Accelerating $\ell^1 - \ell^2$ deblurring using wavelet expansions of operators

Paul Escande [*] Pierre Weiss [†]

August 10, 2017

Abstract

Image deblurring is a fundamental problem in imaging, usually solved with computationally intensive optimization procedures. The goal of this paper is to provide new efficient strategies to reduce computing times for simple deblurring models regularized using orthogonal wavelet transforms. We show that the minimization can be significantly accelerated by leveraging the fact that images and blur operators are compressible in the same orthogonal wavelet basis. The proposed methodology consists of three ingredients: i) a sparse approximation of the blur operator in wavelet bases, ii) a diagonal preconditioner and iii) an implementation on massively parallel architectures. Combing the three ingredients leads to acceleration factors ranging from 4 to 250 on a typical workstation. For instance, a 1024×1024 image can be deblurred in 0.15 seconds.

Keywords: sparse wavelet expansion, preconditioning, GPU programming, image deblurring, inverse problems.

AMS: 65F50, 65R30, 65T60, 65Y20, 42C40, 45Q05, 45P05, 47A58

[*]Département d'Ingénierie des Systèmes Complexes (DISC), Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), Toulouse, France, `paul.escande@gmail.com`

[†]Institut des Technologies Avancées en Sciences du Vivant, ITAV-USR3505 and Institut de Mathématiques de Toulouse, IMT-UMR5219, CNRS and université de Toulouse, Toulouse, France, `pierre.armand.weiss@gmail.com`

1 Introduction

Most imaging devices produce blurry images. This degradation very often prevents the correct interpretation of image contents and sometimes ruins expensive experiments. One of the most advertised examples of that type is Hubble space telescope [*], which was discovered to suffer from severe optical aberrations after being launched. Such situations occur on a daily basis in fields such as biomedical imaging, astronomy or conventional photography.

Starting from the seventies, a large number of numerical methods to deblur images was therefore developed. The first methods were based on linear estimators such as the Wiener filter [40]. They were progressively replaced by more complicated nonlinear methods, incorporating prior knowledge on the image contents. We refer the interested reader to the following review papers [34, 38, 30] to get an overview of the available techniques.

Despite providing better reconstruction results, the most efficient methods are often disregarded in practice, due to their high computational complexity, especially for large 2D or 3D images. The goal of this paper is to develop new numerical strategies that significantly reduce the computational burden of image deblurring. The proposed ideas yield a *fast deblurring method*, compatible with *large data* and *routine use*. They allow handling both *stationary and spatially varying blurs* with an identical complexity. The proposed algorithm does not reach the state-of-the-art in terms of image quality, because the prior is too simple, but still performs well in short computing times.A real-time deblurring experiment is provided here `https://www.youtube.com/watch?v=oHnGNRc9Qeo&feature=youtu.be`.

1.1 Image formation and image restoration models

In this paper, we assume that the observed image u_0 reads:

$$u_0 = Hu + b, \tag{1}$$

where $u \in \mathbb{R}^N$ is the clean image we wish to recover, $b \sim \mathcal{N}(0, \sigma^2 I_N)$ is a white Gaussian noise of standard deviation σ and $H \in \mathbb{R}^{N\times N}$ is a known blurring operator. Loosely speaking, a blurring operator replaces the value of a pixel by a mean of its neighbors. A precise definition will be given in Section 4.

Let $\Psi \in \mathbb{R}^{N\times N}$ denote an orthogonal wavelet transform and let $A = H\Psi$. A standard variational formulation to restore u consists of solving:

$$\min_{x\in\mathbb{R}^N} E(x) = F(x) + G(x). \tag{2}$$

In this equation, $F(x)$ is a quadratic data fidelity term defined by

$$F(x) = \frac{1}{2}\|Ax - u_0\|_2^2. \tag{3}$$

[*]The total cost of Hubble telescope is estimated at 10 billions US Dollars [2].

The regularization term $G(x)$ is defined by:

$$G(x) = \|x\|_{1,w} = \sum_{i=1}^{N} w[i]|x[i]| \tag{4}$$

The vector of weights $w \in \mathbb{R}_+^N$ is a regularization parameter that may vary across subbands of the wavelet transform. The weighted ℓ^1-norm is well known to promote sparse vectors. This is usually advantageous since images are compressible in the wavelet domain. Overall, problem (2) consists of finding an image Ψx consistent with the observed data u_0 with a sparse representation x in the wavelet domain.

Many groups worldwide have proposed minimizing similar cost functions in the literature, see e.g. [16, 26, 36, 37]. The current trend is to use frames Ψ such as the undecimated wavelet transforms or learned transforms instead of orthogonal transforms [33, 10, 7, 6]. This usually allows reducing reconstruction artifacts. We focus here on the case where Ψ is orthogonal. This property will help designing much faster algorithms traded for some image quality.

1.2 Standard optimization algorithms

A lot of algorithms based on proximity operators were designed in the last decade to solve convex problems of type (2). We refer the reader to the review papers [4, 13] to get an overview of the available techniques. A typical method is the accelerated proximal gradient descent, also known as FISTA (Fast Iterative Soft Thresholding Algorithm) [3]. By letting $\|A\|_2$ denote the largest singular value of A, it takes the form described in Algorithm 1. This method got very popular lately due to its ease of implementation and relatively fast convergence.

Algorithm 1 Accelerated proximal gradient descent

```
1: input: Initial guess $x^{(0)} = y^{(1)}$, $\tau = 1/\|A\|_2^2$ and $Nit$.
2: for $k = 1$ to $Nit$ do
3:     Compute $\nabla F(y^{(k)}) = A^*(Ay^{(k)} - u_0)$.                ▷ 99.35″
4:     $x^{(k)} = \mathrm{Prox}_{\tau G}\left(y^{(k)} - \tau \nabla F(y^{(k)})\right)$.   ▷ 2.7″
5:     $y^{(k+1)} = x^{(k)} + \frac{k-1}{k+2}(x^{(k)} - x^{(k-1)})$.       ▷ 1.1″
6: end for
```

Let us illustrate this method on a practical deconvolution experiment. We use a 1024×1024 image and assume that $Hu = h \star u$, where $\star$ denotes the discrete convolution product and h is a motion blur described on Figure 4b. In practice, the deblurred image stabilizes visually after 500 iterations. The computing times on a workstation with Matlab and mex-files is around $103''15$. The result is shown on Figure 5. Profiling the code leads to the computing times shown on the right-hand-side of Algorithm 1. As can be seen, 96%

of the computing time is spent in the gradient evaluation. This requires computing two wavelet transforms and two fast Fourier transforms. This simple experiment reveals that two approaches can be used to reduce computing times:

- *Accelerate gradients computation.*
- *Use more sophisticated minimization algorithms to accelerate convergence.*

2 Main ideas

The method proposed in this paper relies on three ideas. First, function F in equation (3) can be approximated by another function F_K such that ∇F_K is inexpensive to compute. Second, we characterize precisely the structure of the Hessian of F_K, allowing to design efficient preconditioners. Finally, we implement the iterative algorithm on a Graphical Processing Unit (GPU). The first two ideas, which constitute the main contribution of this paper, are motivated by our recent observation that spatially varying blur operators are compressible and have a well characterized structure in the wavelet domain [15]. We showed that matrix

$$\Theta = \Psi^* H \Psi, \tag{5}$$

which differs from H by a change of basis, has a particular banded structure, with many negligible entries. Therefore, one can construct a K-sparse matrix Θ_K (i.e. a matrix with at most K non zero entries) such that $\Theta_K \simeq \Theta$.

Problem approximation Using the fact that Ψ is an orthogonal transform allows writing that:

$$\min_{x\in\mathbb{R}^N} \frac{1}{2}\|H\Psi x - u_0\|_2^2 + \|x\|_{1,w} \tag{6}$$

$$= \min_{x\in\mathbb{R}^N} \frac{1}{2}\|\Psi^*(H\Psi x - u_0)\|_2^2 + \|x\|_{1,w} \tag{7}$$

$$= \min_{x\in\mathbb{R}^N} \frac{1}{2}\|\Theta x - x_0\|_2^2 + \|x\|_{1,w}, \tag{8}$$

where $x_0 = \Psi^* u_0$ is the wavelet decomposition of u_0. Problem (8) is expressed entirely in the wavelet domain, contrarily to problem (2). However, matrix-vector products with Θ might be computationally expensive. We therefore approximate the variational problem (8) by:

$$\min_{x\in\mathbb{R}^N} \frac{1}{2}\|\Theta_K x - x_0\|_2^2 + \|x\|_{1,w}. \tag{9}$$

Now, let $F_K(x) = \frac{1}{2}\|\Theta_K x - x_0\|_2^2$. The gradient of F_K reads:

$$\nabla F_K(x) = \Theta_K^*(\Theta_K x - x_0), \tag{10}$$

and therefore requires computing two matrix-vector products with sparse matrices. Computing the approximate gradient (10) is usually much cheaper than computing $\nabla F(x)$ exactly using Fast Fourier transforms and fast wavelet transforms. This may come as a surprise since they are respectively of complexity $O(N \log(N))$ and $O(N)$. In fact, we will see that in favorable cases, the evaluation of $\nabla F_K(x)$ may require about 2 operations per pixel!

Preconditioning The second ingredient of our method relies on the observation that the Hessian matrix $H_{F_K}(x) = \Theta_K^* \Theta_K$ has a near diagonal structure with decreasing entries on the diagonal. This allows designing *efficient preconditioners*, which reduces the number of iterations necessary to reach a satisfactory precision. In practice preconditioning leads to acceleration factors ranging from 2 to 5.

GPU implementation Finally, using massively parallel programming on graphic cards still leads to an acceleration factor of order 10 on an NVIDIA K20c. Of course, this factor could be improved further by using more powerful graphic cards. Combining the three ingredients leads to algorithms that are from 4 to 250 times faster than FISTA or the Alternating Direction Method of Multipliers (ADMM) [18] applied to (2), which arguably constitute the current state-of-the-art.

2.1 Related works

The idea of characterizing integral operators in the wavelet domain appeared nearly at the same time as wavelets, at the end of the eighties. Y. Meyer characterized many properties of Calderón-Zygmund operators in his seminal book [24]. Later, Beylkin, Coifman and Rokhlin [5], showed that those theoretical results may have important consequences for the fast resolution of partial differential equations and the compression of matrices. Since then, the idea of using multi-scale representations has been used extensively in numerical simulation of physical phenomena. The interested reader can refer to [11] for some applications.

Quite surprisingly, it seems that very few researchers attempted to apply them in imaging. In [9, 22], the authors proposed to approximate integral operators by matrices diagonal in the wavelet domain. Our experience is that diagonal approximations are too crude to provide sufficiently good approximations. In [17], the authors use wavelets as a tool to prove convergence of a domain decomposition method with ideas similar to those presented in this paper. However, they do not really explore the consequences for the fast resolution of inverse problems. More recently the authors of [39, 15] independently proposed to compress operators in the wavelet domain. However they did not explore its implications for the fast resolution of inverse problems.

On the side of preconditioning, the two references [36, 37] are closely related to our work. The authors designed a few preconditioners to accelerate the convergence of the proximal gradient descent (also called thresholded Landweber algorithm or Iterative Soft Thresholding Algorithm). Overall, the idea of preconditioning is therefore not new. To the best of our knowledge, our contribution is however the first that is based on a precise understanding of the structure of Θ.

2.2 Contributions

The main contributions of this paper are as follows.

- We show that the ideas proposed in [15] make the new formulation (8) particularly attractive for the design of optimization algorithms.
- We provide a fast $O(N\log(N)^2)$ algorithm to compute matrix Θ in the case where H is a convolution operator.
- We design preconditioners based on clear understanding of the problem geometry. To the best of our knowledge, existing attempts to precondition the problem are rather heuristic [36, 37].
- We provide a gallery of numerical comparisons and examples to show the well-foundedness of the proposed approach.

2.3 Paper outline

The paper is structured as follows. We first provide some notation and definitions in section 3. We then provide a few results characterizing the structure of blurring operators in section 4. This section is mostly based on our previous work [15]. We propose two simple explicit preconditioners in section 5. Finally, we perform numerical experiments and comparisons in section 6.

3 Notation

In this paper, we consider d dimensional images. To simplify the discussion, we use periodic boundary conditions and work on the d-dimensional torus $\Omega = \mathbb{T}^d$, where $\mathbb{T}^d = \mathbb{R}^d/\mathbb{Z}^d$. The space $L^2(\Omega)$ denotes the space of squared integrable functions defined on Ω.

