HAL
open science

Fast Wavelet Decomposition of Linear Operators through Product-Convolution Expansions

Paul Escande, Pierre Weiss

To cite this version:

Paul Escande, Pierre Weiss. Fast Wavelet Decomposition of Linear Operators through Product-Convolution Expansions. IMA Journal of Numerical Analysis, 2020, 10.1093/imanum/draa072 . hal-02612434v3

HAL Id: hal-02612434

https://hal.science/hal-02612434v3

Submitted on 5 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fast Wavelet Decomposition of Linear Operators through Product-Convolution Expansions

Paul Escande * Pierre Weiss †

July 22, 2020

Abstract

Wavelet decompositions of integral operators have proven their efficiency in reducing computing times for many problems, ranging from the simulation of waves or fluids to the resolution of inverse problems in imaging. Unfortunately, computing the decomposition is itself a hard problem which is oftentimes out of reach for large scale problems. The objective of this work is to design fast decomposition algorithms based on another representation called product-convolution expansion. This decomposition can be evaluated efficiently assuming that a few impulse responses of the operator are available, but it is usually less efficient than the wavelet decomposition when incorporated in iterative methods. The proposed decomposition algorithms, run in quasi-linear time and we provide some numerical experiments to assess its performance for an imaging problem involving space varying blurs.

1 Introduction

The efficient computation of linear operators and their inverses is paramount for nearly any scientific computing problem. A large number of numerical approaches have been developed over the years, such as low rank decompositions, product-convolution expansions [12], hierarchical matrices [20] or wavelet decompositions [24, 3]. They can yield huge speed-ups for the practical resolution of problems ranging from partial differential equations [6, 28] to inverse problems [11]. We can also expect these ideas to play a growing role in the design of efficient neural networks [14].

A serious hindrance to their popularization is however the high cost to perform the decomposition in a large scale setting. For a square matrix in $\mathbb{R}^{N\times N}$, the cost of a naive decomposition is typically $O(N^3)$, which is incompatible with many current numerical challenges which frequently satisfy $N \gg 10^6$. The main objective of this paper is to reduce the computational burden of decomposing an operator in an orthogonal wavelet basis using an alternative decomposition called product-convolution.

Product-convolution expansions Let $\mathcal{E}$ denote a finite dimensional vector space of discrete d dimensional functions. The functions in $\mathcal{E}$ are defined on $\Omega = \{1, \dots, N_1\} \times \dots \times \{1, \dots, N_d\}$ with $N = \prod_{i=1}^d N_i$. Throughout the paper we will work with periodic boundary conditions, i.e. consider that Ω is a discretization of the torus $\mathbb{T}^d$. Consider a linear

*Aix Marseille Univ, CNRS, Centrale Marseille, I2M, Marseille, France, paul.escande@univ-amu.fr

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

operator G defined for all $f \in \mathcal{E}$ by

$$G : f \mapsto \sum_{y \in \Omega} K(\cdot, y) f(y), \tag{1.1}$$

where $K \in \mathbb{R}^{N \times N}$ is the matrix form (the kernel) of the operator. If all the impulse responses $S(\cdot, y) = K(\cdot + y, y)$ are well approximated by their projections on a low-dimensional subspace $\mathcal{U} = \operatorname{span}(u_k, 1 \leq k \leq m)$, it is possible to construct a product-convolution expansion H of G defined as

$$Hf = \sum_{k=1}^{m} u_k \star (v_k \odot f), \tag{1.2}$$

where $\star$ denotes the convolution operator, $u_k \in \mathcal{E}$ is a convolution filter, $\odot$ denotes the point-wise multiplication and $v_k \in \mathcal{E}$ is a multiplier. The multipliers $\mathcal{V} = (v_1, \ldots, v_m)$ can be defined as the projections of the impulse responses on $\mathcal{U}$, i.e. $v_k(y) = \langle S(\cdot, y), u_k \rangle$ for all $y \in \Omega$, see [12]. From a numerical perspective, the expansion (1.2) can be evaluated efficiently using fast Fourier transforms with a complexity $O(mN \log(N))$.

Product-convolution expansions have appeared at least 3 decades ago and have been given different names in various fields, see e.g. [5, 25, 15, 17, 10, 1]. A remarkable aspect of these expansions is their ability to interpolate linear operators from scattered impulse responses: the bases $\mathcal{U}$ and $\mathcal{V}$ can be evaluated efficiently for operators with slowly varying impulse responses. Assuming that a few impulse responses $S(\cdot, y_i)$ are known at scattered locations $y_i \in \Omega$, it is possible to evaluate the basis $\mathcal{U}$ using a principal component analysis and to interpolate the projection coefficients over the domain Ω to obtain the multipliers $\mathcal{V}$. This process not only provides a minimax estimate of the operator G [4], but also a compact representation compatible with fast numerical algorithms. We refer the reader to [12] and Section 3.3 of this paper for more insight on the construction and approximation properties of this decomposition.

Wavelet decompositions On the other hand, multi-scale representations of integral operators have led to practical breakthroughs in the theoretical and numerical analysis of partial differential equations and inverse problems. Despite their considerable impact for the compact representation of functions, especially in the field of signal processing, the role of wavelets for coding operators still seems marginal, at least at an industrial level.

A possible explanation is the high cost related to the computation of the wavelet representation of an operator. Let $(\psi_\lambda)_{\lambda \in \Lambda}$ denote an orthogonal wavelet basis and $\Psi^* : \mathcal{E} \to \mathbb{R}^N$ denote the associated forward wavelet transform. The wavelet representation of G in the wavelet basis is the matrix $\Theta = (\theta[\lambda, \mu])_{\lambda, \mu \in \Lambda}$ of coefficients $\theta[\lambda, \mu] = \langle G\psi_\lambda, \psi_\mu \rangle$. For an arbitrary operator G, computing these coefficients has a complexity of order $O(N^3)$ operations since the operator G needs to be applied to each of the N wavelets. This cost is prohibitive for large N.

The contribution We address this issue by providing a fast decomposition algorithm for product-convolution operators. Its complexity roughly scales as $O(mN \log_2^2 N \eta^{-\alpha})$ for any arbitrary precision $\eta > 0$, where $\alpha > 0$ is a quantity that depends on the smoothness of the operator and on the number of vanishing moments of the wavelet basis.

One may wonder what is the interest of using a wavelet decomposition if a product-convolution is already available. The reason comes from the higher numerical efficiency of wavelet decompositions when incorporated in iterative methods. For instance, we showed in [11, 13] that the simultaneous sparsity of operators and images in the same orthogonal

wavelet basis could be leveraged to accelerate the resolution of inverse problems by one or two orders of magnitude. The gain comes from two complementary facts:

- the method can operate in the wavelet domain only and avoid to continuously swap between different domains such as the spatial domain, the Fourier domain and the wavelet domain.
- In addition, efficient diagonal preconditioners can be designed since the matrix Θ is concentrated mostly along its diagonal. This idea is often referred to as a multi-level preconditioner [7].

We will illustrate the power of these ideas on a practical image deblurring problem.

2 Preliminaries

Let $\mathcal{E}$ denote the linear space of d-dimensional discretized functions of N samples defined on $\Omega = \{1, \ldots, N_1\} \times \ldots \times \{1, \ldots, N_d\}$ with $\prod_{i=1}^{d} N_i = N$. This space $\mathcal{E}$ will be often identified with $\mathbb{R}^N$ through a bijective mapping $\omega : \Omega \to \{1, \ldots, N\}$. To simplify the discussion, we assume that the numbers $N_1 = \ldots = N_d = 2^J$ and we use circular boundary conditions. Assuming that the number of pixels is large enough, methods and analyses proposed in this work can be extended to any boundary conditions by using boundary wavelets.

We let $f(x)$ denote the value of a function f at x, $v[i]$ denote the i-th coefficient of a vector v and $A[i, j]$ denote the (i, j)-th element of a matrix A.

Let A be an $N \times N$ matrix, we let $\operatorname{supp} A$ denote its support i.e. the set of indexes associated to a non-zero coefficient: $\operatorname{supp} A = \big\{(i, j) \in \{1 \ldots N\}^2 \,|\, A[i, j] \neq 0\big\}$. The approximation rates stated in this paper will be expressed with respect to the spectral norm $\| \cdot \|_{2\to 2}$ defined by

$$\|A\|_{2\to 2} = \sup_{f \in \mathbb{R}^N, \|f\|_2 = 1} \|Af\|_2.$$

2.1 One dimensional orthogonal wavelet bases

We first recall the construction of wavelets on the continuous interval $[0, 1]$. Let ϕ and ψ denote the scaling and mother wavelets. Translated and dilated versions of the wavelets are defined, for $j \geq 0$, as follows

$$\phi_{j,n} = 2^{j/2} \phi\left(2^j \cdot - n\right),$$
$$\psi_{j,n} = 2^{j/2} \psi\left(2^j \cdot - n\right),$$

with $n \in \{0, \ldots, 2^j - 1\}$. A mother wavelet ψ is said to have M vanishing moments when

$$\forall 0 \leq m < M, \quad \int_{[0,1]} t^m \psi(t) dt = 0.$$

We now detail the construction of a 1D discrete orthogonal wavelet transform. A function $f : [0, 1] \to \mathbb{R}$ can be discretized leading to a vector $v \in \mathbb{R}^{2^J}$ samples. The Fast Wavelet Transform is derived from the observation that wavelets ϕ and ψ are associated to a filter bank (h, g) [23, Theorem 7.7, p. 348]. From these filters (h, g) the discrete wavelets ϕ_j and ψ_j can be defined recursively using convolutions and sub-sampling. Setting $\phi_{J,n} = \mathbb{1}_{\{n\}}$, we get for $0 \leq j < J$

$$\phi_{j,l} = \sum_{n \in \mathbb{Z}} h[n - 2l] \phi_{j+1,n}, \quad \text{and} \quad \psi_{j,l} = \sum_{n \in \mathbb{Z}} g[n - 2l] \phi_{j+1,n}.$$

We assume that the wavelets are compactly supported i.e. $\mathrm{supp}(\psi) = [-\delta+1, \delta]$. Note that δ is related to the number of vanishing moments of the mother wavelet. Let M be this number, we have $\delta \geq M$, with equality for Daubechies wavelets, see e.g. [23, Theorem 7.9, p. 294]. Together with [23, Theorem 7.5, p. 286] and $g[i] = (-1)^{1-i}h[1-i]$, we deduce that $\mathrm{supp}\, h = \mathrm{supp}\, \phi = [0, 2\delta - 1]$ and $\mathrm{supp}\, g = [-2(\delta-1), 1]$. Therefore for $0 \leq j \leq J$,

$$\begin{aligned} \mathrm{supp}\, \phi_{j,0} &= \left[0, (2^{J-j}-1)(2\delta-1)\right] \quad \text{and} \\ \mathrm{supp}\, \psi_{j,0} &= \left[-(2^{J-j}-1)2(\delta-1), (2^{J-j}-1)\right]. \end{aligned}$$

2.2 Orthogonal wavelet bases on $\mathcal{E}$

In dimension d, we use isotropic separable wavelet bases, see, e.g., [23, Theorem 7.26, p. 348]. Let $l = (l_1, \ldots, l_d)$. Define $\rho^0_{j,n} = \phi_{j,n}$ and $\rho^1_{j,n} = \psi_{j,n}$. Let $e = (e_1, \ldots, e_d) \in \{0,1\}^d$ and $\mathcal{T}_j = \{0, \ldots, 2^j - 1\}^d$. For the ease of reading, we will use the shorthand notation $\lambda = (j, e, l)$ and $|\lambda| = j$. We also let $J = \frac{1}{d}\log_2 N$ and

$$\Lambda = \left\{(j,e,l) \mid 0 \leq j \leq J-1,\ l \in \mathcal{T}_j,\ e \in \{0,1\}^d\right\}. \tag{2.1}$$

The wavelet ψ_λ is defined by $\psi_\lambda(x_1, \ldots, x_d) = \psi^e_{j,l}(x_1, \ldots, x_d) = \rho^{e_1}_{j,l_1}(x_1) \ldots \rho^{e_d}_{j,l_d}(x_d)$.

