Distribution strategies for very large 3D image deconvolution algorithms

Céline Meillier[a,*], Rita Ammanouil[b], André Ferrari[b], Pascal Bianchi[a]

[a] *Laboratoire Traitement et Communication de l'Information, 46, rue Barrault - 75013 Paris*
[b] *Laboratoire Lagrange, Observatoire de la Côte dAzur - Boulevard de l'Observatoire CS 34229 - F 06304 NICE Cedex 4*

Abstract

This paper proposes three different distribution strategies for very large 3D image deconvolution algorithms. The deconvolution problem is generic and tailored for spatio-spectral 3D image reconstruction. The three proposed algorithms for large-scale data are distributed in the sense that both the storage and the computations are distributed over several compute nodes. As a result, the workload is drastically reduced compared to a centralized approach where the storage and the computations are handled by a single compute node. The proposed algorithms are validated through experiments on simulated astronomical images.

Keywords: Image deconvolution, distributed optimization algorithms, large 3D data.

1. Introduction

The development of imaging devices that produce ever more data, with ever finer resolutions and ever wider fields of view, leads to an increase in the size of the data to be processed. Big data instruments in astronomy such as MUSE [1, 2, 3], LOFAR [4], and SKA [5] gather massive amounts of images of the sky. Imaging devices in other areas such as hyperspectral remote sensing [6], confocal microscopy [7], and wide-field fluorescence microscopy [8] also fall into

*Corresponding author : meillier@unistra.fr, present address: ICube, Universit de Strasbourg, CNRS (UMR 7357),300 boulevard Sbastien Brant, CS 10413, 67412 Illkirch, France

the category of big imaging data instruments. Big imaging data sets are typically tremendous two dimensional (2D) images or three dimensional (3D) cubes consisting of a stack of 2D images acquired at different wavelength bands. To provide perspective, the radio-interferometric array SKA is designed to produce images of the sky with millions of pixels having arcsec spatial resolution and with over hundreds of frequency samples per pixel estimated at the lower frequency part of the electromagnetic spectrum [9, 10]. Data sets generated by this instrument can size up to several hundreds of Pbytes/yr. Exploiting these big imaging data sets in astronomy, remote sensing, and microscopy will bring significant advancements in their respective application fields. However, this comes at the cost of increasing the computational complexity needed to process the data. Even with a powerful computing machine, storing and analyzing these images are computationally expensive tasks.

Images are unavoidably distorted during the acquisition process. More precisely, they are affected by convolutional effects and corrupted by noise. Image deconvolution is aimed at removing the various distortions and improving the signal to noise ratio, hence improving the extraction of scientific content from the data. The deconvolution step is a fundamental step in any image processing pipeline, and it has been extensively studied in the literature in astronomy and microscopy, see for example [11, 12, 13]. However, the deconvolution step is becoming an increasingly challenging one in the case of big imaging data. The issue of big data has triggered further research and motivated the development of new deconvolution strategies more suitable for large-scale data. In the field of microscopic deconvolution, this has led to the development of commercial and open-source softwares hosting various imaging techniques such as the Huygens Remote Manager [14] and DeconvolutionLab [15]. These softwares are web-based processors allowing to remotely run deconvolution algorithms of large scale microscopic images on high computing facilities. Nevertheless, even with powerful computing machines, the deconvolution of large-scale images can still be an expensive task. In astronomy, recent works focused on the design of scalable deconvolution algorithms based on convex optimization. In astronomy,

more precisely in the context of 2D radio-interferometric imaging, the authors of [16] developed a distributed algorithm for image reconstruction where computations are distributed over several computing nodes. The proposed algorithm is based on the simultaneous direction method of multipliers (SDMM) which decomposes the original problem into smaller sub-problems that can be solved in parallel [17]. In [18], the authors exploit block randomisation in order to alleviate the memory and the computational complexity. More precisely, the data is decomposed into several blocks and only a fraction of the data is processed at each iteration. In the context of 3D radio-interferometric imaging, the spectral dimension further increases the size of the deconvolution problem and hence the need for scalable algorithms. The work in [19, 20] propose a method called MUFFIN (MUlti-Frequency image reconstruction For radio INterferometry) which performs spatio-spectral reconstruction. The proposed algorithms are based on the alternating direction method of multipliers (ADMM) and on the primal-dual method proposed in [21] respectively, both known for their ability to be distributed [22]. In both of the aforementioned works, scalability comes from the fact that the most computationally expensive steps are performed in parallel on different computing nodes.

In this work, we mainly focus on the spatio-spectral image deconvolution problem, i.e. we consider the case of 3D images, and propose three distributed algorithmic structures for deconvolution. In order to fix the ideas, each 2D image, at a given frequency, is considered to be convolved by a 2D known kernel referred to as the point spread function (PSF). In particular, we adopt the 3D deconvolution model proposed in [19] to develop our distributed optimization algorithms because it proposes the challenge of combining constraints both on the images and on the spectra in the regularized optimization problem. The spatio-spectral regularization term makes classical optimization algorithms difficult to parallelize: it becomes impossible to simply parallelize only according to the images or only according to the spectra. The optimization problem is generic and targeted towards large-scale 3D images. We have therefore focused on the distribution of the algorithm on a set of machines gathered within a clus-

ter. Each node of the cluster will perform some of the calculations necessary to find the problem's solution and communicate its results to the other nodes so that they can in turn perform calculations and communicate their results to the other nodes. Most importantly, since disk space and RAM can limit the scalability of the computations, the data storage is also distributed among the different nodes, i.e. each node only stores a part of the overall data set. We also provide an in-depth study and comparison of the theoretical computational and storage complexity of each one of the proposed algorithms.

The paper is organized as follows: section II describes the deconvolution context and the adopted optimization problem, section III proposes three algorithms for solving the proposed optimization problem. The strategy to distribute the algorithms' implementation is detailed in section IV, and finally experiments on synthetic data are presented in section V.

2. Optimization problem

2.1. Data description

Hyperspectral data is a data cube $\mathcal{Y}$ with two spatial dimensions (the observed scene) and a spectral dimension which provides physical information about the observed sources. Calligraphic symbol $\mathcal{Y}$ is dedicated to 3D data, and bold symbol $\mathbf{y}$ stands for the vectorized form of the 3D data cube. Assuming that the observations are made on L frequency bands, the 3D data cube can be seen as a collection of L monochromatic images $\mathcal{Y}_l$ with $l \in \{1, \cdots, L\}$. Each image $\mathcal{Y}_l$ is composed of N_c-by-N_l pixels, where N_c is the number of columns and N_l is the number of lines. Let $N = N_c \times N_l$ be the total number of pixels in each monochromatic image, the vectorized form of $\mathcal{Y}_l$ is written $\mathbf{y}_l \in \mathbb{R}^N$. Similarly to [19], the image observed at the frequency $\nu_l \in \{\nu_1, \cdots, \nu_L\}$, also called the "dirty image" at ν_l, is given by the equation

$$\mathbf{y}_l = \mathbf{H}_l \mathbf{x}_l + \mathbf{n}_l \in \mathbb{R}^N \tag{1}$$

where $\mathbf{x}_l \in \mathbb{R}^N$ is the "true" monochromatic image vector at frequency ν_l, $\mathbf{n}_l$ is the noise vector and $\mathbf{H}_l$ is a convolution matrix representing the point spread function (PSF) of the observation device at frequency ν_l. Let $N_{\mathbf{H}}$ be the radius of the PSF, so that the convolution kernel is a $(2N_{\mathbf{H}}+1) \times (2N_{\mathbf{H}}+1)$ matrix. The full hyperspectral vectorized image is obtained by stacking the L vectorized dirty images $\mathbf{y}_l \in \mathbb{R}^N$ in the vector $\mathbf{y} = [\mathbf{y}_1^T, \cdots, \mathbf{y}_L^T]^T \in \mathbb{R}^M$ where $M = N \times L$ is the total number of pixels composing the 3D data cube. Model (1) can be extended to the complete hyperspectral data cube

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \in \mathbb{R}^M \tag{2}$$

by defining the global convolution matrix $\mathbf{H}$ as a block diagonal matrix:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{H}_L \end{pmatrix}$$

where each block $\mathbf{H}_l$ applies at a given frequency ν_l and $\mathbf{x} = [\mathbf{x}_1^T, \cdots, \mathbf{x}_L^T]^T \in \mathbb{R}^M$ is the vectorized true multispectral image.

2.2. Optimization problem formulation

Denoting respectively as $\|\cdot\|_2$ and $\|\cdot\|_1$ the Euclidean and the ℓ_1 norms, we consider the following the optimization problem, initially proposed in [19]:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \frac{\mu_\epsilon}{2}\|\mathbf{x}\|_2^2 + \imath_{\mathbb{R}^+}(\mathbf{x}) + \|\mathbf{W}\mathbf{x}\|_1, \tag{3}$$

where the first term is the data fidelity term, the second is a Tikhonov regularization term controlled by the parameter $\mu_\epsilon > 0$. The third term is a positivity constraint, since we are recovering sky brightnesses, where $\imath_{\mathbb{R}^+}(\mathbf{x}) = \infty$ if at least one of the elements of $\mathbf{x}$ is negative, and 0 otherwise. The last term is a sparsity term in some 3D image transformation domains. In this work, the sparsity is induced in the multiresolution (wavelet) domain for each monochromatic

image, and in the Discrete Cosine Transform (DCT) domain at each multifrequency pixel as in [19, 20]. More precisely, we have

$$\mathbf{W} = \begin{pmatrix} \mu_s \tilde{\mathbf{W}}_s \\ - - - \\ \mu_\nu \tilde{\mathbf{W}}_\nu \end{pmatrix}$$

where $\mu_s > 0$ and $\mu_\nu > 0$ are regularization parameters, and

$$\tilde{\mathbf{W}}_s = \begin{pmatrix} \mathbf{W}_s & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{W}_s \end{pmatrix} \in \mathbb{R}^{m_s M \times M}$$

is a block-diagonal matrix where each block $\mathbf{W}_s \in \mathbb{R}^{m_s N \times N}$, acting on a monochromatic image, is the concatenation of m_s orthogonal wavelet bases. Similarly to [16, 19], we identify $\mathbf{W}_s$ with a dictionary consisting in the concatenation of the first eight Daubechies wavelet bases ($m_s = 8$). Finally,

$$\tilde{\mathbf{W}}_\nu = \begin{pmatrix} \mathbf{W}_\nu & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{W}_\nu \end{pmatrix} \mathbf{P} \in \mathbb{R}^{M \times M}$$

where $\mathbf{P}$ is the orthonormal permutation matrix that rearranges the elements of the vector $\mathbf{x}$ spectrum by spectrum, each of these spectra being represented by a vector of L frequencies, and where each block $\mathbf{W}_\nu$ is an $L \times L$ matrix representing the DCT. Note that $\tilde{\mathbf{W}}_s^T \tilde{\mathbf{W}}_s = m_s \mathbf{I}_M$ and $\tilde{\mathbf{W}}_\nu^T \tilde{\mathbf{W}}_\nu = \mathbf{I}_M$. The last term in equation (3) is a spatio-spectral regularization and can written as $\|\mathbf{W}\mathbf{x}\|_1 = \mu_s \|\tilde{\mathbf{W}}_s \mathbf{x}\|_1 + \mu_\nu \|\tilde{\mathbf{W}}_\nu \mathbf{x}\|_1$. Note that the permutation matrix $\mathbf{P}$ in $\tilde{\mathbf{W}}_\nu$ makes the optimization problem a non-separable problem because it rearranges elements of $\mathbf{x}$ in order to select spectra rather than 2D images as in $\tilde{\mathbf{W}}_s$.