Let $\alpha = (\alpha_1, \ldots, \alpha_d)$ denote a multi-index. The sum of its components is denoted $|\alpha| = \sum_{i=1}^d \alpha_i$. The Sobolev spaces $W^{M,p}$ are defined as the set of functions $f \in L^p$ with partial derivatives up to order M in L^p where $p \in [1, +\infty]$ and $M \in \mathbb{N}$. These spaces, equipped with the following norm are Banach spaces

$$\|f\|_{W^{M,p}} = \sum_{|\alpha|\leq M} \|\partial^\alpha f\|_{L^p}\,, \tag{11}$$

where $\partial^\alpha f = \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \cdots \frac{\partial^{\alpha_d}}{\partial x_d^{\alpha_d}} f$.

Let us now define a wavelet basis on $L^2(\Omega)$. To this end, we first introduce a 1D wavelet basis on $\mathbb{T}$. Let ϕ and ψ denote the scaling and mother wavelets and assume that the mother wavelet ψ has M vanishing moments, i.e.

$$\text{for all } 0 \le m < M, \quad \int_{[0,1]} t^m \psi(t)dt = 0. \tag{12}$$

We assume that $\mathrm{supp}(\psi) = [-c(M)/2, c(M)/2]$. Note that $c(M) \ge 2M - 1$, with equality for Daubechies wavelets, see, e.g., [23, Theorem 7.9, p. 294].

Translated and dilated versions of the wavelets are defined, for $j \ge 0$, as follows

$$\phi_{j,l} = 2^{j/2}\phi\left(2^j \cdot - l\right), \tag{13}$$

$$\psi_{j,l} = 2^{j/2}\psi\left(2^j \cdot - l\right), \tag{14}$$

with $l \in \mathcal{T}_j$ and $\mathcal{T}_j = \{0, \ldots, 2^j - 1\}$.

In dimension d, we use isotropic separable wavelet bases, see, e.g., [23, Theorem 7.26, p. 348]. Let $m = (m_1, \ldots, m_d)$. Define $\rho^0_{j,l} = \phi_{j,l}$ and $\rho^1_{j,l} = \psi_{j,l}$. Let $e = (e_1, \ldots, e_d) \in \{0,1\}^d$. For the ease of reading, we will use the shorthand notation $\lambda = (j, m, e)$ and $|\lambda| = j$. We also let

$$\Lambda_0 = \left\{(j, m, e) \mid j \in \mathbb{Z},\ m \in \mathcal{T}_j,\ e \in \{0,1\}^d\right\} \tag{15}$$

and

$$\Lambda = \left\{(j, m, e) \mid j \in \mathbb{Z},\ m \in \mathcal{T}_j,\ e \in \{0,1\}^d \setminus \{0_{\mathbb{R}^d}\}\right\}. \tag{16}$$

Wavelet ψ_λ is defined by $\psi_\lambda(x_1, \ldots, x_d) = \psi^e_{j,m}(x_1, \ldots, x_d) = \rho^{e_1}_{j,m_1}(x_1) \ldots \rho^{e_d}_{j,m_d}(x_d)$. Elements of the separable wavelet basis consist of tensor products of scaling and mother wavelets at the same scale. Note that if $e \ne 0$ wavelet $\psi^e_{j,m}$ has M vanishing moments in $\mathbb{R}^d$. Let $I_{j,m} = \cup_e \operatorname{supp} \psi^e_{j,m}$ and $I_\lambda = \operatorname{supp} \psi_\lambda$. The distance between the supports of ψ_λ and ψ_μ is defined by

$$\begin{aligned} \mathrm{dist}(I_\lambda, I_\mu) &= \inf_{x \in I_\lambda,\, y \in I_\mu} \|x - y\|_\infty & (17)\\ &= \max\left(0, \left\|2^{-j}m - 2^{-k}n\right\|_\infty - (2^{-j} + 2^{-k})\frac{c(M)}{2}\right). & (18) \end{aligned}$$

This distance will play an important role to describe the structure of matrix Θ.

With these definitions, every function $f \in L^2(\Omega)$ can be written as

$$u = \left\langle u, \psi^0_{0,0}\right\rangle \psi^0_{0,0} + \sum_{e \in \{0,1\}^d \setminus \{0\}} \sum_{j=0}^{+\infty} \sum_{m \in \mathcal{T}_j} \left\langle u, \psi^e_{j,m}\right\rangle \psi^e_{j,m} \tag{19}$$

PSF	Point Spread Function or impulse response of the blurring operator
ADMM	Alternating Direction Method of Multipliers [18]
FISTA	Fast Iterative Soft Thresholding Algorithm [3]
pSNR	peak Signal To Noise Ratio, defined in (22).
GPU	Graphics Processing Unit.
CPU	Central Processing Unit.
SPAI	SParse Approximate Inverse, a preconditioning technique, see section 5.3.

Table 1: Table of acronyms.

$$= \left\langle u, \psi^0_{0,0} \right\rangle \psi^0_{0,0} + \sum_{\lambda \in \Lambda} \langle u, \psi_\lambda \rangle \psi_\lambda \tag{20}$$

$$= \sum_{\lambda \in \Lambda_0} \langle u, \psi_\lambda \rangle \psi_\lambda. \tag{21}$$

We let $\Psi^* : L^2(\Omega) \to l^2(\mathbb{Z})$ denote the wavelet decomposition operator and $\Psi : l^2(\mathbb{Z}) \to L^2(\Omega)$ its associated reconstruction operator. The discrete wavelet transform is also denoted $\Psi : \mathbb{R}^N \to \mathbb{R}^N$. We refer to [23, 14, 12] for more details on the construction of wavelet bases.

The peak Signal to Noise Ration (pSNR) is expressed in decibels (dB). It is defined for two vector $(u_1, u_2) \in \mathbb{R}^n \times \mathbb{R}^n$ by:

$$pSNR(u_1, u_2) := 10 \log_{10} \left(\frac{\max(u_1) - \min(u_1)}{\|u_1 - u_2\|_2^2} \right). \tag{22}$$

Table 1 summarizes all acronyms used in the paper.

4 Wavelet decompositions of blurring operators

In this section we remind some results on the decomposition of blurring operators in the wavelet domain.

4.1 Definition of blurring operators

A blurring operator H can be modeled by a linear integral operator $H : L^2(\Omega) \to L^2(\Omega)$:

$$\forall x \in \Omega, \quad Hu(x) = \int_\Omega K(x, y) u(y) dy. \tag{23}$$

The function $K : \Omega \times \Omega \to \mathbb{R}$ is called kernel of the integral operator and defines the Point Spread Function (PSF) $K(\,\cdot\,, y)$ at location $y \in \Omega$. The image Hu is the blurred version of u. Following our recent paper [15], we define blurring operators as follows.

Definition 1 (Blurring operators [15])**.** Let $M \in \mathbb{N}$ and $f : [0,1] \to \mathbb{R}_+$ denote a non-increasing bounded function. An integral operator is called a blurring operator in the class $\mathcal{A}(M,f)$ if it satisfies the following properties:

1. Its kernel $K \in W^{M,\infty}(\Omega \times \Omega)$;
2. The partial derivatives of K satisfy:

   (a)
   $$\forall\, |\alpha| \leq M,\ \forall (x,y) \in \Omega \times \Omega, \quad |\partial_x^\alpha K(x,y)| \leq f\left(\|x-y\|_\infty\right), \tag{24}$$

   (b)
   $$\forall\, |\alpha| \leq M,\ \forall (x,y) \in \Omega \times \Omega, \quad \left|\partial_y^\alpha K(x,y)\right| \leq f\left(\|x-y\|_\infty\right). \tag{25}$$

Condition (24) means that the PSF is smooth, while condition (25) indicates that the PSFs vary smoothly. These regularity assumptions are met in a large number of practical problems. In addition, they allow deriving theorems similar to those of the seminal papers of Y. Meyer, R. Coifman, G. Beylkin and V. Rokhlin [25, 5]. Those results basically state that an operator in the class $\mathcal{A}(M,f)$ can be represented and computed efficiently when decomposed in a wavelet basis. We make this key idea precise in the next paragraph.

4.2 Decomposition in wavelet bases

Since H is defined on the Hilbert space $L^2(\Omega)$, it can be written as $H = \Psi\Theta\Psi^*$, where $\Theta : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$ is the infinite matrix representation of the operator H in the wavelet domain. Matrix Θ is characterized by the coefficients:

$$\theta_{\lambda,\mu} = \Theta[\lambda,\mu] = \langle H\psi_\lambda, \psi_\mu\rangle. \tag{26}$$

The following result provides a good upper-bound on their amplitude.

Theorem 1 (Representation of blurring operator in wavelet bases [15])**.** *Let $f_{\lambda,\mu} = f\left(\mathrm{dist}(I_\lambda, I_\mu)\right)$ and assume that:*

- *Operator H belongs to the class $\mathcal{A}(M,f)$ (see Definition 1).*
- *The mother wavelet is compactly supported with M vanishing moments.*

Then for all $\lambda = (j,m,e) \in \Lambda$ and $\mu = (k,n,e') \in \Lambda$, with $e, e' \neq 0$:

$$|\theta_{\lambda,\mu}| \leq C_M 2^{-\left(M+\frac{d}{2}\right)|j-k|} 2^{-\min(j,k)(M+d)} f_{\lambda,\mu}, \tag{27}$$

where C_M is a constant that does not depend on λ and μ.

The coefficients of Θ decay exponentially with respect to the scale difference and also as a function of the distance between the two wavelets supports.

4.3 Approximation in wavelet bases

In order to get a representation of the operator in a finite dimensional setting, the wavelet representation can be truncated at scale J. Let $\Theta^{(J)}$ denote the infinite matrix defined by:

$$\Theta^{(J)}[\lambda, \mu] = \begin{cases} \theta_{\lambda,\mu} & \text{if } |\lambda| \leq J \text{ and } |\mu| \leq J, \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

This matrix contains at most N^2 non-zero coefficients, where $N = 1 + \sum_{j=0}^{J-1}(2^d - 1)2^{dj}$ denotes the numbers of wavelets kept to represent functions. The operator $H^{(J)} = \Psi\Theta^{(J)}\Psi^*$ is a Galerkin approximation of H. It is possible to control the discretization error $\|H^{(J)} - H\|_{X\to Y}$, where $\|\cdot\|_{X\to Y}$ is some operator norm between two function spaces X and Y. We refer to [15] for more details.

The following theorem is a variation of [5]. Loosely speaking, it states that $H^{(J)}$ can be well approximated by a matrix containing only $O(N)$ coefficients.

Theorem 2 (Computation of blurring operators in wavelet bases [15])**.** *Set* $0 \leq \eta \leq \log_2(N)^{-(M+d)/d}$. *Let* $\Theta_\eta^{(J)}$ *be the matrix obtained by zeroing all coefficients in* $\Theta^{(J)}$ *such that*

$$2^{-min(j,k)(M+d)} f_{\lambda,\mu} \leq \eta. \tag{29}$$

Define $H_\eta^{(J)} = \Psi\Theta_\eta^{(J)}\Psi^*$. *Under the same hypotheses as Theorem* (1)*, the number of coefficients needed to satisfy* $\left\|H^{(J)} - H_\eta^{(J)}\right\|_{2\to 2} \leq \epsilon$ *is bounded above by*

$$C'_M N \, \epsilon^{-\frac{d}{M}} \tag{30}$$

where $C'_M > 0$ *is independent of* N.

This theorem has important consequences for numerical analysis. It states that evaluations of H can be obtained with an ϵ accuracy using only $O(N\epsilon^{-\frac{d}{M}})$ operations. Note that the smoothness M of the kernel is handled automatically. Of interest, let us mention that [25, 5] proposed similar results under less stringent conditions. In particular, similar inequalities may still hold true if the kernel blows up on the diagonal.

4.4 Discretization

In the discrete setting, the above results can be exploited as follows. Given a matrix $H \in \mathbb{R}^{N\times N}$ that represents a discretized version of H, we perform the change of basis:

$$\Theta = \Psi^* H \Psi, \tag{31}$$

where $\Psi \in \mathbb{R}^{N\times N}$ is the discrete isotropic separable wavelet transform. Similarly to the continuous setting, matrix Θ is essentially concentrated along the diagonals of the wavelet sub-bands (see Figure 1).

Probably the main reason why this decomposition has very seldom been used in practice is that it is very computationally demanding. Computing the whole set of coefficients $(\langle H\psi_\lambda, \psi_\mu\rangle)_{\lambda,\mu}$ is an $O(N^3)$ algorithm! To evaluate this set, one has to apply H to each of the N discrete wavelets. The cost of evaluating one vector $H\psi_\lambda$ is $O(N^2)$ operations since it is a matrix-vector product with an $N \times N$ matrix. Then, evaluating $(\langle H\psi_\lambda, \psi_\mu\rangle)_{\lambda,\mu}$ for all μ and a fixed λ costs $O(N)$ operations (a discrete wavelet transform). The overall complexity is therefore dominated by the evaluation of N vectors $H\psi_\lambda$, with complexity $O(N^3)$.

