n(l-1)+1:nl) = \hat{G}_l(1:n, :)^{\mathrm{T}}$
        $C = \hat{G}_l(1:n, :)Z(n(l-1)+1:\tilde{N}, n(l-1)+1:\tilde{N})^{\mathrm{T}}$
        $G_l = \begin{bmatrix} C(:, n+1:\tilde{N}-n(l-1)) \\ \hat{G}_{l-1}(n+1:2(n+p), n+1:\tilde{N}-n(l-1)) \end{bmatrix} \in \mathbb{R}^{2(n+p)\times(\tilde{N}-nl)}$
end
for l = 1 : p,
    find a $J_{n,p}$-orthogonal matrix $Q_{n+l}$ such that
    $\hat{G}_{n+l} = Q_{n+l} G_{n+l-1}$ is in proper form
    $L(n^2+p(l-1)+1:\tilde{N}, n^2+p(l-1)+1:n^2+pl) = \hat{G}_l(1:p, :)^{\mathrm{T}}$
    $C = \hat{G}_{n+l}(1:p, :)Z(n^2+p(l-1)+1:\tilde{N}, n^2+p(l-1)+1:\tilde{N})^{\mathrm{T}}$
    $G_l \begin{bmatrix} C(:, p+1:p^2-p(l-1)) \\ \hat{G}_{n+l-1}(p+1:2(n+p), p+1:p^2-p(l-1)) \end{bmatrix} \in \mathbb{R}^{2(n+p)\times(p-l)p}$
end
```

Step $l.1$ can be accomplished by applying n hyperbolic Householder transformations $\tilde{Q}_{l,k}$ to the left of G_{l-1}, determined to annihilate the entries in the nondense rows of the kth column below the main diagonal, for $k = 1, \ldots, n$. Hence $\tilde{Q}_l \equiv \tilde{Q}_{l,n}\tilde{Q}_{l,n-1}\ldots\tilde{Q}_{l,1}$ and $\tilde{G}_l \equiv G_{l,n}$. We observe that the generator submatrix $G_{l-1}([n+1:n+p, 2n+p+1:2n+2p], :)$ is not involved in this reduction. Moreover, only the first np and the last p^2 entries of the involved generators are different from zero. Step $l.2$ must be implemented in such a way that the latter structure is preserved, for $l = 1, \ldots, n$. In this case, any of these steps can be accomplished in $\mathrm{O}(pn^3 + p^2n^2)$ flops.

Note that the submatrix $\tilde{G}_l(n+p+1:2n+p, :)$ is already in proper form after step $l.1$, and hence it is not involved in step $l.2$. An efficient implementation of step $l.2$ can be accomplished by the following procedure:

```
% Step $l.2$
$\hat{G}_l \equiv \tilde{G}_l$
for i = 1 : n,
    for k = 1 : p,
        Step $l.i.k$
        find a $J_{n,p}$-orthogonal matrix $Q_{lik}$ such that
        $\breve{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))$
            $= Q_{lik}\hat{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))$
        is in proper form
        $\hat{G}_l \equiv \breve{G}_l$
    end
end
```

The generators $i, n+k, 2n+p+k$ involved in Step have the following structure

(1) The entries of $\hat{G}_l(i, np+1 : n^2 - n(l-1))$ are zero.
(2) $\hat{G}_l(n+k, :)$ and $\hat{G}_l(2n+p+k, :)$ differ only for the entry $n^2 + k - n(l-1)$.