3. Optimization algorithms

We investigate three optimization algorithms in order to solve problem (3). The first one is based on the alternating direction method of multipliers (ADMM) that is well suited for distributed optimization [22]. The second and the third ones are based on the projected gradient (PG) and on its accelerated version the fast iterative shrinkage-thresholding algorithm (FISTA) proposed in [23].

3.1. ADMM

In order to solve the optimization problem (3), we reformulate it under the ADMM formalism:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to: } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{0} \end{aligned}$$

where:

$$\begin{aligned} f(\mathbf{x}) &= (1/2)\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + (\mu_\epsilon/2)\|\mathbf{x}\|_2^2, \\ g(\mathbf{z}) &= \imath_{\mathbb{R}^+}(\mathbf{p}) + \mu_s\|\mathbf{t}\|_1 + \mu_\nu\|\mathbf{v}\|_1 \end{aligned}$$

and:

$$\begin{aligned} &\mathbf{z}^T = (\mathbf{p}^T, \mathbf{t}^T, \mathbf{v}^T) \in \mathbb{R}^M \times \mathbb{R}^{m_s M} \times \mathbb{R}^M, \\ &\mathbf{A} = \begin{pmatrix} \mathbf{I}_M \\ \tilde{\mathbf{W}}_s \\ \tilde{\mathbf{W}}_\nu \end{pmatrix}, \text{ and } \mathbf{B} = \begin{pmatrix} -\mathbf{I}_M & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_{m_s M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I}_M \end{pmatrix}. \end{aligned}$$

The associated augmented Lagrangian for $\rho > 0$ is:

$$\begin{aligned} \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}, \boldsymbol{\gamma}) &= f(\mathbf{x}) + g(\mathbf{z}) \\ &+ \boldsymbol{\gamma}_{\mathbf{p}}^T(\mathbf{x} - \mathbf{p}) + \boldsymbol{\gamma}_{\mathbf{t}}^T(\tilde{\mathbf{W}}_s\mathbf{x} - \mathbf{t}) + \boldsymbol{\gamma}_{\mathbf{v}}^T(\tilde{\mathbf{W}}_\nu\mathbf{x} - \mathbf{v}) \\ &+ \frac{\rho}{2}\|\mathbf{x} - \mathbf{p}\|_2^2 + \frac{\rho}{2}\|\tilde{\mathbf{W}}_s\mathbf{x} - \mathbf{t}\|_2^2 + \frac{\rho}{2}\|\tilde{\mathbf{W}}_\nu\mathbf{x} - \mathbf{v}\|_2^2 \end{aligned} \tag{4}$$

where $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_{\mathbf{p}}^T, \boldsymbol{\gamma}_{\mathbf{t}}^T, \boldsymbol{\gamma}_{\mathbf{v}}^T]^T \in \mathbb{R}^{(2+m_s)M}$ is the vector of Lagrange multipliers, decomposed in accordance with the right hand side of (4) and ρ is the ADMM penalty parameter. Note that the dimension of this vector is smaller than in [19], where four sets of Lagrange mutlipliers were used instead of three here.

The well known ADMM consists in the following iterations:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\mathbf{x}, \mathbf{z}^k, \boldsymbol{\gamma}^k) && (5)\\
\mathbf{z}^{k+1} &= [\mathbf{p}^{k+1^T}, \mathbf{t}^{k+1^T}, \mathbf{v}^{k+1^T}]^T \\
&= \underset{\mathbf{z}}{\operatorname{argmin}}\ \mathcal{L}_\rho(\mathbf{x}^{k+1}, \mathbf{z}, \boldsymbol{\gamma}^k) && (6)\\
\boldsymbol{\gamma}^{k+1} &= [\boldsymbol{\gamma}_{\mathbf{p}}^{k+1^T}, \boldsymbol{\gamma}_{\mathbf{t}}^{k+1^T}, \boldsymbol{\gamma}_{\mathbf{v}}^{k+1^T}]^T \\
&= \boldsymbol{\gamma}^k + \rho(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^{k+1}). && (7)
\end{aligned}$$

We now write $\mathbf{x}^k = [\mathbf{x}_1^{k^T}, \ldots, \mathbf{x}_L^{k^T}]^T$ where each subvector is of size N and thus corresponds to a monochromatic image estimated at iteration k. We do the same decomposition for $\mathbf{p}^k$ and $\boldsymbol{\gamma}_{\mathbf{p}}^k$. Similar decompositions are also done for $\mathbf{t}^k$ and $\boldsymbol{\gamma}_{\mathbf{t}}^k$ where the dimensions of the subvectors are now equal to $m_s N$. We also write $\mathbf{v}^k = [\mathbf{v}_1^{k^T}, \ldots, \mathbf{v}_N^{k^T}]^T$ and $\boldsymbol{\gamma}_{\mathbf{v}}^k = [\boldsymbol{\gamma}_{\mathbf{v},1}^{k^T}, \ldots, \boldsymbol{\gamma}_{\mathbf{v},N}^{k^T}]^T$ where the blocks within these vectors have the size L (they are homogeneous to a spectrum).

Let us consider the minimization equation (5). Solving the equation shows that the minimization w.r.t. $\mathbf{x}$: $\mathbf{x}^{k+1} = \mathbf{Q}^{-1}\mathbf{b}^k$, is separable with respect to the frequencies thanks to the block diagonal structure of $\tilde{\mathbf{W}}_s$ and to the orthogonality of $\tilde{\mathbf{W}}_\nu$. Let $\mathbf{Q}$ be a block diagonal matrix where $diag(\mathbf{Q}) = [\mathbf{Q}_1, \cdots, \mathbf{Q}_l, \cdots, \mathbf{Q}_L]$. After a straightforward computation, we obtain that $\mathbf{x}_l^{k+1} = \mathbf{Q}_l^{-1}\mathbf{b}_l^k$ for each $l \in \{1, \ldots, L\}$ where:

$$\begin{aligned}
\mathbf{Q}_l &= \mathbf{H}_l^T\mathbf{H}_l + (\mu_\epsilon + (2+m_s)\rho)\mathbf{I}_N \quad \text{and} \\
\mathbf{b}_l^k &= \mathbf{H}_l^T\mathbf{y}_l - \boldsymbol{\gamma}_{\mathbf{p},l}^k - \mathbf{W}_s^T\left(\boldsymbol{\gamma}_{\mathbf{t},l}^k - \rho\mathbf{t}_l^k\right) \qquad (8)\\
&\quad + \rho\left(\mathbf{p}_l^k + \left(\mathbf{P}^T(\tilde{\mathbf{W}}_\nu^T\mathbf{v}^k)\right)_l\right) - \left(\mathbf{P}^T(\tilde{\mathbf{W}}_\nu^T\boldsymbol{\gamma}_{\mathbf{v}}^k)\right)_l,
\end{aligned}$$

where the notation $(\mathbf{u})_l$ denotes the l^{th} subvector of size N of a vector $\mathbf{u}$. The

computations leading to the expression of $\mathbf{Q}_l$ make use of the isometric nature of $\tilde{\mathbf{W}}_s$. Note that $\mathbf{Q}_l$ can be computed once at the beginning of the algorithm while the vectors $\mathbf{b}_l^k$ need to be computed at every iteration. Finally, since $\mathbf{H}_l$ is a convolution operator, $\mathbf{H}_l^T$ is a matched filtering operation, so $\mathbf{Q}_l$ can be seen as a filter matrix. The computation of $\mathbf{Q}_l^{-1}\mathbf{b}_l^k$ can be approximated in practice by using the Fast Fourier Transform (FFT) operator.

Minimizations w.r.t. the vectors $\mathbf{p}$ and $\mathbf{t}$ in (6) are structurally separable with respect to the frequencies. Writing $\tilde{\mathbf{p}}_l^{k+1} = \rho^{-1}\boldsymbol{\gamma}_{\mathbf{p},l}^k + \mathbf{x}_l^{k+1}$, we get:

$$\begin{aligned}\mathbf{p}_l^{k+1} &= \arg\min_{\mathbf{u}\in\mathbb{R}^M} \imath_{\mathbb{R}^+}(\mathbf{u}) + {\boldsymbol{\gamma}_{\mathbf{p},l}^k}^T(\mathbf{x}_l^{k+1} - \mathbf{u}) \\ &\quad + \frac{\rho}{2}\|\mathbf{x}_l^{k+1} - \mathbf{u}\|_2^2 \\ &= \max(0, \tilde{\mathbf{p}}_l^{k+1})\end{aligned}$$

where max is taken elementwise. Writing: $\tilde{\mathbf{t}}_l^{k+1} = \mathbf{W}_s\mathbf{x}_l^{k+1} + \rho^{-1}\boldsymbol{\gamma}_{\mathbf{t},l}^k$, we also have:

$$\begin{aligned}\mathbf{t}_l^{k+1} &= \arg\min_{\mathbf{u}\in\mathbb{R}^{m_s M}} \mu_s\|\mathbf{u}\|_1 + \frac{\rho}{2}\|\mathbf{u} - \tilde{\mathbf{t}}_l^{k+1}\|_2^2 \\ &= \tilde{\mathbf{t}}_l^{k+1} \bullet \max\Big(0, 1 - \frac{\rho^{-1}\mu_s}{|\tilde{\mathbf{t}}_l^{k+1}|}\Big)\end{aligned}$$

where $\bullet$ is the elementwise product. We recognize here the usual soft thresholding operator.

The update of the vector $\mathbf{v}^k$ is done at each multifrequency pixel. Let $\tilde{\mathbf{v}}_i^{k+1} = (\tilde{\mathbf{W}}_\nu\mathbf{x}^{k+1})_i + \rho^{-1}\boldsymbol{\gamma}_{\mathbf{v},i}^k$, $i \in \{1, \ldots, N\}$, where $(\mathbf{u})_i$ is the i^{th} subvector of size L of $\mathbf{u}$, we get:

$$\begin{aligned}\mathbf{v}_i^{k+1} &= \arg\min_{\mathbf{u}\in\mathbb{R}^L} \mu_\nu\|\mathbf{u}\|_1 + \frac{\rho}{2}\|\mathbf{u} - \tilde{\mathbf{v}}_i^{k+1}\|_2^2 \\ &= \tilde{\mathbf{v}}_i^{k+1} \bullet \max\Big(0, 1 - \frac{\rho^{-1}\mu_\nu}{|\tilde{\mathbf{v}}_i^{k+1}|}\Big).\end{aligned}$$

Finally, the inspection of Equation (7) shows that $\boldsymbol{\gamma}_{\mathbf{p}}^k$ and $\boldsymbol{\gamma}_{\mathbf{t}}^k$ are updated at the level of the monochormatic images while $\boldsymbol{\gamma}_{\mathbf{v}}^k$ is updated at the pixel (spec-

trum) level. Algorithm 1 summarizes the main steps required for the ADMM algorithm.

