Updating preconditioners for nonlinear deblurring and denoising image restoration[☆]

Daniele Bertaccini[a,*], Fiorella Sgallari[b]

[a] *Università di Roma "Tor Vergata", Dipartimento di Matematica and Centro Interdipartimentale per la Biostatistica e la Bioinformatica , viale della Ricerca Scientifica, 00133 Roma, Italy.*
[b] *Università di Bologna, Dipartimento di Matematica and CIRAM, Via Saragozza, 8, 40123 Bologna, Italy.*

Abstract

Image restoration is the recovery of images that have been degraded by blur and noise. Nonlinear degenerate diffusion partial differential equation models for image restoration requires often the solution of a challenging discrete problem. We consider solving the related discrete models in the time-scale steps by Krylov iterative solvers accelerated by updating a preconditioner based on incomplete factorizations which presents a global computational cost slightly more than linear in the number of the image pixels. We demonstrate the efficiency of the strategy by denoising and deblurring some images with a generalized Alvarez-Lions-Morel-like partial differential equation model discretized by a semi implicit complementary volume scheme.

Keywords: preconditioners, image restoration, PDE models, iterative solvers
2000 MSC: 65M22, 65F08, 65F10, 68U10

1. Introduction

Image restoration is the process which aims to recover the original image from noisy and blurred data. Often, a degraded observed image u^0 of a reference image u is considered coming from some physical process decribed by a linear model

$$u^0 = K * u + n,$$

where K is a known blurring operator and n is the noise. However, the setting proposed here does not need any assumption on the blur (symmetry/spatially invariant or even not necessarily coming from a convolution). In recent years, many models for noise removal and deblurring have been proposed and studied. Models based on partial differential equations, or PDE for short, can be discretized by finite differences, finite elements or finite volumes by using explicit (see, e.g., [16], semi implicit (see, e.g., [13], [12]) and fully implicit (see, e.g., [21]) time stepping in order to obtain a discrete model. If the model is for denoising only, its discrete counterpart can be solved by using an *additive operator splitting* technique (see [14], [22]); then it is hard to solve it faster because of the optimality of the technique in terms of the computational complexity (linear in the number of the image pixels). However, in presence of a blur operator, one should resort to alternative strategies, provided efficient enough. Unfortunately, it is not straightforward to design efficient and robust preconditioners to solve linear systems generated by discretizing PDE image restoration models. In particular, two families among the most popular, algebraic multigrids and incomplete factorization techniques, does not give always the desired answers. The first ones are very efficient whenever the model has linear and isotropic diffusion while it is less obvious how to get robustness and convergence with a global computational cost of $O(N)$, where N is the number of the image pixels; see [8] and implementation plays a nontrivial role in the

[☆] Work partially supported by PRIN, grant 20083KLJEZ.
[*] Corresponding author
Email addresses: bertaccini@mat.uniroma2.it (Daniele Bertaccini), sgallari@dm.unibo.it (Fiorella Sgallari)
URL: http://www.mat.uniroma2.it/bertaccini (Daniele Bertaccini), http://www.ciram.unibo.it/~sgallari (Fiorella Sgallari)

game. The second ones can be computationally expensive whenever the blur operator enlarges the bandwidth of the linear systems generated by the discretized models. Moreover, are often difficult to parallelize, because incomplete factorization techniques require solving triangular linear systems. On the other hand, sparse approximate inverse preconditioners are usually much more suitable for parallel environments; see, e.g., [18]. It should be noticed that today some degree of parallelism capabilities is present even in our desktop or laptop PCs; see, e.g., AMD or Intel multicore architectures and software supports it, often in a way that is transparent to the user, see, e.g., Matlab and more recent Intel Math Kernel Library (Intel MKL) etc.

In view of these facts, we analyze the impact of a technique updating sparse approximate inverse factorizations in image restoration with a theoretical computational cost, both in time and space, linear in the number of the N image pixels and $O(N \log N)$ globally, the same asymptotic cost of a very successful multilevel solver; see [7]. Moreover, we apply the underlying approach to two efficient sparse approximate inverse preconditioning techniques, AINV (see [4]) and an inverse ILU technique (see, e.g., [18] and [20]).

The underlying technique is suitable for parallel computation whenever used for updating a factorized sparse approximate inverse preconditioner.

Preconditioners for solving sequences of shifted linear systems arising in partial differential equations of evolutionary type were proposed in [15], [3] and [5]. Here, we generalize the preconditioners introduced in [3] and [5] in order to update incomplete factorizations for a generic sequence of symmetric positive definite matrices by using band updates. We apply the underlying inverse updated incomplete factorizations to the sequences of linear systems generated in a semi implicit complementary volume discretization of an Alvarez-Lions-Morel-like model for selective smoothing and deblurring (see [1] and [16]) analyzing performance and effects of its use in comparison with recomputing and freezing incomplete Cholesky preconditioners with different thresholds at each step. The efficiency of the proposed framework is shown by denoising and deblurring some images.

The proposed ideas can be easily implemented and adapted to other incomplete factorizations that can be given in the form LDL^T. We are investigating their use with other image restoration models like, e.g., total variation. Moreover, our algorithms do not need any delicate parameters settings.

In Section 2 we briefly describe the model, discretized with a semi implicit complementary volume scheme. Section 3 describes how to update approximate factorization preconditioners for the discrete operators in Section 2. A theoretical analysis of the quality of the approximation, of the computational complexity and convergence with some remarks can be found in Section 4. Section 5 includes tests on images with noise and blur and comparisons. Finally, in Section 6, some conclusions and works in progress.

2. Nonlinear selective smoothing

We consider two variants of an Alvarez-Lions-Morel-like nonlinear model for selective smoothing (see [1]) generalized as in [16] for deblurring

$$u_t = |\nabla u|_\epsilon \nabla \cdot \left(g(|\nabla G_\sigma * u|) \frac{\nabla u}{|\nabla u|_\epsilon} \right) - \alpha |\nabla u|_\epsilon K^T \left(K\,u - u^0 \right), \quad (t,x) \in I \times \Omega, \tag{1}$$

and

$$u_t = \nabla \cdot (g(|\nabla G_\sigma * u|) \nabla u) - \alpha K^T \left(K\,u - u^0 \right), \quad (t,x) \in I \times \Omega, \tag{2}$$

where K is the blur operator, α is a parameter controlling the blur term, Ω can be assumed as a bounded rectangular domain, I is a scale (time) interval, g is a nonincreasing real function which tends to zero as its argument tends to infinity, G is a smoothing kernel and $|\nabla \cdot|_\epsilon$ means that we regularize in the sense of Evans and Spruck [11] to avoid zero in the denominators

$$|\nabla u|_\epsilon = \sqrt{|\nabla u|^2 + \epsilon^2}. \tag{3}$$

The equation (1) is coupled with initial and boundary conditions

$$\begin{aligned} \frac{\partial u}{\partial \mathbf{n}} &= 0, \quad (t,x) \in I \times \partial\Omega \\ u(0,x) &= u^0(x), \quad x \in \Omega \end{aligned} \tag{4}$$

and can be discretized in "time"by an implicit or semi-implicit scale-step integrator. As usual, $u(t,x)$ is the unknown image function, u^0 is the grey level of the image to be processed, $\mathbf{n}$ is the unit normal to the boundary of Ω.

