IOP Publishing Inverse Problems
Inverse Problems 37 (2021) 014001 (28pp) https://doi.org/10.1088/1361-6420/abb541

An ADMM-LAP method for total variation myopic deconvolution of adaptive optics retinal images

Xiaotong Chen[1], James L Herring[2], James G Nagy[3,*], Yuanzhe Xi[3] and Bo Yu[1]

[1] School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning 116024, People's Republic of China
[2] Department of Mathematics, University of Houston, Houston, TX 77204, United States of America
[3] Department of Mathematics, Emory University, Atlanta, GA 30322, United States of America

E-mail: chenxiaotong@mail.dlut.edu.cn, jnagy@emory.edu, herrinj@gmail.com, yxi26@emory.edu and yubo@dlut.edu.cn

Received 23 December 2019, revised 30 August 2020
Accepted for publication 4 September 2020
Published 3 December 2020

Abstract

Adaptive optics corrected flood imaging of the retina is a popular technique for studying the retinal structure and function in the living eye. However, the raw retinal images are usually of poor contrast and the interpretation of such images requires image deconvolution. Different from standard deconvolution problems where the point spread function (PSF) is completely known, the PSF in these retinal imaging problems is only partially known which leads to the more complicated myopic (mildly blind) deconvolution problem. In this paper, we propose an efficient numerical scheme for solving this myopic deconvolution problem with total variational (TV) regularization. First, we apply the alternating direction method of multipliers (ADMM) to tackle the TV regularizer. Specifically, we reformulate the TV problem as an equivalent equality constrained problem where the objective function is separable, and then minimize the augmented Lagrangian function by alternating between two (separated) blocks of unknowns to obtain the solution. Due to the structure of the retinal images, the subproblems with respect to the fidelity term appearing within each ADMM iteration are tightly coupled and a variation of the linearize and project method is designed to solve these subproblems efficiently. The proposed method is called the ADMM-LAP method. Theoretically, we establish the subsequence convergence of the ADMM-LAP method to a stationary point.

*Author to whom any correspondence should be addressed.

Both the theoretical complexity analysis and numerical results are provided to demonstrate the efficiency of the ADMM-LAP method.

Keywords: total variation, retinal imaging, image restoration, myopic deconvolution

(Some figures may appear in colour only in the online journal)

1. Introduction

Image restoration is an important topic in image processing which is widely used in many areas, such as astronomical imaging, medical imaging and restoring aging and deteriorated films. The goal of image restoration is to reconstruct the best possible approximation of the clean, original image from an observed, blurred and noisy image. The basic image restoration problem can be described as a linear inverse problem

$$\boldsymbol{d} = A\boldsymbol{x} + \boldsymbol{e}, \tag{1}$$

where $A \in \mathbb{R}^{n^2 \times n^2}$ is an ill-conditioned blurring matrix defined by the point spread function (PSF) [9], $\boldsymbol{d} \in \mathbb{R}^{n^2}$ represents the observed, blurred and noisy image, $\boldsymbol{e} \in \mathbb{R}^{n^2}$ denotes the additive noise and $\boldsymbol{x} \in \mathbb{R}^{n^2}$ denotes the unknown true image to be restored.

The PSF is generally assumed to be perfectly known in standard image restoration techniques. However, this is not always the case. In many applications, the true PSF (and therefore the blurring matrix A) is unknown or partially known. This results in blind deconvolution problems where image restoration also requires recovering or approximating the PSF. If the PSF can be parameterized by a small number of unknown parameters, the problem can be considered mildly/partially blind, or *myopic*. Adaptive optics (AO) corrected flood imaging of the retina is one such myopic deconvolution problem that has received much attention. In [2, 4], the authors present an imaging model that can transfer the 3D model to a 2D model with the global PSF being an unknown linear combination of a few PSFs. Thus, problem (1) requires estimating both the combination coefficients of A and the true image $\boldsymbol{x}$ in this model.

In order to guarantee the fidelity of the recovery, it is necessary to add a regularizer. There are two well-known types of regularizers for problem (1): one is Tikhonov and the other is total variation (TV). Tikhonov regularization was first proposed in [20] with a quadratic penalty added to the objective function. Due to its quadratic property, it is inexpensive to minimize the objective function. Thus, Tikhonov regularization is computationally efficient and has been widely used. However, one disadvantage of the Tikhonov approach is that it tends to over smooth the image and fails to preserve important image details such as sharp edges. The TV regularizer, first proposed by Rudin, Osher and Fatemi in [16], has also been widely adopted in image reconstruction problems [11, 17, 19, 21]. Since the TV approach uses the summation of the variation of the image $\boldsymbol{x}$ at all pixels to control the norm (or semi-norm) and the smoothness of the solution, it has been shown both experimentally and theoretically that the TV approach can effectively preserve sharp edges and keep the important features of the restored image.