Algorithm 1: ADMM

Initialization
Evaluate $\mathbf{Q}$
Initialize $\mathbf{p}^0$, $\mathbf{t}^0$, $\mathbf{v}^0$, $\boldsymbol{\gamma}_{\mathbf{p}}^0$, $\boldsymbol{\gamma}_{\mathbf{t}}^0$, $\boldsymbol{\gamma}_{\mathbf{v}}^0$ to 0

at iteration $k+1$**:**
Evaluate $\mathbf{b}^{k+1}$
Solve $\mathbf{Q}\mathbf{x}^{k+1} = \mathbf{b}^{k+1}$
$\mathbf{p}^{k+1} = \max(0, \tilde{\mathbf{p}}^{k+1})$
$\mathbf{t}^{k+1} = \tilde{\mathbf{t}}^{k+1} \bullet \max\left(0, 1 - \frac{\rho^{-1}\mu_s}{|\tilde{\mathbf{t}}^{k+1}|}\right)$
$\mathbf{v}^{k+1} = \tilde{\mathbf{v}}^{k+1} \bullet \max\left(0, 1 - \frac{\rho^{-1}\mu_\nu}{|\tilde{\mathbf{v}}^{k+1}|}\right)$
$\boldsymbol{\gamma}_{\mathbf{p}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{p}}^{k} + \rho(\mathbf{x}^{k+1} - \mathbf{p}^{k+1})$
$\boldsymbol{\gamma}_{\mathbf{t}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{t}}^{k} + \rho(\mathbf{W}_s\mathbf{x}^{k+1} - \mathbf{t}^{k+1})$
$\boldsymbol{\gamma}_{\mathbf{v}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{v},i}^{k} + \rho(\tilde{\mathbf{W}}_\nu\mathbf{x}^{k+1} - \mathbf{v}^{k+1})$

3.2. Projected gradient algorithm for solving the dual problem

3.2.1. Reformulation of the minimization problem

Problem (3) can be rewritten in the following equivalent manner:

$$\min_{\mathbf{x},\mathbf{p}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \frac{\mu_\epsilon}{2}\|\mathbf{x}\|_2^2 + \imath_{\mathbf{p}=\mathbf{x}}(\mathbf{p}) + \imath_{\mathbb{R}^+}(\mathbf{p}) + \|\mathbf{W}\mathbf{x}\|_1 \tag{9}$$

Compared to the optimization problem in (3), the positivity constraint in (9) is imposed on the intermediate variable $\mathbf{p}$, and an indicator function (the third term in (9)) is introduced in order to enforce the consensus between the intermediate variable $\mathbf{p}$ and the original variable $\mathbf{x}$. The relevance of the intermediate variable $\mathbf{p}$ is that it allows to write problem (3) in an equivalent form which has a solvable dual problem. This is in contrast with our previous work [24]. This new formulation can be summarized by the following expression:

$$\min_{\mathbf{x},\mathbf{p}} f(\mathbf{x}, \mathbf{p}) + g(\mathbf{W}\mathbf{x}, \mathbf{p}) \tag{10}$$

where:

$$f(\mathbf{x},\mathbf{p}) = \frac{1}{2}\|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \frac{\mu_\epsilon}{2}\|\mathbf{x}\|_2^2 + \imath_{\mathbf{p}=\mathbf{x}}(\mathbf{p}) \tag{11}$$

$$\begin{aligned} g(\mathbf{W}\mathbf{x},\mathbf{p}) &= \|\mathbf{W}\mathbf{x}\|_1 + \imath_{\mathbb{R}^+}(\mathbf{p}) \\ &= h_1(\mathbf{W}\mathbf{x}) + h_2(\mathbf{p}) \end{aligned} \tag{12}$$

with $h_1 = \|\cdot\|_1$ and $h_2 = \imath_{\mathbb{R}^+}(\cdot)$

3.2.2. Dual problem

The solution of the primal problem (10) can be deduced from the solution of the corresponding dual problem:

$$\min_{\boldsymbol{\lambda}_\mathbf{x},\boldsymbol{\lambda}_\mathbf{p}} f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) + g^*(\boldsymbol{\lambda}_\mathbf{x},\boldsymbol{\lambda}_\mathbf{p}) \tag{13}$$

where f^* and g^* denote the Fenchel-Legendre convex conjugate functions of f and g defined by equations (11) and (12) [25]. Details of the calculation of this dual problem are given in appendix Appendix .1. From the expressions of f^* and g^* given in appendix Appendix .1, the dual problem is:

$$\begin{aligned} \min_{\boldsymbol{\lambda}_\mathbf{x},\boldsymbol{\lambda}_\mathbf{p}} \; & \frac{1}{2}(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p}) \\ & + (-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y} \\ & + \imath_{\mathcal{B}_\infty}(\boldsymbol{\lambda}_\mathbf{x}) + \imath_{\mathbb{R}^-}(\boldsymbol{\lambda}_\mathbf{p}) \end{aligned} \tag{14}$$

with $\Delta = (\mathbf{H}^T\mathbf{H} + \mu_\epsilon\mathbf{I}_M)$ where $\mathbf{I}_M$ is the identity matrix of size $M \times M$. Equation (14) is equivalent to:

$$\begin{aligned} \min_{\substack{\|\boldsymbol{\lambda}_\mathbf{x}\|_\infty \leqslant 1 \\ \boldsymbol{\lambda}_\mathbf{p} \leqslant 0}} \; & \frac{1}{2}(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p}) \\ & + (-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y} \end{aligned} \tag{15}$$

Looking at the structure of the minimization dual problem (15), it can be noted that projected gradient algorithm is particularly well suited for solving it.

3.2.3. Projected gradient algorithm

Using the projected gradient algorithm (PG) to solve the dual problem (15) leads to the following iterative update equation

$$\begin{pmatrix} \boldsymbol{\lambda}_{\mathbf{x}}^{k+1} \\ --- \\ \boldsymbol{\lambda}_{\mathbf{p}}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathcal{P}_{\infty}\left(\boldsymbol{\lambda}_{\mathbf{x}}^{k} - \kappa \nabla_{\boldsymbol{\lambda}_{\mathbf{x}}} f^{*}(-\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}^{k}, -\boldsymbol{\lambda}_{\mathbf{p}}^{k})\right) \\ ---------- \\ \mathcal{P}_{\mathbb{R}^{-}}\left(\boldsymbol{\lambda}_{\mathbf{p}}^{k} - \kappa \nabla_{\boldsymbol{\lambda}_{\mathbf{p}}} f^{*}(-\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}^{k}, -\boldsymbol{\lambda}_{\mathbf{p}}^{k})\right) \end{pmatrix}$$

where κ is the parameter of PG algorithm, $\mathcal{P}_{\infty}$ is the projection operator on $\mathcal{B}_{\infty} = \{\mathbf{u} \ : \|\mathbf{u}\|_{\infty} \leqslant 1\}$, being the proximity operator of h_1^* and $\mathcal{P}_{\mathbb{R}^-}$ is the projection operator on $\mathbb{R}^{-n}$ (proximity operator of h_2^*).

From the expression of the composed function $(f \circ -\mathbf{W}^T)^*$, the gradient expression according to the $\boldsymbol{\lambda}_{\mathbf{x}}$ variable is:

$$\begin{aligned} &\nabla_{\boldsymbol{\lambda}_{\mathbf{x}}}\left[f^{*}(-\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}, -\boldsymbol{\lambda}_{\mathbf{p}})\right] \\ &\quad = \mathbf{W}\boldsymbol{\Delta}^{-1}\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}} + \mathbf{W}\boldsymbol{\Delta}^{-1}\boldsymbol{\lambda}_{\mathbf{p}} - \mathbf{W}\boldsymbol{\Delta}^{-1}\mathbf{H}^{T}\mathbf{y} \\ &\quad = \mathbf{W}\boldsymbol{\Delta}^{-1}\left(\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}} + \boldsymbol{\lambda}_{\mathbf{p}} - \mathbf{H}^{T}\mathbf{y}\right) \end{aligned} \tag{16}$$

In the same way, the gradient expression according to the $\boldsymbol{\lambda}_{\mathbf{p}}$ variable is:

$$\begin{aligned} &\nabla_{\boldsymbol{\lambda}_{\mathbf{p}}}\left[f^{*}(-\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}, -\boldsymbol{\lambda}_{\mathbf{p}})\right] \\ &\quad = \boldsymbol{\Delta}^{-1}\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}} + \boldsymbol{\Delta}^{-1}\boldsymbol{\lambda}_{\mathbf{p}} - \boldsymbol{\Delta}^{-1}\mathbf{H}^{T}\mathbf{y} \\ &\quad = \boldsymbol{\Delta}^{-1}\left(\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}} + \boldsymbol{\lambda}_{\mathbf{p}} - \mathbf{H}^{T}\mathbf{y}\right) \end{aligned} \tag{17}$$

Finally, the update of the dual vectors $\boldsymbol{\lambda}_{\mathbf{x}}$ and $\boldsymbol{\lambda}_{\mathbf{p}}$ at each iteration $k+1$ is:

$$\begin{pmatrix} \boldsymbol{\lambda}_{\mathbf{x}}^{k+1} \\ -- \\ \boldsymbol{\lambda}_{\mathbf{p}}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathcal{P}_{\infty}\Big(\boldsymbol{\lambda}_{\mathbf{x}}^{k} - \frac{1}{L}\mathbf{W}\boldsymbol{\Delta}^{-1}\Big(\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}^{k} + \boldsymbol{\lambda}_{\mathbf{p}}^{k} - \mathbf{H}^{T}\mathbf{y}\Big)\Big) \\ ---------- \\ \mathcal{P}_{\mathbb{R}^{-}}\Big(\boldsymbol{\lambda}_{\mathbf{p}}^{k} - \frac{1}{L}\boldsymbol{\Delta}^{-1}\Big(\mathbf{W}^{T}\boldsymbol{\lambda}_{\mathbf{x}}^{k} + \boldsymbol{\lambda}_{\mathbf{p}}^{k} - \mathbf{H}^{T}\mathbf{y}\Big)\Big) \end{pmatrix}$$

where $\kappa = 1/\mathcal{L}$ and $\mathcal{L} = \frac{m_s\mu_s^2 + \mu_\nu^2 + 1}{\mu_\epsilon}$ is chosen higher than the Lipschitz constant

of the gradient vector:

$$\begin{pmatrix} \nabla_{\boldsymbol{\lambda}_\mathbf{x}} \left[f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) \right] \\ - - - - - - - - - - - - \\ \nabla_{\boldsymbol{\lambda}_\mathbf{p}} \left[f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) \right] \end{pmatrix}$$

By developing the gradient expression $\nabla_{\boldsymbol{\lambda}_\mathbf{x}} \left[f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) \right]$, it can be seen that $\boldsymbol{\lambda}_\mathbf{x}$ can be decomposed into two subvectors $\boldsymbol{\lambda}_s \in \mathbb{R}^{m_s M}$ and $\boldsymbol{\lambda}_\nu \in \mathbb{R}^M$ respectively related to the sparse constraint on the wavelet decomposition of the images $\mathbf{x}_l$, $l \in \{1, \cdots, L\}$ and to the sparse constraint on the DCT of each spectrum. The developed expression is:

$$\mathbf{W}\boldsymbol{\Delta}^{-1}\mathbf{W}^T \begin{pmatrix} \boldsymbol{\lambda}_s \\ -- \\ \boldsymbol{\lambda}_\nu \end{pmatrix} = \begin{pmatrix} \mu_s \mathbf{W}_s \boldsymbol{\Delta}^{-1}\big(\mu_s \mathbf{W}_s^T \boldsymbol{\lambda}_s + \mu_\nu \mathbf{W}_\nu^T \boldsymbol{\lambda}_\nu\big) \\ - - - - - - - - - - - - - - - - - \\ \mu_\nu \mathbf{W}_\nu \boldsymbol{\Delta}^{-1}\big(\mu_s \mathbf{W}_s^T \boldsymbol{\lambda}_s + \mu_\nu \mathbf{W}_\nu^T \boldsymbol{\lambda}_\nu\big) \end{pmatrix}$$

Note that the term $\boldsymbol{\Delta}^{-1}\big(\mu_s \mathbf{W}_s^T \boldsymbol{\lambda}_s + \mu_\nu \mathbf{W}_\nu^T \boldsymbol{\lambda}_\nu\big)$ appears twice and some operations are easily parallelized according to images: $\mathbf{W}_s^T \boldsymbol{\lambda}_s$ and the product by $\boldsymbol{\Delta}^{-1}$ while the last term $\mathbf{W}_\nu^T \boldsymbol{\lambda}_\nu$ is easily parallelized according to spectra.