This computational burden is hardly tractable for large signals, unless it can be computed once for all. This happens when the blur is device dependent but not sample dependent. We will see that computing times are drastically reduced for *convolution operators* in section 4.6.

4.5 Illustration

In order to illustrate the various results provided so far, let us consider an operator acting on 1D signals, with kernel defined by

$$K(x,y) = \frac{1}{\sigma(y)\sqrt{2\pi}} \exp\left(-\frac{(x-y)^2}{2\sigma^2(y)}\right), \tag{32}$$

where $\sigma(y) = 4 + 10y$. All PSFs are Gaussian with a variance that increases linearly. The matrix is displayed in linear scale (resp. log scale) in Figure 1 top-left (resp. top-right). The wavelet representation of the matrix is displayed in linear scale (resp. log scale) in Figure 1 bottom-left (resp. bottom-right). As can be seen, the matrix expressed in the wavelet basis is sparser than in the space domain. It has a particular banded structure captured by Theorem 1.

4.6 Decomposition of convolutions

From now on, we assume that H is a convolution with a kernel h. The results below hold both in the continuous and the discrete setting. We establish them in the discrete setting to ease the implementation. Matrix Θ can be decomposed into its wavelet sub-bands:

$$\Theta = \left(\Theta_{j,k}^{e,e'}\right)_{j,k}, \quad \text{with } \Theta_{j,k}^{e,e'} = \left(\left\langle H\psi_{j,m}^{e}, \psi_{k,n}^{e'}\right\rangle\right)_{m\in\mathcal{T}_j, n\in\mathcal{T}_k}. \tag{33}$$

For instance, on 1D signals with $J = 2$, matrix Θ can be decomposed as shown in Figure 2, left. Let us now describe the specific structure of the sub-bands $\Theta_{j,k}^{e,e'}$ for convolutions. We will need the following definitions.

Definition 2 (Translation operator)**.** Let $a \in \mathbb{R}^N$ denote a d-dimensional image and $m \in \mathbb{Z}^d$ denote a shift. The translated image $b = \tau_m(a)$ is defined for all $i_1, \ldots, i_d$ by:

$$b[i_1, \ldots, i_d] = a[i_1 - m_1, \ldots, i_d - m_d] \tag{34}$$

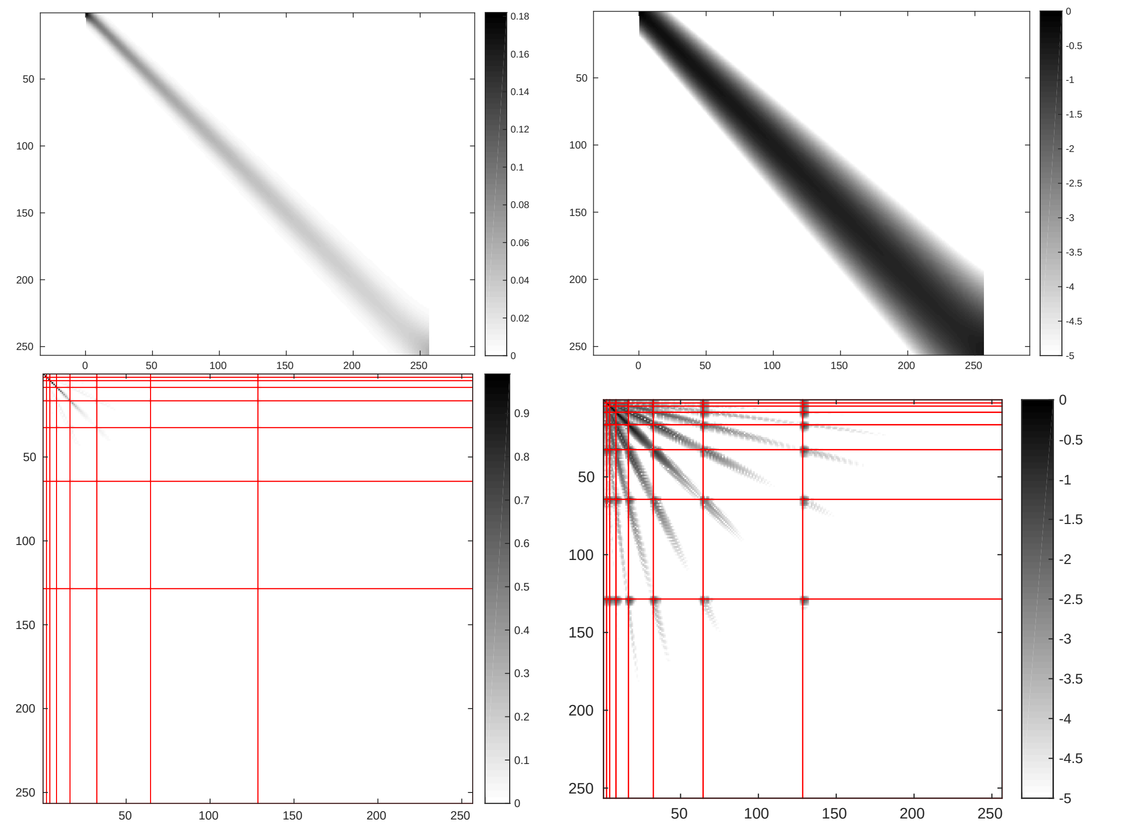


Figure 1: An illustration of the compression of a spatially varying blur in the wavelet domain. Top-left: H. Top-right: H in $\log_{10}$-scale. Bottom-left: Θ obtained using Daubechies wavelets with 10 vanishing moments and a decomposition level $J = 7$. Bottom-right: Θ in $\log_{10}$-scale.

with circular boundary conditions.

Definition 3 (Rectangular circulant matrices)**.** Let $A \in \mathbb{R}^{2^j \times 2^k}$ denote a rectangular matrix. It is called circulant if and only if:

- When $k \geq j$: there exists $a \in \mathbb{R}^{2^k}$, such that, for all $0 \leq l \leq 2^j - 1$,

$$A[l, :] = \tau_{2^{k-j}l}(a).$$

- When $k < j$: there exists $a \in \mathbb{R}^{2^j}$, such that, for all $0 \leq l \leq 2^k - 1$,

$$A[:, l] = \tau_{2^{j-k}l}(a).$$

As an example, a 4×8 circulant matrix is of the form:

$$A = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \\ a_7 & a_8 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ a_5 & a_6 & a_7 & a_8 & a_1 & a_2 & a_3 & a_4 \\ a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_1 & a_2 \end{pmatrix}.$$

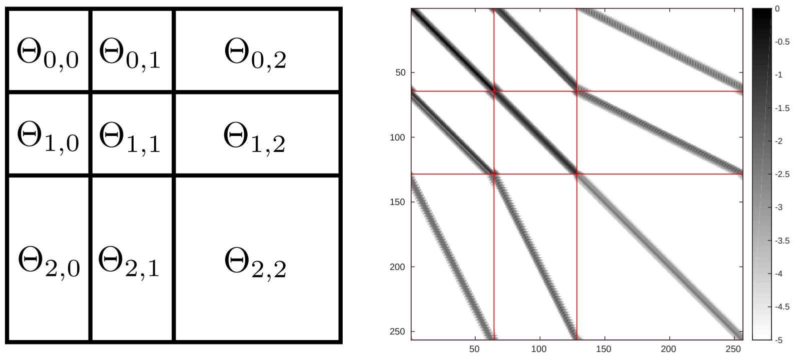


Figure 2: Left: structure of Θ. Right: Θ in $\log_{10}$-scale when H is a convolution with a Gaussian kernel.

Theorem 3 states that all the wavelet sub-bands of Θ are circulant for convolution operators. This is illustrated on Figure 2, right.

Theorem 3 (Circulant structure of Θ)**.** *Let H be a convolution matrix and define $\Theta = \Psi^* H \Psi$ be its wavelet representation. Then, for all $j, k \in [0, J]$ and $e, e' \in \{0,1\}^d$, the sub-matrices $\Theta_{j,k}^{e,e'}$ are circulant.*

Proof. We only treat the case $j \geq k$, since the case $j < k$ is similar. Let $a \in \mathbb{R}^{2^j}$ be defined by:

$$a[m] = \left\langle h \star \psi^e_{j,m}, \psi^{e'}_{k,0} \right\rangle . \tag{35}$$

We have:

$$\begin{aligned}
\left\langle h \star \psi^e_{j,m}, \psi^{e'}_{k,n} \right\rangle &= \left\langle h \star \psi^e_{j,m}, \tau_{2^k n}\left(\psi^{e'}_{k,0}\right) \right\rangle & (36)\\
&= \left\langle \tau_{-2^k n}\left(h \star \psi^e_{j,m}\right), \psi^{e'}_{k,0} \right\rangle & (37)\\
&= \left\langle h \star \tau_{-2^k n}\left(\psi^e_{j,m}\right), \psi^{e'}_{k,0} \right\rangle & (38)\\
&= \left\langle h \star \psi^e_{j,m-2^{j-k}n}, \psi^{e'}_{k,0} \right\rangle & (39)\\
&= a[m - 2^{j-k}n]. & (40)
\end{aligned}$$

The sub-matrix $\Theta^{e,e'}_{j,k}$ is therefore circulant. In this list of identities, we only used the fact that the adjoint of $\tau_{2^k n}$ is $\tau_{-2^k n}$ and the fact that translations and convolution commute. □

The main consequence of Theorem 3 is that the computation of matrix Θ reduces to computing one column or one row of each matrix $\Theta^{e,e'}_{j,k}$. This can be achieved by computing $(2^d - 1)J$ wavelet transforms (see Algorithm 2). The complexity of computing Θ therefore reduces to $O\left((2^d-1)JN\right)$ operations instead of $O(N^3)$ operations for spatially varying operators.

Algorithm 2 An algorithm to compute Θ for convolution operator

```
1: input: $h \in \mathbb{R}^N$, the convolution kernel of $H$.
2: output: $\Theta$, the wavelet representation of $H$
3: for $(j,e) \in [0,J] \times \{0,1\}^d$ do
4:     Compute the wavelet $\psi_\lambda$ with $\lambda = (j,e,0)$.
5:     Compute the blurred wavelets $H\psi_\lambda$ and $H^*\psi_\lambda$.
6:     Compute $(\langle H\psi_\lambda, \psi_\mu \rangle)_\mu$ using one forward wavelet transform.
7:     Compute $(\langle H^*\psi_\lambda, \psi_\mu \rangle)_\mu$ using one forward wavelet transform.
8:     for $(k,e') \in [0,J] \times \{0,1\}^d$ do
9:         if $k \geq j$ then
10:            $\Theta^{e,e'}_{j,k}$ is the circulant matrix with column: $\left(\left\langle H\psi_\lambda, \psi^{e'}_{k,n} \right\rangle\right)_n$
11:        else
12:            $\Theta^{e,e'}_{j,k}$ is the circulant matrix with row: $\left(\left\langle H^*\psi_\lambda, \psi^{e'}_{k,n} \right\rangle\right)_n = \left(\left\langle \psi_\lambda, H\psi^{e'}_{k,n} \right\rangle\right)_n$
13:        end if
14:    end for
15: end for
```

4.7 Thresholding strategies

Theorem 2 ensures that one can construct good sparse approximations of Θ. However, the thresholding strategy suggested by the theorem turns out to be impractical.

In this section, we propose efficient thresholding strategies. Most of the proposed ideas come from our recent work [15], and we refer to this paper for more details. The algorithm specific to convolution operators is new.

Let us define the operator norm:

$$\|H\|_{X\to Y} = \sup_{\|u\|_X\leq 1} \|Hu\|_Y, \tag{41}$$

where $\|\cdot\|_X$ and $\|\cdot\|_Y$ denote two norms on $\mathbb{R}^N$. A natural way to obtain a K-sparse approximation Θ_K of Θ consists of finding the minimizer of:

$$\min_{\Theta_K,\ K\text{-sparse}} \|H_K - H\|_{X\to Y}, \tag{42}$$

where $H_K = \Psi\Theta_K\Psi^*$. The most naive thresholding strategy consists of constructing a matrix Θ_K such that:

$$\Theta_K[\lambda,\mu] = \begin{cases} \Theta[\lambda,\mu] & \text{if } |\Theta[\lambda,\mu]| \text{ is among the } K \text{ largest values of } |\Theta|, \\ 0 & \text{otherwise.} \end{cases} \tag{43}$$

This thresholding strategy can be understood as the solution of the minimization problem (42), by setting $\|\cdot\|_X = \|\Psi^*\cdot\|_1$ and $\|\cdot\|_Y = \|\Psi^*\cdot\|_\infty$.