With these definitions, every signal $f \in \mathcal{E}$ can be written as

$$\begin{aligned} u &= \langle u, \psi^0_{0,0}\rangle \psi^0_{0,0} + \sum_{e \in \{0,1\}^d \setminus \{0\}} \sum_{j=0}^{J-1} \sum_{l \in \mathcal{T}_j} \langle u, \psi^e_{j,l}\rangle \psi^e_{j,l} \\ &= \sum_{\lambda \in \Lambda} \langle u, \psi_\lambda \rangle \psi_\lambda. \end{aligned}$$

Finally, we let $\Psi^* : \mathcal{E} \to \mathbb{R}^N$ denote the discrete forward wavelet transform and Ψ its inverse. We refer to [23, 9, 8] for more details on the construction of wavelet bases.

3 Main results

3.1 Assumptions

The main result will be stated under mild regularity and decay assumptions on the kernels u_k defined below.

Definition 3.1 (Smoothness class $\mathcal{A}_\alpha$ [7, p. 281])**.** A convolution kernel u_k is said to be α-asymptotically smooth if there exists constants $C \geq 0$, $\alpha > 0$ and a compactly supported wavelet basis with $\lceil \alpha \rceil$ vanishing moments such that the following inequality holds:

$$|\langle u_k \star \psi_\lambda, \psi_\mu \rangle| \leq C 2^{-(d/2+\alpha)||\lambda|-|\mu||} \vartheta(\lambda, \mu)^{-(d+\alpha)}, \tag{3.1}$$

where

$$\vartheta(\lambda, \mu) := 1 + 2^{\min(|\lambda|,|\mu|)} \mathrm{dist}(\mathrm{supp}\, \psi_\lambda, \mathrm{supp}\, \psi_\mu)$$

measures the distance between the wavelets supports.

Remark 3.1 (Asymptotic smoothness in a continuous setting). In a continuous setting, it can be shown that sufficient conditions for the kernels u_k to be α-asymptotically smooth is $u_k \in W^{\alpha,\infty}(\mathbb{R}^d)$ with the following decay property

$$|\partial^p u_k(x)| \leq C_k \left(1 + \|x\|_2\right)^{-(|p|+d)}, \tag{3.2}$$

for all multi-indexes p with $|p| \leq \alpha$. Hence, the α smoothness property mostly boils down to regularity properties of the kernel. The larger α, the more regular the kernels.

Our main results will be stated under the following assumptions.

Assumption 3.1 (Regularity of the kernel)**.**

- *the operator H has the form* (1.2)*.*
- *there exists $\alpha > 0$ such that $u_k \in \mathcal{A}_\alpha$ for all $1 \leq k \leq m$.*
- *the mother wavelet is compactly supported on a hypercube of sidelength 2δ.*

Overall the set of assumptions in Assumption 3.1 describes a fairly large variety of operators comprising blurring operators [11], singular integral operators and pseudo-differential operators [7].

3.2 The algorithm and its guarantees

The main objective of this paper is to efficiently compute a sparse approximation $\widetilde{\Theta}$ of the wavelet representation Θ of H defined by

$$\Theta = \Psi^* H \Psi. \tag{3.3}$$

The proposed algorithm heavily relies on the peculiar structure of product-convolution expansions. In what follows, we let U_k denote the convolution operator with u_k, $V_k = \mathrm{diag}(v_k)$ denote the k-th multiplier, $A_k = \Psi^* U_k \Psi$ denote the wavelet representation of U_k and $B_k = \Psi^* V_k \Psi$ denote the wavelet representation of the multiplier. The proposed methodology is described in Algorithm (1).

Algorithm 1 Decomposition of product-convolution in an orthogonal wavelet basis

Require: A precision $\eta > 0$, the
  Set $\epsilon_k = \frac{\eta}{m\|v_k\|_\infty}$ for all $1 \leq k \leq m$
  Set $\widetilde{\Theta}$ an empty sparse matrix
  for all $k = 1 \to m$ **do**
    Compute A_k (by exploiting the convolution structure) ▷ $O(N \log^2 N)$
    Threshold A_k to get an ϵ_k approximation $\widetilde{A}_k$ ▷ $O(N \log N)$
    Construct B_k (using a sparse cascade algorithm) ▷ $O(N \log N)$
    Compute $C_k = \widetilde{A}_k B_k$ ▷ $O(N \log N \epsilon_k^{-d/t})$
    Accumulate $\widetilde{\Theta} \leftarrow \widetilde{\Theta} + C_k$ ▷ $O(N \log N \epsilon_k^{-d/t})$
  end for
  return $\widetilde{\Theta}$ an η-approximation of Θ

Our main theoretical result reads as follows.

Theorem 3.1. *Under Assumption 3.1, Algorithm 1 produces a matrix $\widetilde{\Theta}$ satisfying $\|\Theta - \widetilde{\Theta}\|_{2\to 2} \leq \eta$ in $O\left(\delta^{d+1} m^{\max(1,d/\alpha)} N \log_2^2 N \eta^{-d/\alpha}\right)$ operations. Furthermore, the number of coefficients in $\widetilde{\Theta}$ can also be bounded above by $O\left(\delta^d N \log_2^2 N \eta^{-d/\alpha}\right)$.*

Remark 3.2. In many applications, the kernels of integral operators are smooth. In particular, for $\alpha \geq d$ the term $\max(1, d/\alpha) = 1$. The complexity is thus linear in m. Remark 3.1 shows that this is satisfied when the kernels u_k are of class $W^{d,\infty}$.

Remark 3.3. The bound on the number of coefficients in $\widetilde{\Theta}$ provided by Theorem 3.1 allows to bound the complexity of matrix-vector products with $\widetilde{\Theta}$. If we assume that the operator G belongs to the class $\mathcal{A}_\alpha$, a direct application of [3, 11] (see also Theorem 5.1), shows

that one can construct an η approximation $\widetilde{G}$ of G, containing no more than $O(\delta^d N \eta^{-d/\alpha})$ coefficients. The result of Theorem 3.1 is therefore optimal up to the $\log_2 N$ factor. We believe that it could be discarded by assuming further regularity of the multipliers v_k. We could then use an additional approximation in place of B_k, in the exact same fashion as for the A_k's.

3.3 Getting a product-convolution expansion

While the last two items in Assumption 3.1 are now well established, the combination with the first one may seem problematic. We review some numerical approaches to design product-convolution expansions below.

3.3.1 Naive interpolation of impulse responses

The simplest and probably the most widespread product-convolution expansions in image processing are based on the assumption of slow variations of the impulse responses of the operator in the domain Ω. Under this assumption, it is possible to set

$$u_k = S(\cdot, y_k)$$

i.e. $(u_k)_{k=1}^m$ are the impulse responses of the operator at some locations $(y_k)_{k=1}^m$. By setting the locations $(y_k)_{k=1}^m$ as a coarse Euclidean grid [25], it is possible to keep a small value for m. The functions $(v_k)_{k=1}^m$ are then used to interpolate the impulse responses i.e. they define a partition of unity with the constraints $v_k(y_k) = 1$ for all $1 \leq k \leq m$. The choice of these v_k are discussed in many works, without particular guarantees on the approximation obtained. We refer the interested reader to [10] for a nice overview and [12] for approximation rates.

In the context of PDEs, product-convolution expansions are encountered in Schur complement methods for solving PDEs, Dirichlet-to-Neumann maps and in PDE-constrained optimization as Hessians. In [1], the functions $(u_k, v_k)_{k=1}^m$ are chosen as above, but instead of being defined on a Euclidean grid, locations $(y_k)_{k=1}^m$ are adaptively sampled to best approximate the operator with a fixed number m.

3.3.2 Interpolating expansions of impulse responses

Under mild regularity assumptions of the impulse responses, product-convolution expansions can be designed from scattered impulse responses in a more efficient fashion than what was just described [15, 16, 4].

The main idea is to use a principal component analysis of the observed impulse responses $S(\cdot, y_k)$ to obtain a low-dimensional orthogonal basis (u_k). The coefficient maps (v_k) can then be computed by interpolating the known projection coefficients at the positions (y_k). We showed in [4] that this approach was optimal from an approximation theoretic point of view.

3.3.3 Singular value decomposition

When the operator G in (1.1) is fully known, the optimal way (in Frobenius norm) for fixed m to get a product-convolution expansion of an operator (1.1) is to construct a low-rank approximation of the *SVIR* S (and not the kernel K). For many practical problems, the SVIR S has a much lower rank than the kernel K. A typical example is the identity operator which has a kernel of rank N and an SVIR of rank 1. We refer to [12] for more examples and mathematical insights.

The low-rank decomposition of space varying impulse response can be achieved by performing a singular value decomposition (SVD) on S. For most problems, this approach is however intractable due to the size of matrices S.

Alternatively, we may just assume that the action of an operator with a matrix form S can be evaluated on any vector. The couples $(u_k, v_k)_{k=1}^m$ can now be obtained using a randomized SVD of the matrix S [22]. The complexity of such algorithms scales as $O(m^2 N)$, where m is the number of matrix-product evaluations.

4 Numerical experiments – Spatially varying deblurring

4.1 Description of the operators

To assess the performance of the algorithm, we propose to perform numerical experiments on a large scale image deblurring problem. We synthesize two different blurring operators:

- The first one, in Figure 1a, is made of isotropic Gaussian impulse responses with a variance $\sigma(y_1, y_2)$ increasing along the vertical direction only (i.e. $\sigma(y_1, y_2) = 3y_2$ for $(y_1, y_2) \in [0, 1]^2$). The impulse responses are then truncated out, so that 99% of the Gaussian's energy is kept. This allows to encode the operator as a sparse matrix to accelerate the explicit computation of Θ. Notice that this is however unnecessary to apply our algorithms. As an indication, the largest support has a size 19×19 pixels.

- The second operator, in Figure 1b models realistic degradations appearing in optical systems [29]. The impulse responses are rotated and skewed Gaussian, with parameters depending on their location with respect to the center of the field. As an indication, the support of the impulse responses are of size 25×25.

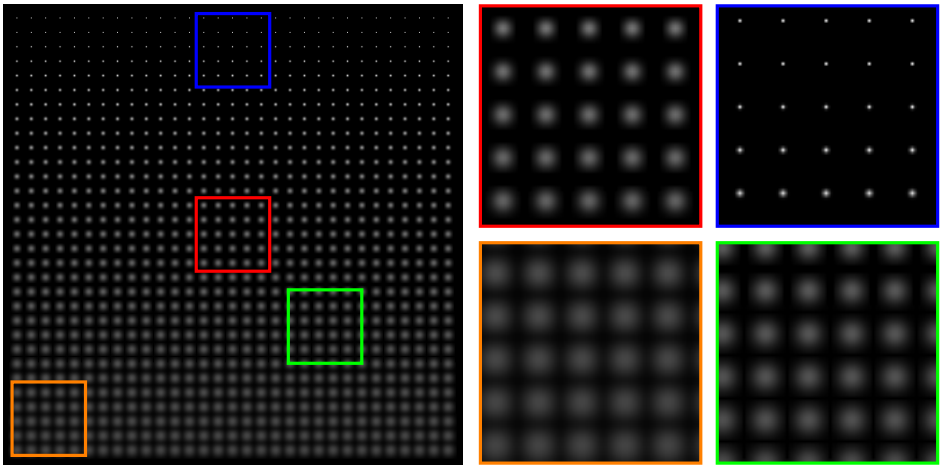
We compute the product-convolution expansions of each of these operators using a randomized SVD of the matrix S as explained in Section 3.3.3. The order m of the expansion has been minimized so as to obtain a fixed deblurring performance, see paragraph 4.3.1. The first operator - Figure 1a - is decomposed with $m = 5$ coefficients, while the second one - Figure 1b - with $m = 25$ coefficients.