3.2.4. Solution of the primal problem

PG algorithm converges to the optimal dual solution $(\boldsymbol{\lambda}_x^*, \boldsymbol{\lambda}_p^*)$. The optimal primal solution $(\mathbf{x}^*, \mathbf{p}^*)$ can be deduced from $(\boldsymbol{\lambda}_x^*, \boldsymbol{\lambda}_p^*)$ using the following equations:

$$\begin{aligned} \mathbf{x}^* &= \nabla_{\boldsymbol{\lambda}_\mathbf{x}} f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}^*, -\boldsymbol{\lambda}_\mathbf{p}^*) \\ &= \boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}^* + \boldsymbol{\lambda}_\mathbf{p}^* - \mathbf{H}^T\mathbf{y}\right) \end{aligned} \tag{18}$$

$$\begin{aligned} \mathbf{p}^* &= \nabla_{\boldsymbol{\lambda}_\mathbf{p}} f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}^*, -\boldsymbol{\lambda}_\mathbf{p}^*) \\ &= \boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}^* + \boldsymbol{\lambda}_\mathbf{p}^* - \mathbf{H}^T\mathbf{y}\right) \end{aligned} \tag{19}$$

Note that the estimated solutions $\mathbf{x}^*$ and $\mathbf{p}^*$ for problem (9) are equal. This

is due to the indicator function $\imath_{\mathbf{p}=\mathbf{x}}(\mathbf{p})$ in (9) enforcing consensus between the two variables. Algorithm 2 summarizes the main steps required for the PG algorithm.

Algorithm 2: PG

Initialization

$\boldsymbol{\lambda}_{\mathbf{x}}^0 = 0$, $\boldsymbol{\lambda}_{\mathbf{p}}^0 = 0$

at iteration $k+1$**:**

$\boldsymbol{\lambda}_{\mathbf{x}}^{k+1} = \mathcal{P}_{\infty}\left(\boldsymbol{\lambda}_{\mathbf{x}}^k - \frac{1}{L}\mathbf{W}\boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\lambda}_{\mathbf{x}}^k + \boldsymbol{\lambda}_{\mathbf{p}}^k - \mathbf{H}^T\mathbf{y}\right)\right)$

$\boldsymbol{\lambda}_{\mathbf{p}}^{k+1} = \mathcal{P}_{\mathbb{R}^-}\left(\boldsymbol{\lambda}_{\mathbf{p}}^k - \frac{1}{L}\boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\lambda}_{\mathbf{x}}^k + \boldsymbol{\lambda}_{\mathbf{p}}^k - \mathbf{H}^T\mathbf{y}\right)\right)$

3.3. FISTA

The dual problem (15) can be solved using FISTA proposed in [23] which can be considered as an acceleration of the previous PG algorithm. The main steps of FISTA are reported in algorithm 3.

Algorithm 3: FISTA

Initialization

$\xi_0 = 0$

$\boldsymbol{\lambda}_{\mathbf{x}}^0 = 0$, $\boldsymbol{\lambda}_{\mathbf{p}}^0 = 0$

$\boldsymbol{\alpha}_{\mathbf{x}}^0 = 0$, $\boldsymbol{\alpha}_{\mathbf{p}}^0 = 0$

at iteration $k+1$**:**

$\xi_{k+1} = \frac{1+\sqrt{1+4\xi_k^2}}{2}$

$\gamma_{k+1} = \frac{1-\xi_k}{\xi_{k+1}}$

$\boldsymbol{\lambda}_{\mathbf{x}}^{k+1} = \mathcal{P}_{\infty}\left(\boldsymbol{\alpha}_{\mathbf{x}}^k - \frac{1}{L}\mathbf{W}\boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\alpha}_{\mathbf{x}}^k + \boldsymbol{\alpha}_{\mathbf{p}}^k - \mathbf{H}^T\mathbf{y}\right)\right)$

$\boldsymbol{\lambda}_{\mathbf{p}}^{k+1} = \mathcal{P}_{\mathbb{R}^-}\left(\boldsymbol{\alpha}_{\mathbf{p}}^k - \frac{1}{L}\boldsymbol{\Delta}^{-1}\left(\mathbf{W}^T\boldsymbol{\alpha}_{\mathbf{x}}^k + \boldsymbol{\alpha}_{\mathbf{p}}^k - \mathbf{H}^T\mathbf{y}\right)\right)$

$\boldsymbol{\alpha}_{\mathbf{x}}^{k+1} = (1-\gamma_{k+1})\boldsymbol{\lambda}_{\mathbf{x}}^{k+1} + \gamma_k\boldsymbol{\lambda}_{\mathbf{x}}^k$

$\boldsymbol{\alpha}_{\mathbf{p}}^{k+1} = (1-\gamma_{k+1})\boldsymbol{\lambda}_{\mathbf{p}}^{k+1} + \gamma_k\boldsymbol{\lambda}_{\mathbf{p}}^k$

The operations required in FISTA are strictly the same as those required in the PG algorithm. The main difference is the introduction of intermediate

variables $\boldsymbol{\alpha}_{\mathbf{x}}$ and $\boldsymbol{\alpha}_{\mathbf{p}}$ defined as linear combination of current and previous values of $\boldsymbol{\lambda}_{\mathbf{x}}$ and $\boldsymbol{\lambda}_{\mathbf{p}}$ and allowing to have faster convergence. Note that similarly to $\boldsymbol{\lambda}_{\mathbf{x}}$, the vector $\boldsymbol{\alpha}_{\mathbf{x}}$ can be separated into two subvectors $\boldsymbol{\alpha}_s \in \mathbb{R}^{m_s M}$ and $\boldsymbol{\alpha}_\nu \in \mathbb{R}^M$. Similarly to PG algorithm, the primal solution $\mathbf{x}^*$ of the minimization problem (3) is given by equation (18).

4. Distributed architecture and computational complexity

A straightforward implementation of the ADMM, PG, and FISTA algorithms described in the previous section leads to a centralized processing. Each one of the three algorithms, namely algorithms 1, 2, and 3, is run on a single computer (or processor) which handles all the required data storage and the data processing. The main drawback of the centralized approach is that a single computer has limited storage capacity and limited performance which can be prohibitive when running an algorithm with a relatively large data set. In order to overcome this drawback, distributed processing divides the original algorithm into many smaller tasks that are handled by different computers. The various computers run at the same time, complete their corresponding task, and communicate their results, for example at the end of each iteration. The main challenge in the distributed framework is to efficiently "distribute" the tasks required by the original algorithm among the different computers (compute nodes). The final estimation result obtained by the distributed algorithm should be the same as the one obtained by running the original centralized algorithm on a single computer. Furthermore, the memory load required by each compute node and the overall processing time should be at least smaller than the loads required by a single computer running the centralized version of the algorithm. In what follows we will propose such a distributed architecture allowing to efficiently distribute the execution of the ADMM, PG, and FISTA algorithms.

4.1. Distributed architecture

Given the inherent spatial-spectral structure of the multispectral image data cube, the calculations in the previously developed three algorithms can be divided into those performed on spectra and those performed on monochromatic images. As a result, we propose to use a cluster of machines divided into two groups: one for the calculations w.r.t. the frequencies (group A), the other for the calculations w.r.t. the spectra (group B). Figure 1 illustrates the two groups of nodes architecture and exchanges between nodes. For the sake of simplicity, in Figure 1, we assume that there are as many nodes in group A (resp. in group B) as frequency bands (resp. spectra) in the multispectral image. Note that in practice, each node is in charge of several frequencies or pixels, depending on the capacity of the cluster and the image size. Let F be the number of nodes in group A and P the number of nodes in group B. The set of frequencies $\{\nu_1, \cdots, \nu_L\}$ is partioned into $\{\sigma_1, \cdots, \sigma_F\}$ such as $\forall i \neq j, \sigma_i \bigcap \sigma_j = \emptyset$ and $\bigcup_{i=1}^{F} \sigma_i = \{\nu_1, \nu_2, \cdots, \nu_L\}$. In the same way, let $\{\pi_1, \cdots, \pi_P\}$ be the partition of the set of the N observed spectra such as $\forall i \neq j, \pi_i \bigcap \pi_j = \emptyset$ and $\bigcup_{i=1}^{P} \pi_i = \{1, 2, \cdots, N\}$. The proportion of nodes in group A (resp. in group B) must be chosen in order to have globally the same number of operations carried out by each node when computation can be done in a parallel way (see table 1 for operations carried out by each group of nodes).