2.1. Semi implicit complementary volume scheme

The linear semi implicit fully discrete complementary volume scheme for equation (1), proposed in [12] without the data fidelity term (i.e., $\alpha = 0$ in (1)), is generated by the discretization of (1) with Neumann boundary conditions, choosing a uniform discrete time-scale increment τ, and replacing the time-scale derivative by backward differences. To simplify the exposition, we concentrate on the model (1) in this section. Similar arguments can be used for the variant (2). In order to generate a semi implicit scheme, nonlinear terms are computed on the basis of the previous scale step while linear terms on the current one:

$$\frac{1}{|\nabla u^{n-1}|_\epsilon}\frac{u^n - u^{n-1}}{\tau} - \nabla \cdot \left(g^{n-1}\frac{\nabla u^n}{|\nabla u^{n-1}|_\epsilon}\right) = 0 \tag{5}$$

where

$$g^{n-1} = g\left(|\nabla G_\sigma * u^{n-1}|\right),$$

$g^{n-1} > 0$ for the smoothing properties of convolution [6]. A 2D digital image is given on a structure of pixels with rectangular shape, in general. In the complementary volume method, approximation of solution is assumed to be a piecewise linear function. The values of discrete image intensity are considered as approximations of continuous image intensity function in centers of pixels. These centers of pixels will correspond to the nodes of a triangulation. We can get such triangulation simply by connecting the centers of pixels by new rectangular mesh and then dividing every rectangle into two triangles. In complementary volume method, together with the triangulation also the so-called dual mesh is used. The dual mesh consists of cells V_i (called also complementary volume, control volumes or co-volumes) associated with the ith node, $i = 1, ..., N$, of the given triangulation T_h. The co-volume V_i is bounded by the lines that bisect and are perpendicular to the edges emanating from the node. Let us note, that in image processing the dual mesh again corresponds to pixel structure, see [12], [9] for more details. In order to derive complementary volume spatial discretization we integrate (5) over a co-volume V_i. Using the divergence theorem, we get the discrete system to be solved for u_i^n:

$$u_i^n = u_i^{n-1} + \frac{\tau}{b_i^{n-1}}\sum_{j\in C_i} a_{i,j}^{n-1}\left(u_j^n - u_i^n\right), \quad i = 1, ..., N, \quad n = 1, 2, ..., \tag{6}$$

where N are the nodes of the triangulation and

$$b_i^{n-1} = \frac{|V_i|}{g(|\nabla G_\sigma * u^{n-1}|_i)|\nabla u_i^{n-1}|_\epsilon},$$

$$a_{i,j}^{n-1} = \frac{1}{h_{i,j}}\sum_{T\in\varepsilon_{i,j}}\frac{|c_{i,j}|}{|\nabla u_T^{n-1}|_\epsilon}, \tag{7}$$

where ∇u_T^{n-1} is related to the value assumed by the gradient of the piecewise linear function which is constant on every simplex T of the triangulation while $|c_{i,j}|$ is the length of the portion of the co-edge that is the perpendicular bisector of the edge $\sigma_{i,j}$ connecting the nodes i and j and the length of $\sigma_{i,j}$ is $h_{i,j}$; see [12, section 2]. By rewriting the discrete system (6) as a linear system $C_n u_n = f_n$ whose solution vector is

$$u_n = (u_1^n, ..., u_N^n)^T,$$

the matrix C_n has as entries $\phi_{i,j}^n$, $i, j = 1, ..., N$, $n = 1, 2, ...$, where

$$\phi_{i,j}^n = b_i^{n-1} + \tau\sum_{j\in C_i} a_{i,j}^{n-1},\ i = j; \quad \phi_{i,j}^n = -\tau a_{i,j}^{n-1},\ i \neq j, \tag{8}$$

we have the result.

Theorem 1. *The matrices of the linear systems generated at each time scale by the discrete complementary volume scheme (6) are symmetric and diagonally dominant M-matrices.*

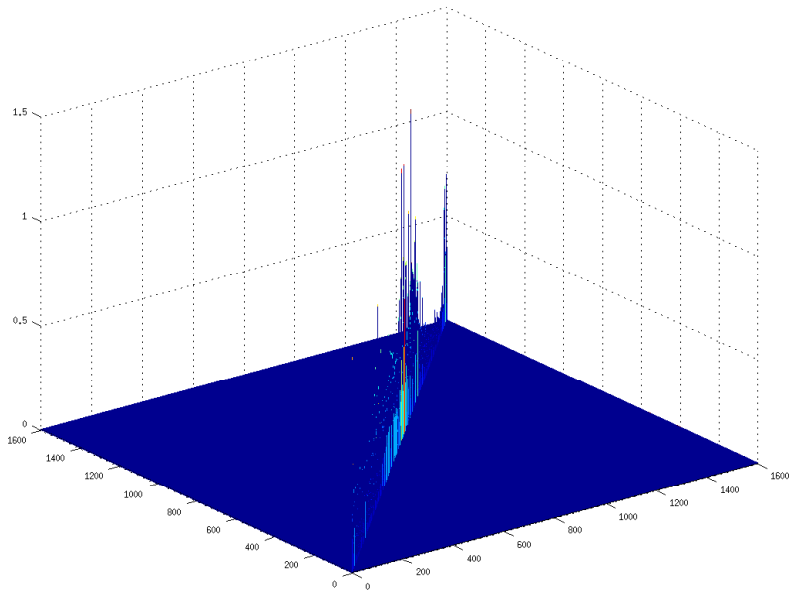

Figure 1: Typical decay of the entries of the inverse of a matrix in $\{C_j\}$ with the blur operator based on the Gaussian kernel described in Section 5, parameter *band* = 5, *sigma* = 3 and 20% gaussian noise.

Proof. See [12, Proposition 1]. □

The above property is a promising starting point for reasonable performances of our updating strategy. Indeed, by [10], we can expect a fast decay of the entries of the inverse of the matrices $\{C_n\}$ (8) by moving away from the main diagonal; see Figure 1.

More on the effect of parameters ϵ, τ, and the number of image pixels on the convergence of iterations can be found in Sections 4.1 and 4.2.

By including the *data fidelity* term

$$\alpha K^T(Ku - u^0) \tag{9}$$

in (6), the discretized model can be rewritten as a sequence of linear system whose matrices have a larger and more dense band with respect to those with $\alpha = 0$ in (1) and (2). The data fidelity term (9) in both variants of the models (1), (2) is treated implicitly and it is part of the reaction term of these equations.