4.2 Comparing decomposition timings

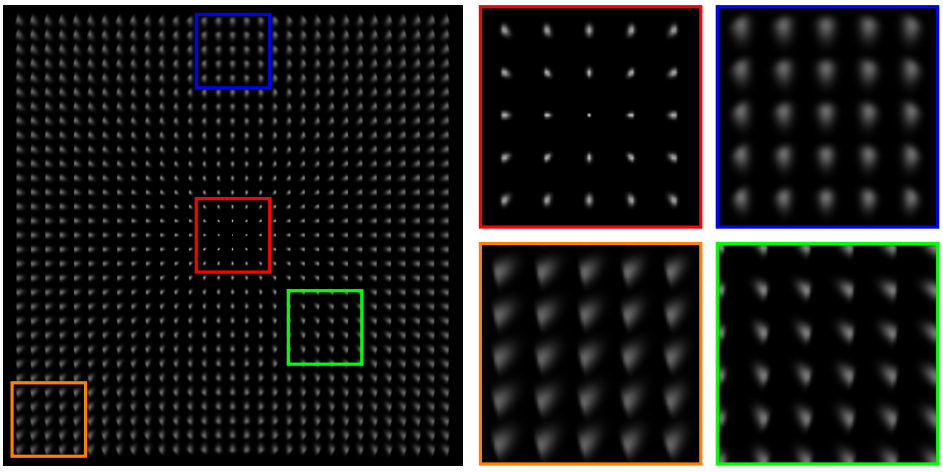
In this section, we compare Algorithm 1 to standard decomposition methods. The direct algorithm to compute Θ from a product-convolution expansion is given in Algorithm 2. It consists in using a matrix-vector product and a wavelet transform for each column. Its complexity is therefore in $O(N^3)$ operations (this complexity is reduced here to $O(N^2)$ since we use the fact that the impulse responses are compactly supported).

Algorithm 2 Naive decomposition of matrices in an orthogonal wavelet basis

for all $\lambda \in \Lambda$ **do**
  Get ψ_λ and compute $w_\lambda = H\psi_\lambda$
  Compute $\Psi^* w_\lambda$ to obtain $(\langle H\psi_\lambda, \psi_\mu \rangle)_{\mu \in \Lambda}$. Set it as the λ-th column of Θ
end for



(a) Operator 1 (applied to a 1024×1024 image).



(b) Operator 2 (applied to a 1024×1024 image)

Figure 1: An illustration of the spatially varying blurs used in the numerical experiments. Blurring operators are applied to a Dirac comb to obtain the impulse responses at various locations.

In this comparison, we used parallel implementations running on 12 cores on a workstation with 256GB of RAM in double precision. Figure 2 compares the computing times using 3 Algorithms:

- Algorithm 2 with a direct computation with a sparse matrix (Alg. 2 – exact).
- Algorithm 2 with a product-convolution expansion (Alg. 2 – PC).
- The proposed Algorithm 1 (Alg. 1).

We use square images containing $N = 2^n \times 2^n$ pixels with $n \in \{7, \dots, 12\}$. The maximal image size is therefore 4096×4096. We run the algorithms with a Symmlet wavelet basis of order 6. This choice is explained in Remark 4.1. The precision $\eta = 5.10^{-4}$ has been chosen so as to obtain good performance for a deblurring experiment. This choice is further detailed in Section 4.3.1.

In this setting, Algorithm 1 is much faster than any instance of Algorithm 2. For $n = 12$, corresponding to images of 16 millions pixels, the speed-up from Alg. 2 – product-convolution to Alg. 1 is 4680 for the operator on Figure 1a and 2330 for the operator on Figure 1b.

We also evaluate the decomposition times for various precisions with $N = 256 \times 256$ being fixed, see Figure 3. Assuming that the clock time is proportional to the number of operations, we deduce that the number of operations is bounded by a quantity proportional to $\eta^{-d/\alpha}$. We fit the curves with a line to obtain an approximation of the smoothness α of each operator.

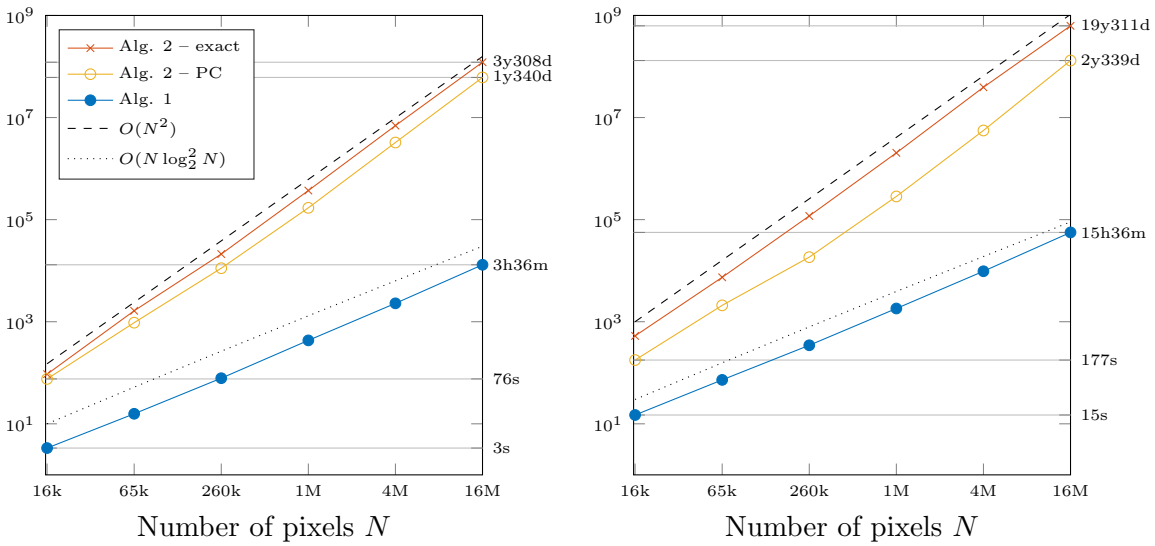


Figure 2: Running times for various number of pixels $N = 2^n \times 2^n$ with $n \in \{7, \dots, 12\}$. Left: for the blurring operator on Figure 1a and Right: for the one on Figure 1b.

Remark 4.1. The choice of the wavelet basis is of crucial importance as it has to simultaneously provide a good representation for the image and for the operator. A key parameter is the number of vanishing moments of the wavelet basis. As suggested by Definition 3.1, the number of vanishing moments should be at least equal to the smoothness α of the convolution kernels u_k in order to get the best possible approximation.

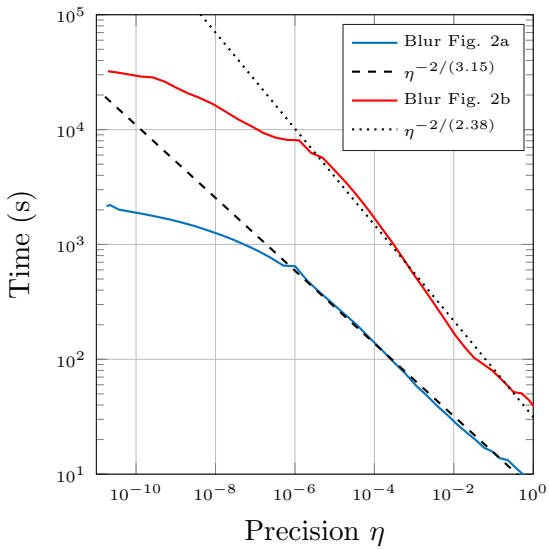


Figure 3: Decomposition times for various precisions and $N = 256 \times 256$.

In [11] we performed extensive numerical experiments of the approximation properties of a wavelet basis w.r.t. the number of vanishing moments. As expected, taking as many moments as possible was preferable for the approximation rate. However, increasing it too much led to insignificant approximation quality while deteriorating the numerical complexity significantly. This is mostly due to the constants of Theorem 5.1 which increase with the number of vanishing moments.

We performed additional numerical simulations - not reported here - to assess the behavior of the complexity of Algorithm 1 w.r.t. the sidelength δ of the wavelet basis. This sidelength is related to the number of vanishing moments M since $\delta \geq M$ [23, Theorem 5.7, p. 286]. Numerically, we observed that the algorithm actually performs better than the rate δ^{d+1} in Theorem 3.1.

4.3 A deblurring experiment

The setting We assume that an observed image $f_0 \in \mathcal{E}$ reads

$$f_0 = Hf + b,$$

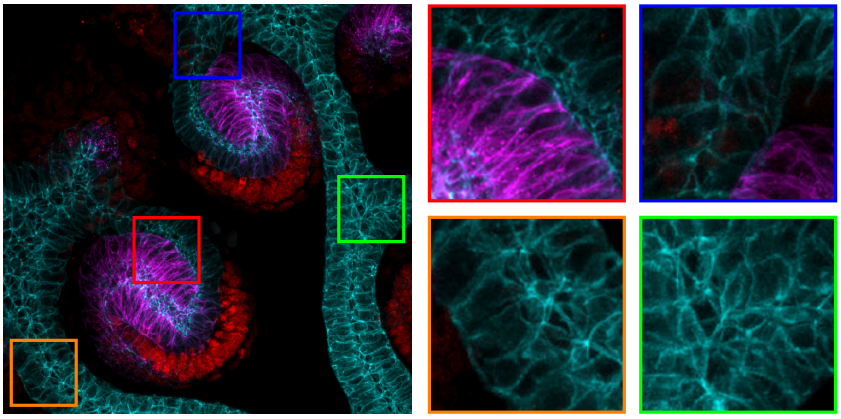
where $f \in \mathcal{E}$ is the clean image to recover, $b \sim \mathcal{N}(0, \sigma^2 \mathrm{Id}_N)$ is a white Gaussian noise of standard deviation σ and $H \in \mathbb{R}^{N \times N}$ is the blurring operator. The image f used in this experiment is shown in Figure 4a and Figure 4 contains the degraded version f_0 with the two operators considered.