Following the AO retinal imaging model in [2, 4], in this paper, we consider the myopic deconvolution model with TV regularization as follows

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathcal{C}_{\boldsymbol{x}},\boldsymbol{w}\in\mathcal{C}_{\boldsymbol{w}}} \ & \Phi(\boldsymbol{x},\boldsymbol{w}) = \frac{\mu}{2}\|A(\boldsymbol{w})\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \mathrm{TV}(\boldsymbol{x}) \\ \text{s.t.} \ & \sum_{j=1}^{p} w_j = 1, \end{aligned} \tag{2}$$

where $\mathcal{C}_{\boldsymbol{x}} = \left\{\boldsymbol{x} | x_i \geqslant 0 \text{ for } i = 1, \ldots, n^2\right\}, \mathcal{C}_{\boldsymbol{w}} = \{\boldsymbol{w} | w_j \geqslant 0 \text{ for } j = 1, \ldots, p\}$, μ is a positive parameter that is used to balance the two terms in the objective function, TV denotes the TV regularization term, $\boldsymbol{x} \in \mathbb{R}^{n^2}$ is a vectorized version of the unknown $n \times n$ image to be recovered, $\boldsymbol{w} \in \mathbb{R}^p$ denotes unknown weights parameterizing the blurring matrix $A(\boldsymbol{w}) \in \mathbb{R}^{n^2 \times n^2}$, and $\boldsymbol{d} \in \mathbb{R}^{n^2}$ is the observed, blurred noisy image (data). For the imaging model we use, the blurring matrix $A(\boldsymbol{w})$ is a weighted sum of p known blurring matrices A_j with the form

$$A(\boldsymbol{w}) = \sum_{j=1}^{p} w_j A_j = w_1 A_1 + w_2 A_2 + \cdots + w_p A_p.$$

Note that the fidelity term in the objective function in (2) is nonconvex and the TV regularization term is nondifferentiable and nonlinear. This poses some computational challenges in the optimization problem. In particular, due to the existence of the parameters $\boldsymbol{w}$ and the nonconvexity of the objective function, approaches such as the fast iterative shrinkage-thresholding algorithm [1, 6] cannot be applied directly to solve the problem. However, alternative approaches have been proposed to solve the optimization problem with TV regularization in the literature. One particularly efficient scheme is the alternating direction method of multipliers (ADMM) [19, 21, 28]. ADMM was first developed to solve convex optimization problems by breaking them into subproblems, each of which are then easier to handle. Recent work has shown that ADMM can also perform well for a variety of applications involving nonconvex objective functions or nonconvex sets [5, 13, 22, 26, 27]. Inspired by the success of ADMM on nonconvex problems, we consider using it as the optimization method to tackle problem (2).

The main contribution of this paper can be summarized as follows:

- We implement an efficient algorithm called ADMM-LAP for myopic deconvolution problems with TV regularization arising from the AO retinal image restoration. Specifically, we first apply ADMM as an outer optimization method to tackle the TV regularizer and then apply the linearize and project (LAP) method as an inner optimization method to solve the tightly coupled subproblems arising within each ADMM iteration. We establish the subsequence convergence of ADMM-LAP to a stationary point and conduct a complexity analysis of ADMM-LAP to demonstrate its computational efficiency for myopic deconvolution problems with TV regularization.
- We present extensive numerical experiments to illustrate the effectiveness of the ADMM-LAP method. In addition, we compare the performance of ADMM-LAP with a benchmark method called ADMM-BCD where ADMM is applied to tackle the TV regularization while block coordinate descent (BCD) is applied to solve the coupled subproblems. Compared to ADMM-BCD, ADMM-LAP converges faster and can reach smaller relative errors for both the restored image and the obtained parameters.

The paper is organized as follows. In section 2, we introduce a general formulation of the AO retinal imaging problem. In section 3, we first briefly review the iteration format of ADMM and introduce a variation of LAP for our problem, then propose the ADMM-LAP method and present the convergence results. A benchmark method ADMM-BCD is also discussed. The computational complexity of both methods are analyzed in this section. Numerical results are given in section 4, and concluding remarks are drawn in section 5.

2. Retinal imaging problem

2.1. AO retinal imaging model

AO is a well-known optoelectronic technique that compensates for the time-varying aberrations of the eye [3]. However, AO flood imaging suffers from an intrinsic limitation that leads to a loss in resolution because the object is three-dimensional, the image contains information from both the in-focus plane and the out-of-focus planes of the object. Hence, interpretation of such images requires an appropriate post-processing, including image deconvolution. In this paper, we focus on the myopic image deconvolution problem that arises from the AO retinal image restoration and propose the efficient ADMM-LAP method to solve the problem.