We stress that the contribution of (9) to the models can in principle destroy the diagonal dominance of the matrices generated by the semi implicit complementary volume discretization. The contribution of the data fidelity term to the matrix of the underlying linear algebraic systems can be viewed as adding the matrix $B = \tau\alpha K^T K$ (nonnegative definite) to the positive definite, diagonally dominant matrix C_n given by the regularization term; see Theorem 1. On the other hand, we experienced that the correct values of the step τ (from 10^{-2} to 10^{-5}, say) for all real image restoration tested are relatively small and therefore they force the sum $C_n = C_n + B$ to remain diagonally dominant.

3. Updating (incomplete) factorizations

In order to describe the proposed updated preconditioners, we begin with an outline of the basic ideas and then how we apply them, a basic algorithm, some convergence results and notes. Finally, the computational complexity is discussed.

3.1. Motivations

Let us denote the sequence of large linear algebraic systems that we need to solve as

$$C_n u_n = f_n, \quad n = 1, 2, ..., \tag{10}$$

where C_n is the n-th matrix that can be generated by either the quasi-Newton step of an implicit solver as, e.g., in [21] or a semi-implicit formulation of the discretization in time as, e.g., in [1], [9], [12].

We assume C_1 as the reference matrix $C_{reference}$ and suppose we can compute the LDL^T factorization of C_1, i.e.

$$C_{reference} = C_1 = L\,D\,L^T, \tag{11}$$

where L is unit lower triangular and D is diagonal. If C_1 is symmetric and positive definite, then the factorization (11) exists and the entries of D are positive. It is worth to note that the factorization (11) will *never* be computed in practice, we need to state this only to justify our approximations in the sequel. In the style of [5], by denoting with C_j the matrix we want to precondition, we can look for a formal factorization for C_j, that, assuming C_j factorizable, can be written as

$$C_j = L\,(D + E_j)\,L^T, \tag{12}$$

where E_j can be seen as an update of the diagonal matrix D in (11). By a generalization of an argument from [3] and [5], we get

$$E_j = Z^T\,\Delta_j\,Z, \quad Z = \left(L^T\right)^{-1} \tag{13}$$

where

$$\Delta_j = C_j - C_{reference}, \quad j = 1, 2, \ldots \tag{14}$$

The inverse of C_n can be written as

$$C_n^{-1} = Z\left(D + E_j\right)^{-1} Z^T, \quad Z = \left(L^T\right)^{-1}. \tag{15}$$

To obtain a sparse preconditioner from the formal expression of the inverse of the updated factorization (15), it is necessary to replace the factors Z and the inverse of $D + E_j$ in (15) with sparse approximations that can be computed reliably and efficiently. In view of these requirements, we propose two alternative approaches we found viable: factorized stabilized approximate inverse preconditioners (SAINV) proposed in [4] and the scalable approximate inverse of the LDL^T factorization proposed in, e.g., [20], dropping fill as soon as it occurs, i.e., within a sparse vector update and called here $IILDL^T$ for brevity. Both strategies work fine on shared memory parallel machines such as the multicore architectures and provide an approximate factorization for the inverse of $C_{reference}$ of the form

$$\tilde{Z}\,\tilde{D}^{-1}\,\tilde{Z}^T, \tag{16}$$

where $\tilde{Z}$ is upper triangular and $\tilde{D}$ is diagonal. Therefore, the updated preconditioner, in inverse factorized form, is

$$M_j = \tilde{Z}\left(\tilde{D} + \tilde{E}_j\right)^{-1} \tilde{Z}^T, \tag{17}$$

where the $\tilde{Z}$, $\tilde{D}$ and $\tilde{E}_j$ are the underlying sparse approximations.

The computational cost is expected to be lower using (17) in a machine with multithreading capabilities because no triangular linear systems are needed to be solved in order to apply the inverse preconditioner if, e.g., $\tilde{E}_j$ is a diagonal approximation for the matrix (13) because the matrix $\tilde{D} + \tilde{E}_j$ is diagonal and the matrix-vector product is easily performed in parallel in a shared memory machine. More details and general updates are given in Section 4.3.

3.2. The approximate factorizations in our setting

In order to solve the underlying discretized problems, the sequence of linear systems (6), we use Krylov subspace algorithms with the updated preconditioners. Here are the main steps we perform in order to provide the updated inverse preconditioner (17) in practice.

1. The first step depends on the preconditioning strategy used for $C_{reference}$.
   - By using SAINV, we generate $\tilde{Z}$ and $\tilde{D}$ by incomplete A-orthogonalization of the vectors of the canonical basis of $\mathcal{R}^n$ dropping off-diagonal entries below a prescribed tolerance during the orthogonalization process; see [4]. The input matrix $C_{reference}$ is symmetrically scaled so that it has unit diagonal; this tends to improve the conditioning of the matrix and it allows for the use of an absolute drop tolerance.

- By using the approximate inverse $IILDL^T$ as the preconditioner for $C_{reference}$, we generate the $\tilde{L}\tilde{D}\tilde{L}^T$ incomplete factorization of $C_{reference}$ by a threshold-Cholesky algorithm (see, e.g., [18, chapter 10] for a review). We compute a sparsified version of $Z = \left(L^T\right)^{-1}$ by inverting the sparse unit lower triangular matrix $\tilde{L}^T$ dropping fill as soon as it occurs, see [20, Algorithm 3]. The input matrix $C_{reference}$ is scaled as for SAINV. The overall cost of the procedure is similar to the cost of SAINV; see Section 4.3 for more details.

Note that in general, there is no guarantee to get a sparse matrix by inverting a sparse upper triangular L^T, it can be upper triangular but full. However, under some hypotheses on the sequence of the matrices to be approximated by the preconditioners, (for example, if the matrices generated by (6) are diagonally dominant) we get a fast decay of the entries away from the main diagonal for C_n^{-1} and then the (exact) matrix $Z = \left(L^T\right)^{-1}$ will have the largest entries in modulus near to the main diagonal; see [3, Theorem 4.2].