The optimization problem In this experiment we will seek to recover the image f by solving the following variational problem

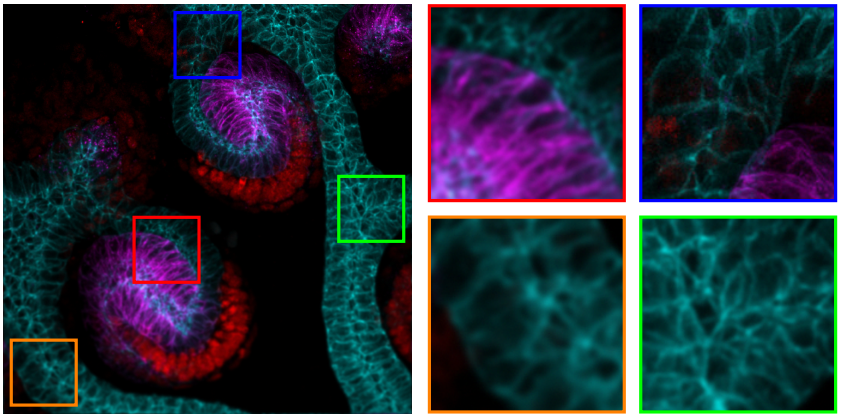
$$\arg\min_{f \in \mathcal{E}} E(f) = \frac{1}{2} \|Hf - f_0\|_2^2 + \|\Psi^* f\|_{1,w}, \tag{4.1}$$

where $\|z\|_{1,w}$ is a weighted ℓ_1-norm with weights $w \in \mathbb{R}^N$ and defined by

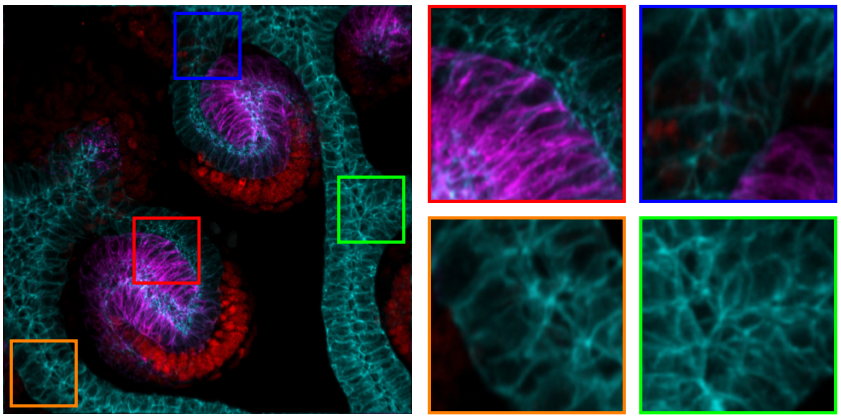
$$\|z\|_{1,w} = \sum_{\lambda \in \Lambda} w[\lambda] \, |z[\lambda]| \, .$$



(a) Original image f – 1024×1024 pixels



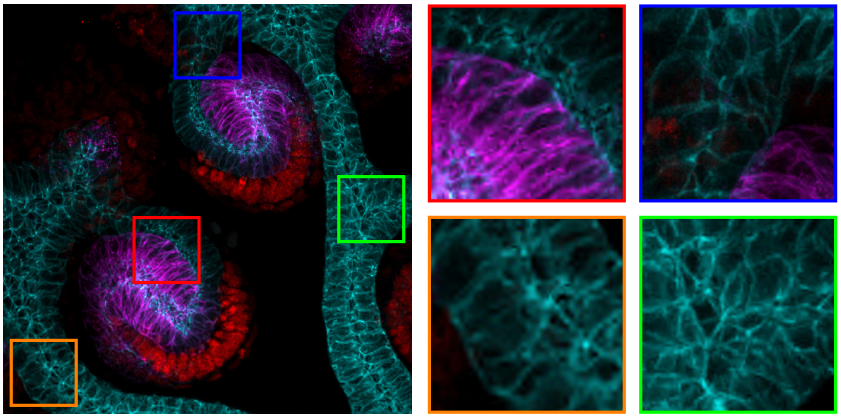
(b) Blurred with blur Figure 1a – SNR = 25.30 dB



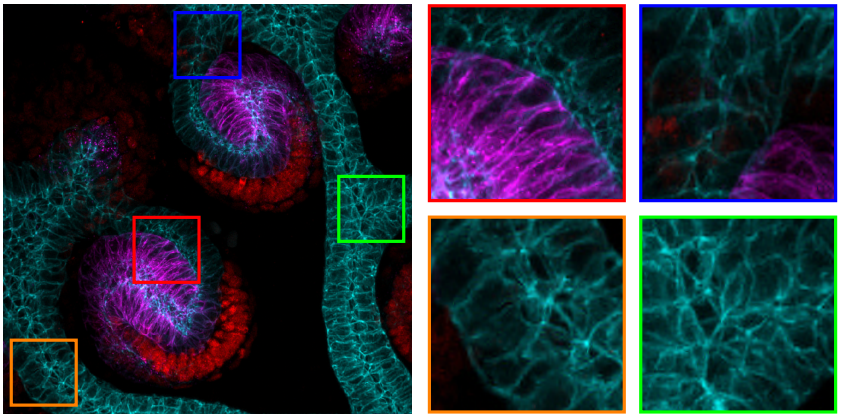
(c) Blurred with blur Figure 1b – SNR = 22.72 dB

Figure 4: Images f_0, degraded versions of f for the two operators considered. Noise level $\sigma = 5.10^{-3}$

The solution of this variational problem does not correspond to the state-of-the-art in terms of image quality since the prior is simple, but it is known to perform well in short computation times. Figure 5 shows the solutions of (4.1) for the two blurs, with Ψ being a Symmlet wavelet basis of order 6, since it offers a good compromise between computing times and visual quality of the results [13]. We use this basis in all the experiments.



(a) Restored image – SNR = 27.66 dB



(b) Restored image – SNR = 29.71 dB

Figure 5: Solutions of (4.1) for the two operators considered. The exact operator is used with $w[\lambda] = 2 \cdot 10^{-2} \cdot |\lambda|$.

The proposed algorithms Many algorithms were designed to solve non-smooth convex problems of the form (4.1). The two predominant algorithms are the (accelerated) proximal gradient descent [26] also known as FISTA [2] and the alternating descent method of multipliers (ADMM) [18].

As shown in [13], a very efficient version of FISTA can be derived by approximating (4.1) in the wavelet domain i.e.

$$\arg\min_{z \in \mathbb{R}^N} \frac{1}{2} \|\Theta_L z - z_0\|_2^2 + \|z\|_{1,w}, \tag{4.2}$$

where $z_0 = \Psi^* f_0$, and Θ_L is an L-sparse approximation of $\Theta = \Psi^* H \Psi$. This idea allows to avoid constantly swapping between the wavelet and spatial domains, where most of the time is spent in traditional solvers. In addition, the matrix $\Theta_L^* \Theta_L$ - which appears in the gradient of the quadratic term - is mostly concentrated around its diagonal that decreases across sub-bands. An efficient diagonal preconditioner can be designed exploiting

this property and further reduce the number of iterations by a factor roughly equal to 2 without compromising image quality.

The ADMM on its side requires the inversion of symmetric positive definite linear systems with matrix $\Theta_L^* \Theta_L + \tau \mathrm{I}_N$. This step could also be accelerated using the same ideas, but we observed that it was less efficient in practice and will not further report about this.

Table 1 compares the performance of various algorithms detailed in more details in Appendix A.

Algorithm	Description	Complexity			
		MVP	FFT	FWT	Diag
FISTA-Spa	FISTA with the exact operator	2	-	2	-
FISTA-PC	FISTA using a PC expansion	-	$2m$	2	$2m$
FISTA-W	FISTA with sparse Θ_L	2	-	-	-
FISTA-WP	FISTA with Θ_L and Jacobi preconditioner	2	-	-	1
ADMM-PC	See Appendix A.2	-	$3m$	3	$6m+2$

Table 1: List of algorithms compared in the numerical experiments with their complexity per iteration in terms of matrix vector products (MVP), fast Fourier transforms (FFT), wavelet transforms (WT) and product with diagonal matrices (Diag).

4.3.1 Choice of m and η

The approximation properties of product-convolution expansions are investigated in [12] in terms of Frobenius distance on the operators. We are dealing here with an inverse problem thus the approximation needs are different. Because of the regularization term in (4.1), fine approximations of the operator are unnecessary, allowing to reduce the computational burden, see Figure 6.

We choose the order of expansion m in a such a way that the pSNR of the deblurred images does not differ from more than 0.01dB from the pSNR of the solution of (4.1). From Figure 6, we obtain $m = 5$ for the operator on Figure 1a and $m = 25$ for the operator on Figure 1b. Note that m is larger for the second blur since the impulse response variations are more complex.

In a similar fashion, we select the precision η in such a way that the pSNR of the solution of (4.2) does not differ from more that 0.04dB from the pSNR of the solution of (4.1). We do not report the plots, but we found that $\eta = 5.10^{-4}$ was sufficient to ensure this requirement.

4.3.2 Computing times

In this paragraph, we precisely compare the different methods designed to solve (4.1). The methodology consists in:

- finding the number L for which the solution of (4.2) has a decrease of pSNR of less than 0.2dB w.r.t. the pSNR of the solution of (4.1),

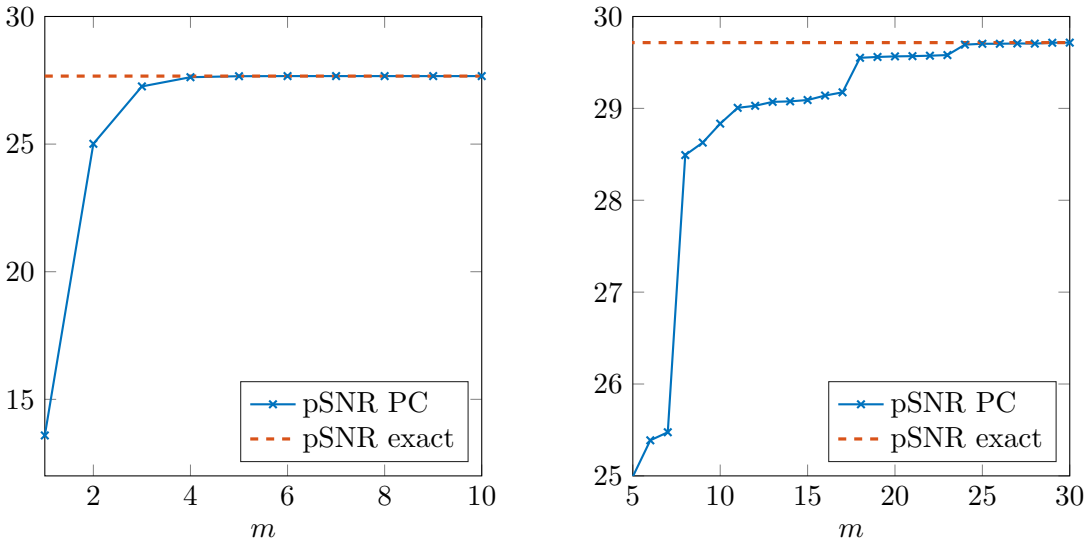


Figure 6: pSNR of the solution of (4.1) obtained with the product-convolution expansion of order m (blue) and the exact operator (red). Left: for the operator on Figure 1a. Right: for the operator on Figure 1b.

- for each algorithm, finding a number of iteration Nit leading to a precision.

$$E(f^{(Nit)}) - E(f^{(0)}) \leq 10^{-3} E(f^{(0)}). \tag{4.3}$$

The cost functions are reported in Figure 7, while the timings are reported in Tables 2 and 3. Deblurring were performed on Matlab with automatic multi-threading disabled (using the -singleCompThreah option), to avoid uncontrolled behaviors while timing. To further demonstrate the efficiency of FISTA-WP, we implement it in CUDA and run it on a GeForce GTX Titan X in double precision.

	FISTA-Spa	FISTA-PC	ADMM-PC	FISTA-W	FISTA-WP	FISTA-WP (GPU)
# iterations	84	80	32	84	38	38
Time (s)	58.3	44.4	37.6	3.6	1.9	0.087
Speed-up	-	1.3	1.5	16.2	30.7	670

Table 2: Timings and iterations number to reach criterion (4.3) for the blur in Figure 1a for the algorithms in Table 1 and in Figure4a ($N = 1024 \times 1024$).

	FISTA-Spa	FISTA-PC	ADMM-PC	FISTA-W	FISTA-WP	FISTA-WP (GPU)
# iterations	58	54	34	54	28	28
Time (s)	616	119.1	172.6	9.9	5.1	0.164
Speed-up	-	5.2	3.6	62	120	4089

Table 3: Timings and iterations number to reach criterion 4.3 for the blur in Figure 1b for the algorithms in Table 1 and in Figure4a ($N = 1024 \times 1024$).