First, we describe the structure of the AO retinal imaging model. In the continuous setting, retinal imaging is typically modeled as a three-dimensional (3D) convolution [2, 4]:

$$\boldsymbol{d}_{3\mathrm{D}} = \boldsymbol{h}_{3\mathrm{D}} *_{3\mathrm{D}} \boldsymbol{x}_{3\mathrm{D}} + \boldsymbol{e}, \tag{3}$$

where $\boldsymbol{d}_{3\mathrm{D}}$ is the observed image, $\boldsymbol{h}_{3\mathrm{D}}$ is the PSF, $*_{3\mathrm{D}}$ is the three-dimensional convolution operator, $\boldsymbol{x}_{3\mathrm{D}}$ is the true object, and $\boldsymbol{e}$ is the additive noise. If the true object is assumed to be shift invariant along the optical axis, i.e., the z-axis, then $\boldsymbol{x}_{3\mathrm{D}}$ becomes separable with

$$\boldsymbol{x}_{3\mathrm{D}}(x, y, z) = \boldsymbol{x}_{2\mathrm{D}}(x, y)\, \boldsymbol{w}(z),$$

where $\boldsymbol{w}(z)$ is the normalized flux emitted by the plane at depth z such that $\int \boldsymbol{w}(z)\mathrm{d}z = 1$. For reasonable optical setups, this shift invariance along the optical axis can be guaranteed to a sufficient degree to make this separability assumption meaningful [2].

In practice, retinal flood imaging systems typically image along a single plane of interest, a departure from the 3D model in (3). This results in a 2D data image taken at a single depth. For depth $z = 0$, this gives the observed image

$$\boldsymbol{d}_{2\mathrm{D}}(x, y) = \boldsymbol{d}_{3\mathrm{D}}(x, y, 0).$$

The shift invariance assumption for the true image $\boldsymbol{x}_{3\mathrm{D}}$ then implies that the two-dimensional PSF for the observed image $\boldsymbol{d}_{2\mathrm{D}}(x, y)$ can be expressed as

$$\boldsymbol{h}_{2\mathrm{D}}(x, y) = \int \boldsymbol{w}(-z)\, \boldsymbol{h}_{3\mathrm{D}}(x, y, z)\ \mathrm{d}z.$$

Discretizing the above integral using a quadrature rule (a simple rectangle rule is often sufficient) we obtain the PSF for the observed retinal flood images as

$$\boldsymbol{h}_{2\mathrm{D}}(x, y) \approx \sum_{j=1}^{p} w_j \boldsymbol{h}_j(x, y),$$

where $\boldsymbol{h}_j(x, y) = \boldsymbol{h}_{3\mathrm{D}}\left(x, y, z_j\right)$ is the 2D PSF taken at depth z_j and $w_j = \boldsymbol{w}\left(z_j\right)\Delta z_j$ are weights with Δz_j the thickness of the jth slice of the 3D image in the quadrature sum. Thus, in our myopic deconvolution model (2), the problem becomes determining the unknown weights w_j that parameterize the PSF with the constraints that $\sum_{j=1}^{p} w_j = 1$ and $w_j \geqslant 0$ for all j. We point out that it can be seen from numerical experiments that the global PSF in (2) is normalized (i.e., all entries sum to approximately 1). Therefore, the myopic deconvolution model satisfies the fact that the PSF is energy preserving.

2.2. Myopic deconvolution model with TV regularization

For TV regularization in (2), we follow the notations used in [19, 21]: the discrete form of TV for a grayscale image $\boldsymbol{x} \in \mathbb{R}^{n^2}$ is defined as

$$\mathrm{TV}(\boldsymbol{x}) = \sum_{i=1}^{n^2} \|D_i \boldsymbol{x}\|_2, \tag{4}$$

where for each i, $D_i \boldsymbol{x} \in \mathbb{R}^2$ represents the first-order finite difference of $\boldsymbol{x}$ at pixel i in both horizontal and vertical directions. We note that the 2-norm in (4) can be replaced by the 1-norm. If the 2-norm is used, then we obtain the isotropic version of TV, and if the 1-norm is used, we obtain the anisotropic version. In this paper, we will only treat the isotropic case for simplicity, but the treatment for the anisotropic case is completely analogous.

Notations: the two first-order global finite difference operators in horizontal and vertical directions are, respectively, denoted by $D^{(1)}, D^{(2)} \in \mathbb{R}^{n^2 \times n^2}$. $D_i \in \mathbb{R}^{2 \times n^2}$ is a two-row matrix formed by stacking the ith row of $D^{(1)}$ on top of the ith row of $D^{(2)}$ and $D := \left(D^{(1)}; D^{(2)}\right) \in \mathbb{R}^{2n^2 \times n^2}$ is the global first-order finite difference operator.