The ℓ^1-norm of the wavelet coefficients is not adapted to the description of images. Natural images are often modeled as elements of Besov spaces or the space of bounded variation functions [29, 1]. These spaces can be characterized by the decay of their wavelet coefficients [11] across sub-bands. This motivates to set $\|\cdot\|_X = \|\Sigma\Psi^*\cdot\|_1$ where $\Sigma = \mathrm{diag}(\sigma) \in \mathbb{R}^{N\times N}$ is a diagonal matrix and where $\sigma\in\mathbb{R}^N$ is constant by levels. The thresholding strategy using this metric can be expressed as the minimization problem:

$$\min_{\Theta_K,\ K\text{- sparse}} \sup_{\|\Sigma x\|_1\leq 1} \|(\Theta-\Theta_K)x\|_\infty. \tag{44}$$

Its solution is given in closed-form by:

$$\Theta_K[\lambda,\mu] = \begin{cases} \Theta[\lambda,\mu] & \text{if } |\sigma_\mu\Theta[\lambda,\mu]| \text{ is among the } K \text{ largest values of } |\Theta\Sigma|, \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

The weights σ_μ must be adapted to the class of images to recover. In practice we found that setting $\sigma_\mu = 2^{-k}$ for $\mu = (k, e', n)$ is a good choice. These weights can also be trained from a set of images belonging the class of interest. Finally let us mention that we also proposed greedy algorithms when setting $\|\cdot\|_Y = \|\cdot\|_2$ in [15]. In practice, it turns out that both approaches yield similar results.

We illustrate the importance of the thresholding strategy in section 6.1, Figure 7.

5 On the design of preconditioners

Let $P \in \mathbb{R}^{N\times N}$ denote a Symmetric Positive Definite (SPD) matrix. There are two equivalent ways to understand preconditioning: one is based on a change of variable, while the other is based on a metric change.

For the change of variable, let z be defined by $x = P^{1/2}z$. A solution x^* of problem (9) reads $x^* = P^{1/2}z^*$, where z^* is a solution of:

$$\min_{z\in\mathbb{R}^N} \frac{1}{2}\|\Theta_K P^{1/2}z - x_0\|_2^2 + \|P^{1/2}z\|_{1,w}. \tag{46}$$

The convergence rate of iterative methods applied to (46) is now driven by the properties of matrix $\Theta_K P^{1/2}$ instead of Θ_K. By choosing P adequately, one can expect to significantly accelerate convergence rates. For instance, if Θ_K is invertible and $P^{1/2} = \Theta_K^{-1}$, one iteration of a proximal gradient descent provides the exact solution of the problem.

For the metric change, the idea is to define a new scalar product defined by

$$\langle x, y\rangle_P = \langle Px, y\rangle, \tag{47}$$

and to consider the associated norm $\|x\|_P = \sqrt{\langle Px, x\rangle}$. By doing so, the gradient and proximal operators are modified, which leads to different dynamics. The preconditioned FISTA algorithm is given in Algorithm 3.

Algorithm 3 Preconditioned accelerated proximal gradient descent

```
1: input: Initial guess $x^{(0)} = y^{(1)}$, $\tau = 1/\|\Theta_K^*\Theta_K P^{-1}\|_2$ and $Nit$.
2: for $k = 1$ to $Nit$ do
3:     Compute $\nabla F(y^{(k)}) = \Theta_K^*(\Theta_K y^{(k)} - u_0)$.
4:     $x^{(k)} = \mathrm{Prox}_{\tau G}^P\left(y^{(k)} - \tau P^{-1}\nabla F(y^{(k)})\right)$.
5:     $y^{(k+1)} = x^{(k)} + \frac{k-1}{k+2}(x^{(k)} - x^{(k-1)})$.
6: end for
```

In this algorithm

$$\mathrm{Prox}_{\tau G}^P(z_0) = \arg\min_{z\in\mathbb{R}^N} \frac{1}{2}\|z - z_0\|_P^2 + \tau G(z). \tag{48}$$

Unfortunately, it is impossible to provide a closed-form expression of (48), unless matrix P has a very simple structure (e.g. diagonal). Finding an efficient preconditioner therefore requires: i) defining a structure for P compatible with fast evaluations of the proximal operator (48) and ii) improving some "properties" of $\Theta_K P^{1/2}$ using this structure.

5.1 What governs convergence rates?

Good preconditioners are often heuristic. The following sentence is taken from a reference textbook about the resolution of linear systems by Y. Saad [31]: "Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science. Theoretical results are rare and some methods work surprisingly well, often despite expectation." In what follows, we will first show that existing convergence results are indeed of little help. We then provide two simple diagonal preconditioners.

Let us look at the convergence rate of Algorithm 1 applied to problem (46). The following theorem appears in [27, 3] for instance.

Theorem 4. *Let $A = P^{-1/2}\Theta_K^*\Theta_K P^{-1/2}$ and set $L = \lambda_{\max}(A^*A)$. The iterates in Algorithm 3 satisfy:*

$$E(x^{(k)}) - E(x^*) \leq L\|x - x_0\|_2^2 \cdot \min\left(\frac{1}{k^2}, \frac{1}{2}\left(\frac{\sqrt{\kappa(A)-1}}{\sqrt{\kappa(A)+1}}\right)^{2k}\right), \tag{49}$$

where $\kappa(A)$ designs the condition number of A:

$$\kappa(A) = \begin{cases} \sqrt{\frac{\lambda_{\max}(A^*A)}{\lambda_{\min}(A^*A)}} & \text{if } \lambda_{\min}(A^*A) > 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{50}$$

When dealing with ill-posed inverse problems, the condition number $\kappa(A)$ is huge or infinite and bound (49) therefore reduces to

$$E(x^{(k)}) - E(x^*) \leq \frac{L\|x - x_0\|_2^2}{k^2}, \tag{51}$$

even for a very large number of iterations. Unfortunately, this bound tells very little about which properties of A characterize the convergence rate. Only the largest singular value of A seems to matter. The rest of the spectrum does not appear, while it obviously plays a key role.

Recently, more subtle results were proposed in [35] for the specific $\ell^1 - \ell^2$ problem and in [21] for a broad class of problems. These theoretical results were shown to fit some experiments very well. Let us state a typical result.

Theorem 5. *Assume that problem (46) admits a unique minimizer x^*. Let $S^* = \mathrm{supp}(x^*)$ denote the solution's support. Then Algorithm 3 generates a sequence $(x^{(k)})_{k\in\mathbb{N}}$ satisfying the following properties:*

- *The sequence $(x^{(k)})_{k\in\mathbb{N}}$ converges to x^*.*
- *There exists an iteration number k^* such that, for $k \geq k^*$, $\mathrm{supp}(x^{(k)}) = \mathrm{supp}(x^*)$.*

- *If in addition*
$$\langle Ax, Ax\rangle \geq \alpha \|x\|_2^2, \ \forall x \ s.t. \ \mathrm{supp}(x) \subseteq S^*, \tag{52}$$
then the sequence of iterates $(x^{(k)})_{k\in\mathbb{N}}$ *converges linearly to* x^**: there exists* $0 \leq \rho < 1$ *s.t.*
$$\|x^{(k)} - x^*\|_2 = O\left(\rho^k\right). \tag{53}$$

Remark 1. The uniqueness of a solution x^* is not required if the algorithm converges. This can be ensured if the algorithm is slightly modified [8].

The main consequence of Theorem (5) is that good preconditioners should depend on the support S^* of the solution. Obviously, this support is unknown at the start of the algorithm. Moreover, for compact operators, the value α in equation (52) is null or very small. Therefore - once again - Theorem (5) seems to be of little help to find well founded preconditioners.

In this paper we therefore restrict our attention to two standard heuristic preconditioners: Jacobi and Sparse Approximate Inverses (SPAI) [19, 31]. The overall idea is to cluster the eigenvalues of A^*A.

5.2 Jacobi preconditioner

The Jacobi preconditioner is one of the most popular diagonal preconditioner, it consists of setting
$$P = \max(\mathrm{diag}(\Theta_K^*\Theta_K), \epsilon), \tag{54}$$
where ϵ is a small constant and the maximum should be understood component-wise. The parameter ϵ guarantees the invertibility of P.

The idea of this preconditioner is to make the Hessian matrix $P^{-1/2}\Theta_K^*\Theta_K P^{-1/2}$ "close" to the identity. This preconditioner has a simple analytic expression and is known to perform well for diagonally dominant matrices. Blurring matrices expressed in the wavelet domain have a fast decay away from the diagonal, but are usually not diagonally dominant. Moreover, the parameter ϵ has to be tuned. In our numerical experiments, this parameter was hand-tuned so as to ensure the fastest convergence.

5.3 SPAI preconditioner

The preconditioned gradient in Algorithm 3 involves matrix $P^{-1}\Theta_K^*\Theta_K$. The idea of sparse approximate inverses is to cluster the eigenvalues of $P^{-1}\Theta_K^*\Theta_K$ around 1. To improve the clustering, a possibility is to solve the following optimization problem:
$$\underset{P,\ \text{diagonal}}{\arg\min} \left\|\mathrm{Id} - P^{-1}\Theta_K^*\Theta_K\right\|_F^2, \tag{55}$$
where $\|\cdot\|_F$ denotes the Frobenius norm. This formulation is standard in the numerical analysis community [19, 31].

Lemma 6. *Let $M = \Theta_K^* \Theta_K$. The set of solutions of* (55) *reads:*

$$P[i,i] = \begin{cases} \frac{M^2[i,i]}{M[i,i]} & \textit{if } M[i,i] \neq 0, \\ \textit{an arbitrary positive value otherwise.} \end{cases} \tag{56}$$

Proof. First notice that problem (55) can be rewritten as

$$\underset{P,\text{ diagonal}}{\arg\min} \left\|\mathrm{Id} - MP^{-1}\right\|_F^2, \tag{57}$$

by taking the transpose of the matrices, since M is symmetric and P diagonal. The Karush-Kuhn-Tucker optimality conditions for problem (57) yield the existence of a Lagrange multiplier $\mu \in \mathbb{R}^{N\times N}$ such that:

$$M(MP^{-1} - I) + \mu = 0_{\mathbb{R}^N}, \tag{58}$$

with

$$\mu[i,j] = \begin{cases} 0 & \text{if } i = j, \\ \text{an arbitrary value otherwise.} \end{cases} \tag{59}$$

Therefore, for all i,

$$(M^2P^{-1})[i,i] = M[i,i], \tag{60}$$

which can be rewritten as

$$M^2[i,i]P^{-1}[i,i] = M[i,i], \tag{61}$$

since P is diagonal. If $M^2[i,i] = 0$, then $M[i,i] = 0$ since $M^2[i,i]$ is the squared norm of the i-th column of M. In that case, $P^{-1}[i,i]$ can take an arbitrary value. Otherwise $P^{-1}[i,i] = M[i,i]/M^2[i,i]$, finishing the proof. □

5.4 GPU implementation

Algorithm 3 has a structure that is near ideal for parallelization on GPU. It consists of iteratively:

- Evaluating matrix-vector products with sparse matrices. This was achieved with the cuSPARSE library from CUDA.
- Evaluating the action of shrinkage operators. This is a simple operation for a GPU since it suffices to unroll a *for* loop.

All computations are done on the GPU: the only exchanges with the CPU are i) loading the wavelet coefficients and the sparse matrix at the beginning of the iterative process and ii) returning the restored wavelet coefficients at the end. The reported computing times do not include the transfer time.

6 Numerical experiments

In this section we propose a set of numerical experiments to illustrate the proposed methodology and to compare its efficiency with respect to state-of-the-art approaches. In all experiments, we assume that the blur kernel and the noise variance are known (non blind deblurring).

The numerical experiments are performed on two 1024×1024 images with values rescaled in $[0,1]$, see Figure 3. We also consider two different blurs, see Figure 4. The PSF in Figure 4a is an anisotropic 2D Gaussian with kernel defined for all $(t_1, t_2) \in [0,1]^2$ by

$$k(t_1,t_2) = \begin{cases} \exp\left(-\frac{t_1^2}{2\sigma^2} - \frac{t_2^2}{2\sigma^2}\right) & \text{if } t_1 \geq 0, \\ \exp\left(-\frac{4t_1^2}{2\sigma^2} - \frac{t_2^2}{2\sigma^2}\right) & \text{otherwise,} \end{cases}$$

with $\sigma = 5$. This PSF is smooth, which is a favorable situation for our method, see Theorem 2. The PSF in Figure 4b is a simulation of motion blur. This PSF is probably one of the worst for the proposed technique since it is singular. The PSF is generated from a set of $l = 5$ points drawn at random from a Gaussian distribution with standard deviation $\sigma_1 = 8$. Then the points are joined using a cubic spline and the resulting curve is blurred using a Gaussian kernel of standard deviation $\sigma_2 = 1$.