Exploiting the particular structure of $\hat{G}_l$, a technique has been developed in [14, 13] to compute the reduction to proper form of the matrix involved in step in $\mathrm{O}(np + p^3)$ flops, by preserving the initial structure in the generator matrix. Each of these steps is accomplished choosing Q_{lik} as the product of the hyperbolic rotation H_{lik} with the Givens rotation G_{lik} computed such that $H_{lik}G_{lik}\hat{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))$ is in proper form. In particular,

$$G_{lik} = \begin{bmatrix} c_{lik} & s_{lik} & \\ -s_{lik} & c_{lik} & \\ & & 1 \end{bmatrix}$$

is computed such that

$$\begin{aligned}&\tilde{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))\\ &\quad = G_{lik}\hat{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))\end{aligned}$$

and $\tilde{G}_l(n+k, i) = 0$. It turns out that

$$\begin{aligned}&\tilde{G}_l(i, i) = \sqrt{\hat{G}_l^2(i, i) + \hat{G}_l^2(n+k, i)},\\ &c_{lik} = \frac{\hat{G}_l(i, i)}{\tilde{G}_l(i, i)}, \quad s_{lik} = \frac{\hat{G}_l(n+k, i)}{\tilde{G}_l(i, i)},\end{aligned} \tag{13}$$

Moreover,

$$\tilde{G}_l(i, np+1 : n^2 - n(l-1)) = s_{lik}\hat{G}_l(n+k, np+1 : n^2 - n(l-1))$$

and

$$\tilde{G}_l(n+k, np+1 : n^2 - n(l-1)) = c_{lik}\hat{G}_l(n+k, np+1 : n^2 - n(l-1)).$$

To complete the step, the hyperbolic rotation

$$H_{lik} = \begin{bmatrix} \tilde{c}_{lik} & & -\tilde{s}_{lik} \\ & 1 & \\ -\tilde{s}_{lik} & & \tilde{c}_{lik} \end{bmatrix}$$

is computed such that

$$\begin{aligned}&\breve{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1)) \\ &\quad = H_{lik}\tilde{G}_l([i, n+k, 2n+p+k], i : \tilde{N} - n(l-1))\end{aligned}$$

and $\breve{G}_l(2n+p+k, i) = 0$. It turns out that

$$\tilde{c}_{lik} = \frac{\tilde{G}_l(i,i)}{\sqrt{\tilde{G}_l^2(i,i) - \tilde{G}_l^2(2n+p+k,i)}} = \frac{\tilde{G}_l(i,i)}{\hat{G}_l(i,i)},$$

$$\tilde{s}_{lik} = \frac{\hat{G}_l(2n+p+k,i)}{\sqrt{\tilde{G}_l^2(i,i) - \tilde{G}_l^2(2n+p+k,i)}} = \frac{\hat{G}_l(2n+p+k,i)}{\hat{G}_l(i,i)},$$

and

$$\breve{G}_l(n+k,i) = \hat{G}_l(n+k,i).$$

Moreover, taking (13) into account,

$$\breve{G}_l(i, np+1 : n^2 - n(l-1)) = \hat{G}_l(i, np+1 : n^2 - n(l-1)),$$

$$\begin{aligned}&\breve{G}_l(2n+p+k, np+1 : n^2 - n(l-1)) \\ &\quad = c_{lik}\hat{G}_l(n+k, np+1 : n^2 - n(l-1)).\end{aligned}$$

and

$$\begin{aligned}&\breve{G}_l(2n+p+k, np+i+1 : n^2 - n(l-1)) \\ &\quad = \tilde{G}_l(n+k, np+i+1 : n^2 - n(l-1)).\end{aligned} \tag{14}$$

Hence, at the end of the kth step, the entries in the ith generator from $np+1$ up to $n^2 - n(l-1)$ are still equal to zero. Moreover, we observe that it is not necessary to update the whole vectors in (14). In fact, at the end of each iteration $l, l = 1, \ldots, n$, the first n positive generators, according to GSA, are computed by first multiplying them to the right by $Z(n(l-1)+1 : \tilde{N}, n(l-1)+1 : \tilde{N})$ (the effect of this multiplication is to displace the first np entries of these generators n positions to the right), and then removing the first n columns of the whole generator matrix. At iteration $l+1$, the first n positive and the first n negative generators have the entries from $np+1$ up to $n^2 - ln$ equal to zero. Hence, to proceed it is necessary only to update the entries of the generators (14) corresponding to the nonzero entries of the first n positive and the first n negative generators and store the product of the coefficients c_{lik} into a temporary variable in order to dynamically update the latter generators at each iteration.

Therefore, applying the above mentioned technique, Step *l*.2 can be accomplished in $O(n^2p + np^3)$ flops and the computational complexity of the first n iterations of GSA is reduced to $O(n^4)$.

The implementation of GSA is weakly stable [16] if each of the hyperbolic rotations is performed in a stable way [1] and only stabilized hyperbolic Householder transformations [20] are considered.

The solution of (6) is computed solving the following triangular linear systems

$$L\hat{z} = P_2 M^{\mathrm{T}} v,$$
$$L^{\mathrm{T}}(P_2 z) = \hat{z}$$

with $O(n^3 + n^2p^2)$ flops, since only the first $n + p$ subdiagonals and the last p^2 rows of L are different from zero.

The `matlab` files of the RSTLS algorithm, with the fast implementation of the GSA described in this section, can be obtained from the authors upon request.

4. Numerical results

We demonstrate the accuracy of the RSTLS approach on two image deblurring problems. In our implementation, we stop the RSTLS algorithm after 200 iterations, or sooner if

$$\|[\Delta\alpha^{\mathrm{T}} \Delta x^{\mathrm{T}}]\|_2 < 0.01.$$

Moreover, the reduction of the computational complexity by means of the fast implementation of RSTLS is shown in Example 3.