4.2. Distributed implementation

All of the three algorithms are implemented in Python and the distribution of the algorithms on a cluster (instructions and communications between nodes) is managed by *message passing interface* (MPI) which is a communication protocol designed for programming parallel machines. Within this protocol, memory is supposed to be distributed, exchanges of information are done by point-to-point or collective communications. The same rules of parallelization and distribution apply for ADMM, PG and FISTA algorithms: operations carried out by the nodes of group A (resp. nodes of group B) are parallelized over images (resp. over spectra) on each node. Moreover between two MPI synchronization

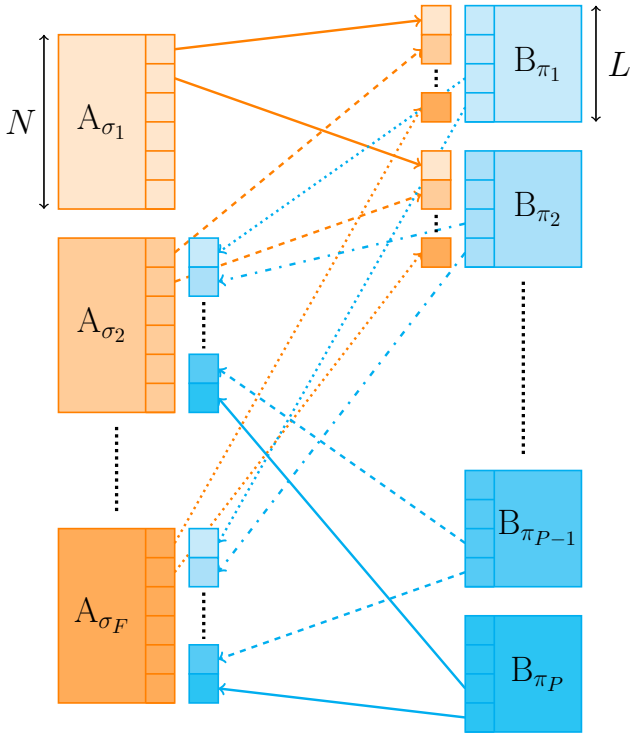

Figure 1: Distributed architecture divided into two groups of nodes and exchanges between nodes of group A and nodes of group B. Note that within a group, the nodes do not exchange data. For the sake of simplicity, F = L and P = N for the graphical representation.

barriers, the operations performed by the group A take place simultaneously to the operations performed by group B. Details of ADMM distributed implementation are given in algorithm 4, while algorithm 5 describes PG implementation. Algorithm 6 details the iterations handled by nodes of group A (resp. group B) of the FISTA implementation on the distributed architecture. Compared to the PG algorithm implementation, it requires additional steps for updating the intermediate variables $\boldsymbol{\alpha}_s$, $\boldsymbol{\alpha}_\nu$ and $\boldsymbol{\alpha}_\mathbf{p}$.

In the case of the ADMM algorithm, the different computations required by the algorithm (steps $6-13$ in algorithm 1) can be divided into two groups: those acting on images and those acting on spectra. Computations related to the least squares term in the optimization problem, the positivity constraint

and the spatial regularization act on images, and those related to the spectral regularization act on spectra. More precisely, steps $6-9$, 11 and 12 act on images, and steps 10 and 13 act on spectra. As mentioned previously, steps $6-9$, 11 and 12 acting on all the spectral bands will be distributed among the nodes in group A where each node is responsible for a subset of the spectral bands. This corresponds to the first and second for loop in algorithm 4. Steps 10 and 13 in algorithm 1 acting on the pixels will be distributed among the nodes in group B where each node in responsible for a subset of the pixels in the images. This corresponds to the third for loop in algorithm 4. Compared to algorithm 1, the distributed algorithm requires two additional steps, namely steps 6 and 18 in algorithm 4 which consist of sending intermediate results necessary for the computations at the receiver node side. For example, in step 6 of algorithm 4, group A nodes send to group B nodes the current estimate of $\mathbf{x}$ which is necessary for the computation of $\tilde{\mathbf{p}}$ in step 10. Note that each node only loads its corresponding subset of the variables, and no node has to load the overall variables as in the centralized case. Nevertheless, the communication between the nodes creates duplicates of the some of variables at different nodes, making the overall memory load of the distributed algorithm slightly higher than the centralized algorithm. Finally, note that the same strategy used for the ADMM was used in order to develop the distributed versions of the PG and FISTA algorithms, described in algorithms 5 and 6.

4.3. Computational complexity and memory load

Table 1 summarizes the main operations and computational complexity handled by nodes of groups A and B for ADMM, PG and FISTA. All of the three algorithms exhibit approximately the same complexity, which is principally related to matrix products. The difference concerns the memory load, since ADMM works with as many primal and dual variables as the number of constraints and FISTA requires to introduce intermediate variables $\boldsymbol{\alpha}_s$, $\boldsymbol{\alpha}_\nu$ and $\boldsymbol{\alpha}_{\mathbf{p}}$ compared to PG algorithm as detailed in table 2.

Algorithm 4: Distributed ADMM algorithm

Initialize $\mathbf{x}$, $\mathbf{p}$, $\mathbf{t}$, $\mathbf{v}$, $\boldsymbol{\gamma}_{\mathbf{p}}$, $\boldsymbol{\gamma}_{\mathbf{t}}$, $\boldsymbol{\gamma}_{\mathbf{v}}$;
while $\delta_{\mathbf{x}}^{k} \geqslant 10^{-5}$ **do**
  for *nodes in group A* **do**
    Evaluate $\mathbf{b}^{k+1}$;
    Solve $\mathbf{Q}\mathbf{x}^{k+1} = \mathbf{b}^{k+1}$;
    Send $\mathbf{x}^{k+1}$ to nodes of group B ;
  end
  MPI synchronization barrier;
  for *nodes in group A* **do**
    $\mathbf{p}^{k+1} = \max(0, \tilde{\mathbf{p}}^{k+1})$;
    $\mathbf{t}^{k+1} = \tilde{\mathbf{t}}^{k+1} \bullet \max\left(0, 1 - \frac{\rho^{-1}\mu_s}{|\tilde{\mathbf{t}}^{k+1}|}\right)$;
    $\boldsymbol{\gamma}_{\mathbf{p}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{p}}^{k} + \rho(\mathbf{x}^{k+1} - \mathbf{p}^{k+1})$;
    $\boldsymbol{\gamma}_{\mathbf{t}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{t}}^{k} + \rho(\mathbf{W}_s\mathbf{x}^{k+1} - \mathbf{t}^{k+1})$
  end
  for *nodes in group B* **do**
    $\mathbf{v}^{k+1} = \tilde{\mathbf{v}}^{k+1} \bullet \max\left(0, 1 - \frac{\rho^{-1}\mu_\nu}{|\tilde{\mathbf{v}}^{k+1}|}\right)$;
    $\boldsymbol{\gamma}_{\mathbf{v}}^{k+1} = \boldsymbol{\gamma}_{\mathbf{v},i}^{k} + \rho(\tilde{\mathbf{W}}_\nu\mathbf{x}^{k+1} - \mathbf{v}^{k+1})$;
    Send $\mathbf{v}^{k+1}$ and $\boldsymbol{\gamma}_{\mathbf{v}}^{k+1}$ to nodes of group A ;
  end
  MPI synchronization barrier ;
  k = k+1 ;
end
return $\mathbf{x}^{k-1}$

5. Simulation results

3D images deconvolution is a typical problem met with hyperspectral data in astronomy [1], [2], in fluorescence microscopy [8], and confocal microscopy [7]. In all of these applications, the blur introduced into the observations can be expressed as a convolution by a point spread function (psf) as expressed in (1). In the experiments, we consider the particular case of astronomical data.

5.1. Synthetic data

We simulated a data cube of size $2048 \times 2048 \times 32$ from one image of the radio emission of an HII region in the M31 galaxy, see Figure 2 (left). The original image is a simple 2D image with normalized intensity between 0 and 1.

We extended the 2D image to a 3D data cube by constructing a spectrum of 32 samples for each pixel in the image as shown in Figure 2 (right). For each pixel the spectrum is a smooth portion of a combination of 3 sine waves. For the sake of simplicity, the combination is the same for all of the pixels. As a result, the spectral form is the same for all pixels and the amplitude of each spectrum is multiplied by the corresponding amplitude in the 2D image. Note that the dependance between spectra does not affect the optimization problem since the sparsity constraint on the spectrum discrete cosine transform is formulated for each spectrum taken independently. The size of the resulting $2048 \times 2048 \times 32$ data cube is 1.1GB, which means that the necessary memory for each algorithm is tens of GB, depending on the number of intermediate variables required. The three distributed algorithms detailed previously have been implemented on a distributed architecture as shown in Figure 1 with 8 nodes in group A and 8 nodes in group B. The simulations were performed on a cluster consisting of 2 machines each with 2 Intel Xeon processors E5430 @ 2.66GHz (8 cores per machine), and 32 GB of memory.

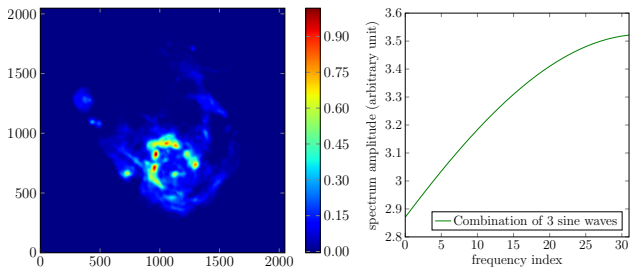

Figure 2: Left: M31 radio emission (2D image). Right: spectrum composed of three sine waves.

5.2. Parameters calibration

The optimization problem (3) depends on different parameters μ_ϵ, μ_s and μ_ν. Setting these parameters to their optimal values remains an open problem, we use the strategy proposed in [20] which consists in decoupling the calibration in two steps: first, μ_ν is set to 0 and different values of μ_s are tested, second

μ_s is set to the value that provides the best signal to noise ratio (SNR) for the reconstruction and different values of μ_ν are tested. The parameter μ_ϵ corresponding to the Tikhonov regularization is set to 10 for all the simulations in order to favor smooth solutions. During the first step, different values between 0.5 and 10 are tested for the parameter μ_s. Figure 3 shows that with larger values of μ_s, the SNR increases faster. Experimentation also shows that values greater than 10 do not increase significantly the SNR. As a result, in the rest of the experiments the parameter μ_s is set to 10.

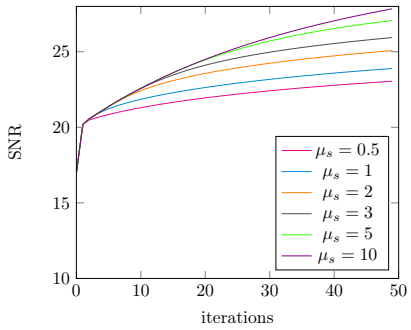

Figure 3: Evolution of the signal to noise ratio for the image reconstruction with different values of μ_s parameter (μ_ν is set to 0 and $\mu_\epsilon = 10$).

Given the optimal parameter value for μ_s, the second step consists of finding the optimal value for the parameter μ_ν. Similarly to the previous step, different values between 0 and 5 have been tested. Figure 4 shows that $\mu_\nu = 2$ and $\mu_\nu = 3$ gives highest SNR after 50 iterations. For value of μ_ν equal to 5 the SNR increases in the first few iterations before it starts falling down and it reaches zero. For $\mu_\nu = 4$ the SNR curve oscillates, this is mainly due to the fact that for very large values of μ_ν the spectral smoothing becomes excessive. In order to prevent too rigorous smoothing, the value μ_ν equal 2 rather than μ_ν equal 3 is chosen. Furthermore, compared to the case where only the spatial regularization was triggered (Figure 3), Figure 4 shows that the SNR is higher when both the spatial and the spectral regularizations are triggered (i.e. μ_s and μ_ν both different than zero). Finally, the calibration of the parameters μ_s and

μ_ν using the ADMM algorithm leads to the following optimal parameter values: $\mu_s = 10$ and $\mu_\nu = 2$. In all of the following experiments, we use these values when solving the deconvolution problem (3).