All our numerical experiments are based on Symmlet 6 wavelets decomposed $J = 6$ times. This choice offers a good compromise between computing times and visual quality of the results. The weights w in function G in (4) were defined by $w[i] = j(i)$, where $j(i)$ denotes the scale of the i-th wavelet coefficient. This choice was hand tuned so as to produce the best deblurring results. Figures 5 and 6 display two typical deconvolution results using these parameters.

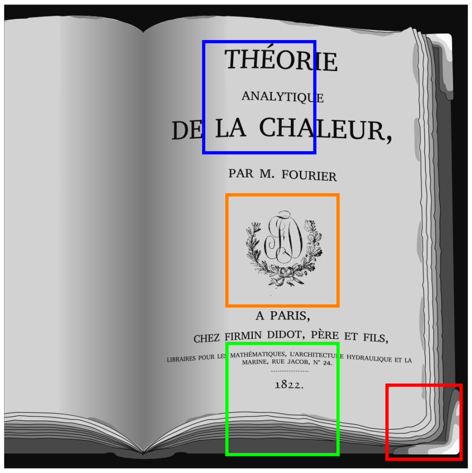
The numerical experiments were performed on Matlab2014b on an Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz with 200Gb RAM in double precision. Automatic multi-threading was disabled by launching Matlab with:

```
>>  matlab -singleCompThread
```

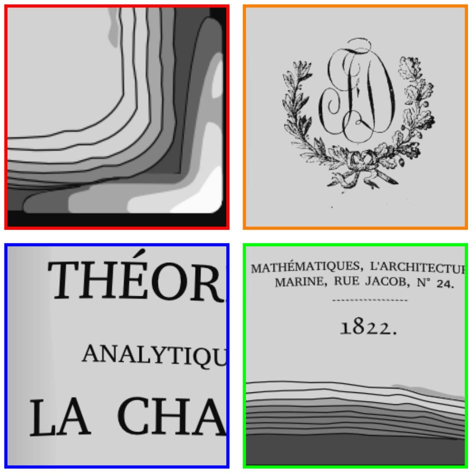
For the experiments led on GPU, we use a NVIDIA Tesla K20c containing 2496 CUDA cores and 5GB internal memory. All computations were performed in double precision.

6.1 On the role of thresholding strategies

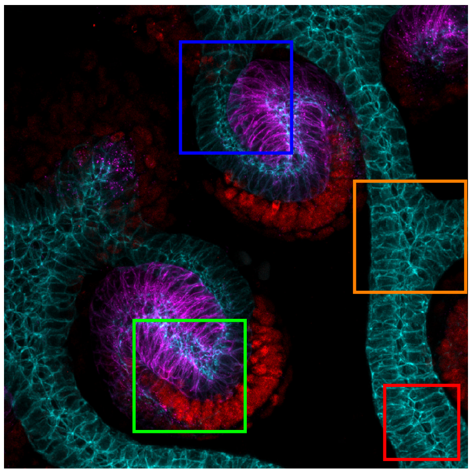
We first illustrate the influence of the thresholding strategy discussed in Section 4.7. We construct two matrices having the same number of coefficients but built using two different thresholding rules: the naive thresholding given in equation (43) and the weighted thresholding given in equation (45). Figure 7 displays the images restored with each of these two matrices. It is clear that the weighted thresholding strategy significantly outperforms the simple one: it produces less artifacts and a higher pSNR. In all the following experiments, this thresholding scheme will be used.


THÉORIE
ANALYTIQUE
DE LA CHALEUR,
PAR M. FOURIER
A PARIS,
CHEZ FIRMIN DIDOT, PÈRE ET FILS,
LIBRAIRES POUR LES MATHÉMATIQUES, L'ARCHITECTURE HYDRAULIQUE ET LA
MARINE, RUE JACOB, N° 24.
1822.

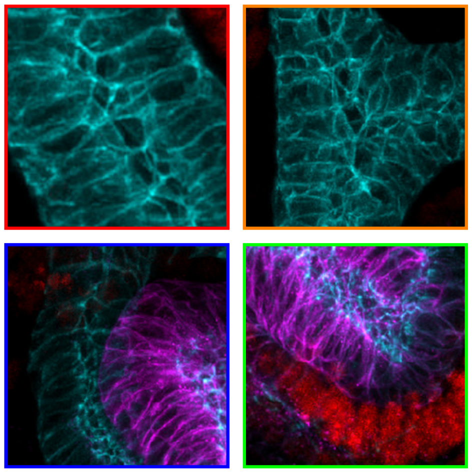
(a)



(b)



(c)



(d)

Figure 3: Original images 1024×1024.

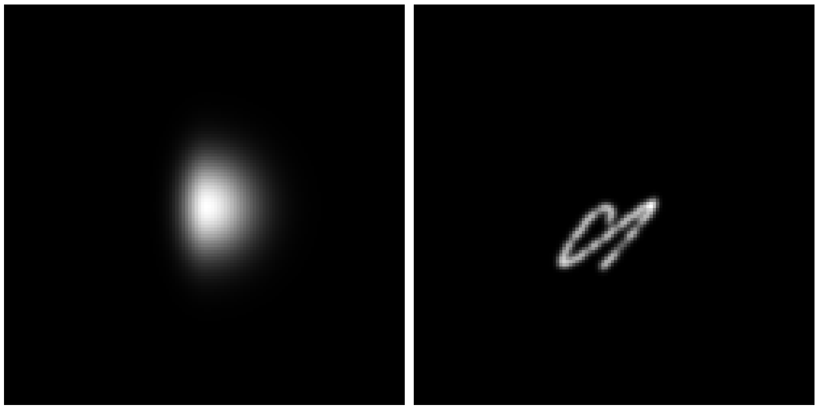

(a) (b)

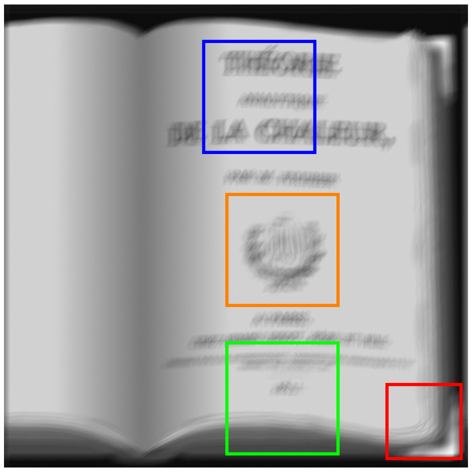
Figure 4: PSFs used in the paper. (4a) is a skewed Gaussian. (4b) is a motion blur.

6.2 Approximation in wavelet bases

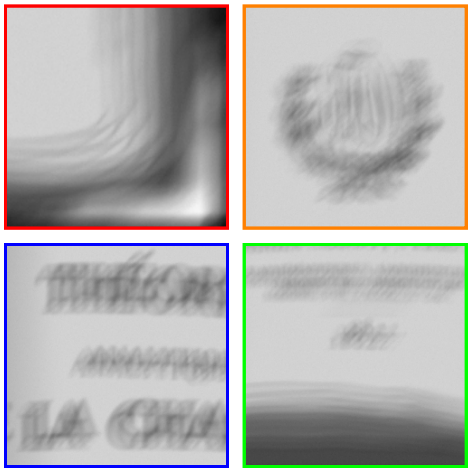
In this paragraph, we illustrate the influence of the approximation on the deblurring quality. We compare the solution of the original problem (2) with the solution of the approximated problem (9) for different numbers of coefficients K. Computing the exact gradient $\nabla F = \Psi^* H^* H \Psi$ requires two fast Fourier transforms, two fast wavelet transforms and a multiplication by a diagonal matrix. Its complexity is therefore: $2N\log_2(N) + 2lN + N$, with l denoting the wavelet filter size. The number of operations per pixel is therefore $2\log_2(N) + 2l + 1$. The approximate gradient ∇F_K requires two matrix-vector products with a K-sparse matrix. Its complexity is therefore $2\frac{K}{N}$ operations per pixel. Figure (8) displays the restoration quality with respect to the number of operations per pixel.

For the smooth PSF in Figure 4a, the standard approach requires 89 operations per pixel, while the wavelet method requires 20 operations per pixel to obtain the same pSNR. This represents an acceleration of a factor 4.5. For users ready to accept a decrease of pSNR of 0.2dB, K can be chosen even significantly lower, leading to an acceleration factor of 40 and around 2.2 operations per pixels! For the less regular PSF 4b, the gain is less important. To obtain a similar pSNR, the proposed approach is in fact slower with 138 operations per pixel instead of 89 for the standard approach. However, accepting a decrease of pSNR of 0.2dB, our method leads to an acceleration by a factor 1.1. To summarize, the proposed approximation does not really lead to interesting acceleration factors for motion blurs. Note however that the preconditioners can be used even if the operator is not expanded in the blur domain.

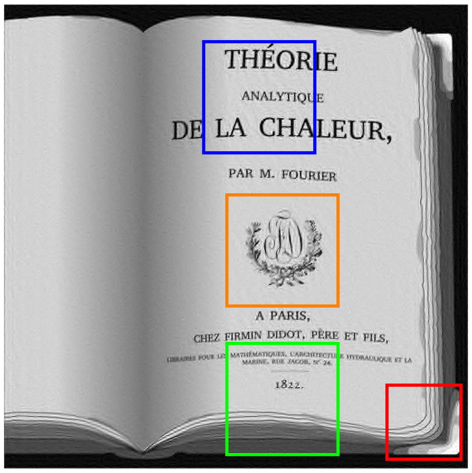
The different behavior between the two blurs was predicted by Theorem 2, since the compressibility of operators in wavelet bases strongly depends on the regularity M of the PSF.



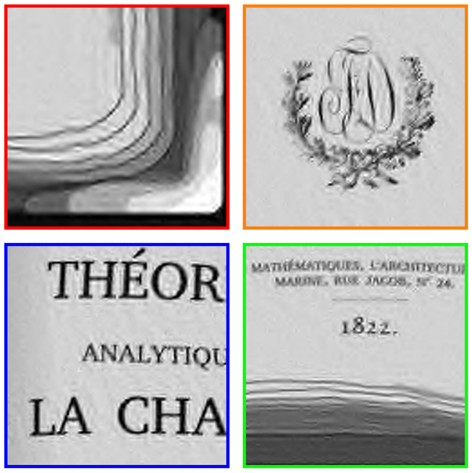
(a)



(b)

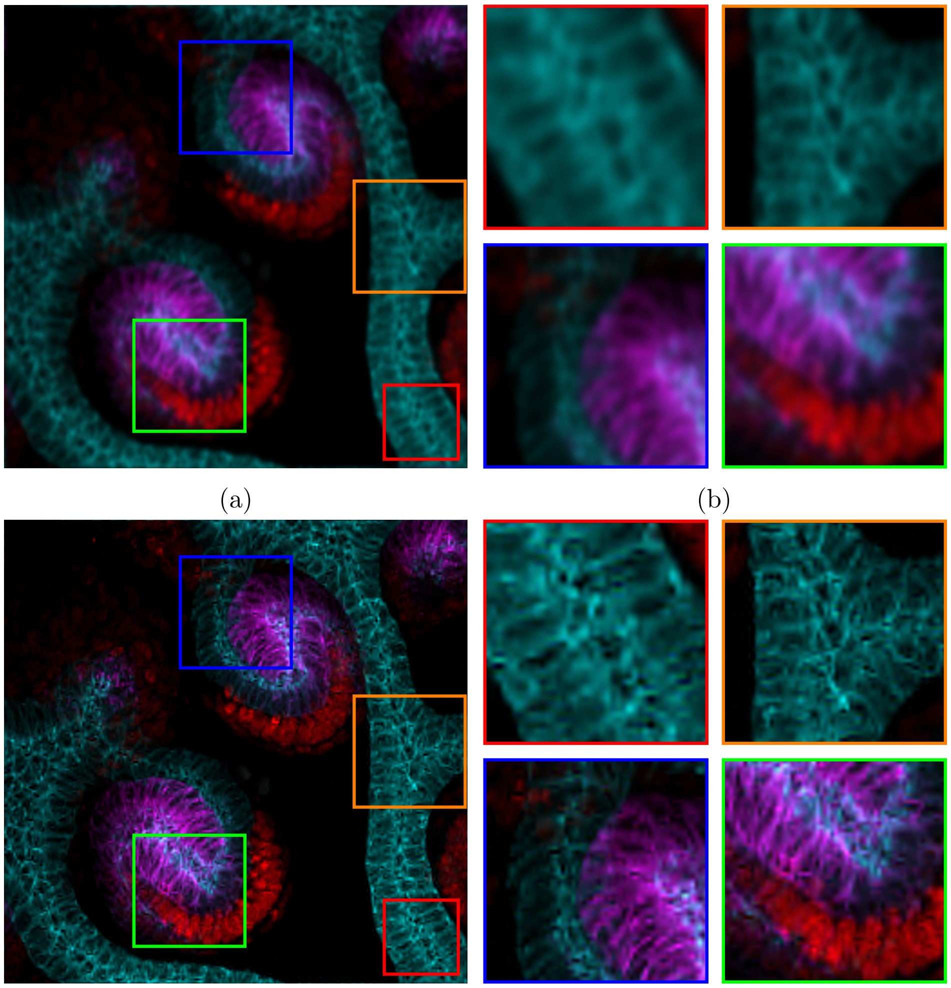


(c)