2. The perturbation $\tilde{E}_j$ that we add to $\tilde{D}$ of the reference incomplete factorization can be an approximation of $E_j = Z^T \Delta_j Z$ that could in principle be given by $\tilde{E}_j = \tilde{Z}^T \Delta_j \tilde{Z}$ itself. However, to apply the updated preconditioner, we need to solve a linear system with matrix $\tilde{D} + \tilde{E}_j$ even if we use the preconditioner in the inverse factorized form (17). Note that the decay of the entries in Z is fast enough in all examples where the matrices C_n are diagonally dominant (in all tests we consider, reported and not). A typical behavior for the entries of the inverse is shown in Figure 1. Therefore, by taking $\tilde{E}_j$ as a *diagonal* or a tight banded approximation of $\tilde{Z}^T \Delta_j \tilde{Z}$ (or of $\tilde{Z}^T \tilde{\Delta}_j \tilde{Z}$ with $\tilde{\Delta}_j$ approximating Δ_j; in our tests we take $\tilde{\Delta}_j = \Delta_j$) can give a potentially effective and very cheap approximation. This is what we do here in the numerical experiments with good results; see Section 5.
3. We used absolute drop tolerances for both SAINV and IILDL^T (in the ILDL^T and during the inversion of triangular factor phase for the latter) experiencing that 0.1 is satisfactory in order to avoid too much fill-in and fast convergence of the updates; see tables in Section 5. The same tolerance has been found confortable for the incomplete Cholesky freezed and recomputed preconditioners too.
4. The inclusion of the *data fidelity* term requires some care. In particular, the reference approximate factorization of the discrete operator including both the data fidelity and the regularization terms can be performed matrix-free by using AINV algorithm that needs to perform only matrix-vector products with the underlying matrix; see [4]. We remark that our approach can manage both spatially variant and spatially invariant blur.
5. The updates of the preconditioner is performed matrix-free for the *data fidelity* term because we note that with a blur operator K in the models (1), (2), in order to generate the updated approximate factorization M_j, we do not need to explicitly form the difference of $C_j - C_{reference} = \Delta_j$. Indeed, we have

$$\Delta_j = C_j - C_{reference} = C_j - C_{reference}, \tag{18}$$

where C_j and $C_{reference}$ are the discretized regularization operators at step j and zero, i.e., *not* including the data fidelity term (9) and therefore are band matrices sparser than C_j, $C_{reference}$, respectively.

We experienced that zero fill-in incomplete Cholesky factorizations (i.e., computed by discarding entries of the factors that are zero in the originating matrix) are not appropriate here because the quality of the approximations is low and the results are not satisfactory.

3.3. The algorithm

Here is a simplified version based on AINV of the algorithm using diagonal updates and $\Delta_j = C_j - C_{reference}$. This include a matrix-free treatment of the data fidelity term (9) and the update. Some parameters such as those concerning the stopping criteria for conjugate gradient are not included for brevity. We remark that `pcg` stands for *preconditioned conjugate gradients*.

1. Input u^0, K, α, σ, τ (ϵ), droptol, maxstep
2. Compute u^1 by (6) and (9). Store $C_{reference}$ (the discretization of the regularization term, does not include $K^T K$ matrix)
3. Apply AINV to $C_{reference} + \alpha\tau K^T K$ with drop tolerance droptol
4. Store $\tilde{Z}$ and $\tilde{D}$, set $M_0 = \tilde{Z}\tilde{D}^{-1}\tilde{Z}^T$ [1].

[1] The *reference* approximate factorization (16).

5. For j=1,...,maxstep
 5.1 Compute C_j and $\Delta_j = C_j - C_{reference}$ (see (18) above)
 5.2 Compute $\tilde{E}_j = \mathrm{diag}(\tilde{Z}^T \Delta_j \tilde{Z})$
 5.3 If $\tilde{D} + \tilde{E}_j$ is nonsingular
  5.3.1 Update $M_j = \tilde{Z}(\tilde{D} + \tilde{E}_j)^{-1}\tilde{Z}^T$
 5.4 Else $M_j = M_0$
 5.5 End if
 5.6 Solve with pcg $(C_j + \alpha\tau K^T K)u^{j+1} = b + \alpha\tau K^T u^0$ using M_j
6. End for

4. Analysis of the proposed methods

Let us state some formal results for our preconditioners that can help their use with the models (1) and (2) in practice. We remark that the proposed discussions can in principle be extended to different image restoration models like *total variation*.

4.1. Convergence of updated preconditioned iterations

The bounds in the statements below are given mostly in order to understand the effect of the updates on the reference preconditioner. In practice, they could require some restrictions of the parameters τ and tolerances of the incomplete factorizations. It is intended that the matrices in the sequence $\{C_j\}$ are generated by a time-scale discretization process that depends on a scale parameter τ such that

$$\lim_{\tau \to 0} C_j = C_{reference}, \quad j = 1, 2, ...$$

Let us introduce the operator $\mathcal{A}_k(\cdot)$ approximating its matrix argument by extracting $k \geq 1$ upper and lower main diagonals. Note that $\mathcal{A}(0) = 0$, where 0 is the null matrix. $P_{reference}$ is the preconditioner computed for $C_{reference}$ ($M_{reference}$ in inverse form), P_j is the preconditioner for C_j (M_j in inverse form).

Theorem 2. *Let $C_{reference}$ be symmetric and positive definite such that an incomplete factorization $\tilde{L}\tilde{D}\tilde{L}^T$ for $C_{reference}$ is well defined, i.e., there exist δ_L, $\delta_D > 0$ and $\zeta > 0$ of moderate size such that $\|\tilde{L} - L\| \leq \delta_L$, $\|\tilde{D} - D\| \leq \delta_D$ $\|\tilde{Z}\| \leq \zeta$, $\tilde{E}_j = \mathcal{A}_k\left(\tilde{Z}^T \tilde{\Delta}_j \tilde{Z}\right)$ with $k \geq 1$. Then, there exists $\delta_{\max} > 0$ such that for all matrices of the sequence obeying*

$$\|C_j - C_{reference}\| \leq \delta_{\max}, \quad j = 1, 2, ...,$$

the updates of the reference preconditioner (17) are well defined and the spectrum of $P_j^{-1}C_j$ (M_jC_j), the matrix preconditioned by updating $P_{reference}$ given by

$$P_j^{-1}C_j = I + P_j^{-1}R_j \quad (M_jC_j = I + M_jR_j), \quad R_j = C_j - P_j, \tag{19}$$

has a cluster of eigenvalues in 1*, and*

$$\|P_j^{-1}\|_2 \leq \frac{\zeta^2}{l_{\min}} \quad (\|M_j\|_2 \leq \frac{\zeta^2}{l_{\min}}), \quad 0 < l_{\min} \leq \lambda_{\min}(\tilde{D} + \tilde{E}_j),$$

$$\|R_j\| \leq O(\delta_L + \delta_D) + c_k\delta_{\max}. \tag{20}$$

and c_k is a constant dependent on the norm used and on the operator $\mathcal{A}_k(\cdot)$.