In order to deal with the restriction on the sum of the weights, we introduce an appropriate regularizer on $\boldsymbol{w}$:

$$\min_{\boldsymbol{x} \in \mathcal{C}_{\boldsymbol{x}}, \boldsymbol{w} \in \mathcal{C}_{\boldsymbol{w}}} \frac{\mu}{2} \|A(\boldsymbol{w})\boldsymbol{x} - \boldsymbol{d}\|_2^2 + \sum_{i=1}^{n^2} \|D_i \boldsymbol{x}\|_2 + S(\boldsymbol{w}) \tag{5}$$

where $\mathcal{C}_{\boldsymbol{x}} = \left\{\boldsymbol{x} | x_i \geqslant 0 \quad \text{for } i = 1, \ldots, n^2\right\}, \mathcal{C}_{\boldsymbol{w}} = \{\boldsymbol{w} | w_j \geqslant 0 \quad \text{for } j = 1, \ldots, p\}$ and $A(\boldsymbol{w}) = \sum_{j=1}^{p} w_j A_j$. The regularizer $S(\boldsymbol{w})$ is defined as

$$S(\boldsymbol{w}) = \frac{\xi}{2}(\boldsymbol{e}^\top \boldsymbol{w} - 1)^2,$$

where $\boldsymbol{e} \in \mathbb{R}^p$ is the vector all ones and $\xi > 0$ is a weighting parameter. This regularizer penalizes solutions where the weights w_k do not sum to 1, so for appropriately large ξ, it can effectively enforce the summation constraint. We choose this option because it fits conveniently within the LAP framework in section 3.2. Another possible alternative is a Lagrangian multiplier approach, which would strictly enforce the summation [15].

3. Optimization schemes

In this section, we first briefly review the ADMM, discuss how to adapt ADMM to solve (5) and discuss how to adapt a variation of the LAP method to solve the related coupled subproblems. We then propose the ADMM-LAP method for solving (5) and give the convergence analysis. A benchmark method called ADMM-BCD is also introduced in this section. We compare their computational complexity and show that ADMM-LAP is more efficient when solving (5).

3.1. ADMM splitting

The classical ADMM is designed to solve the following two-block optimization problem with linear constraints

$$\begin{aligned} &\min_{\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{n^2}} f(\boldsymbol{u}) + g(\boldsymbol{v}) \\ &\text{s.t.} \quad B\boldsymbol{u} + C\boldsymbol{v} = \boldsymbol{z}, \end{aligned} \tag{6}$$

where $f(\boldsymbol{u}):\mathbb{R}^{n^2}\to(-\infty,+\infty]$, $g(\boldsymbol{v}):\mathbb{R}^{n^2}\to(-\infty,+\infty]$ are convex functions, and $B,C\in\mathbb{R}^{n^2\times n^2}$ and $\boldsymbol{z}\in\mathbb{R}^{n^2}$ are given.

The augmented Lagrangian function for (6) is given by

$$\mathcal{L}_\beta(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda})=f(\boldsymbol{u})+g(\boldsymbol{v})+(\boldsymbol{\lambda},B\boldsymbol{u}+C\boldsymbol{v}-\boldsymbol{z})+\frac{\beta}{2}\|B\boldsymbol{u}+C\boldsymbol{v}-\boldsymbol{z}\|_2^2,$$

where $\boldsymbol{\lambda}\in\mathbb{R}^{n^2}$ denotes the Lagrange multiplier, $\beta>0$ is the penalty parameter.

Given $\left(\boldsymbol{u}^0,\boldsymbol{v}^0,\boldsymbol{\lambda}^0\right)\in\mathbb{R}^{n^2}\times\mathbb{R}^{n^2}\times\mathbb{R}^{n^2}$, the penalty parameter $\beta>0$, ADMM iterates as follows:

$$\begin{cases}\boldsymbol{u}^{k+1}=\underset{\boldsymbol{u}}{\operatorname{argmin}}\,\mathcal{L}_\beta\left(\boldsymbol{u},\boldsymbol{v}^k,\boldsymbol{\lambda}^k\right),\\ \boldsymbol{v}^{k+1}=\underset{\boldsymbol{v}}{\operatorname{argmin}}\,\mathcal{L}_\beta\left(\boldsymbol{u}^{k+1},\boldsymbol{v},\boldsymbol{\lambda}^k\right),\\ \boldsymbol{\lambda}^{k+1}=\boldsymbol{\lambda}^k-\beta\left(B\boldsymbol{u}^{k+1}+C\boldsymbol{v}^{k+1}-\boldsymbol{z}\right).\end{cases}$$