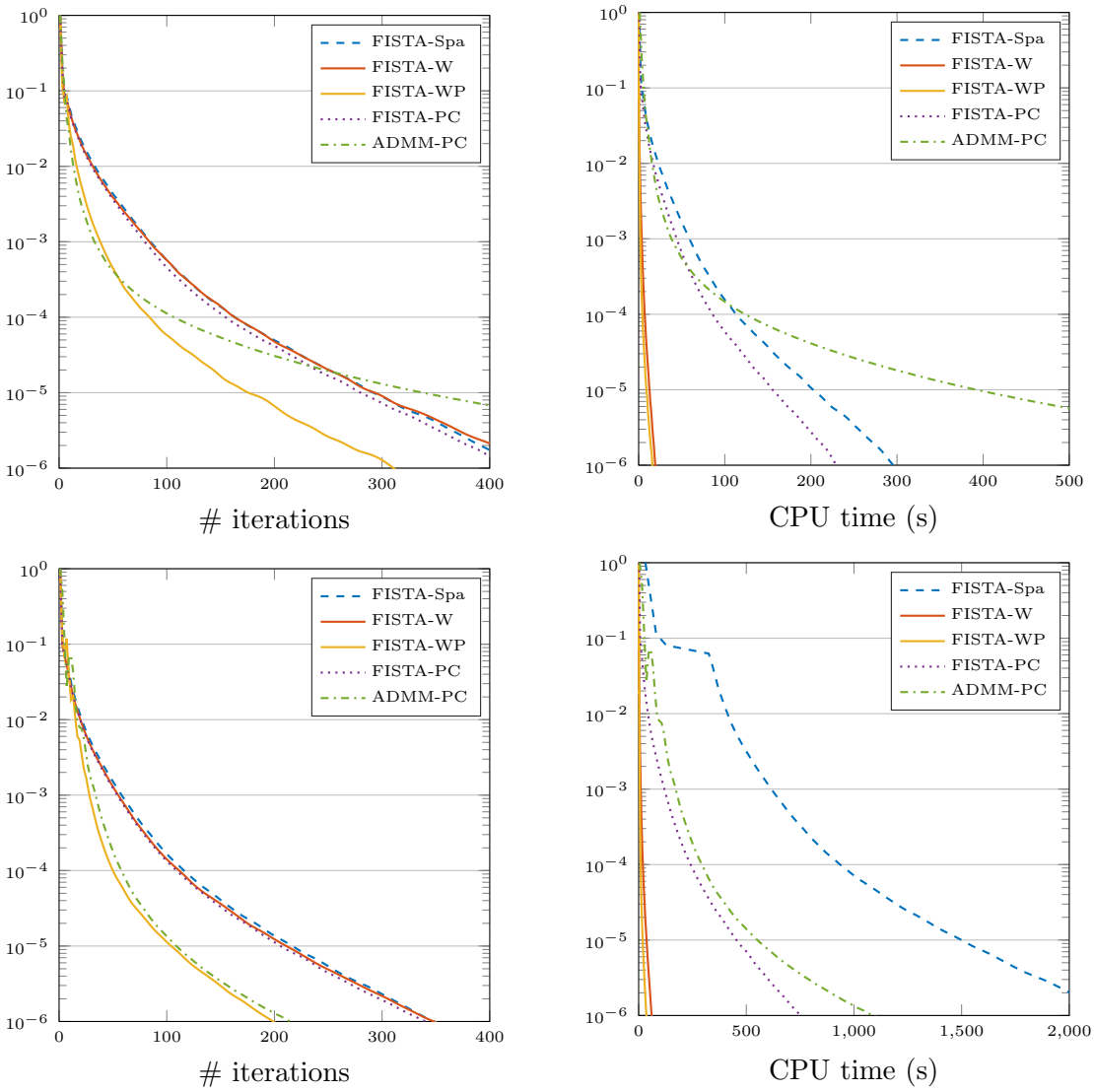


Figure 7: Performance of the deblurring methods in Table 1 for an image of size 1024×1024. The cost functions are displayed with respect to the number of iterations on the left column and the time on the right column. The first row corresponds to the blur on Figure 1a and the second one to the blur on Figure 1b.

This example confirms that solving problem (4.1) by an approximation in the wavelet domain (4.2) leads to dramatic reduction of computation times ($\times 16$ and $\times 62$ speed-up factors) with little loss in the quality (less than 0.2dB). Furthermore, the use of the structure of Θ_L allows to derive efficient diagonal preconditioners which lead to an extra $\times 2$ speed-up.

5 Proof of the main result

The proof exploits the peculiar structure of product-convolution expansions:

- the convolution operator $U_k = u_k \star \cdot$ can be decomposed in the wavelet basis and yields a matrix $A_k = \Psi^* U_k \Psi$ in $O(N \log_2^2 N)$ operations. As long as U_k belongs to the smoothness class described in Definition 3.1, the matrix A_k can be thresholded to obtain a sparse approximation $\widetilde{A}_k$ of A_k.
- the multipliers $V_k = v_k \odot \cdot$ are diagonal. Therefore the sparsity pattern of $B_k = \Psi^* V_k \Psi$ is included in the set of wavelet with supports intersecting the diagonal and contain at most $O(N \log_2 N)$ non-zero coefficients, with known locations. They can be computed in $O(N \log_2 N)$ operations using a cascade algorithm.

The general approach is therefore to compute each couple (A_k, B_k), compute the products $\widetilde{A}_k B_k$ and accumulate the sum to get an approximation $\widetilde{\Theta} = \sum_{k=1}^m \widetilde{A}_k B_k$ of Θ. The last two steps are also critical since in the general case, the number of coefficients can explode in the product $\widetilde{A}_k B_k$ or when summing them all. However, relying again on the peculiar structure of the matrices $\widetilde{A}_k$ and B_k, one can prove that the complexity of the algorithm stays quasi-linear. In what follows we precisely describe each of these steps.

5.1 Structure of A_k

The matrix $A_k = \Psi U_k \Psi^*$ is the wavelet representations of a convolution operator U_k. As such, it inherits a peculiar structure. As observed in [13] it has circulant sub-bands. To make it precise, we recall that a wavelet representation Θ can be decomposed into its wavelet sub-bands:

$$\Theta = \left(\Theta_{j,j'}^{e,e'}\right)_{j,j' \leq J-1}^{e,e' \in \{0,1\}^d}, \quad \text{with } \Theta_{j,j'}^{e,e'}[l,l'] = \Theta[\lambda, \mu].$$

for $l \in \mathcal{T}_j$, $l' \in \mathcal{T}_{j'}$ and $\lambda = (j, e, l)$, $\mu = (j', e', l')$. For instance on 1D signals with $J = 2$, the sub-bands of Θ are shown in Figure 8.

$\Theta_{0,0}$	$\Theta_{0,1}$	$\Theta_{0,2}$
$\Theta_{1,0}$	$\Theta_{1,1}$	$\Theta_{1,2}$
$\Theta_{2,0}$	$\Theta_{2,1}$	$\Theta_{2,2}$

Figure 8: Sub-band structure of convolution matrices in the wavelet domain. Here on 1D signals with $J = 2$.

The sub-bands $\Theta_{j,k}^{e,e'}$ themselves have a specific structure captured by the following definition.

Definition 5.1 (Rectangular circulant matrices)**.** Let $P \in \mathbb{R}^{2^{jd} \times 2^{kd}}$ denote a rectangular matrix and τ_a be the translation operator by vector $a \in \Omega$. The matrix P is called circulant if and only if

- When $k \geq j$: there exists $p \in \mathbb{R}^{2^{kd}}$ such that $P[l,:] = \tau_{2^{k-j}l} p$ for all $l \in \mathcal{T}_j$.
- When $k < j$: there exists $p \in \mathbb{R}^{2^{jd}}$ such that $P[:,l] = \tau_{2^{j-k}l} p$ for all $l \in \mathcal{T}_k$.

Lemma 5.1 ([13, Theorem 3])**.** *The sub-bands of matrix A_k are circulant. Therefore, only $O\left((2^d-1)N\log_2 N\right)$ coefficients are needed to encode A_k. They can be computed with no more than $O\left((2^d-1)N\log_2^2 N\right)$ operations.*

Furthermore, the matrices A_k are assumed to belong to the smoothness class $\mathcal{A}_\alpha$ governing the decay of its coefficients. This properties allows to obtain efficient approximations of A_k as described by Theorem 5.1.

Theorem 5.1 ([7, Theorem 4.6.2])**.** *Let $A \in \mathcal{A}_\alpha$. For all $L \geq 0$, one can construct a matrix A_L with at most L non-zero entries of A per row and column such that*

$$\|A - A_L\|_{2\to 2} \lesssim L^{-\alpha/d}. \tag{5.1}$$

As a summary, Lemma 5.1 together with Theorem 5.1 ensure that the computation of the matrices A_k and their approximations at any precision are tractable in large dimensions.

Remark 5.1. Note that the approximation step is crucial to obtain a quasi-linear complexity of the overall algorithm. If no approximation were made, the complexity of the product $A_k B_k$ could become quasi-quadratic.

Remark 5.2. The proof of Theorem 5.1 gives an explicit way of building A_L from A. We recall it here for sake of completeness. The matrix A_L is obtained by zeroing all entries satisfying either (i) $||\lambda| - |\mu|| > J(L)$ or (ii) $\vartheta(\lambda,\mu) > 2^{J(L)}$ where $J(L)$ is linked to L through the relation $L \lesssim J(L)2^{dJ(L)}$. A straightforward algorithm building A_L consists in looping along all rows λ and rows μ and discard elements satisfying either (i) or (ii). This algorithm requires $O(N^2)$ operations in the general case. Due to the particular structure of matrix A - only $\log_2(N)$ coefficient are non-zero per row (Lemma 5.1) - this algorithm has $O(N\log_2 N)$ complexity. In practice, many authors propose to simply threshold A. This comes from the observation that keeping coefficients that are larger than $2^{-(d/2+\alpha)J}$ do not satisfy (i) and (ii) (from Definition 3.1). Again, since A is encoded with $O(N\log_2 N)$ coefficients, thresholding A costs $O(N\log_2 N)$ instead of $O(N^2)$ in the general case.

5.2 Structure of B_k

We now turn on discussing the structures of the matrices $B_k = \Psi V_k \Psi^*$. Since the operators V_k are diagonal, the coefficients $B_k[\lambda,\mu] = \langle v_k \odot \psi_\lambda, \psi_\mu\rangle$ necessarily vanish as long as $\operatorname{supp}\psi_\lambda$ and $\operatorname{supp}\psi_\mu$ do not intersect. Define

$$\mathcal{I} = \{(\lambda,\mu) \in \Lambda \times \Lambda \mid \operatorname{supp}\psi_\lambda \cap \operatorname{supp}\psi_\mu \neq \emptyset\},$$

we thus have $\operatorname{supp} B_k \subseteq \mathcal{I}$ for all $1 \leq k \leq m$. This property allows to derive upper bounds for the number of coefficients in a column and a row of B_k (Lemma 5.2) as well as for the cardinality of $\operatorname{supp} B_k$ (Lemma 5.3).