Proof. Let us write $P_j^{-1}R_j$ (M_jR_j) as

$$P_j^{-1}R_j = M_j^{(1)} + M_j^{(2)} \quad (M_jR_j = M_j^{(1)} + M_j^{(2)}),$$

where $M_j^{(1)}$ is of small rank (with respect to the size of the underlying matrices) and $\|M_j^{(2)}\|$ has small norm in order to get clustering of the preconditioned spectrum, in particular we can require that $\|M_j^{(2)}\| < 1$. Then, in order to get a cluster of eigenvalues in 1 for (19), let us observe that

$$R_j = \left(C_{reference} - \tilde{L}\tilde{D}\tilde{L}^T\right) + \left(\Delta_j - \tilde{L}\,\mathcal{A}_k\left(\tilde{Z}^T\tilde{\Delta}_j\tilde{Z}\right)\tilde{L}^T\right). \tag{21}$$

and, by

$$\tilde{E}_j = \mathcal{A}_k\left(\tilde{Z}^T\tilde{\Delta}_j\tilde{Z}\right)$$

we get

$$\|\tilde{E}_j\| \le \delta_{\max}\|\tilde{Z}\|^2.$$

Moreover, from the expression of the updated preconditioner, we get

$$\|P_j^{-1}\| \le \|\tilde{Z}\|^2\left(\lambda_{\min}(\tilde{D} + \tilde{E}_j)\right)^{-1} \quad \left(\|M_j\| \le \|\tilde{Z}\|^2\left(\lambda_{\min}(\tilde{D} + \tilde{E}_j)\right)^{-1}\right),$$

and, in order to get a well defined preconditioner, $\lambda_{\min}(\tilde{D} + \tilde{E}_j)$ should be such that $\lambda_{\min}(\tilde{D} + \tilde{E}_j) \ge l_{\min} > 0$ ($l_{\min} > 0$ can be increased by reducing $\delta_{\max}$ by acting on τ).

If k is such that

$$\mathcal{A}_k\left(\tilde{Z}^T\Delta_j\tilde{Z}\right)\tilde{L}^T = \tilde{Z}^T\tilde{\Delta}_j\tilde{Z}, \tag{22}$$

and $\tilde{Z} = \tilde{L}^{-T}$, then we get that the variable part (with respect to j) in (21) is given by

$$\|\Delta_j - \tilde{\Delta}_j\| \le \|\Delta_j\| + \|\tilde{\Delta}_j\| \le 2\|\Delta_j\|,$$

because $\|\tilde{\Delta}_j\| \le \|\Delta_j\|$ and $c_k = 1$ in (20). On the other hand, if $\tilde{Z}$ is a sparsified approximation for $\tilde{L}^{-T}$, $\mathcal{A}_k$ is not trivial and (22) does not hold, we get that the result (20) holds with $c_k \ne 1$. □

Some comments on Theorem 2.

- We note that $\|M_j^{(2)}\|$ can be made arbitrarily small by restricting the maximum admissible value of $\delta_{\max}$ because $\delta_{\max} = O(\tau)$ and therefore by acting on the scale step.
- Same results hold if $C_{reference}$ is a small rank perturbation of a symmetric matrix due to, e.g., Neumann boundary conditions. In this case, the incomplete factorization can be computed from a symmetrization of $C_{reference}$.
- Same arguments apply for a right or split left-right preconditioning.

4.2. Convergence results and the model parameters

The parameters of the PDE model (1) and the original image itself (size, presence of sharp edges or not, etc.) influence the characteristics of the discretized model and then the convergence.

- We have that $\|C_j - C_{j-1}\| = O(\tau)$, therefore by decreasing τ, $\delta_{\max}$ decreases[2]. This does not mean that the proposed algorithms are suitable for small scale step only. On the other side, our experience suggests that realistic choices of τ usually provide good performances.
- We never experienced indefinite or singular updated preconditioners for all test examples we performed, an issue that can occur updating a generic sequence of matrices.
- The condition number of the sequence of unpreconditioned (or preconditioned with the same preconditioner computed in the first step) matrices $\{C_j\}$ is nondecreasing with j in the considered PDE model; this is related to the smoothing effect that increases with the scale steps. Therefore, the number of nonpreconditioned iterations usually increases with j, the scale step index. On the other hand, the number of updated preconditioned iterations seems to be almost insensitive to the number of scale steps, i.e., remains almost constant with j.

[2]The choice of the scale step parameter and the number of steps are a delicate matter not discussed here that depends on the given image and the entity of the selective smoothness induced on the image.

- The number of the scale steps is provided by our package automatically in order to maximimize the quality of the image restoration; see Section 5 for details.

- Resolution or the number of pixels in the image. For simplicity, let us consider it square, i.e., a matrix of $m \times m$ pixels and $\alpha = 0$ in (1). We observe that the condition number of the discrete operator C_j is increases at most quadratically with m. If the function $g(\cdot)$ does not vary too much and its value is nonzero, then the general PDE model (1) shows the typical character of the discrete diffusion operator, i.e., a condition number of the order of $O(1/h^2)$, and therefore of $O(m^2)$, linear with the number of image pixels, because $h = 1/m$. In general, we can expect an unpreconditioned condition number that increases with m more than linearly. On the other hand, updated preconditioned discrete operators does not show this issue and we observe a number of updated preconditioned iterations not increasing or only slightly with m.

- The parameter ϵ regularizing gradients in the model (1) to avoid zero in the denominators is expected to influence the condition number of the discrete operators C_j like $O(1/\epsilon)$. Indeed, in case of small or zero gradients we observe from (7) that $\min_r\{\lambda_r(C_j)\} \geq O(1/\epsilon)$, $r = 1, ..., m^2$, $j = 1, 2, ...$

We recall that the convergence is expected fast provided that there is a tight cluster of eigenvalues far away from zero or the condition number of the (preconditioned) matrix is not large; see [2] and [18, Chapter 6] for details. Therefore, the notes above with Theorem 2 can help in order to explain qualitatively the performances of nonpreconditioned and preconditioned iterations, with updated or recomputed factorizations. However, the above considerations does not consider that we do not start iterations from an arbitrary guess but by using a valuable and precious information: the restored image of the previous step $j - 1$.

Finally, a brief discussion on breakdowns in updating preconditioners for sequence of general matrices, an important issue that was not considered in other recent works; see, e.g., [19]. Two main types of breakdowns can destroy the effectiveness of updated factorizations (approximate or not), in general:

- *severe breakdown*, the update is numerically singular or very ill-conditioned;

- *consistency breakdown*, the update is indefinite whenever the matrix C_j is definite.

The matrix-free algorithms for updating preconditioners recently proposed in [3] and [5] does not suffer from these shortcomings because of the hypothesis on the sequence of matrices C_j considered there ($C_j = C_{reference} + D_j$, D_j multiple of the identity matrix or diagonal with positive real part in [3], [5]). On the other hand, if the updates (for general sequences) $\tilde{E}_j$ are diagonal matrices, it is easy to control the quality of the updates by checking diagonal entries in $\tilde{E}_j$. If one of the two breakdowns listed above occurs for a few or some diagonal updates, we provide three choices: (a) regularizing the update, for example by setting to $\theta > 0$ the negative diagonal entries of the update; (b) recalculating the incomplete factorization from scratch, i.e. setting a new $C_{reference}$ or (c) using the previous update or even the preconditioner computed for $C_{reference}$. The choice (a) seems to be the best compromise between computational cost and performances. However, in this image restoration setting, we *never* experienced any breakdown or indefinite updates.