Example 1. In this example a 20×20 image is considered. Fig. 2 shows the original (a) and blurred (b) images. The point-spread function was Gaussian with $p = 5$. The noise added to each element of the blurred image and the point spread function was normally distributed with mean zero and standard deviation $\sigma = 0.02$. In Fig. 2 the images reconstructed by using the STLS algorithm (c) and the RSTLS algorithm, with regularization parameter $\lambda = 0.18$ (d) are depicted. The STLS algorithm converges within 22 iterations, the RSTLS algorithm within 29 iterations.

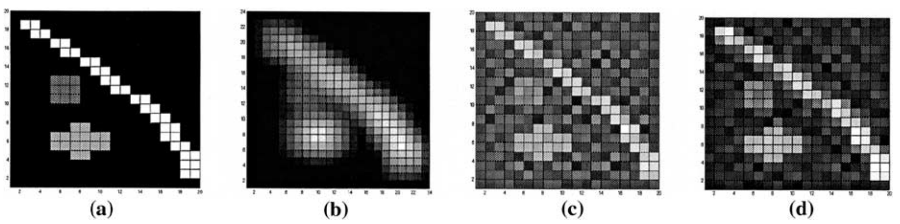

Fig. 2. (a) True 20×20 image. (b) Blurred image with $p = 5$. (c) Reconstructed image by the STLS algorithm. (d) Reconstructed image by the RSTLS algorithm with $\lambda = 0.18$.

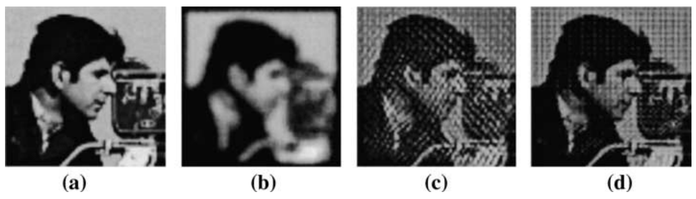

Fig. 3. (a) True 64×64 image. (b) Blurred image with $p = 5$. (c) Reconstructed image by the STLS algorithm. (d) Reconstructed image by the RSTLS algorithm with $\lambda = 0.175$.

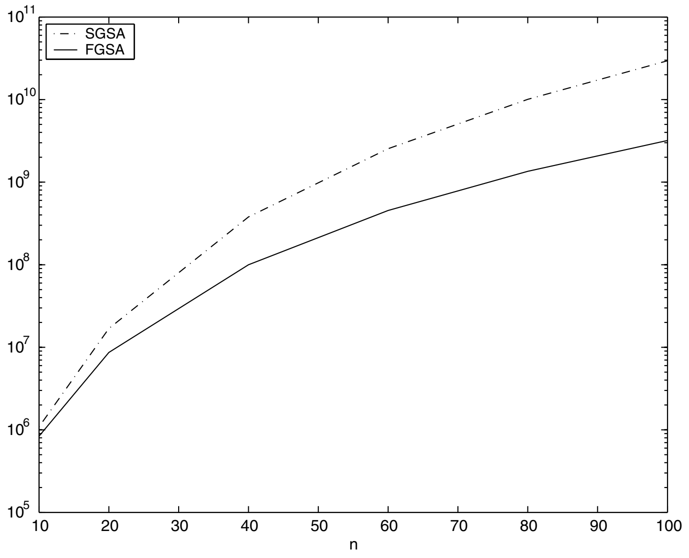

Fig. 4. Plot of the flops required to compute the R factor of the QR factorization of M for $p = 5$ and for different values of n by the fast implementation of GSA (continuous line) and the implementation of GSA without exploiting the structure of the generators (dashed line) in logarithmic scale.

Example 2. In this example a 64×64 image is considered. Fig. 3 shows the original (a) and blurred (b) images. The point-spread function was Gaussian with $p = 5$. The noise added to each element of the blurred image and the point spread function was normally distributed with mean zero and standard deviation $\sigma = 0.12$. In Fig. 3 the images reconstructed by using the STLS algorithm (c) and the RSTLS algorithm, with regularization paramenter $\lambda = 0.175$ (d) are depicted. For this image, STLS algorithm requires 82 iterations to converge, RSTLS algorithm requires 167 iteration to converge.

Example 3. In this example we show the reduction of the computational complexity by means of the fast implementation of RSTLS algorithm described in this paper.

The number of flops needed to compute the R factor of the QR factorization of the matrix M, by the GSA algorithm implemented without the exploitation of the structure of the generators (SGSA), and the algorithm described in the paper, the GSA algorithm implemented exploiting the structure of the generators (FGSA), for different values of n, is depicted in Fig. 4. The value of p is equal to 5. These simulations were performed in Matlab 5.3.