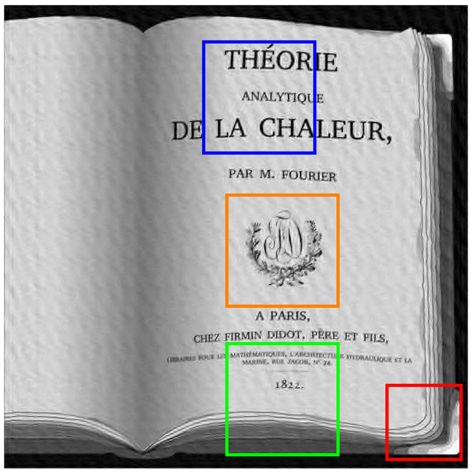
(d)

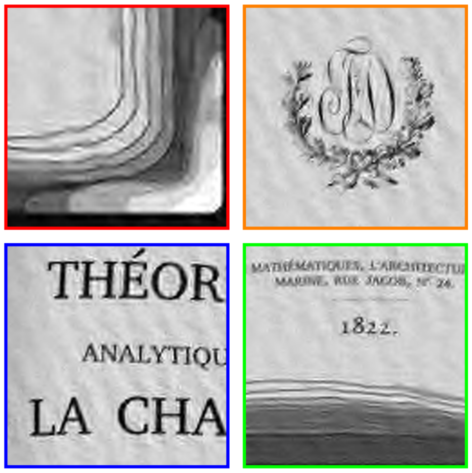
Figure 5: A deconvolution example. The book image in Figure 3 is blurred with the motion blur Figure 4b and degraded with a noise level of 5.10^{-3}. The pSNR of the degraded image (on top) is 17.85dB. Problem (2) is solved using the exact operator, $\lambda = 10^{-4}$, 500 iterations and Symmlet 6 wavelets decomposed 6 times. The pSNR of the restored image is 24.14dB.

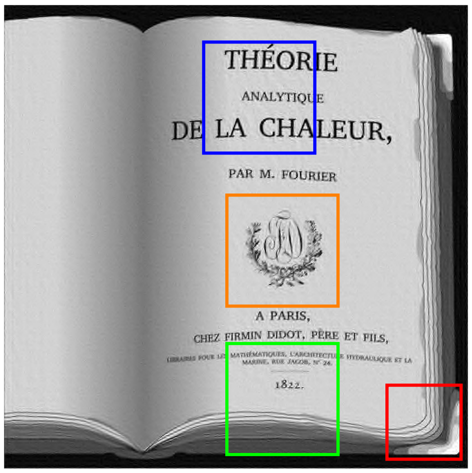


(a) (b) (c) (d)

Figure 6: A deconvolution example. The confocal image Figure 3 has been blurred with blur Figure 4a and degraded with a noise level of 5.10^{-3}. The pSNR of the degraded image (on top) is 23.94dB. Problem (2) is solved using the exact operator, $\lambda = 10^{-4}$, 500 iterations and Symmlet 6 wavelets decomposed 6 times. The pSNR of the restored image is 26.33dB.


(a)


(b)


(c)

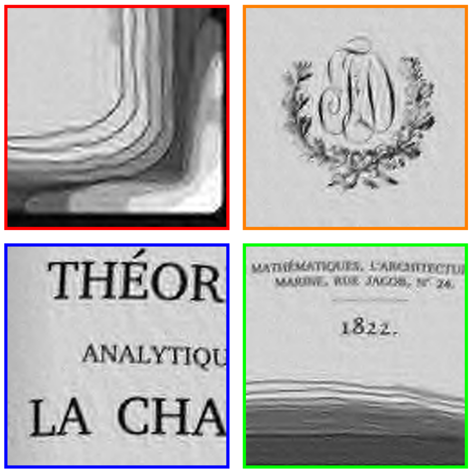

(d)

Figure 7: A deconvolution example showing the importance of the thresholding strategy. The book image on Figure 3 is blurred with the kernel in Figure 4b and degraded with a noise level of 5.10^{-3} (see Figure 5). Matrices have been constructed with the same number of coefficients that corresponds to 57 operations per pixel. Top: the result for the simple thresholding strategy, pSNR = 23.71dB. Bottom: the weighted strategy pSNR = 24.07dB.

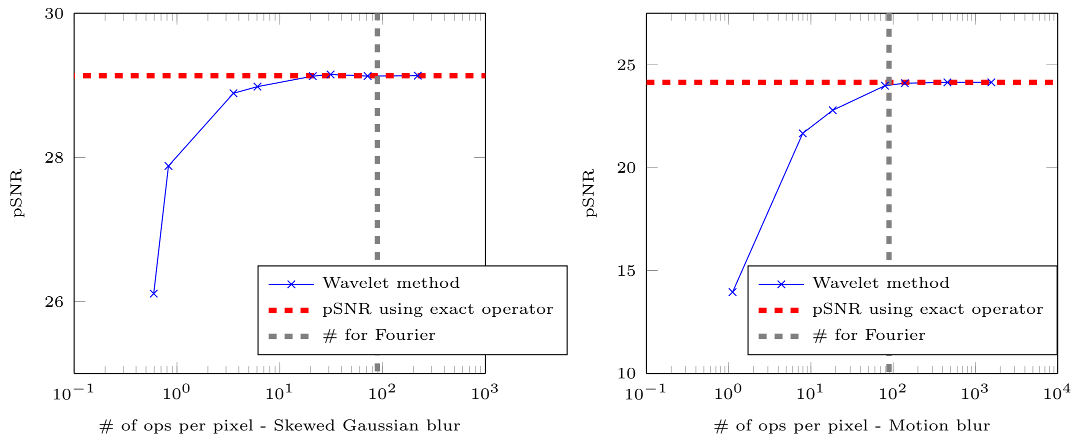


Figure 8: Evolution of the pSNR of the deconvolved image w.r.t. the number of operations per pixel per iteration. The gray vertical line gives the number of operations per pixel per iteration to solve the exact ℓ^1-problem 2 using fast Fourier transforms and fast wavelet transforms. The horizontal line gives the pSNR obtained using the exact operator.

6.3 A competitor: the ADMM

FISTA is probably one of the most popular approaches to solve the ℓ^1 problem (2). Another very effective and popular approach is the alternating direction of multipliers (ADMM) [18, 28]. We therefore compare the proposed approach to the ADMM too.

The problem we wish to solve reads:

$$\min_{u\in\mathbb{R}^N} \frac{1}{2}\|Hu-u_0\|_2^2 + \|\Psi^T u\|_{1,w}. \tag{62}$$

The ADMM can be applied as follows. First use the following splitting:

$$\min_{u\in\mathbb{R}^N, x\in\mathbb{R}^N, x=\Psi^T u} \frac{1}{2}\|Hu-u_0\|_2^2 + \|x\|_{1,w}. \tag{63}$$

The augmented Lagrangian associated to problem (63) is:

$$\mathcal{L}(x,u,\lambda) = \frac{1}{2}\|Hu-u_0\|_2^2 + \|x\|_{1,w} + \langle \lambda, x-\Psi^T u\rangle + \frac{\beta}{2}\|x-\Psi^T u\|_2^2. \tag{64}$$

The ADMM then takes the following algorithmic form:

The first step is a soft-thresholding operation on x. The second step consists of solving a linear system of type:

$$(H^T H + \beta I)u = c, \tag{65}$$

with $c = H^T u_0 - \Psi\lambda + \beta\Psi x$. Solving this linear system can be achieved in $O(N\log(N))$ operations for convolution operators, since $(H^T H + \beta I)$ can be diagonalized using the

Algorithm 4 ADMM to solve (2)

```
1: input: Initial guesses $x^{(0)}$, $u^{(0)}$, $\lambda^{(0)}$ and $Nit$.
2: for $k = 1$ to $Nit$ do
3:     Solve $x^{(k+1)} = \underset{x\in\mathbb{R}^N}{\arg\min}\, \mathcal{L}\left(x, u^{(k)}, \lambda^{(k)}\right)$.
4:     Solve $u^{(k+1)} = \underset{u\in\mathbb{R}^N}{\arg\min}\, \mathcal{L}\left(x^{(k+1)}, u, \lambda^{(k)}\right)$.
5:     $\lambda^{(k+1)} = \lambda^{(k)} + \beta\left(\Psi^T u^{(k+1)} - x^{(k+1)}\right)$.
6: end for
```

discrete Fourier transform. This is no longer the case for spatially varying blurs. In that case, iterative methods such as a conjugate gradient descent can be used.

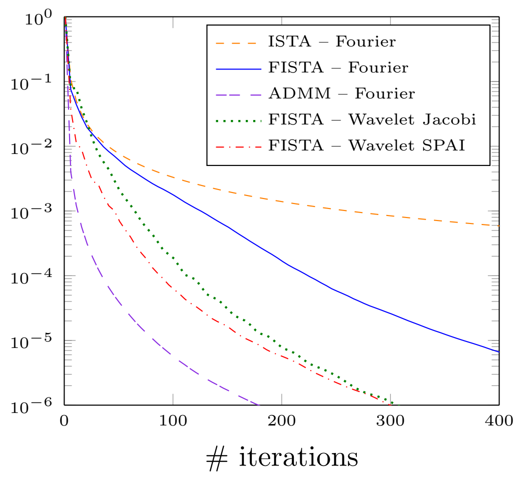
Algorithm 4 can be implemented on the GPU. The linear system is solved with the parallel version of the fast Fourier transform called cuFFT. We used the open-source parallel implementation of the wavelet transform available here `https://github.com/pierrepaleo/PDWT`.

6.4 Comparing preconditioners and ADMM

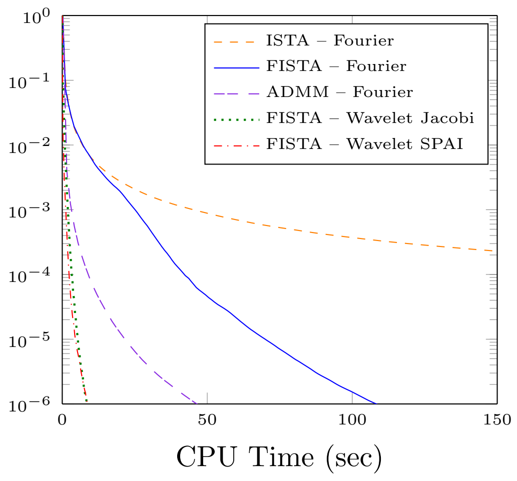
We now illustrate the interest of using the preconditioners described in Section 5. We compare the cost function w.r.t. the iterations number for different methods: ISTA, FISTA, FISTA with a Jacobi preconditioner (see (54)), FISTA with a SPAI preconditioner (see (56)) and the ADMM described in paragraph 6.3. For the Jacobi preconditioner, we optimized ϵ by trial and error in order to get a fixed accuracy in a minimum number of iterations.

As can be seen in Figure 9, the Jacobi and SPAI preconditioners allow reducing the iterations number significantly. We observed that the SPAI preconditioner outperformed the Jacobi preconditioner for all blurs we tested and thus recommend SPAI in general. From a practical point view, a speed-up of a factor 3 is obtained for both blurs.