4.3. Computational complexity

Let us consider computational complexity *in space* first. The updated preconditioners discussed here can be implemented performing just matrix-vector products and the only (sparse-banded) matrices we need to store for the preconditioner are $\tilde{Z}$ and the part of $\tilde{\Delta}_j$ generated by the discretization of (1) for $\alpha = 0$ at scale step j, requiring globally the space for allocating a few vectors with many entries as the number of image pixels. Indeed, the discretization of the reaction part of the equation (1), i.e. the *data-fidelity term*, does not play any role in Δ_j or in its approximation $\tilde{\Delta}_j$.

For the computational complexity *in time* for updated preconditioners in inverse factorized form, let us suppose that $\tilde{Z}$ in (17) is an upper triangular matrix with nonzero entries only in the first d_Z nonzero upper diagonals and let G be the Cholesky factor (or, e.g., a banded approximation of it) of $\tilde{D} + \tilde{E}_j$ with nonzero entries only in the first d_G nonzero lower diagonals and we consider an $m \times m$ input image. The application of an updated preconditioner requires:

- 2 matrix-vector products with $\tilde{Z}$ and with $\tilde{Z}^T$: $2\, d_Z\, m^2$ flops;

- generating Cholesky factorization of the banded matrix $\tilde{D} + \tilde{E}_j$: $O(d_G^2\, m^2)$ flops;
- solving banded triangular linear systems with matrices G and G^T by using a solver for sparse triangular matrices: $2 \cdot O(d_G m^2)$ flops.
- In case of $d_Z \gg d_G$ ($d_Z > d_G^2$), then the computational cost is dominated by the cost for generating products with matrices $\tilde{Z}$ and $\tilde{Z}^T$, thus banded or diagonal updates have a similar computational cost.

By the arguments in Section 4.1, the number of updated preconditioned iterations increases in theory at most with m. In theory, the total computational cost per iteration is at most of $m \cdot O(m^2) = O(m^3)$ flops, i.e., $O(N\sqrt{N})$ flops, $N = m^2$ is the number of image pixels. In practice, we experience convergence within a number of iterations not increasing with m or increasing at most with $O(\log m)$. Therefore, the global computational cost is observed between $O(N)$ and $O(N \log N)$, the latter is the same cost of successful algebraic multigrid preconditioners; see, e.g., [7].

We remark that a shared memory parallel implementation of the proposed techniques can improve very much timings because most of the operations are easy to parallelize such as the matrix-vector products, the construction of SAINV preconditioner for the reference matrix and the sparse triangular matrix inversions of IILUT.

5. Numerical experiments

To illustrate and validate the methods described in the previous sections, we report some tests and comparisons with a preliminary Matlab version of our ideas on an Intel T9550 processor laptop, second core deliberately always idle, equipped with 2Gb Ram.

In particular, the tests here will focus on model (2) because the performances of the updates are similar for the other.

Here, a blur- and noise-contaminated gray-scale image $u^0 \in \mathbb{R}^N$, $N = m^2$, is obtained by blurring and adding an error vector e to the original image u. Thus, $u^0 = K\,u + n$, where K represents the blur operator. The corrupted image u_0 is assumed to be available and the aim is to restore the blur- and noise-free image u. In our experiments, the noise n has normally distributed entries with mean zero, scaled to yield a desired noise-level $\nu = \|n\|/\|u\|$, where $\|\cdot\|$ is the Euclidean norm. The parameter α in (1) is set to 1 for all experiments with blur contaminated images.

The blur contribution is realized by convolution with the Gaussian kernel

$$h(x,y) = \frac{1}{2\pi\, sigma^2} \exp\left(-\frac{x^2+y^2}{2 sigma^2}\right)$$

Our Matlab blur function has parameters `band` and `sigma`. The former specifies the half-bandwidth inside the blocks and the latter the variance of the Gaussian point spread function. The resulting matrix is block-Toeplitz plus Hankel with Toeplitz plus Hankel blocks and can be diagonalized by two-dimensional discrete cosine transforms; see [17].

We stress a few important general facts concerning the experiments.

- Updated preconditioned iterative solvers need more iterations to get the approximations to the required tolerance with respect to incomplete Cholesky factorization recomputed at each time step (*incomplete Cholesky* in the tables) but the situation is reversed for what concern the timings.
- We deliberately avoid a very accurate choice of the parameters (e.g., adaptive stopping tolerances, thresholds for incomplete factorizations, etc.) for each test in order to show that the proposed techniques are robust enough and can give interesting performances even without a fine tuning.

The notation

Here is a legenda for the tables below.

- *inv updated prec (band)*: inverse updated preconditioner (17) updating the inverse drop-threshold incomplete Cholesky preconditioner IILDL^T with *band* upper and lower diagonals for the updates. *Band*= 0 means a diagonal approximation for E_j (13).
- *SAinv updated prec*: inverse updated preconditioner updating the SAINV threshold preconditioner.

- *IC (δ)*: threshold incomplete Cholesky factorization preconditioner (drop tolerance δ) recomputed at each scale step.
- *IC static (δ)*: threshold incomplete Cholesky factorization preconditioner (drop tolerance δ) computed for the first matrix in the sequence is used in all steps until iterations are below 400.
- *Unpreconditioned*: no preconditioner is used, a failure is declared if convergence does not take place within 400 iterations.

We report the global number of iterations, the global and preconditioning time (in seconds) needed for solving the underlying linear systems and the ratio of the number of nonzero entries of the triangular factor of the preconditioner (i.e., of $\tilde{Z}$ or $\tilde{L}$) over the nonzero entries of C_j, $nnz(P_j)/nnz(C_j)$ in the tables. We stress that for the updated preconditioners $nnz(P_j)$ is constant for all j here because only one reference incomplete factorization preconditioner was computed for each test. On the other hand, the fill-in parameter $nnz(P_j)/nnz(C_j)$ for the incomplete Cholesky factorization recomputed at each step strategy changes for each step j because both the triangular factors and the matrix they are computed from change at each step. For the latter, we report an average of the ratios. The timings in columns $Time_P$ include only the time in seconds for calculating the preconditioners (the *reference* preconditioners for the *updated preconditioners*). The cost for updating is enclosed in the total time *Time*.