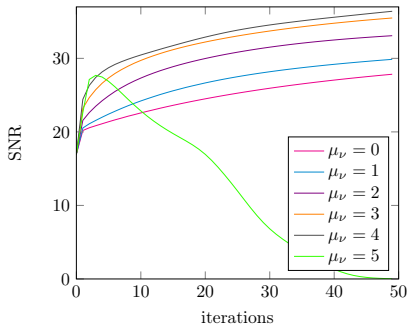

Figure 4: Evolution of the signal to noise ratio for the image reconstruction with different values of μ_ν parameter. Parameter μ_s is set to the value $\mu_s = 10$ which provided the better reconstruction SNR during the first step of calibration and $\mu_\epsilon = 10$.

5.3. Deconvolution results

Figure 5 presents the comparison of reconstruction SNR for the three proposed algorithms: ADMM, PG and FISTA, and for the MUFFIN algorithm, the regularisation parameters being set to the optimal values found previously in all of the algorithms. Figure 5 shows that FISTA and ADMM have quite the

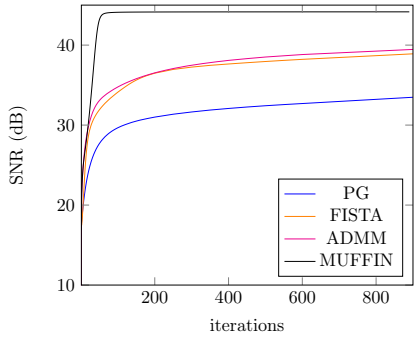

Figure 5: Comparison of signal to noise ratio (SNR) for the reconstruction by ADMM, PG and FISTA. Model parameters are set to $\mu_\epsilon = 10$, $\mu_s = 10$ and $\mu_\nu = 2$.

same performances, they score 39 and 40 dB after 900 iterations respectively. As

expected, the PG algorithm is the slowest algorithim, and it gives the smallest SNR of approximately 34 dB. Finally, MUFFIN gives the best results in terms of the SNR, it scores 44 dB. Nevertheless, recall that the advantage of the proposed algorithms compared to MUFFIN is that in addition to the distribution of the computations, they also distribute the memory storage. Furthermore, it is important to note that, theoretically, problem (3) being a strictly convex optimization problem, all the algorithms should converge to the same optimal solution. However, unlike in MUFFIN, in the ADMM, PG, and FISTA algorithms large matrix inversions are required (see for example step 7 in algorithm 1 and step 10 in algorithm 2). In the simulations, the matrix inversions are not carried out exactly and are avoided using FFT given that the matrices to be inverted correspond to convolution operators. However, the circular convolution assumption is not exact and induces approximation errors along the iterations which justifies the fact that the three algorithms do not converge to the exact same image, and hence do not yield the exact same SNR. Figure 6 presents some results of the deconvolution methods (ADMM and FISTA). Finally, Figure 7 shows the ability of ADMM, FISTA, and MUFFIN to restore small and faint details spread by the PSF after 900 iterations.

5.4. Centralized versus distributed implementation

We test the proposed centralized and distributed ADMM and FISTA algorithms on a second real data set in order to compare the memory usage and the runtime of the different algorithms. The considered real dataset corresponds to an hyperspectral image acquired by the Multi Unit Spectroscopic Explorer (MUSE) instrument known as the Hubble Deep Field South dataset (DATACUBE-HDFS-1.34.fits.gz) available at `http://muse-vlt.eu/science/hdfs-v1-0/`. We consider a subset of the original DATACUBE-HDFS-1.34 consisting of 256×256 pixels and 512 bands in order to make feasible running the centralized versions of the algorithms. The first image in Figure 8 shows the considered MUSE scene at the 379^{th} spectral band. It can be seen that the image is contaminated by noise and requires deconvolution. The corresponding

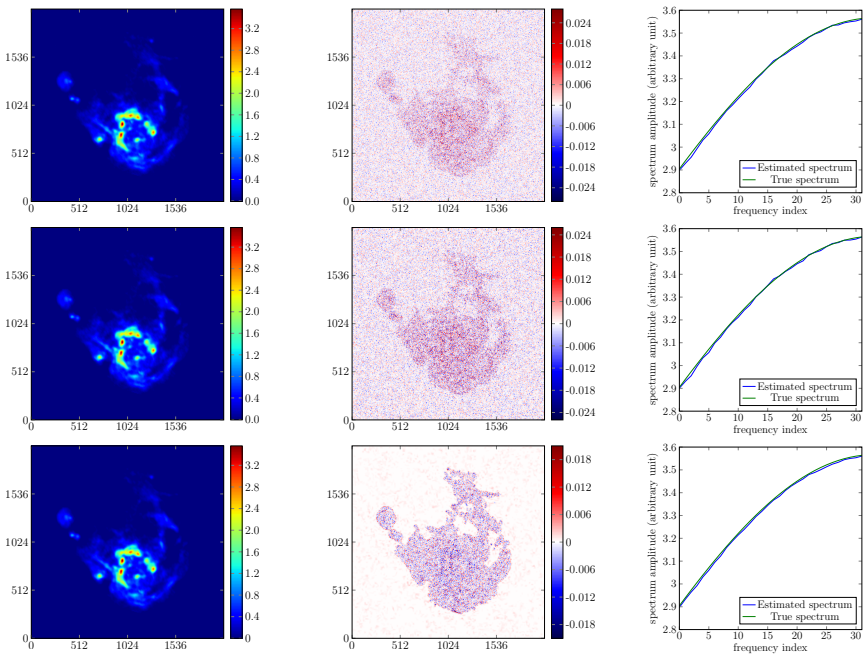

Figure 6: From top to bottom: ADMM, FISTA, and MUFFIN results. From left to right: reconstructed image at the highest frequency, reconstruction error (bias) at the highest frequency, and reconstructed spectrum of the brightest pixel (blue) compared to the truth spectrum (green).

PSF is provided in the reference paper associated to this dataset [3]. For the centralized algorithms, typically one node is used to handle the computations and the memory load. For the distributed algorithms, we consider the following architecture consisting of 7 nodes, 4 belonging to group A and 3 belonging to group B. The nodes have the following specifications: Intel Core i7-6700,16 Gb RAM, 3.40GHz.

The middle and right images in Figure 8 show the deconvolution results of the MUSE dataset at the 379^{th} spectral band obtained with the ADMM and the FISTA algorithms. Given the lack of ground truth, it is not possible to analyse the deconvolution performance in terms of the SNR. Nevertheless, the restored images are rather satisfying, they contain less noise and show spatially close and well defined galaxies. The main purpose of the simulations is to study the memory load and the execution time for the various distributed and centralized

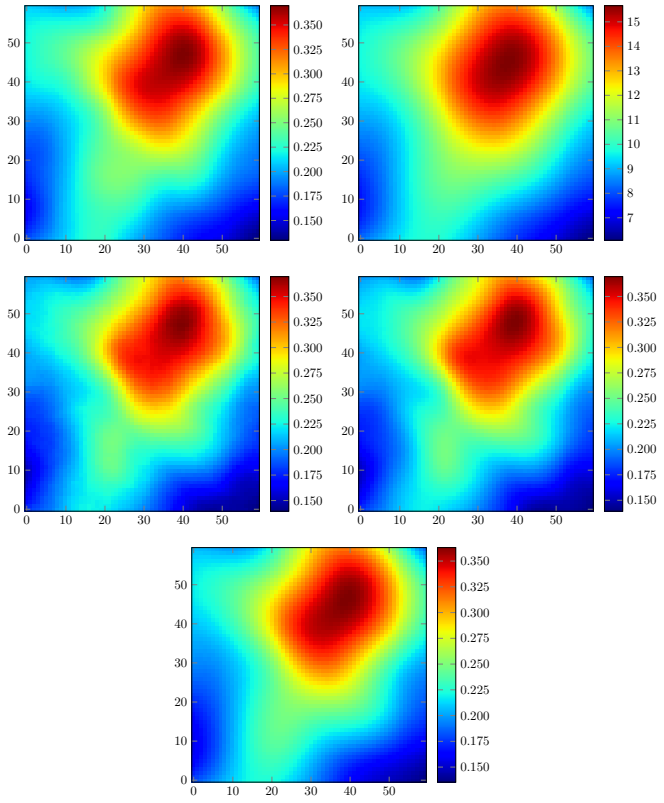

Figure 7: Zoom on a small and faint detail of the m31 galaxy emission at the highest frequency. Top: ground truth image (left) and dirty image (right). Middle: ADMM deconvolution result (left) and FISTA deconvolution result (right) after 900 iterations. Bottom: MUFFIN deconvolution result after 900 iterations.

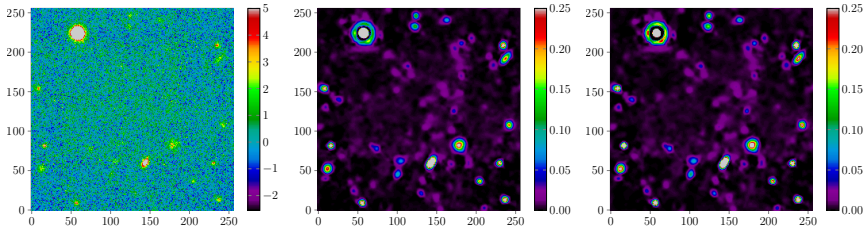

Figure 8: Left: MUSE images corresponding to the 379^{th} spectral band, middle: same image extracted from the ADMM solution, and right: same image extracted from the FISTA solution. Deconvolution results are denoised and show spatially close and well defined galaxies. Approximations due to the circulant convolution lead to artefacts particularly visible around the brightest sources.

algorithms. Table 3 reports the memory usage per node for the centralized and the distributed ADMM and FISTA algorithms. This table shows that the centralized algorithms require that a high memory capacity is handled by a single computing node. Whereas in the distributed algorithms, the memory load per node is smaller than in the centralized case. This is due to the fact that the storage of the data cube and the algorithm variables is distributed among several nodes instead of one. Note that, the total memory load is higher for the distributed case due to duplicated variables among the different nodes. Table 4 reports the runtime per iteration for the centralized and the distributed ADMM and FISTA algorithms. This table shows that the distributed implementation is faster than the centralized implementation. Finally, note that both, the memory load per node and the runtime, could be further reduced by adding additional nodes.

6. Conclusion and discussion

In this paper we presented three optimization algorithms designed to solve the deconvolution problem (3), and implemented these algorithms according to the distributed architecture illustrated in Figure 1. The code distribution on the different nodes of the cluster is handled using the MPI interface which is available in Python, and with other programming languages.

The main strength of all these algorithms is the distributed architecture: the whole data cube is never loaded in memory by the nodes, each node loads only a few number of images (group A) or a few number of spectra (group B). A table is created at the initialization step to indicate the correspondance between images/spectra indexes and nodes identifiers. Consequently, data exchanges are always carried out by only two pre-defined nodes, which limits the communication times between nodes during the iterative algorithms. Comparing the deconvolution results and the memory charge between ADMM and FISTA, the first one seems to be slightly better than the second one: the mean square error is smaller, the reconstruction SNR is higher for the same number of iterations and

the number of variables is lower. However ADMM has an additional parameter ρ to be set at the best value for the convergence of the algorithm. Calibration of the regularization parameters μ_ϵ, μ_s and μ_ν is a difficult task when there is no ground truth image, then the addition of an algorithm parameter increases this difficulty. Finding the regularization parameter values is the purpose of the ongoing work [26]. Both algorithms use the PSF matrix inversion by Fast Fourier Transform. Approximations have a few influence on the deconvolution result if the size of the PSF is limited. If the PSF have the same size than the image to be deconvolved, approximations lead to large errors. The second version of MUFFIN algorithm [20] presents the advantage of avoiding the inversion of the PSF matrix by using Vu-Condat algorithm [27, 28], it will systematically overperform ADMM and FISTA if the size of the PSF is not small compared to the image size. Note that MUFFIN consists in a Vu-Condat optimization algorithm that solves problem (3) with an architecture similar to the one presented in Figure 1 where the number of nodes of group B is limited to 1 (this only node of group B plays the role of the master node in a master/slaves scheme).