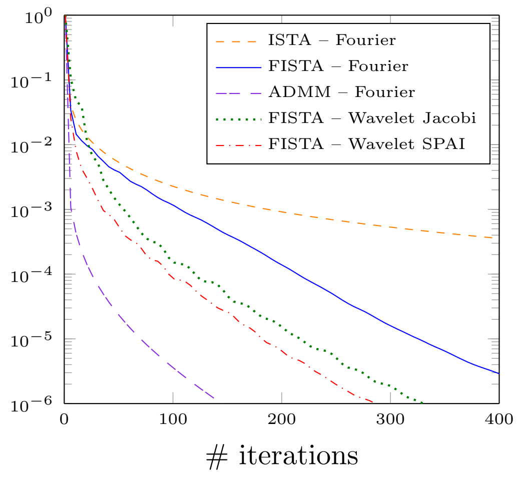
As can be seen, ADMM is significantly faster than preconditioned FISTA for both blurs, at least in the first iterations. The number of iterations necessary to reach a relative accuracy of 10^{-3} is roughly divided by 4. Notice that ADMM and FISTA handle the quadratic term $\frac{1}{2}\|Hu-u_0\|_2^2$ in a very different way: the minimization is basically performed exactly with the ADMM, while it is solved iteratively in FISTA algorithm. This being said, the proposed approach is still significantly faster for smooth kernels. For the skewed Gaussian blur, the proposed algorithm is 4 times faster, since each iteration of the proposed FISTA approach has a significantly lower cost. For the motion blur, the ADMM is twice faster since the number of non-zero components in Θ_K is much higher.



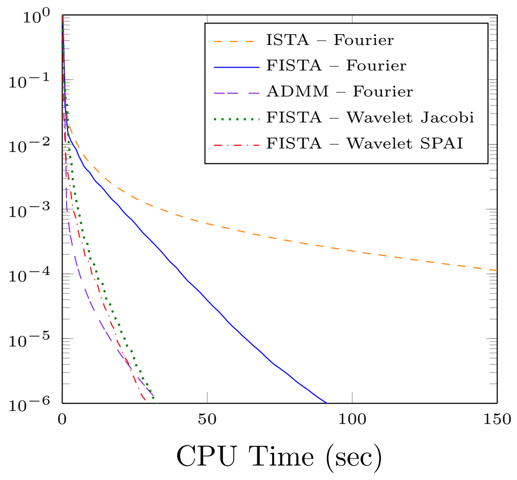
(a) Skewed Gaussian



(b) Skewed Gaussian



(c) Motion blur



(d) Motion blur

Figure 9: Cost function with respect to iterations and time for different preconditioners.

	Exact / GPU	FISTA / GPU	Jacobi / GPU	SPAI / GPU	ADMM / GPU
Iterations number	117	127	55	43	16
Time (in seconds)	24.30 / 2.47	2.57 / 0.43	1.31 / 0.19	1.16 / 0.15	4.22 / 0.59

Table 2: Timing and iterations number for a deconvolution experiment with different methods. The number of operations per pixel is 2.46. This experiment corresponds to the Skewed Gaussian blur in Figure 6.

6.5 Computing times

In this paragraph, we time precisely what can be gained using the proposed approach on the two examples of Figure 6 and 5. The proposed approach consists of:

- Finding a number K such that deconvolving the image with matrix Θ_K instead of Θ leads to a decrease of pSNR of less than 0.2dB.
- For each optimization method, finding a number of iterations Nit leading to a precision

$$E(x^{(Nit)}) - E(x^*) \leq 10^{-3} E(x^{(0)}). \tag{66}$$

In all experiments, matrix Θ_K is computed off-line, meaning that we assume it is known beforehand. In all the paper, the "speed-up" is defined as the ratio between the time needed by the standard FISTA algorithm to reach a given accuracy, divided by the time needed by the proposed algorithm to reach the same accuracy.

The results are displayed in Table 2 for the skewed Gaussian blur and in Table 3 for the motion blur. For the skewed Gaussian, the total speed-up is roughly 162, which can be decomposed as: sparsification = 7.8, preconditioning = 2.7, GPU = 7.7. For the motion blur, the total speed-up is roughly 32, which can be decomposed as: sparsification = 1.01, preconditioning = 3, GPU = 10.5.

As can be seen from this example, the proposed sparsification may accelerate computations significantly for smooth enough blurs. On these these two examples, the preconditioning led to an acceleration of a factor 3. Finally, GPU programming allows accelerations of a factor 7-8, which is on par with what is usually reported in the literature.

Note that for the smooth blurs encountered in microscopy, the total computing time is 0.17 seconds for a 1024×1024 image, which can be considered as real-time.

6.6 Dependency on the blur kernel

In this paragraph, we analyze the method behavior with respect to different blur kernels. We consider 5 different types of kernels commonly encountered in applications: Gaussian blur, skewed Gaussian blur, motion blur, Airy pattern and defocus blur. For each type, we consider two different widths ($\sigma = 2.5$ and $\sigma = 5$). The blurs are shown in Figure 10.

	Exact	GPU	FISTA	GPU	Jacobi	GPU	SPAI	GPU	ADMM	GPU
Iterations number	107		107		52		36		8	
Time (in seconds)	20.03	1.67	16.7	1.82	7.48	0.89	6.54	0.62	1.95	0.28

Table 3: Timing and iterations number depending on the method. The number of operations per pixel is 39.7. This experiment corresponds to the motion blur in Figure 5.

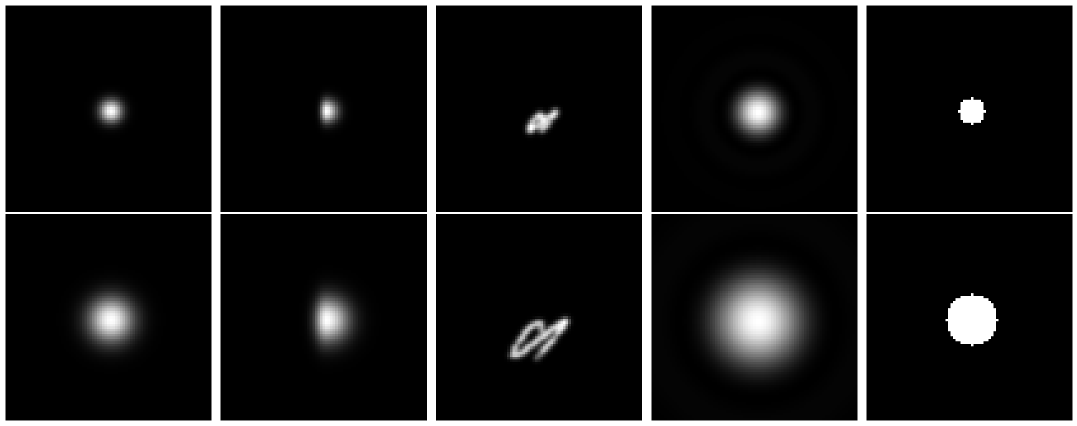

(a) Gaussian (b) Skewed (c) Motion (d) Airy (e) Defocus

Figure 10: The different blurs used to analyze the method's efficiency.

Table 4 summarizes the acceleration provided by using simultaneously the sparse wavelet approximation, SPAI preconditioner and GPU programming. We used the same protocol as Section 6.5. The acceleration varies from 218 (large Airy pattern) to 19 (large motion blur). As expected, the speed-up strongly depends on the kernel smoothness. Of interest, let us mention that the blurs encountered in applications such as astronomy or microscopy (Airy, Gaussian, defocus) all benefit greatly from the proposed approach. The acceleration factor for the least smooth blur, corresponding to the motion blur, still leads to a significant acceleration, showing that the proposed methodology can be used in nearly all types of deblurring applications.

6.7 Dependency on resolution

In this paragraph, we aim at illustrating that the method efficiency increases with resolution. To this end, we deconvolve the phantom in [20] with resolutions ranging from 512×512 to 4096×4096. The convolution kernel is a Gaussian. Its standard deviation is chosen as $\sigma = 2^L/200$, where 2^L is the number of pixels in each direction. This choice is the natural scaling that ensures resolution invariance. We then reproduce the experiment of the previous section to evaluate the speed-up for each resolution. The results are dis-

Blur	Time (Fourier)	Time (Proposed)	Speed-up	# ops per pixels
Gaussian (small)	14.8	0.15	99	4
Gaussian (large)	17.8	0.14	127	2
Skewed (small)	11.2	0.11	100	10
Skewed (large)	10.5	0.1	102	4
Motion (small)	6.0	0.26	23	80
Motion (large)	9.7	0.51	19	80
Airy (small)	15.16	0.081	187	4
Airy (large)	18.6	0.085	218	2
Defocus (small)	20.2	0.23	87	20
Defocus (large)	21.89	0.20	110	10

Table 4: Speed-up of ℓ^1-ℓ^2 deconvolution with respect to the different blur kernels, see Figure 10.

Resolution	512	1024	2048	4096
Time (Fourier)	3.19	17.19	76	352
Time Wavelet + GPU + SPAI	0.07	0.25	0.55	1.35
Total Speed-up	44	70	141	260
Speed-up sparse	4.1	4.5	9.6	9.7
Speed-up SPAI	2.4	2.2	2.1	2.7
Speed-up GPU	4.5	7.1	7.0	10.0

Table 5: Speed-up of ℓ^1-ℓ^2 deconvolution with respect to the image resolution.

played in Table 5. As can be seen, the speed-up increases significantly with the resolution, which could be expected, since as resolution increases, the kernel's smoothness increases. Of interest, note that 1.35 seconds is enough to restore a 4096×4096 image.

6.8 Spatially varying blurs

Finally, we show that the proposed ideas can be successfully applied to spatially varying blurs. In fact, with the proposed method, *computing times do not depend on whether the blur is spatially varying or spatially invariant.* We work on 512×512 images. The confocal image -Figure 3 - has been degraded with a spatially varying blur presented in Figure 11. This blur models realistic degradations appearing in optical systems [32]. The matrix H associated to this blur contains about $537N$ non-zero coefficients. The aim of this section is to compare the performance of the following methods:

- Problem (6) solved using FISTA algorithm and the exact operator implemented with a sparse matrix, see Algorithm 1.

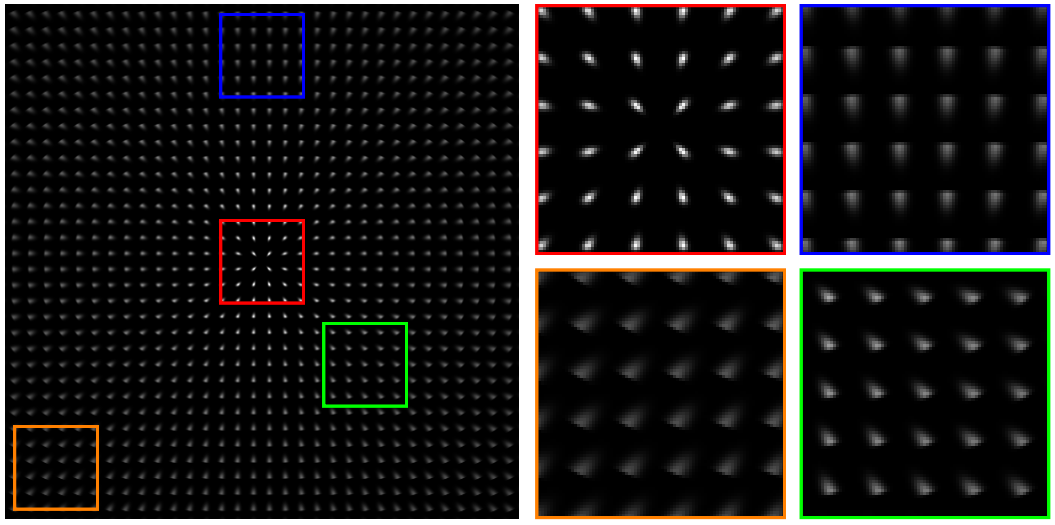
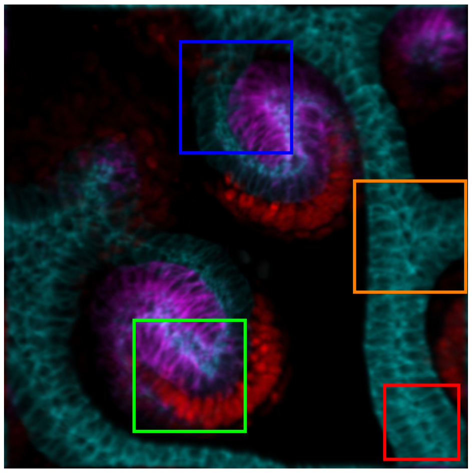

(a) (b)

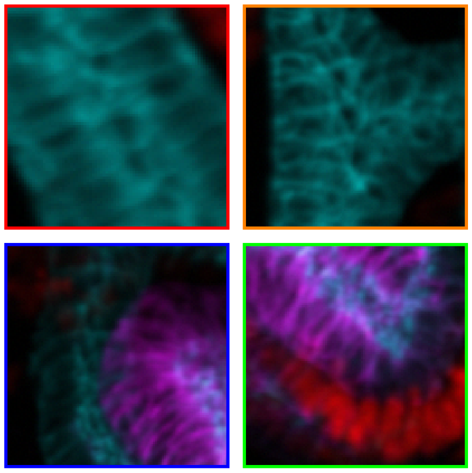
Figure 11: An illustration of the spatially varying blur used in the numerical experiments. The blurring operator is applied to a Dirac comb to obtain the PSFs at various locations. This blur models realistic degradations appearing in optical systems [32].