5. Conclusions

We have shown that image deblurring problems can be formulated as a Structured Total Least Squares problem. Since the involved matrices are often ill-conditioned, regularization is needed to have meaningful results. An iterative algorithm for the Regularized Structured Total Least Squares problem has been designed that requires a solution of a structured least squares problem at each iteration. A fast implementation of the generalized Schur algorithm has been proposed for solving the latter problem. Moreover, the accuracy of the RSTLS estimator with respect to the STLS estimator is shown by means of two different image deblurring problems.

References

[1] A.W. Bojanczyk, R.P. Brent, P. Van Dooren, F.R. de Hoog, A note on downdating the Cholesky factorization, SIAM J. Sci. Statist. Comput. 8 (1987) 210–220.

[2] A.W. Bojanczyk, A.O. Steinhardt, Stabilized hyperbolic householder transformations, IEEE Trans. Acoustics Speech Signal Process. ASSP-37 (1989) 1286–1288.

[3] H. Fu, J.L. Barlow, A regularized structured total least squares algorithm for high resolution image reconstruction, Linear Algebra Appl. 391 (2004) 75–98.

[4] G.H. Golub, C.F. Van Loan, An analysis of the total least squares problem, SIAM J. Numer. Anal. 17 (1980) 883–893.

[5] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., The John Hopkins University Press, Baltimore, MD, 1996.

[6] R.C. Gonzalez, P. Wintz, Digital Image Processing, second ed., Addison-Wesley, 1987.

[7] P.C. Hansen, Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion, SIAM Monographs on Mathematical Modeling and Computation, SIAM, Philadelphia, 1998.

[8] P.C. Hansen, Deconvolution and regularization with Toeplitz matrices, Numerical Algorithms 29 (2002) 323–378.

[9] T. Kailath, Displacement structure and array algorithms, in: T. Kailath, A.H. Sayed (Eds.), Fast Reliable Algorithms for Matrices with Structure, SIAM, Philadelphia, 1999, pp. 1–56.

[10] A. Kalsi, D.P. O'Leary, Algorithms for structured total least squares problems with applications to blind image deblurring, CS-TR-4390, University of Maryland, August 2002.

[11] R.L. Lagendijk, J. Biemond, Iterative Identification and Restoration of Images, Kluwer Academic Publishers, Boston, 1991.

[12] P. Lemmerling, S. Van Huffel, B. De Moor, Structured total least squares problems: formulations, algorithms and applications, in: S. Van Huffel (Ed.), Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling, SIAM, Philadelphia, 1997, pp. 215–223.

[13] N. Mastronardi, Fast and reliable algorithms for structured total least squares and related matrix problems, PhD thesis, Faculty of Engineering, K. U. Leuven, Leuven, Belgium, May 2001.

[14] N. Mastronardi, P. Lemmerling, S. Van Huffel, Fast structured total least squares algorithm for solving the basic deconvolution problem, SIAM J. Matrix Anal. Appl. 2 (22) (2000) 533–553.

[15] N. Mastronardi, P. Lemmerling, S. Van Huffel, Fast regularized structured total least squares algorithm for solving the basic deconvolution problem, Internal Report 02-162, ESAT-SISTA, K. U. Leuven, Leuven, Belgium, 2002. Accepted for publication in Numerical Linear Algebra with Applications.

[16] N. Mastronardi, P. Van Dooren, S. Van Huffel, On the stability of the generalized Schur algorithm, Lecture Notes in Computer Science 1988 (2001) 560–567.

[17] M.K. Ng, R.J. Plemmons, F.A. Pimentel, A new approach to constrained total least squares image restoration, Linear Algebra Appl. 316 (2000) 237–258.

[18] A. Pruessner, D.P. O'Leary, Blind deconvolution using a regularized structured total least norm approach, SIAM J. Matrix Anal. Appl. 24 (2003) 1018–1037.

[19] C.M. Rader, A.O. Steinhardt, Hyperbolic Householder transformations, SIAM J. Matrix Anal. Appl. 9 (1988) 269–290.

[20] C.M. Rader, A.O. Steinhardt, Hyperbolic householder transformations, IEEE Trans. Acoustics Speech Signal Process. ASSP-34 (1986) 1584–1602.

[21] J.B. Rosen, H. Park, J. Click, Total least norm formulation and solution for structured problems, SIAM J. Matrix Anal. Appl. 17 (1996) 110–126.

[22] S. Van Huffel, J. Vandewalle, The total least squares problem: computational aspects and analysis, Frontiers in Applied Mathematics Series, vol. 9, SIAM, Philadelphia, 1991.

[23] S. Van Huffel, H. Park, J.B. Rosen, Formulation and solution of structured total least norm problems for parameter estimation, IEEE Trans. Signal Process. 44 (1996) 2464–2474.