The parameters

The models (1), (2) use $g(s) = 1/(1 + \kappa\, s^2)$, with $\kappa = 0.1$ for both the proposed test image. The convolution is realized as in [12] by using σ less than τ. The incomplete Cholesky factorization is used with a threshold of 0.1 and 0.01 and 0.1 is used in order to generate $\tilde{Z}$ from $(\tilde{L}^T)^{-1}$. The iterations are stopped when the relative residual is less than 10^{-7}. We recall that the space step h is $1/m$. In all experiments, the time scale τ is set to be large enough to get a good and fast improvement in the SNR for the restored images as the time steps proceed. With the parameters as above, we generate the number of scale steps to get the best possible *signal to noise ratio* in deciBel defined as

$$SNR = 10 \cdot \log_{10}\left(\frac{\|x - E(x)\|^2}{\|x - u\|^2}\right), \tag{23}$$

and checking the *improvement to signal to noise ratio* (ISNR)

$$ISNR = 10 \cdot \log_{10}\left(\frac{\|u^0 - x\|^2}{\|u - x\|^2}\right), \tag{24}$$

at each step, where u^0, u, x represent the array of the observed, restored and original image respectively, $E(x)$ is the mean intensity of x.

The *speed up* is computed for each example by $(t_{IC} - t_{upd})/t_{upd}$, where IC and upd stand for the faster incomplete Cholesky test and the faster inverse updated preconditioner reported in the columns of timings for the appropriate table, respectively.

The images

Two different popular test images are considered in this section.

Example 4.1

Lena is a detailed gray-scale photographic image with edges, details and textures. It is represented by a 256×256 pixels image. The noise- and blur-free image is shown in Figure 2 (left). We have compared the performance of the preconditioner with high noise levels, i.e. 20%. The timings and iterations, together with ratio of nonzero entries of the triangular factors of the preconditioner vs. nonzero entries of the matrices C_j are reported in Table 1. We have a gain in time of 137% for the updated preconditioned (3.7) compared to the best non-updated test.

Figure 2: (from left to right) original 256×256 Lena picture; with 20% Gaussian noise and *band* = 5, *sigma* = 3 for the blur; restored using model (2) after one time step; after 14 steps. $\tau = 10^{-5}$.

Table 1: Timings and iterations for denoising the 256×256 Lena picture, Gaussian noise level 20%, $\tau = 10^{-5}$ and blur with parameters *band* = 5, *sigma* = 1. 14 scale steps, $SNR \simeq 12\,dB$ for the restored image in the last step.

Algorithm	Iterations	Time	Time$_P$	$nnz(P)/nnz(C)$
inv updated prec (band=5)	83	11.20	2.6	0.28
inv updated prec (band=1)	83	10.35	2.6	0.28
inv updated prec (band=0)	95	9.68	2.6	0.28
SAinv updated prec	94	9.75	2.6	0.30
IC (10^{-1})	117	48.9	20	$8 \cdot 10^{-3}$
IC static (10^{-1})	333	23	1	$8 \cdot 10^{-3}$
IC (10^{-2})	54	95	45	$1.2 \cdot 10^{-2}$
unpreconditioned	526	32.6	–	–

Example 4.2

The pepper image is a detailed gray-scale photographic image. It is represented by a 256×256 pixels image. The noise- and blur-free image is shown in Figure 3 (left). We have compared the performance of the preconditioner with

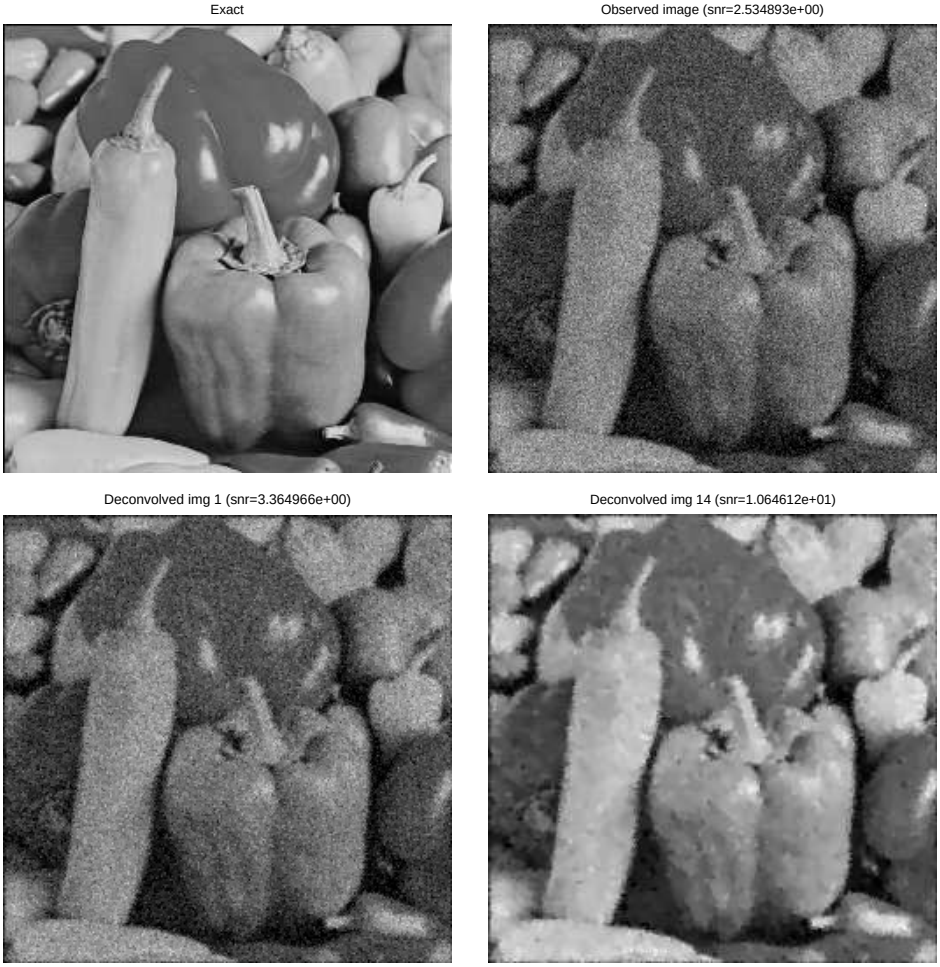

Figure 3: (from left to right) original 256×256 peppers picture; with 20% Gaussian noise and *band* = 5, *sigma* = 3 for the blur; restored using model (2) after one time step; after 14 steps. $\tau = 10^{-5}$.

high noise levels, i.e. 20%. The timings and iterations, together with ratio of nonzero entries of the triangular factors of the preconditioner vs. nonzero entries of the matrices C_j are reported in Table 2. We have a gain in time ranging from 143% to 145% of inverse updated preconditioned (3.7) compared to the best non-updated test.

Table 2: Timings and iterations for denoising the 256×256 peppers picture, Gaussian noise level 20%, $\tau = 10^{-5}$ and blur with parameters *band* = 5, *sigma* = 3. 14 scale steps, $SNR \simeq 11\, dB$ for the restored image in the last step.