In order to apply ADMM to solve (5), we introduce artificial vectors $\mathbf{y}_i\in\mathbb{R}^2$, $i=1,\ldots,n^2$, then we can rewrite (5) in an equivalent form:

$$\begin{aligned}&\min_{\boldsymbol{x},\boldsymbol{w},\mathbf{y}}\ \frac{\mu}{2}\|A(\boldsymbol{w})\boldsymbol{x}-\boldsymbol{d}\|_2^2+S(\boldsymbol{w})+\sum_{i=1}^{n^2}\|\mathbf{y}_i\|_2+\delta_{\mathcal{C}_x}(\boldsymbol{x})+\delta_{\mathcal{C}_w}(\boldsymbol{w}),\\ &\text{s.t.}\quad \mathbf{y}_i=D_i\boldsymbol{x},\quad i=1,\ldots,n^2,\end{aligned}\tag{7}$$

where $\mathcal{C}_x=\left\{\boldsymbol{x}|x_i\geqslant 0\text{ for }i=1,\ldots,n^2\right\}$, $\mathcal{C}_w=\{\boldsymbol{w}|w_j\geqslant 0\text{ for }j=1,\ldots,p\}$, $A(\boldsymbol{w})=\sum_{j=1}^p w_jA_j$ and $\delta_{\mathcal{C}}(\cdot)$ denotes the indicator function of $\mathcal{C}$, i.e.,

$$\delta_{\mathcal{C}}(s)=\begin{cases}0, & s\in\mathcal{C},\\ \infty, & s\notin\mathcal{C}.\end{cases}$$

For convenience, let[4] $\boldsymbol{y}=[\boldsymbol{y}_1;\boldsymbol{y}_2]\in\mathbb{R}^{2n^2}$, where $\boldsymbol{y}_1,\boldsymbol{y}_2$ are vectors of length n^2 and $[(\boldsymbol{y}_1)_i;(\boldsymbol{y}_2)_i]=\mathbf{y}_i\in\mathbb{R}^2$ for $i=1,\ldots,n^2$. The augmented Lagrangian function for (7) is then given by

$$\begin{aligned}\mathcal{L}_\beta(\boldsymbol{x},\boldsymbol{w},\boldsymbol{y},\boldsymbol{\lambda}):=&\frac{\mu}{2}\|A(\boldsymbol{w})\boldsymbol{x}-\boldsymbol{d}\|_2^2+S(\boldsymbol{w})+\delta_{\mathcal{C}_x}(\boldsymbol{x})+\delta_{\mathcal{C}_w}(\boldsymbol{w})\\ &+\sum_{i=1}^{n^2}\left(\|\mathbf{y}_i\|_2-\boldsymbol{\lambda}_i^{\mathrm{T}}\left(\mathbf{y}_i-D_i\boldsymbol{x}\right)+\frac{\beta}{2}\|\mathbf{y}_i-D_i\boldsymbol{x}\|_2^2\right),\end{aligned}$$

where each $\boldsymbol{\lambda}_i\in\mathbb{R}^2$ and $\boldsymbol{\lambda}\in\mathbb{R}^{2n^2}$ is a reordering of $\boldsymbol{\lambda}_i$ similar to $\boldsymbol{y}$.

Consider $(\boldsymbol{x},\boldsymbol{w})$ as one block of variables and $\boldsymbol{y}$ as the other. We can now apply ADMM as the outer optimization method to tackle the TV regularization term in (7). Given the initial point

[4] Borrowing MATLAB notation, we use the semicolon in a vector to denote concatenation of terms.

$\left(\boldsymbol{y}^0, \boldsymbol{x}^0, \boldsymbol{w}^0, \boldsymbol{\lambda}^0\right)$, the ADMM algorithm iteratively solves the following three subproblems:

$$\begin{cases} \boldsymbol{y}^{k+1} = \underset{\boldsymbol{y}}{\operatorname{argmin}}\, \mathcal{L}_\beta\left(\boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{y}, \boldsymbol{\lambda}^k\right), \\ (\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}) = \underset{\boldsymbol{x},\boldsymbol{w}}{\operatorname{argmin}}\, \mathcal{L}_\beta\left(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right), \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \beta\left(\boldsymbol{y}^{k+1} - D\boldsymbol{x}^{k+1}\right). \end{cases}$$

First, for the $\boldsymbol{y}$-subproblem, notice that minimizing $\mathcal{L}_\beta\left(\boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{y}, \boldsymbol{\lambda}^k\right)$ with respect to $\boldsymbol{y}$ is equivalent to minimizing n^2 two-dimensional problems of the form

$$\min_{\mathbf{y}_i \in \mathbb{R}^2} \|\mathbf{y}_i\|_2 + \frac{\beta}{2}\|\mathbf{y}_i - \left(D_i\boldsymbol{x}^k + \frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i\right)\|_2^2, \quad i = 1, 2, \ldots, n^2. \tag{8}$$