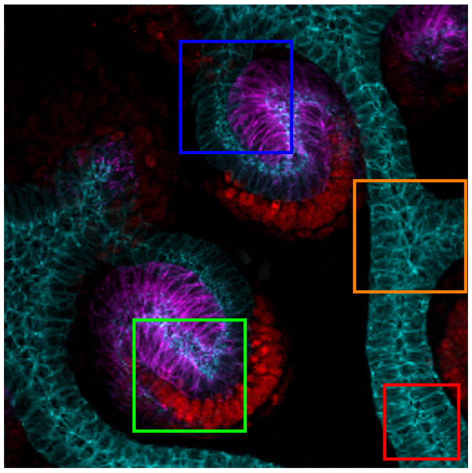
- Problem (6) solved using ADMM algorithm and the exact operator implemented with a sparse matrix. The linear system is solved with a conjugate gradient descent initialized with an initial guess corresponding to the solution at the previous iteration and a stopping criterion optimized so as to get the fastest convergence.
- Problem (46) solved using FISTA algorithm, with an operator approximated in the wavelet domain and a Jacobi preconditioner, i.e. Algorithm 3.

Similarly as the previous section, the number of coefficients K is chosen such that deblurring the image with matrix Θ_K instead of Θ leads to a decrease of pSNR of less than 0.2dB. To reach this we had to set $K = 15N$ which is about 36 times smaller than the number of coefficients in H.

The deblurred image is displayed in Figure 12. The performance of each algorithm is provided on Figure 13. The ADMM algorithm has a faster convergence than FISTA. However each iteration involves the resolution of a linear system with a conjugate gradient descent which considerably increases computing times. FISTA algorithm implemented in the spatial domain involves matrix-vector products with H, which - again - leads to slow computing times. Table 6 shows that *the proposed method is 368 times faster* than a FISTA implementation on a CPU and 73 times faster than FISTA implemented on a CPU. The ADMM is even slower due to expensive inner iterations to solve the linear systems.


(a)


(b)


(c)

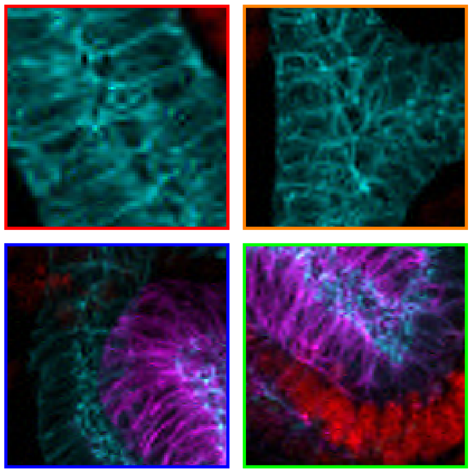

(d)

Figure 12: A restoration of images degraded by spatially varying blurs. The confocal image - 512×512 - in Figure 3 has been blurred with the operator in Figure 11 and degraded with a noise level of 5.10^{-3}. The pSNR of the degraded image (on top) is 22.3dB. Problem (2) is solved using the exact operator, $\lambda = 10^{-4}$, 500 iterations and Symmlet 6 wavelets decomposed 6 times. The pSNR of the restored image is 27.29dB.

	GPU FISTA Spatial	GPU FISTA Wavelet	GPU ADMM
Iterations number	110	44	6
Time (in seconds)	9.54 47.8	0.13 0.73	21.5 107.3

Table 6: Timing and iterations number for a spatially varying blur with different methods.

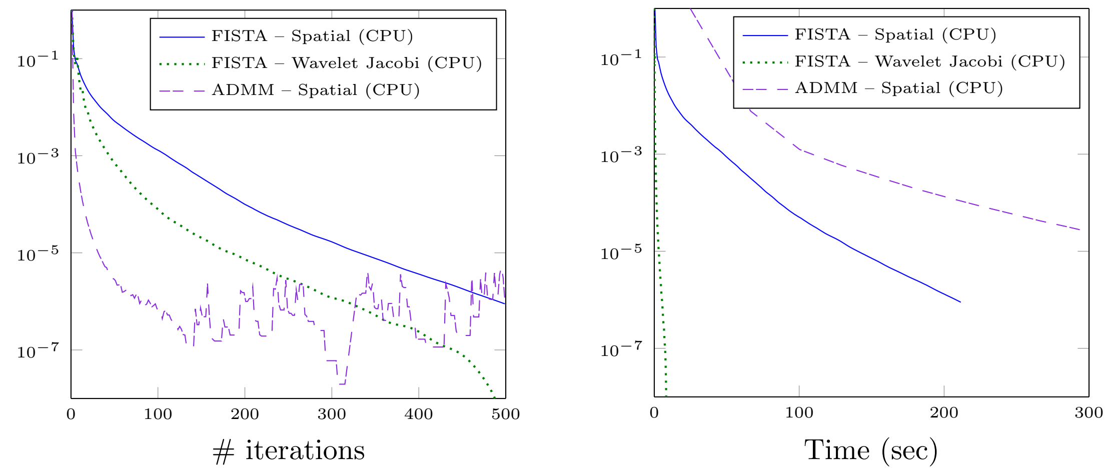


Figure 13: Illustration of the performance of the three deblurring methods. The cost functions are displayed with respect to the number of iterations on the left and the time on the right.

7 Conclusion and outlook

We have shown in this paper that old ideas [25, 5] from applied harmonic analysis provide many insights on deblurring inverse problems. Based on these results and more recent advances [15], we devised a new numerical algorithm to minimize simple $\ell^1 - \ell^2$ functionals involving orthogonal wavelet transforms. It is based on: an operator representation in the wavelet domain, a diagonal preconditioner and a parallel implementation. It solves the deblurring problem with a sufficient accuracy in extremely competitive computing times: of the order of 0.1 second for 1024×1024 images. In addition, the proposed approach allows to deal with spatially varying blurs with the same algorithm and efficiency. The method seems to outperform state-of-the-art algorithms for smooth convolution kernels. It currently might have no rival for spatially varying blurs in terms of computing times.

The methodology is currently limited to orthogonal wavelet transforms without additional constraints on the image such as positivity. Extending the method to frames and additional constraints would make the method more attractive in applications. This will be the scope of future works.

Acknowledgments

The authors wish to thank Manon Dugué for a preliminary numerical study of the proposed idea. They thank Sandrine Anthoine, Caroline Chaux, Hans Feichtinger, Clothilde Mélot and Bruno Torrésani for fruitful discussions and encouragements, which motivated the authors to work on this topic. They thank the anonymous reviewers and the referee for their constructive comments.

References

[1] G. Aubert and P. Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer Science & Business Media, 2006.

[2] W. Ballhaus, J. Casani, S. Dorfman, D. Gallagher, G. Illingworth, J. Klineberg, and D. Schurr. James Webb Space Telescope (JWST) Independent Comprehensive Review Panel (ICRP). 2010.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal recovery. *Convex Optimization in Signal Processing and Communications*, 2009.

[5] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms i. *Communications on pure and applied mathematics*, 44(2):141–183, 1991.

[6] J.-F. Cai and Z. Shen. Framelet based deconvolution. *J. Comput. Math*, 28(3):289–308, 2010.

[7] A. Chai and Z. Shen. Deconvolution: A wavelet frame approach. *Numerische Mathematik*, 106(4):529–587, 2007.

[8] A. Chambolle and C. Dossal. On the convergence of the iterates of FISTA. *Preprint hal-01060130, September*, 2014.

[9] E.-C. Chang, S. Mallat, and C. Yap. Wavelet foveation. *Applied and Computational Harmonic Analysis*, 9(3):312–335, 2000.

[10] C. Chaux, P. L. Combettes, J.-C. Pesquet, and V. R. Wajs. A variational formulation for frame-based inverse problems. *Inverse Problems*, 23(4):1495, 2007.

[11] A. Cohen. *Numerical analysis of wavelet methods*, volume 32. Elsevier, 2003.

[12] A. Cohen, I. Daubechies, and P. Vial. Wavelets on the interval and fast wavelet transforms. *Applied and Computational Harmonic Analysis*, 1(1):54–81, 1993.

[13] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[14] I. Daubechies. *Ten Lectures on Wavelets.* SIAM, June 1992.

[15] P. Escande and P. Weiss. Sparse wavelet representations of spatially varying blurring operators. *SIAM Journal on Imaging Science*, 2015.

[16] M. A. Figueiredo and R. D. Nowak. An EM algorithm for wavelet-based image restoration. *Image Processing, IEEE Transactions on*, 12(8):906–916, 2003.

[17] M. Fornasier, Y. Kim, A. Langer, and C.-B. Schonlieb. Wavelet decomposition method for l_2//tv-image deblurring. *SIAM Journal on Imaging Sciences*, 5(3):857–885, 2012.

[18] R. Glowinski. *Numerical methods for non-linear variational problems.* Springer, 1984.

[19] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[20] M. Guerquin-Kern, L. Lejeune, K. P. Pruessmann, and M. Unser. Realistic analytical phantoms for parallel magnetic resonance imaging. *Medical Imaging, IEEE Transactions on*, 31(3):626–636, 2012.

[21] J. Liang, J. Fadili, and G. Peyré. Activity identification and local linear convergence of forward-backward-type methods. *arXiv preprint arXiv:1503.03703*, 2015.

[22] F. Malgouyres. A Framework for Image Deblurring Using Wavelet Packet Bases. *Applied and Computational Harmonic Analysis*, 12(3):309–331, 2002.

[23] S. Mallat. *A Wavelet Tour of Signal Processing – The Sparse Way.* Third Edition. Academic Press, 2008.

[24] Y. Meyer. Wavelets and operators. *Analysis at Urbana*, 1:256–365, 1989.

[25] Y. Meyer and R. Coifman. *Wavelets: Calderón-Zygmund and multilinear operators*, volume 48. Cambridge University Press, 2000.

[26] R. Neelamani, H. Choi, and R. Baraniuk. Forward: Fourier-wavelet regularized deconvolution for ill-conditioned systems. *Signal Processing, IEEE Transactions on*, 52(2):418–433, 2004.

[27] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[28] M. K. Ng, P. Weiss, and X. Yuan. Solving constrained total-variation image restoration and reconstruction problems via alternating direction methods. *SIAM journal on Scientific Computing*, 32(5):2710–2736, 2010.

[29] P. Petrushev, A. Cohen, H. Xu, and R. A. DeVore. Nonlinear approximation and the space $BV(\mathbb{R}^2)$. *American Journal of Mathematics*, 121(3):587–628, 1999.

[30] N. Pustelnik, A. Benazza-Benhayia, Y. Zheng, and J.-C. Pesquet. Wavelet-based Image Deconvolution and Reconstruction. Jun 2015.

[31] Y. Saad. *Iterative methods for sparse linear systems.* Siam, 2003.

[32] J. Simpkins and R. L. Stevenson. Parameterized modeling of spatially varying optical blur. *Journal of Electronic Imaging*, 23(1):013005–013005, 2014.

[33] J.-L. Starck, M. K. Nguyen, and F. Murtagh. Wavelets and curvelets for image deconvolution: a combined approach. *Signal Processing*, 83(10):2279–2283, 2003.

[34] J.-L. Starck, E. Pantin, and F. Murtagh. Deconvolution in astronomy: A review. *Publications of the Astronomical Society of the Pacific*, 114(800):1051–1069, 2002.

[35] S. Tao, D. Boley, and S. Zhang. Local linear convergence of ISTA and FISTA on the lasso problem. *arXiv preprint arXiv:1501.02888*, 2015.

[36] C. Vonesch and M. Unser. A fast thresholded Landweber algorithm for wavelet-regularized multidimensional deconvolution. *Image Processing, IEEE Transactions on*, 17(4):539–549, 2008.

[37] C. Vonesch and M. Unser. A fast multilevel algorithm for wavelet-regularized image restoration. *Image Processing, IEEE Transactions on*, 18(3):509–523, 2009.

[38] R. Wang and D. Tao. Recent progress in image deblurring. *arXiv preprint arXiv:1409.6838*, 2014.

[39] J. Wei, C. Bouman, J. P. Allebach, et al. Fast space-varying convolution using matrix source coding with applications to camera stray light reduction. *Image Processing, IEEE Transactions on*, 23(5):1965–1979, 2014.

[40] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series*, volume 2. MIT press Cambridge, MA, 1949.