Acknowledgements

This work was partly supported by the Agence Nationale pour la Recherche, France: MAGELLAN project (ANR-14-CE23-0004-01). The authors would like to thank W. Hachem who helped to initiate this work within the framework of the MAGELLAN project and for his expertise on optimization methods.

References

[1] S. Bongard, F. Soulez, É. Thiébaut, É. Pecontal, 3d deconvolution of hyperspectral astronomical data, Monthly Notices of the Royal Astronomical Society 418 (1) (2011) 258–270.

[2] S. Bourguignon, D. Mary, É. Slezak, Processing muse hyperspectral data: Denoising, deconvolution and detection of astrophysical sources, Statistical Methodology 9 (1) (2012) 32–43.

[3] R. Bacon, J. Brinchmann, J. Richard, e. al., The MUSE 3D view of the Hubble Deep Field South, Astronomy & Astrophysics 575.

[4] M. P. van Haarlem, M. W. Wise et al., LOFAR: The LOw-Frequency ARray, Astronomy and Astrophysics 556 (2013) A2.

[5] P. E. Dewdney, P. J. Hall, R. T. Schilizzi, T. J. L. W. Lazio, The Square Kilometre Array, IEEE Proceedings 97 (2009) 1482–1496.

[6] C. Chang, Hyperspectral data exploitation: theory and applications, John Wiley and Sons, 2007.

[7] N. Dey, L. Blanc-Feraud, C. Zimmer, P. Roux, Z. Kam, J.-C. Olivo-Marin, J. Zerubia, Richardson–lucy algorithm with total variation regularization for 3d confocal microscope deconvolution, Microscopy research and technique 69 (4) (2006) 260–266.

[8] F. Soulez, L. Denis, Y. Tourneur, E. Thiébaut, Blind deconvolution of 3d data in wide field fluorescence microscopy, in: 9th IEEE International Symposium on Biomedical Imaging (ISBI), IEEE, 2012, pp. 1735–1738.

[9] P. Dewdney, W. Turner, R. Millenaar, R. McCool, J. Lazio, T. Cornwell, Ska1 system baseline design, Document number SKA-TEL-SKO-DD-001 Revision 1 (1).

[10] A. J. van der Veen, S. J. Wijnholds, Signal processing tools for radio astronomy, in: Handbook of Signal Processing Systems, Springer, 2013, pp. 421–463.

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

[12] P. Sarder, A. Nehorai, Deconvolution methods for 3-d fluorescence microscopy images, IEEE Signal Processing Magazine 23 (3) (2006) 32–45.

[13] A. Griffa, N. Garin, D. Sage, Comparison of deconvolution software in 3d microscopy: A user point of view—part 1, GIT Imaging & Microscopy 12 (EPFL-ARTICLE-163617) (2010) 43–45.

[14] A. Ponti, P. Schwarb, A. Gulati, V. Bäcker, Huygens remote manager, Imaging & Microscopy 9 (2) (2007) 57–58.

[15] D. Sage, L. Donati, F. Soulez, D. Fortun, G. Schmit, A. Seitz, R. Guiet, C. Vonesch, M. Unser, Deconvolutionlab2: An open-source software for deconvolution microscopy, Methods.

[16] R. E. Carrillo, J. D. McEwen, Y. Wiaux, Purify: a new approach to radio-interferometric imaging, Monthly Notices of the Royal Astronomical Society 439 (4) (2014) 3591–3604.

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

[18] A. Onose, R. E. Carrillo, A. Repetti, J. D. McEwen, J.-P. Thiran, J.-C. Pesquet, Y. Wiaux, Scalable splitting algorithms for big-data interferometric imaging in the SKA era, arXiv preprint arXiv:1601.04026.

[19] A. Ferrari, J. Deguignet, C. Ferrari, D. Mary, A. Schutz, O. Smirnov, Multi-frequency image reconstruction for radio interferometry. A regularized inverse problem approach, in: SKA Pathfindes Radio Continuum Surveys (SPARCS), 2015.

[20] J. Deguignet, A. Ferrari, D. Mary, C. Ferrari, Distributed multi-frequency image reconstruction for radio-interferometry, in: European Signal Processing Conference (EUSIPCO), 2016.

[21] L. Condat, A generic proximal algorithm for convex optimization—application to total variation minimization, IEEE Signal Processing Letters 21 (8) (2014) 985–989.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning 3 (1) (2011) 1–122.

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

[24] C. Meillier, P. Bianchi, W. Hachem, Two distributed algorithms for the deconvolution of large radio-interferometric multispectral images, in: European Signal Processing Conference (EUSIPCO), 2016.

[25] H. H. Bauschke, P. L. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, Springer Science & Business Media, 2011.

[26] R. Ammanouil, A. Ferrari, R. Flamary, C. Ferrari, D. Mary, Multi-frequency image reconstruction for radio-interferometry with self-tuned regularization parameters, arXiv preprint arXiv:1703.03608.

[27] B. C. Vũ, A splitting algorithm for dual monotone inclusions involving cocoercive operators, Advances in Computational Mathematics 38 (3) (2013) 667–681.

[28] L. Condat, A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms, Journal of Optimization Theory and Applications 158 (2) (2013) 460–479.

Appendix .1. Dual problem formulation

In this appendix, we derive computation of convex conjugate functions for the dual problem (13). Theory can be found in [25]. The Fenchel-Young's inequality can be written for function f and its convex conjugate f^*:

$$f(\mathbf{x}, \mathbf{p}) + f^*(\boldsymbol{\phi}_\mathbf{x}, \boldsymbol{\phi}_\mathbf{p}) \geqslant \langle \mathbf{x}, \boldsymbol{\phi}_\mathbf{x} \rangle + \langle \mathbf{p}, \boldsymbol{\phi}_\mathbf{p} \rangle \tag{.1}$$

where the Fenchel-Legendre convex conjugate function f^* of f is defined by:

$$f^*(\boldsymbol{\lambda}) = \sup_{\mathbf{x}} \langle \mathbf{x}, \boldsymbol{\lambda} \rangle - f(\mathbf{x})$$

Doing the same for function g, we have:

$$\begin{aligned} g(\mathbf{W}\mathbf{x}, \mathbf{p}) + g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) &\geqslant \langle \mathbf{W}\mathbf{x}, \boldsymbol{\lambda}_\mathbf{x} \rangle + \langle \mathbf{p}, \boldsymbol{\lambda}_\mathbf{p} \rangle \\ &\geqslant \langle \mathbf{x}, \mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x} \rangle + \langle \mathbf{p}, \boldsymbol{\lambda}_\mathbf{p} \rangle \end{aligned} \tag{.2}$$

Adding the two inequalities (.1) and (.2), we obtain:

$$\begin{aligned} f(\mathbf{x}, \mathbf{p}) + f^*(\boldsymbol{\phi}_\mathbf{x}, \boldsymbol{\phi}_\mathbf{p}) + g(\mathbf{W}\mathbf{x}, \mathbf{p}) + g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) \\ \geqslant \langle \mathbf{x}, \boldsymbol{\phi}_\mathbf{x} \rangle + \langle \mathbf{p}, \boldsymbol{\phi}_\mathbf{p} \rangle + \langle \mathbf{x}, \mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x} \rangle + \langle \mathbf{p}, \boldsymbol{\lambda}_\mathbf{p} \rangle \end{aligned} \tag{.3}$$

Let $\boldsymbol{\phi}_\mathbf{x} = -\mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x}$ and $\boldsymbol{\phi}_\mathbf{p} = -\boldsymbol{\lambda}_\mathbf{p}$, it becomes:

$$f(\mathbf{x}, \mathbf{p}) + f^*(-\mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) + g(\mathbf{W}\mathbf{x}, \mathbf{p}) + g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) \geqslant 0 \tag{.4}$$

Under regularity conditions, Fenchel's duality theorem applies:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{p}} f(\mathbf{x}, \mathbf{p}) + g(\mathbf{W}\mathbf{x}, \mathbf{p}) \\ = \max_{\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}} - f^*(-\mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) - g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) \end{aligned} \tag{.5}$$

Finally, solving primal problem (10) is equivalent to solve the dual problem:

$$\min_{\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}} f^*(-\mathbf{W}^T \boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) + g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) \tag{.6}$$

In our case the Fenchel-Legendre convex conjugate function of the function f defined in equation (11) is:

$$
\begin{aligned}
& f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) \\
& \quad = \sup_{\mathbf{x},\mathbf{p}} \big(\langle -\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, \mathbf{x}\rangle + \langle -\boldsymbol{\lambda}_\mathbf{p}, \mathbf{p}\rangle - f(\mathbf{x},\mathbf{p})\big) \qquad (.7) \\
& \quad = \sup_{\mathbf{x}} \left(\langle -\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} - \boldsymbol{\lambda}_\mathbf{p}, \mathbf{x}\rangle - \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 - \frac{\mu_\epsilon}{2}\|\mathbf{x}\|_2^2\right) \qquad (.8)
\end{aligned}
$$

where the equivalence between (.7) and (.8) is a simplification of the equality constraint $\imath_{\mathbf{p}=\mathbf{x}}(\mathbf{p})$. After simplification, we obtain:

$$
\begin{aligned}
f^*(-\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x}, -\boldsymbol{\lambda}_\mathbf{p}) = & \frac{1}{2}(\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} + \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}(\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} + \boldsymbol{\lambda}_\mathbf{p}) \\
& - (\mathbf{W}^T\boldsymbol{\lambda}_\mathbf{x} + \boldsymbol{\lambda}_\mathbf{p})^T\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y} \qquad (.9) \\
& + \frac{1}{2}\mathbf{y}^T(\mathbf{H}\boldsymbol{\Delta}^{-1}\mathbf{H}^T - \mathbf{I})\mathbf{y}
\end{aligned}
$$

where $\boldsymbol{\Delta} = \big(\mathbf{H}^T\mathbf{H} + \mu_\epsilon\mathbf{I}\big)$. Similarly, the convex conjugate function of g is:

$$
g^*(\boldsymbol{\lambda}_\mathbf{x}, \boldsymbol{\lambda}_\mathbf{p}) = \imath_{\mathcal{B}_\infty}(\boldsymbol{\lambda}_\mathbf{x}) + \imath_{\mathbb{R}^-}(\boldsymbol{\lambda}_\mathbf{p}) \qquad (.10)
$$

because the convex conjugate of $\imath_{\mathbb{R}^+}(\cdot)$ is $\imath_{\mathbb{R}^-}(\cdot)$ where $\imath_{\mathbb{R}^-}(\cdot)$ such as $\imath_{\mathbb{R}^-}(\mathbf{x}) = \infty$ if at least one element of $\mathbf{x} \in \mathbb{R}^n$ is positive and 0 else. The unit ball for the ℓ_∞ norm is $\mathcal{B}_\infty = \{\boldsymbol{\lambda}_\mathbf{x} \in \mathbb{R}^{(m_s+1)M} : \|\boldsymbol{\lambda}_\mathbf{x}\|_\infty \leqslant 1\}$. Note that $\boldsymbol{\lambda}_\mathbf{x} \in \mathbb{R}^{(m_s+1)M}$, *i.e.* the vector $\boldsymbol{\lambda}_\mathbf{x}$ is homogeneous to the decomposition of the images on the concatenation of the m_s orthogonal wavelet bases and to the DCT of the spectra. The $\boldsymbol{\lambda}_\mathbf{p} \in \mathbb{R}^M$ vector belongs to the direct image domain.