The solution to (8) is given explicitly by the two-dimensional shrinkage [19]

$$\mathbf{y}_i^{k+1} = \max\left\{\left\|D_i\boldsymbol{x}^k + \frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i\right\|_2 - \frac{1}{\beta}, 0\right\} \frac{D_i\boldsymbol{x}^k + \frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i}{\|D_i\boldsymbol{x}^k + \frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i\|_2}, \quad i = 1, 2, \ldots, n^2. \tag{9}$$

Second, denote

$$R(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) := \sum_{i=1}^{n^2}\left(\|\mathbf{y}_i\|_2 - \boldsymbol{\lambda}_i^\top\left(\mathbf{y}_i - D_i\boldsymbol{x}\right) + \frac{\beta}{2}\|\mathbf{y}_i - D_i\boldsymbol{x}\|_2^2\right),$$

and define

$$\hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}, \boldsymbol{\lambda}) := \frac{\mu}{2}\|A\left(\boldsymbol{w}\right)\boldsymbol{x} - \boldsymbol{d}\|_2^2 + S(\boldsymbol{w}) + R\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}\right).$$

Then it can be shown that the $(\boldsymbol{x}, \boldsymbol{w})$-subproblem is equivalent to the following problem:

$$\left(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}\right) = \underset{\boldsymbol{x}\in\mathcal{C}_{\boldsymbol{x}}, \boldsymbol{w}\in\mathcal{C}_{\boldsymbol{w}}}{\operatorname{argmin}}\, \hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k). \tag{10}$$

Note that this is a tightly coupled optimization problem with element-wise bound constraints.

Finally, update the multiplier,

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \beta\left(\boldsymbol{y}^{k+1} - D\boldsymbol{x}^{k+1}\right). \tag{11}$$

If the termination criterion is met, stop; else, set $k := k + 1$ and go to the $\boldsymbol{y}$-subproblem.

3.2. LAP method

To solve the tightly coupled $(\boldsymbol{x}, \boldsymbol{w})$-subproblem (10), we develop a variation of the LAP method proposed by Herring *et al* [10]. The LAP method is efficient for inverse problems with multiple, tightly coupled blocks of variables such as the problem under consideration. Its strengths include the option to impose element-wise bound constraints on all blocks of variables.

In this paper, we present the LAP method based on the normal equation approach instead of the least squares approach presented in the original paper [10]. First, we consider the unconstrained problem where $\mathcal{C}_{\boldsymbol{x}} = \mathbb{R}^{n^2}$, $\mathcal{C}_{\boldsymbol{w}} = \mathbb{R}^p$. Denote the residual as $\boldsymbol{r}\left(\boldsymbol{x}, \boldsymbol{w}\right) := A\left(\boldsymbol{w}\right)\boldsymbol{x} - \boldsymbol{d}$. Then at the iterate $(\boldsymbol{x}, \boldsymbol{w})$, the Jacobian with respect to the $\boldsymbol{x}$ block of variables is

$$\mathbf{J}_{\boldsymbol{x}} = \nabla_{\boldsymbol{x}}\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{w})^\top = \boldsymbol{A}(\boldsymbol{w})^\top, \tag{12}$$

and the Jacobian with respect to the $\boldsymbol{w}$ block of variables is

$$\mathbf{J}_w = \begin{bmatrix} (A_1\boldsymbol{x})^\top \\ (A_2\boldsymbol{x})^\top \\ \vdots \\ (A_p\boldsymbol{x})^\top \end{bmatrix}. \tag{13}$$

Computing the update step around the current iterate $(\boldsymbol{x}, \boldsymbol{w})$ requires the gradient and Hessian of the objective function $\hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k)$. These are given by

$$\nabla_{x,w}\hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k) = \mu \begin{bmatrix} \mathbf{J}_x^\top \boldsymbol{r} \\ \mathbf{J}_w^\top \boldsymbol{r} \end{bmatrix} + \begin{bmatrix} \nabla_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right) \\ \nabla_w S(\boldsymbol{w}) \end{bmatrix}, \tag{14}$$

$$\nabla^2_{x,w}\hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k) \approx \mu \begin{bmatrix} \mathbf{J}_x^\top \mathbf{J}_x & \mathbf{J}_x^\top \mathbf{J}_w \\ \mathbf{J}_w^\top \mathbf{J}_x & \mathbf{J}_w^\top \mathbf{J}_w \end{bmatrix} + \begin{bmatrix} \nabla^2_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right) & \mathbf{0} \\ \mathbf{0} & \nabla^2_w S(\boldsymbol{w}) \end{bmatrix}, \tag{15}$$

where $\boldsymbol{r} := \boldsymbol{r}(\boldsymbol{x}, \boldsymbol{w})$. Here, the Hessian of the regularizer $R(\boldsymbol{x})$ can be exact or a linearized approximation. Let $\delta\boldsymbol{x}$ and $\delta\boldsymbol{w}$ denote the update step for the image and the parameters, respectively. Then the update steps $\delta\boldsymbol{x}$ and $\delta\boldsymbol{w}$ are given by the solution of the following block linear system