Lemma 5.2. *Assume that the mother wavelet is compactly supported on a hypercube of sidelength 2δ. Let $\lambda \in \Lambda$ and $j = |\lambda|$, there are at most $(2\delta)^d \left((2^d - 1)j + 2^{d(J-j)}\right)$ indexes $\mu \in \Lambda$ such that $\operatorname{supp} \psi_\lambda \cap \operatorname{supp} \psi_\mu \neq \emptyset$.*

Proof. Let $\lambda = (j, e, l) \in \Lambda$. From the definition of ψ_λ we get that

$$\operatorname{supp} \psi_\lambda \subseteq \left[-(2^{J-j} - 1)(\delta - 1) + 2^{J-j} l, (2^{J-j} - 1)\delta + 2^{J-j} l\right].$$

Consider now another $\mu = (j', e', l') \in \Lambda$. Necessary conditions for the intersections of the supports of ψ_λ and ψ_μ are

$$\begin{aligned} -(2^{J-j'} - 1)(\delta - 1) + 2^{J-j'} l' &\leq (2^{J-j} - 1)\delta + 2^{J-j} l \\ -(2^{J-j} - 1)(\delta - 1) + 2^{J-j} l &\leq (2^{J-j'} - 1)\delta + 2^{J-j'} l', \end{aligned}$$

where the $\leq$ has to be understood as a component-wise inequality for $d > 1$. Without loss of generality, we assume that $l = 0$.

We let $\mathcal{N}_{j,j'}$ be the set of $l' \in \Lambda$ satisfying the above conditions i.e.

$$\begin{cases} l' \geq -2^{j'-J}\left(2^{J-j} + 2^{J-j'} - 2\right)\delta + 2^{j'-J}(2^{J-j} - 1) \\ l' \leq 2^{j'-J}\left(2^{J-j} + 2^{J-j'} - 2\right)\delta - 2^{j'-J}(2^{J-j'} - 1) \end{cases}$$

The volume of $\mathcal{N}_{j,j'}$ is therefore

$$\begin{aligned} \#\mathcal{N}_{j,j'} &= \left(2^{j'-J}\left(2^{J-j} + 2^{J-j'} - 2\right)(2\delta - 1) + 1\right)^d \\ &\leq \left(2^{j'}\left(2^{-j} + 2^{-j'}\right)(2\delta - 1) + 1\right)^d \\ &\leq \left(2^{j'}\left(2^{-j} + 2^{-j'}\right)2\delta + 1 - 2^{j'-j} - 1\right)^d \\ &\leq \left(2^{j'}\left(2^{-j} + 2^{-j'}\right)2\delta\right)^d \end{aligned}$$

Therefore,

$$\begin{aligned} &\#\left\{\mu \in \Lambda \mid \operatorname{supp} \psi_\mu \cap \operatorname{supp} \psi_\lambda \neq \emptyset\right\} \\ &\leq \sum_{j'=0}^{J-1} \sum_{e \in \{0,1\}^d \setminus \{0\}} \sum_{l' \in \mathcal{T}_{j'}} \mathbb{1}_{\mathcal{N}_{j,j'}}(l') \\ &= (2^d - 1) \sum_{j'=0}^{J-1} \left(2^{j'}\left(2^{-j} + 2^{-j'}\right)2\delta\right)^d \\ &\leq (2^d - 1)(2\delta)^d \left(\sum_{j'=0}^{j-1} 1 + \sum_{j'=j}^{J-1} 2^{d(j'-j)}\right) \\ &= (2^d - 1)(2\delta)^d \left(j + 2^{-jd}\frac{2^{dJ} - 2^{dj}}{2^d - 1}\right) \\ &\leq (2\delta)^d \left((2^d - 1)j + 2^{d(J-j)}\right) \end{aligned}$$

which gives the upper-bound on the number of intersections. □

Lemma 5.3. *We have* $\#\mathcal{I} \leq c(d)\delta^d J 2^{dJ}$*, where the constant* $c(d) = 2(2^d - 1)2^d$*. Therefore since* $J = \frac{1}{d}\log_2(N)$*, the total number of intersections is at most*

$$\frac{c(d)}{d}\delta^d N \log_2(N).$$

Proof.

$$\begin{aligned}
\#\mathcal{I} &= \#\left\{(\lambda,\mu) \in \Lambda \times \Lambda \mid \operatorname{supp}\psi_\mu \cap \operatorname{supp}\psi_\lambda \neq \emptyset\right\} \\
&\leq \sum_{j=0}^{J-1} (2^d - 1)2^{jd}2^d\delta^d \left((2^d-1)j + 2^{d(J-j)}\right) \\
&\leq (2^d-1)2^d\delta^d \left((2^d-1)\sum_{j=0}^{J-1} j2^{jd} + \sum_{j=0}^{J-1} 2^{dJ}\right) \\
&\leq (2^d-1)2^d\delta^d \left((2^d-1)J\frac{2^{dJ}-1}{2^d-1} + J2^{dJ}\right) \\
&\leq 2(2^d-1)2^d\delta^d J 2^{dJ}
\end{aligned}$$

□

The numerical computation of $B_k = \Psi^* V_k \Psi$ is conceptually simple. It consists in two steps:

Step 1 Compute the wavelet transform of each column of V_k i.e. build a matrix $\widehat{V}_k$ such that $\widehat{V}_k[\lambda, i] = \langle V_k[\cdot, i], \psi_\lambda\rangle$ for all $1 \leq i \leq N$ and $\lambda \in \Lambda$. Each column of V_k is a discrete Dirac mass, i.e. contains only one element. Therefore, its wavelet transform can be computed with a minimal number of operations using a cascade algorithm taking advantage of the signals sparsity. A cascade algorithm consists in applying discrete convolutions with filters h and g and subsampling the outputs in a recursive fashion. It is derived from the multiresolution structure of a wavelet transform. We refer to [23, Section 7.3] for more details.

Step 2 Compute the wavelet transform of each row of $\widehat{V}_k$ i.e. $B_k[\lambda,\mu] = \langle \widehat{V}_k[\lambda,\cdot], \psi_\mu\rangle$ for all $\lambda, \mu \in \Lambda$. Similarly, the λ-th row of $\widehat{V}_k$ is sparse: it contains at most $(2^{J-j}2\delta)^d$ contiguous coefficients. The same sparse cascade algorithm can be used to compute its wavelet transform.

We illustrate the sparsity structure of the matrices $\widehat{V}_k$ and B_k in Figure 9 for the case of 1D signals. Observe that each row of $\widehat{V}_k$ contains only a few contiguous nonzero indexes.

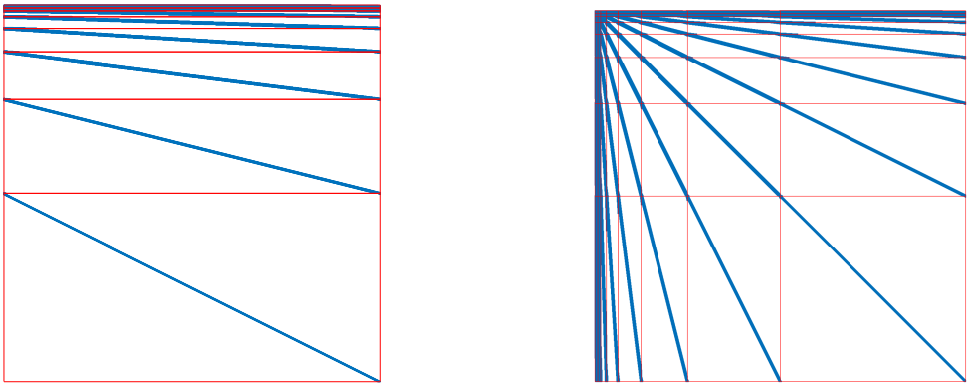
To control the complexity of each step, we will use the following result.

Lemma 5.4. *Let* $f \in \mathcal{E}$ *be supported on the hypercube of* Ω *defined by the lower vertex* $q_0 \in \Omega$ *and the side length* κ*. We call* $q_1 \in \Omega$ *the upper vertex of the hypercube i.e.* $q_1 = q_0 + \kappa$*. Let* $c = \Psi^* f$ *be its wavelet transform and* $c_{j,e}$ *its* (j,e)*-sub-band i.e.* $c_{j,e}[l] = c[\lambda] = \langle f, \psi_\lambda\rangle$ *for all* $l \in \mathcal{T}_j$ *and* $\lambda = (j,e,l)$*. We have*

$$\begin{aligned}
\operatorname{supp} c_{j,e} &= \left[2^{j-J}q_0 - (1-2^{-J}), 2^{j-J}q_1 + (1-2^{-J})2(\delta-1)\right] \\
\#\operatorname{supp} c_{j,e} &\leq 2^d\left(2^{d(j-J)}\kappa^d + (2\delta)^d\right)
\end{aligned}$$

Furthermore, the number of operations to compute c *is bounded above by*

$$2d\delta 2^d\left(\kappa^d + (2^d-1)(2\delta)^d J\right).$$



(a) Sparsity structure of $\widehat{V}_k$ **(b)** Sparsity structure of B_k

Figure 9: Wavelet representations of a diagonal matrix. Here we illustrate the case of a 1D wavelet transform of size 1024×1024 with $J = 3$ decomposition levels.

Proof. Let $\lambda = (j, e, l) \in \Lambda$. From the definition of ψ_λ we get that

$$\operatorname{supp} \psi_\lambda \subseteq \left[-(2^{J-j} - 1)(\delta - 1) + 2^{J-j} l, (2^{J-j} - 1)\delta + 2^{J-j} l\right].$$

Necessary conditions for the intersections of the supports of ψ_λ and f are

$$\begin{cases} q_0 \leq (2^{J-j} - 1)\delta + 2^{J-j} l \\ q_1 \geq -(2^{J-j} - 1)(\delta - 1) + 2^{J-j} l. \end{cases}$$

where the inequalities have to be understood component-wise. These conditions are equivalent to

$$2^{j-J}\left(q_0 - (2^{J-j} - 1)\right) \leq l \leq 2^{j-J}\left(q_1 + (1 - 2^J)2(\delta - 1)\right).$$

The volume of such a set is

$$\begin{aligned} \# \operatorname{supp} c_{j,e} &= \left(2^{j-J} q_1 - 2^{j-J} q_0 + (1 - 2^{-J})2(\delta - 1) + (1 - 2^{-J}) + 1\right)^d \\ &= \left(2^{j-J}(q_1 - q_0 + 1) + (1 - 2^{-J})2\delta\right)^d \\ &\leq 2^d \left(2^{d(j-J)} \kappa^d + (2\delta)^d\right) \end{aligned}$$

Since filters g and h have the same length, the details and average coefficients of the wavelet transforms have the same size in each sub-band - albeit not the same support. Therefore we only analyze the detail coefficients. The computation of one $c_{j,e}$ is obtained from d convolutions of average coefficients at scale $j + 1$ with wavelet filters of size 2δ. Hence the number of operations to compute one $c_{j,e}$ is $d2\delta \# \operatorname{supp} c_{j,e}$ and the total number of operations needed to compute c is

$$\begin{aligned} &\sum_{j=0}^{J-1} (2^d - 1) 2d\delta \# \operatorname{supp} c_{j,e} \\ &\leq 2d\delta(2^d - 1)2^d \sum_{j=0}^{J-1} \left(2^{d(j-J)} \kappa^d + (2\delta)^d\right) \\ &\leq 2d\delta(2^d - 1)2^d \left(\kappa^d \frac{1}{2^d - 1} + (2\delta)^d J\right) \end{aligned} \tag{5.2}$$

which ends the proof. □

We are now ready to bound the complexity of step 1.

Lemma 5.5. *The computation of the matrix $\widehat{V}_k$ requires at most*

$$c(d)(2\delta)^{d+1} N \log_2 N$$

operations. The constant $c(d) = 2^d(2^d - 1)$.

Proof. The i-th column of V is supported on $\{\omega^{-1}(i)\}$. Using Lemma 5.4 together with $J = \frac{1}{d}\log_2 N$ give the result. □

The cost of step 2 is controlled by the following Lemma.

Lemma 5.6. *The computation of B_k from $\widehat{V}_k$ requires at most*

$$c(d)(2\delta)^{d+1} N \log_2 N$$

operations. The constant $c(d) = 2^{d+1}(2^d - 1)$.