Algorithm 5: Distributed PG for dual problem

Initialization ;
for *nodes in group A* **do**
  Evaluate $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$;
  Send $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$ to B ;
  Evaluate and save $\mu_s\mathbf{W}_s\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$;
end
for *nodes in group B* **do**
  Receive $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$ from A ;
  Evaluate and save $\mu_\nu\tilde{\mathbf{W}}_\nu\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$;
end

while $\delta_{\boldsymbol{\lambda}}^k \geqslant 10^{-5}$ **do**
  for *nodes in group B* **do**
    Evaluate $\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k$;
    Send $\mu_\nu\tilde{\mathbf{W}}_\nu^T(\boldsymbol{\lambda}_\nu^k)$ to A ;
  end
  for *nodes in group A* **do**
    Evaluate $\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k$;
    Receive $\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k$ from B ;
  end
  MPI synchronization barrier ;
  for *nodes in group A* **do**
    Evaluate $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$;
    Send $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$ to B ;
    $\boldsymbol{\theta}_s=\mu_s\tilde{\mathbf{W}}_s\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_s^{k+1}=\mathcal{P}_\infty\left(\boldsymbol{\lambda}_s^k-\frac{1}{L}\left(\boldsymbol{\theta}_s-\mu_s\mathbf{W}_s\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
    $\boldsymbol{\theta}_p=\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_{\mathbf{p}}^{k+1}=\mathcal{P}_{\mathbb{R}^-}\left(\boldsymbol{\lambda}_p^k-\frac{1}{L}\left(\boldsymbol{\theta}_p-\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
  end
  for *nodes in group B* **do**
    Receive $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$ from A ;
    $\boldsymbol{\theta}_\nu=\mu_\nu\mathbf{W}_\nu\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k+\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k+\boldsymbol{\lambda}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_\nu^{k+1}=\mathcal{P}_\infty\left(\boldsymbol{\lambda}_\nu^k-\frac{1}{L}\left(\boldsymbol{\theta}_\nu-\mu_\nu\mathbf{W}_\nu\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
  end
  k = k+1 ;
end
return $\boldsymbol{\lambda}_s^{k-1},\boldsymbol{\lambda}_\nu^{k-1},\boldsymbol{\lambda}_{\mathbf{p}}^{k-1}$

Algorithm 6: Distributed FISTA for dual problem

Initialization ;
$\xi_0 = 0$;
for *nodes in group A* **do**
  $\boldsymbol{\lambda}_s^0 = 0$, $\boldsymbol{\lambda}_{\mathbf{p}}^0 = 0$, $\boldsymbol{\alpha}_s^0 = 0$;
  Evaluate $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$;
  Send $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$ to B ;
  Evaluate and save $\mu_s\mathbf{W}_s\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}_l$;
end
for *nodes in group B* **do**
  $\boldsymbol{\lambda}_\nu^0 = 0$, $\boldsymbol{\alpha}_\nu^0 = 0$;
  Receive $\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$ from A ;
  Evaluate and save $\mu_\nu\tilde{\mathbf{W}}_\nu\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}$;
end

while $\delta_{\boldsymbol{\lambda}}^k \geqslant 10^{-5}$ **do**
  $\xi_{k+1} = \frac{1+\sqrt{1+4\xi_k^2}}{2}$;
  $\gamma_{k+1} = \frac{1-\xi_k}{\xi_{k+1}}$;
  for *nodes in group B* **do**
    Evaluate $\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k$;
    Send $\mu_\nu\tilde{\mathbf{W}}_\nu^T(\boldsymbol{\lambda}_\nu^k)$ to A ;
  end
  for *nodes in group A* **do**
    Evaluate $\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\lambda}_s^k$;
    Receive $\mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\lambda}_\nu^k$ from B ;
  end
  MPI synchronization barrier ;
  for *nodes in group A* **do**
    Evaluate $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$;
    Send $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$ to B ;
    $\boldsymbol{\theta}_s = \mu_s\tilde{\mathbf{W}}_s\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_s^{k+1} = \mathcal{P}_\infty\left(\boldsymbol{\alpha}_s^k - \frac{1}{L}\left(\boldsymbol{\theta}_s - \mu_s\mathbf{W}_s\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
    $\boldsymbol{\theta}_p = \boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_{\mathbf{p}}^{k+1} = \mathcal{P}_{\mathbb{R}^-}\left(\boldsymbol{\alpha}_p^k - \frac{1}{L}\left(\boldsymbol{\theta}_p - \boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
    $\boldsymbol{\alpha}_s^{k+1} = (1-\gamma_{k+1})\boldsymbol{\lambda}_s^{k+1} + \gamma_k\boldsymbol{\lambda}_s^k$;
    $\boldsymbol{\alpha}_{\mathbf{p}}^{k+1} = (1-\gamma_{k+1})\boldsymbol{\lambda}_{\mathbf{p}}^{k+1} + \gamma_k\boldsymbol{\lambda}_{\mathbf{p}}^k$;
  end
  for *nodes in group B* **do**
    Receive $\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$ from A ;
    $\boldsymbol{\theta}_\nu = \mu_\nu\mathbf{W}_\nu\boldsymbol{\Delta}^{-1}\left(\mu_s\tilde{\mathbf{W}}_s^T\boldsymbol{\alpha}_s^k + \mu_\nu\tilde{\mathbf{W}}_\nu^T\boldsymbol{\alpha}_\nu^k + \boldsymbol{\alpha}_{\mathbf{p}}^k\right)$;
    $\boldsymbol{\lambda}_\nu^{k+1} = \mathcal{P}_\infty\left(\boldsymbol{\alpha}_\nu^k - \frac{1}{L}\left(\boldsymbol{\theta}_\nu - \mu_\nu\mathbf{W}_\nu\boldsymbol{\Delta}^{-1}\mathbf{H}^T\mathbf{y}\right)\right)$;
    $\boldsymbol{\alpha}_\nu^{k+1} = (1-\gamma_{k+1})\boldsymbol{\lambda}_\nu^{k+1} + \gamma_k\boldsymbol{\lambda}_\nu^k$;
  end
  k = k+1 ;
end
return $\boldsymbol{\lambda}_s^{k-1}, \boldsymbol{\lambda}_\nu^{k-1}, \boldsymbol{\lambda}_{\mathbf{p}}^{k-1}$

Table 1: Operations and corresponding computational complexity handled by each node of group A (resp. group B) for ADMM, PG and FISTA implemented on a distributed architecture where each node of group A works on L' monochromatic images and each node of group B works on N' spectra simultaneously.

ADMM	
Operations on one node of group A	Complexity on one node of group A
$L' \times m_s$ DWT	$L' \times m_s \times \mathcal{O}(N)$
$L' \times m_s$ iDWT	$L' \times m_s \times \mathcal{O}(N)$
L' inversions using FFT (2 FFT + 1 iFFT)	$L' \times 3(N + 16N_{\mathbf{H}}^2)\log((N + 16N_{\mathbf{H}}^2))$
$2 \times N \times L'$ max	$2 \times L' \times \mathcal{O}(N)$
+ addition between vectors	$L' \times \mathcal{O}(N)$
Operations on one node of group B	Complexity on one node of group B
N' DCT	$N' \times L\log(L)$
N' iDCT	$N' \times L\log(L)$
$N' \times L$ max	$N' \times \mathcal{O}(L)$
+ addition between vectors	$N' \times \mathcal{O}(L)$

PG	
Operations on one node of group A	Complexity on one node of group A
$L' \times m_s$ DWT	$L' \times m_s \times \mathcal{O}(N)$
$L' \times m_s$ iDWT	$L' \times m_s \times \mathcal{O}(N)$
L' inversions using FFT (2 FFT + 1 iFFT)	$L' \times 3(N + 16N_{\mathbf{H}}^2)\log((N + 16N_{\mathbf{H}}^2))$
$N \times L'$ orthogonal projections	$L' \times \mathcal{O}(N)$
+ additions between vectors	$L' \times \mathcal{O}(N)$
Operations on one node of group B	Complexity on one node of group B
N' DCT	$N' \times L\log(L)$
N' iDCT	$N' \times L\log(L)$
$N' \times L$ orthogonal projections	$N' \times \mathcal{O}(L)$
+ additions between vectors	$N' \times \mathcal{O}(L)$

FISTA	
Operations on one node of group A	Complexity on one node of group A
$L' \times m_s$ DWT	$L' \times m_s \times \mathcal{O}(N)$
$L' \times m_s$ iDWT	$L' \times m_s \times \mathcal{O}(N)$
L' inversions using FFT (2 FFT + 1 iFFT)	$L' \times 3(N + 16N_{\mathbf{H}}^2)\log((N + 16N_{\mathbf{H}}^2))$
$N \times L'$ orthogonal projections	$L' \times \mathcal{O}(N)$
+ additions between vectors	$L' \times \mathcal{O}(N)$
Operations on one node of group B	Complexity on one node of group B
N' DCT	$N' \times L\log(L)$
N' iDCT	$N' \times L\log(L)$
$N' \times L$ orthogonal projections	$N' \times \mathcal{O}(L)$
+ additions between vectors	$N' \times \mathcal{O}(L)$

Table 2: Number of variables and their size created on each node of group A and group B for the three algorithms.

Algorithm	**Domain**	**Memory**	
		Node of group A	Node of group B
ADMM	primal-dual	$N \times L' : 7$ $m_s N \times L' : 5$ $(N + N_{\mathbf{H}}) \times L' : 2$	$N' \times L : 8$
PG	dual	$N \times L' : 6$ $m_s N \times L' : 4$ $(N + N_{\mathbf{H}}) \times L' : 2$	$N' \times L : 7$
FISTA	dual	$N \times L' : 8$ $m_s N \times L' : 6$ $(N + N_{\mathbf{H}}) \times L' : 2$	$N' \times L : 9$

Table 3: Memory usage in centralized versus distributed implementation. For the distributed implementation, the memory usage is given for one node of group A and group B and for the totality of the cluster.

Algorithm	**Centralized**	**Distributed**		
		Node of group A	Node of group B	Total
ADMM	6.33 Gb	1.6 Gb	0.55 Gb	8.05 Gb
FISTA	17.75 Gb	4.17 Gb	0.93 Gb	22,25 Gb

Table 4: Runtime of one iteration in centralized versus distributed implementation of ADMM and FISTA.

Algorithm	**Centralized**	**Distributed**
ADMM	95s	40s
FISTA	234s	176s