$$\begin{aligned} &\begin{bmatrix} \mathbf{J}_x^\top \mathbf{J}_x + \dfrac{1}{\mu}\nabla^2_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right) & \mathbf{J}_x^\top \mathbf{J}_w \\ \mathbf{J}_w^\top \mathbf{J}_x & \mathbf{J}_w^\top \mathbf{J}_w + \nabla^2_w S(\boldsymbol{w}) \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{w} \end{bmatrix} \\ &\quad = -\begin{bmatrix} \mathbf{J}_x^\top \boldsymbol{r} + \dfrac{1}{\mu}\nabla_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right) \\ \mathbf{J}_w^\top \boldsymbol{r} + \nabla_w S(\boldsymbol{w}) \end{bmatrix}. \end{aligned} \tag{16}$$

Note that omitting the regularizer terms, (16) is the normal equation corresponding to the least-squares problem

$$\frac{\mu}{2}\left\| \begin{bmatrix} \mathbf{J}_x & \mathbf{J}_w \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{x} \\ \delta\boldsymbol{w} \end{bmatrix} + \boldsymbol{r} \right\|_2^2$$

that can be obtained by the *Linearize* step of LAP in [10].

LAP solves (16) by projecting the original problem onto a reduced space. In this paper, we choose to project the problem onto the image space, i.e., we eliminate the block of variables corresponding to $\boldsymbol{w}$. When projecting the problem onto the image space, $\delta\boldsymbol{w}$ can be computed by

$$\delta\boldsymbol{w} = -\left(\mathbf{J}_w^\top \mathbf{J}_w + \nabla^2_w S(\boldsymbol{w})\right)^{-1}\left(\mathbf{J}_w^\top \mathbf{J}_x \delta\boldsymbol{x} + \mathbf{J}_w^\top \boldsymbol{r} + \nabla_w S(\boldsymbol{w})\right). \tag{17}$$

Plug (17) into (16) and get

$$\begin{aligned} &\left(\mathbf{J}_x^\top \mathbf{J}_x + \frac{1}{\mu}\nabla^2_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)\right)\delta\boldsymbol{x} - \mathbf{J}_x^\top \mathbf{J}_w\left(\mathbf{J}_w^\top \mathbf{J}_w + \nabla^2_w S(\boldsymbol{w})\right)^{-1} \\ &\qquad \times \left(\mathbf{J}_w^\top \mathbf{J}_x \delta\boldsymbol{x} + \mathbf{J}_w^\top \boldsymbol{r} + \nabla_w S(\boldsymbol{w})\right) = -\left(\mathbf{J}_x^\top \boldsymbol{r} + \frac{1}{\mu}\nabla_x R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)\right), \end{aligned}$$

which can be simplified as

$$\boldsymbol{M}\delta\boldsymbol{x} = \boldsymbol{b}, \tag{18}$$

the operator and the right-hand side are given by

$$\begin{aligned}\boldsymbol{M} &:= \mathbf{J}_{\boldsymbol{x}}^{\top}(\boldsymbol{I} - \mathbf{J}_{\boldsymbol{w}}(\mathbf{J}_{\boldsymbol{w}}^{\top}\mathbf{J}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}S(\boldsymbol{w}))^{-1}\mathbf{J}_{\boldsymbol{w}}^{\top})\mathbf{J}_{\boldsymbol{x}} + \frac{1}{\mu}\nabla_{\boldsymbol{x}}^{2}R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right),\\ \boldsymbol{b} &:= -\mathbf{J}_{\boldsymbol{x}}^{\top}(\boldsymbol{I} - \mathbf{J}_{\boldsymbol{w}}(\mathbf{J}_{\boldsymbol{w}}^{\top}\mathbf{J}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}S(\boldsymbol{w}))^{-1}\mathbf{J}_{\boldsymbol{w}}^{\top})\boldsymbol{r} - \frac{1}{\mu}\nabla_{\boldsymbol{x}}R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right)\\ &\quad + \mathbf{J}_{\boldsymbol{x}}^{\top}\mathbf{J}_{\boldsymbol{w}}(\mathbf{J}_{\boldsymbol{w}}^{\top}\mathbf{J}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}S(\boldsymbol{w}))^{-1}\nabla_{\boldsymbol{w}}S(\boldsymbol{w}).\end{aligned}$$

Moreover, it is easy to see that the gradient and Hessian of $R(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k})$ satisfy

$$\begin{aligned}\nabla_{\boldsymbol{x}}R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) &= \sum_{i=1}^{n^2}\left(D_i^{\mathrm{T}}(\boldsymbol{\lambda}^{k})_i - \beta D_i^{\mathrm{T}}\left(\boldsymbol{y}_i^{k+1} - D_i\boldsymbol{x}\right)\right)\\ &= D^{\mathrm{T}}\boldsymbol{\lambda}^{k} - \beta D^{\mathrm{T}}\boldsymbol{y}^{k+1} + \beta D^{\mathrm{T}}D\boldsymbol{x},\end{aligned} \tag{19}$$

and

$$\nabla_{\boldsymbol{x}}^{2}R\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) = \sum_{i=1}^{n^2}\beta D_i^{\mathrm{T}}D_i = \beta D^{\mathrm{T}}D. \tag{20}$$

Following the above procedures, we are able to compute the unconstrained update steps $\delta\boldsymbol{x}$ and $\delta\boldsymbol{w}$.