Proof. To bound the number of operations needed to compute B from $\widehat{V}$, we need to know the support of each row of $\widehat{V}$. Let $\lambda = (j, e, l)$. From Lemma 5.4, the necessary condition for $\widehat{V}[\lambda, i]$ to be non zero is

$$l \in \left[2^{j-J}\omega^{-1}(i) - (1 - 2^{-J}), 2^{j-J}\omega^{-1}(i) + (1 - 2^{-J})2(\delta - 1)\right]$$

Equivalently the condition becomes

$$\omega^{-1}(i) \in \left[2^{J-j}l - (2^{J-j} - 1)2(\delta - 1), 2^{J-j}l + (2^{J-j} - 1)\right].$$

Therefore $\operatorname{supp} \widehat{V}[\lambda, \cdot] = \left[2^{J-j}l - (2^{J-j} - 1)2(\delta - 1), 2^{J-j}l + (2^{J-j} - 1)\right]$. Using again Lemma 5.4 we conclude that the number of operations required to compute B from $\widehat{V}$ is bounded above by

$$\begin{aligned}
&\sum_{j=0}^{J-1}(2^d - 1)2^{jd}2d\delta 2^d \left(((2^{J-j} - 1)(2\delta - 1) + 1)^d + (2^d - 1)(2\delta)^d J\right) \\
&\leq (2^d - 1)2d\delta 2^d \sum_{j=0}^{J-1} 2^{jd}\left(2^{d(J-j)}(2\delta)^d + (2^d - 1)(2\delta)^d J\right) \\
&\leq (2^d - 1)2d\delta 2^d \left((2\delta)^d J 2^{dJ} + (2\delta)^d J 2^{dJ}\right) \\
&\leq 22^d(2^d - 1)(2\delta)^{d+1} N \log_2 N
\end{aligned}$$

where we used $J = \frac{1}{d}\log_2 N$ to conclude the proof. □

Overall, the computation of one matrix B_k can be achieved in $O\left(\delta^{d+1} N \log_2 N\right)$ operations. As seen in this two-step algorithm, the crucial ingredient of its efficiency is the sparse cascade algorithm. Its implementation is very similar to the standard algorithm but loops on the non-zero elements of the input signal in $\mathcal{E}$. Furthermore, the indexes of each non-zero coefficient at the next scale have to be tracked and reused at the next filtering stage.

5.3 Products $\widetilde{A}_k B_k$

Two questions are important while investigating the product $\widetilde{A}_k B_k$. First, how many operations are needed for its computation and how many non-zero coefficients are in the product matrix. These questions are answered in the following Lemma.

Lemma 5.7. *The number of operations required to compute $\widetilde{A}_k B_k$ is bounded above by $c(d)\delta^d N \log_2 N \epsilon_k^{-d/\alpha}$. Furthermore, the number of non-zero coefficients is bounded above by the same quantity. The constant $c(d) = \frac{2}{d}(2^d - 1)2^d$.*

The product $\widetilde{A}_k B_k$ could be implemented by building the matrix $\widetilde{A}_k$ from the set of circulant vectors and then rely on a standard implementation of sparse matrix-matrix products. However, the building step could be memory-intensive and long time-wise. This step is avoided by taking advantage of the sub-band circulant structure of $\widetilde{A}_k$: locations of non-zero coefficients can be predicted from the elements of the circulant vectors.

Before proving Lemma 5.7, we present a useful Lemma.

Lemma 5.8 ([19, 30])**.** *Let A and B two $N \times N$ matrices. Let α_i be the number of non-zero entries in the i-th column of A and β_i be the number of non-zeros entries in the i-th row of B. Then, the number of operations required to perform AB is only $\sum_{i=1}^{N} \alpha_i \beta_i$. Furthermore, the number of non-zero coefficients in AB is bounded above by the same quantity.*

We are now ready to prove Lemma 5.7.

Proof of Lemma 5.7. From Theorem 5.1 we know that the number of non-zero coefficients in the λ-th column of A is bounded by

$$\alpha_\lambda = L.$$

Similarly, Lemma 5.2 shows that the number of coefficients in the λ-th row of B is bounded by

$$\beta_\lambda = 2^d \delta^d \left((2^d - 1)j + 2^{d(J-j)} \right).$$

Therefore the total number of operations is bounded above by:

$$\begin{aligned}
\sum_{\lambda \in \Lambda} \alpha_\lambda \beta_\lambda &\leq \sum_{j=0}^{J-1} (2^d - 1) 2^{jd} L 2^d \delta^d \left((2^d - 1)j + 2^{d(J-j)} \right) \\
&= (2^d - 1) 2^d \delta^d L \left((2^d - 1) \sum_{j=0}^{J-1} j 2^{jd} + \sum_{j=0}^{J-1} 2^{dJ} \right) \\
&\leq 2(2^d - 1) 2^d \delta^d L J 2^{dJ} \\
&\leq \frac{2}{d} (2^d - 1) 2^d \delta^d L N \log_2 N.
\end{aligned}$$

Furthermore, L is of the order $\epsilon_k^{-d/\alpha}$ to reach an accuracy ϵ_k on the approximation of A_k. Therefore the total number of operations is bounded above by $c(d)\delta^d N \log_2 N \epsilon_k^{-d/\alpha}$. □

5.4 Accumulation of $\widetilde{A}_k B_k$

The last step of the algorithm consists in summing all the products $C_k = \widetilde{A}_k B_k$ into $\widetilde{\Theta}$. The following results show that the number of coefficients in $\widetilde{\Theta}$ does not explode while adding the C_k and that the returned $\widetilde{\Theta}_k$ reaches the prescribed accuracy.

Lemma 5.9. *There exists $\mathcal{S} \subset \Lambda^2$ s.t.* $\operatorname{supp} \widetilde{A}_k B_k \subset \mathcal{S}$ *for all* $1 \leq k \leq m$. *Moreover,* $\#\mathcal{S} = O\left(c(d)\delta^d N \log_2 N (\min_k \epsilon_k)^{-d/\alpha}\right)$.

Proof. Let k_0 be the index such that $\epsilon_{k_0} = \min_k \epsilon_k$. Since A_{k_0} belongs to the smoothness class $\mathcal{A}_\alpha$, there exists a $\mathcal{F} \subset \Lambda^2$ s.t. $\operatorname{supp} \widetilde{A}_{k_0} \subseteq \mathcal{F}$. Furthermore all A_k belong to the same $\mathcal{A}_\alpha$ we have that $\operatorname{supp} \widetilde{A}_k \subseteq \operatorname{supp} \widetilde{A}_{k_0}$.

On the other hand, since all V_k are diagonal matrices $\operatorname{supp} B_k \subseteq \mathcal{I}$ for all k.

Define I (resp. F) a $N \times N$ matrix with ones on $\mathcal{I}$ (resp. $\mathcal{F}$) and zeros outside. Let $\mathcal{S} = \operatorname{supp} FI$. Then obviously $\operatorname{supp} \widetilde{A}_k B_k \subseteq \mathcal{S}$. Moreover, from Lemma 5.7, the number of non-zero coefficients in FI is bounded above by $c(d)\delta^d N \log_2 N \epsilon_{k_0}^{-d/\alpha}$ which concludes the proof. □

5.5 Gathering everything

We will now gather all the ingredients to prove Theorem 3.1. The first part of the proof certifies the accuracy of the approximation.

Lemma 5.10. *The matrix $\widetilde{\Theta}$ returned by Algorithm 1 satisfies* $\|\Theta - \widetilde{\Theta}\|_{2\to 2} \leq \eta$

Proof. Using triangular inequality and since $\|\cdot\|_{2\to 2}$ is multiplicative we get

$$\begin{aligned}
\|\Theta - \widetilde{\Theta}\|_{2\to 2} &\leq \sum_{k=1}^{m} \|A_k B_k - \widetilde{A}_k B_k\|_{2\to 2} \\
&\leq \sum_{k=1}^{m} \|B_k\|_{2\to 2} \|A_k - \widetilde{A}_k\|_{2\to 2} \leq \sum_{k=1}^{m} \|V_k\|_{2\to 2} \epsilon_k \\
&= \sum_{k=1}^{m} \|v_k\|_\infty \epsilon_k = \eta.
\end{aligned}$$

We used the fact that $\|B_k\|_{2\to 2} = \|V_k\|_{2\to 2} = \|v_k\|_\infty$ since Ψ is an orthogonal wavelet basis and V_k is a diagonal matrix. The last equality is derived from the definition of $\epsilon_k = \frac{\eta}{m\|v_k\|_\infty}$. □

The second part deals with the overall complexity.

Proof of Theorem 3.1. The overall complexity of computing $\widetilde{\Theta}$ is bounded above by a quantity proportional to

$$\begin{aligned}
&c(d)\left(\sum_{k=1}^{m} \underbrace{N \log_2^2 N}_{\text{Lemma 5.1}} + N \log_2 N + \underbrace{\delta^{d+1} N \log_2^2 N}_{\text{Lemma 5.6}} + 2\underbrace{\delta^d N \log_2 N \epsilon_k^{-d/\alpha}}_{\text{Lemma 5.7}}\right) \\
&\leq c(d)\left(3\delta^{d+1} m N \log_2^2 N + 2N \log_2 N \sum_{k=1}^{m} \left(\frac{\eta}{m\|v_k\|_\infty}\right)^{-d/\alpha}\right) \\
&\leq c(d)\left(\delta^{d+1} m N \log_2^2 N + N \log_2 N \eta^{-d/\alpha} m^{d/\alpha} \left(\sum_{k=1}^{m} \|v_k\|_\infty^{d/\alpha}\right)\right)
\end{aligned}$$

The constant $c(d)$ depends only on d. The term $\left(\sum_{k=1}^{m} \|v_k\|_\infty^{d/\alpha}\right)$ depends on the considered operator H. Finally, $\widetilde{\Theta}$ is computed in $O\left(\delta^{d+1} m^{\max(1,d/\alpha)} N \log_2^2 N \eta^{-d/\alpha}\right)$.

The bound on the number of coefficients in $\widetilde{\Theta}$ is obtained using Lemma 5.9. Since there exists a support $\mathcal{S}$ such that all $\operatorname{supp} C_k \subseteq \mathcal{S}$ we get that $\operatorname{supp} \widetilde{\Theta} \subseteq \mathcal{S}$. □

Remark 5.3. The complexity analysis does not take into account that v_k may have some smoothness properties. In this case, the associated B_k might belong to a class $\mathcal{A}_{\alpha'}$ and could therefore be thresholded to further speed up the algorithm.

References

[1] N. Alger, V. Rao, A. Myers, T. Bui-Thanh, and O. Ghattas. Scalable matrix-free adaptive product-convolution approximation for locally translation-invariant operators. *SIAM Journal on Scientific Computing*, 41(4):A2296–A2328, 2019.

[2] 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.

[3] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Commun. on Pure Appl. Math.*, 44(2):141–183, 1991.

[4] J. Bigot, P. Escande, and P. Weiss. Estimation of linear operators from scattered impulse responses. *Applied and Computational Harmonic Analysis*, 47(3):730 – 758, 2019.

[5] R. Busby and H. Smith. Product-convolution operators and mixed-norm spaces. *Transactions of the American Mathematical Society*, 263(2):309–341, 1981.

[6] E. Candes and L. Demanet. Curvelets and fourier integral operators. *Comptes Rendus Mathematique*, 336(5):395–398, 2003.

[7] A. Cohen. *Numerical Analysis of Wavelet Methods*, volume 32. Elsevier, 2003.

[8] 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.