Algorithm	Iterations	Time	Time$_P$	$nnz(P)/nnz(C)$
inv updated prec	92	9.48	2.6	0.26
SAinv updated prec	92	9.55	2.6	0.27
IC (10^{-1})	115	46.5	18.9	$7.4 \cdot 10^{-3}$
IC static (10^{-1})	340	23.2	1	$7.4 \cdot 10^{-3}$
IC (10^{-2})	53	87.3	41	$1.2 \cdot 10^{-2}$
unpreconditioned	506	31.1	–	–

Notes on the results

As suggested by Theorem 2, we experienced that *updated inverse preconditioners* with $\tilde{E}_j$ a suitable banded approximation of $\tilde{Z}^T \tilde{\Delta}_j \tilde{Z}$, instead than a diagonal one, can show a similar number of iterations than recomputing an incomplete Cholesky factorization at each step. A banded $\tilde{E}_j$ requires computing an incomplete factorization of a tight banded matrix and then solving a banded linear system at each scale step, with a computational cost slightly greater than for diagonal updates. However, whenever images are large or very large, this extra cost can be sometimes compensated by the reduction of the number of iterations. We not observe this beneficial in our test (see, e.g., Table 1) but, as for the computation of the reference approximate inverse factorization to be updated by our strategy, the Matlab implementation plays a relevant role here.

The timings does not consider the perfectly scalable structure of part of the updated preconditioner based on inverse incomplete Cholesky threshold factorizations. Preliminary tests with a Matlab (not MEX) multicore implementation show that timings are very interesting for the updating strategy with a *reference* preconditioner based on inverse drop-threshold incomplete Cholesky. On the other hand, in the non-multicore setting of this presentation, we get a very much similar behavior for both *SAINV* and $IILDL^T$.

Updates are implemented starting from a reference incomplete factorization and can be used in an adaptive fashion, i.e., the reference factorized preconditioner can be recomputed whenever the quality of the updated approximation is not acceptable anymore and then the update can be based on the new reference one. For example, recomputing is highly recommended for diagonal approximations of the updates if some of the diagonal entries of $\tilde{E}_j$ are negative or very small with respect to the norm of C_n.

The fill-in of the factor of the preconditioners are always reasonable in all tests performed. Different images, resolutions, noise and blur settings confirm advantages shown in the above experiments.

6. Conclusion

Updated preconditioners provide a simple and potentially efficient way for solving the linear algebraic systems in implicit and semi implicit PDE models for image restoration.

A first spectral analysis and several numerical experiments, a few of them reported here, confirm the advantages in terms of time spent for solving the linear systems with respect to incomplete threshold Cholesky factorization preconditioner recomputed from scratch for each linear system, reusing the same (threshold Cholesky) preconditioner for all the scale steps and nonpreconditioned iterations.

With the considered diagonal or tight band updates (i.e., the bandwidth of the matrices that are used to update the reference incomplete factorizations), the global cost is upper bounded by $O(N\sqrt{N})$ in theory, and often by $O(N \log N)$ in practice.

We plan to use our approach with other image restoration models like, e.g., *total variation*.

References

[1] L. Alvarez, P.L. Lions, J. M. Morel, Image selective smoothing and edge detection by nonlinear diffusion II, SIAM J. Numer. Anal. 29-3 (1992) 845–866.
[2] O. Axelsson, V. A. Barker, Finite Element Solution of Boundary Value Problems, Theory and Computation, Academic Press, Orlando, 1984.
[3] M. Benzi, D. Bertaccini, Approximate inverse preconditioning for shifted linear systems, BIT 43 (2003) 231–244.
[4] M. Benzi, J. K. Cullum, M. Tůma, Robust approximate inverse preconditioning for the conjugate gradient method, SIAM J. Sci. Comput. 22-4 (2000) 1318–1332.
[5] D. Bertaccini, Efficient preconditioning for sequences of parametric complex symmetric linear systems, ETNA 18 (2004) 49–64.
[6] F. Catté, P.L. Lions, J.M. Morel, T. Coll, Image selective smoothing and edge detection by nonlinear diffusion II, SIAM J. Numer. Anal. 129 (1992) 182–193.
[7] K. Chen, Xue-Cheng Tai, A nonlinear multigrid method for total variation minimization from image restoration, J. Sci. Comput. 33 (2007) 115–138.
[8] K. Chen, J. Savage, An accelerated algebraic multigrid algorithm for total variation denoising, BIT 47 (2007) 277–296.
[9] S. Corsaro, K. Mikula, A. Sarti, F. Sgallari, Semi-implicit co-volume method in 3D image segmentation, SIAM J. Sci. Comput. 28 (2006) 2248–2265.
[10] S. Demko, W. F. Moss, P. W. Smith, Decay rates for inverses of band matrices, Math. Comp. 43 (1984) 491–499.
[11] L.C. Evans, J.Spruck, Motion of level sets by mean curvature I, J. Diff. Geom. 33 (1991) 635–681.
[12] A. Handlovičová, K. Mikula, F. Sgallari, Semi-implicit complementary volume scheme for solving level set like equations in image processing and curve evolution, Numer. Math. 93 (2003) 675–695.

[13] G. Kühne, J. Weickert, M. Beier, W. Effelsberg, Fast implicit active contour models, in: L. Van Gool (Ed.), Pattern Recognition, in: Lecture Notes in Computer Science, vol. 2449, Springer, Berlin, 2002, pp. 133–140.
[14] T. Lu, P. Neittaanmäki, X.-C. Tai, A parallel splitting up method and its application to Navier-Stoke equations, Appl. Math. Lett. 4-2 (1991) 25–29.
[15] G. Meurant, On the incomplete Cholesky decomposition of a class of perturbed matrices, SIAM J. Sci. Comput. 23 (2001) 419–429.
[16] A. Marquina, S. Osher, Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal, SIAM J. Sci. Comput. 22 (2000) 387–405.
[17] M. K. Ng, Iterative Methods for Toeplitz Systems, Oxford University Press, Oxford, 2004.
[18] Y. Saad, Iterative methods for sparse linear systems, second ed., SIAM, Philadelphia, 2003.
[19] J. D. Tebbens, M. Tůma, Efficient preconditioning of sequences of nonsymmetric linear systems, SIAM J. Sci. Comput. 29 (2007) 1918–1941.
[20] A.C.N. van Duin, Scalable parallel preconditioning SIAM J. Matrix Anal. Appl. 20 (1999) 987–1006.
[21] N. Walkington, Algorithms for computing motion by mean curvature, SIAM J. Numerical Analysis 33 (1996) 2215–2238.
[22] J. Weickert, B.M. ter Haar Romeny, M.A. Viergever, Efficient and reliable schemes for nonlinear diffusion filtering, IEEE Trans. Image Process. 7 (1998) 398–410.