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

[10] L. Denis, E. Thiébaut, F. Soulez, J.-M. Becker, and R. Mourya. Fast approximations of shift-variant blur. *International Journal of Computer Vision*, 115(3):253–278, 2015.

[11] P. Escande and P. Weiss. Sparse wavelet representations of spatially varying blurring operators. *SIAM Journal on Imaging Sciences*, 8(4):2976–3014, 2015.

[12] P. Escande and P. Weiss. Approximation of integral operators using product-convolution expansions. *Journal of Mathematical Imaging and Vision*, 58(3):333–348, 2017.

[13] P. Escande and P. Weiss. Accelerating ℓ_1-ℓ_2 deblurring using wavelet expansions of operators. *Journal of Computational and Applied Mathematics*, 343:373–396, 2018.

[14] Y. Fan, C. O. Bohorquez, and L. Ying. Bcr-net: A neural network based on the nonstandard wavelet form. *Journal of Computational Physics*, 384:1–15, 2019.

[15] R. C. Flicker and F. J. Rigaut. Anisoplanatic deconvolution of adaptive optics images. *Journal of the Optical Society of America A*, 22(3):504–513, 2005.

[16] M. Gentile, F. Courbin, and G. Meylan. Interpolating point spread function anisotropy. *Astronomy & Astrophysics*, 549:A1, 2013.

[17] E. Gilad and J. Von Hardenberg. A fast algorithm for convolution integrals with space and time variant kernels. *Journal of Computational Physics*, 216(1):326–336, 2006.

[18] R. Glowinski. *Lectures on Numerical Methods for Non-linear Variational Problems*. Springer Science & Business Media, 2008.

[19] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Transactions on Mathematical Software*, 4(3):250–269, 1978.

[20] W. Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[21] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[22] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

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

[24] Y. Meyer and R. Coifman. *Wavelets, Calderón-Zygmund and Multilinear Operators*. 1997.

[25] J. G. Nagy and D. P. O'Leary. Restoring images degraded by spatially variant blur. *SIAM Journal on Scientific Computing*, 19(4):1063–1082, 1998.

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

[27] D. O'Connor and L. Vandenberghe. Total variation image deblurring with space-varying kernel. *Computational Optimization and Applications*, 67(3):521–541, 2017.

[28] K. Schneider and O. V. Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 42:473–503, 2010.

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

[30] R. Yuster and U. Zwick. Fast sparse matrix multiplication. *ACM Transactions On Algorithms*, 1(1):2–13, 2005.

A Deblurring algorithms

We briefly describe the deblurring algorithms used in the numerical experiments.

A.1 FISTA

Let $\Psi : \mathbb{R}^N \to \mathbb{R}^N$ denote an orthogonal wavelet transform and $H : \mathbb{R}^N \to \mathbb{R}^N$ denote a linear operator. We then have

$$\arg\min_{f\in\mathbb{R}^N} \frac{1}{2}\|Hf - f_0\|_2^2 + \|\Psi^* f\|_{1,w} = \Psi\left(\arg\min_{z\in\mathbb{R}^N} \frac{1}{2}\|H\Psi z - f_0\|_2^2 + \|z\|_{1,w}\right). \tag{A.1}$$

The Forward-Backward algorithm applied to the right hand-side of (A.1) starts with an initial guess $z^{(0)} = y^{(1)} \in \mathbb{R}^N$ and iterates for $i \geq 1$ as

$$\begin{cases} z^{(i)} = \mathrm{Prox}_{\|\cdot\|_{1,w}} \left(y^{(i)} - \tau \Psi^* H^* (H \Psi y^{(i)} - f_0) \right) \\ y^{(i+1)} = z^{(i)} + \dfrac{i-1}{i+2} \left(z^{(i)} - z^{(i-1)} \right) \end{cases}$$

with a step-size $\tau \leq \|H\|_{2\to 2}^{-2}$. This algorithm can be applied verbatim with the exact spatial operator (coined as FISTA-Spa). In the same fashion, we design another algorithm, named FISTA-PC, using the product-convolution expansion.

One can also use FISTA to solve (4.2), the approximation of (4.1) in the wavelet domain. It starts by setting $z^{(0)} = y^{(1)} \in \mathbb{R}^N$ and iterates for $i \geq 1$ as

$$\begin{cases} z^{(i)} = \mathrm{Prox}^{\Sigma}_{\|\cdot\|_{1,w}} \left(y^{(i)} - \tau \Sigma \Theta_L^* (\Theta_L y^{(i)} - z_0) \right) \\ y^{(i+1)} = z^{(i)} + \dfrac{i-1}{i+2} \left(z^{(i)} - z^{(i-1)} \right) \end{cases}$$

where Σ is a diagonal preconditioner equal to the identity or to the Jacobi preconditioner $\mathrm{diag}(\Theta_L^* \Theta_L)$. Note that in this alternative, one iteration costs only 2 matrix-vector products with the sparse matrix Θ_L. We will refer to this algorithm as FISTA-W when $\Sigma = \mathrm{I}_N$ and FISTA-WP when $\Sigma = \mathrm{diag}(\Theta_L^* \Theta_L)$ (see [13] for more details).

A.2 ADMM for product-convolution

The resolution of (4.1) using an ADMM or a Douglas-Rachford is expensive due to the resolution of non trivial linear systems at each iteration. We can adapt ideas proposed in [27] and exploit the product-convolution structure to accelerate the resolution of the linear systems and improve its efficiency. We describe this idea below.

A product-convolution operator H can be decomposed as:

$$Hf = \sum_{k=1}^{m} u_k \star v_k \odot f = \mathcal{U}\mathcal{V} f,$$

with $\mathcal{V} = \left(V_1 \dots V_m \right)^T \in \mathbb{R}^{mN \times N}$ and $\mathcal{U} = \left(U_1 \dots U_m \right) \in \mathbb{R}^{N \times mN}$. This decomposition allows to split the problem (A.1) as

$$\min_{\substack{f \in \mathbb{R}^N, g \in \mathbb{R}^{mN}, z \in \mathbb{R}^N \\ z = \Psi^* f, g = \mathcal{V} f}} \frac{1}{2} \|\mathcal{U} g - f_0\|_2^2 + \|z\|_{1,w}$$

The augmented Lagrangian associated to this problem reads

$$\mathcal{L}(f, g, z, \xi, \zeta) = \frac{1}{2} \|\mathcal{U} g - f_0\|_2^2 + \|z\|_{1,w} + \langle \xi, g - \mathcal{V} f \rangle + \langle \zeta, z - \Psi^* f \rangle + \frac{\beta_1}{2} \|g - \mathcal{V} f\|_{2,\gamma}^2 + \frac{\beta_2}{2} \|z - \Psi^* f\|_2^2.$$

The norm $\|\cdot\|_{2,\gamma}$ is just a weighted l^2-norm for the augmented term of $g = \mathcal{U} f$. Weighting the influence of each U_k allows accelerating the convergence of the ADMM, i.e. it acts as a preconditioner. This norm is defined by $\|\cdot\|_{2,\gamma}^2 = \langle D_\gamma \cdot, \cdot \rangle_{\mathbb{R}^{mN}}$, with

$$D_\gamma = \begin{pmatrix} \gamma_1 \mathrm{Id}_N & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \gamma_m \mathrm{Id}_N \end{pmatrix} \in \mathbb{R}^{mN \times mN}$$

a block diagonal matrix where $\gamma_k > 0$ and $\sum_{k=1}^m \gamma_k = 1$. The choice $\gamma_k = \|U_k\|_{2\to 2}$ led to good practical behaviors.

The ADMM amounts to iterating:

- $g^{(i+1)} = \underset{g\in\mathbb{R}^{mN}}{\arg\min}\, \mathcal{L}\left(f^{(i)}, g, z^{(i)}, \xi^{(i)}, \zeta^{(i)}\right)$.

  The solution of this sub-problem is given by

  $$g^{(i+1)} = (\mathcal{U}^*\mathcal{U} + \beta_1 D_\gamma)^{-1}\left(\mathcal{U}^* f_0 - \xi^{(i)} + \beta_1 D_\gamma \mathcal{V} f^{(i)}\right).$$

  The matrix $\mathcal{U}^*\mathcal{U}$ contains $m \times m$ blocks of size $N \times N$ with the (k,l)-th block populated with $U_k^* U_l$. The direct inversion of this matrix will be inefficient even in the Fourier domain. However, using the Woodbury matrix identity [21], the solution can be expressed as

  $$g^{(i+1)} = \left(D_\gamma^{-1} - D_\gamma^{-1}\mathcal{U}^*(\mathrm{Id}_N + \mathcal{U}D_\gamma^{-1}\mathcal{U}^*)^{-1}\mathcal{U}D_\gamma^{-1}\right)\left(\mathcal{U}^* f_0 - \xi^{(i)} + \beta_1 D_\gamma \mathcal{V} x^{(i)}\right).$$

  Matrix $(\mathrm{Id}_N + \mathcal{U}D_\gamma^{-1}\mathcal{U}^*) = \mathrm{Id}_N + \sum_{k=1}^m \gamma_k U_k U_k^* \in \mathbb{R}^{N\times N}$ is diagonal in the Fourier domain and can be efficiently inverted. In this step, the computation of $3m$ FFTs and $4m+1$ products with diagonal matrices are involved.

- $z^{(i+1)} = \underset{z\in\mathbb{R}^N}{\arg\min}\, \mathcal{L}\left(f^{(i)}, g^{(i+1)}, z, \xi^{(i)}, \zeta^{(i)}\right) = \mathrm{Prox}_{\frac{1}{\beta_2}\|\cdot\|_{1,w}}\left(\Psi^* f^{(i)} - \frac{1}{\beta_2}\zeta^{(i)}\right)$.

  This step costs one wavelet tranform plus a thresholding operation.

- $f^{(i+1)} = \underset{f\in\mathbb{R}^N}{\arg\min}\, \mathcal{L}\left(f, g^{(i+1)}, z^{(i+1)}, \xi^{(i)}, \zeta^{(i)}\right)$.

  The solution of this sub-problem reads

  $$f^{(i+1)} = (\beta_1 \mathcal{V}^* D_\gamma \mathcal{V} + \beta_2 \mathrm{Id}_N)^{-1}\left(\mathcal{V}^*(\xi^{(i)} + \beta_1 D_\gamma g^{(i+1)}) + \Psi(\beta_2 z^{(i+1)} + \zeta^{(i)})\right).$$

  Matrix $\beta_1 \mathcal{V}^* D_\gamma \mathcal{V} + \beta_2 \mathrm{Id}_N = \beta_1 \sum_{i=1}^m \gamma_i V_i V_i + \beta_2 \mathrm{Id}_N$ is $N \times N$ diagonal matrix and can be efficiently inverted.

  In this step, one wavelet transform and $m+1$ products with diagonal matrices are involved.

- Update Lagrange multipliers

  $$\xi^{(i+1)} = \xi^{(i)} + \beta_1 D_\gamma(g^{(i+1)} - \mathcal{V} f^{(i+1)})$$
  $$\zeta^{(i+1)} = \zeta^{(i)} + \beta_2(z^{(i+1)} - \Psi^* f^{(i+1)}).$$

  This step requires m products with diagonal matrices and one wavelet transform.

To summarize, one iteration of the ADMM costs $3m$ FFTs, 3 wavelet transforms and $6m+2$ products with diagonal matrices. This is substantially more than one interation of FISTA with the product-convolution expansion i.e. $2m$ FFTs, 2 wavelet transforms and m products with diagonal matrices.

Acknowledgments

P. Weiss is supported by the ANR JCJC Optimization on Measures Spaces ANR-17-CE23-0013-01 and the ANR-3IA Artificial and Natural Intelligence Toulouse Institute. Both authors thank the GDR ISIS its support.