Next, we consider modifying the above procedures in order to handle the element-wise bound constraints on $\boldsymbol{x}$ and $\boldsymbol{w}$. We know that a simple extension to the Gauss–Newton method does not work [12], however, a simple correction, i.e. the projected Gauss–Newton method [8] can be made convergent and it works well for many inverse problems. To this end, the variables are divided into active set variables and inactive set variables and the step $\delta\boldsymbol{x}$ and $\delta\boldsymbol{w}$ are computed through these two separate sets. Let the feasible index set be defined as

$$\mathcal{N} := \left\{ q \in \mathbb{N} \;\middle|\; \begin{bmatrix}\boldsymbol{x}_q\\ \boldsymbol{w}_q\end{bmatrix} \geqslant \boldsymbol{0} \right\}.$$

Define the active and inactive sets as

$$\begin{aligned}\mathcal{A} &:= \left\{ q \in \mathbb{N} \;\middle|\; \begin{bmatrix}\boldsymbol{x}_q\\ \boldsymbol{w}_q\end{bmatrix} = \boldsymbol{0} \right\},\\ \mathcal{I} &:= \mathcal{N}\backslash\mathcal{A}.\end{aligned}$$

Then, we can divide the variables into the active set variables $\begin{bmatrix}\boldsymbol{x}_{\mathcal{A}}\\ \boldsymbol{w}_{\mathcal{A}}\end{bmatrix}$ and the inactive set variables $\begin{bmatrix}\boldsymbol{x}_{\mathcal{I}}\\ \boldsymbol{w}_{\mathcal{I}}\end{bmatrix}$. Denote the steps taken on $\begin{bmatrix}\boldsymbol{x}_{\mathcal{A}}\\ \boldsymbol{w}_{\mathcal{A}}\end{bmatrix}$ by $\begin{bmatrix}\delta\boldsymbol{x}_{\mathcal{A}}\\ \delta\boldsymbol{w}_{\mathcal{A}}\end{bmatrix}$ and the steps taken on $\begin{bmatrix}\boldsymbol{x}_{\mathcal{I}}\\ \boldsymbol{w}_{\mathcal{I}}\end{bmatrix}$ by $\begin{bmatrix}\delta\boldsymbol{x}_{\mathcal{I}}\\ \delta\boldsymbol{w}_{\mathcal{I}}\end{bmatrix}$, respectively.

$\delta \boldsymbol{x}_{\mathcal{I}}$ and $\delta \boldsymbol{w}_{\mathcal{I}}$ can be computed in a similar way as the unconstrained case except that the variables need to be projected onto the inactive set. That is, $\delta \boldsymbol{x}_{\mathcal{I}}$ at the current iterate $(\boldsymbol{x}, \boldsymbol{w})$ is computed as:

$$\hat{\boldsymbol{M}} \delta \boldsymbol{x}_{\mathcal{I}} = \hat{\boldsymbol{b}}, \tag{21}$$

the operator and right-hand side are given by

$$\begin{aligned}
\hat{\boldsymbol{M}} &\coloneqq \hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top})\hat{\mathbf{J}}_{\boldsymbol{x}} + \frac{1}{\mu}\nabla_{\boldsymbol{x}}^{2}\hat{R}\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right), \\
\hat{\boldsymbol{b}} &\coloneqq -\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top})\boldsymbol{r} - \frac{1}{\mu}\nabla_{\boldsymbol{x}}\hat{R}\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) \\
&\quad + \hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}\hat{S}(\boldsymbol{w}))^{-1}\nabla_{\boldsymbol{w}}\hat{S}(\boldsymbol{w}),
\end{aligned}$$

where $\hat{\mathbf{J}}_{\boldsymbol{x}}$, $\hat{\mathbf{J}}_{\boldsymbol{w}}$, $\nabla_{\boldsymbol{x}}\hat{R}$, $\nabla_{\boldsymbol{x}}^{2}\hat{R}$, $\nabla_{\boldsymbol{w}}\hat{S}$ and $\nabla_{\boldsymbol{w}}^{2}\hat{S}$ represent $\mathbf{J}_{\boldsymbol{x}}$, $\mathbf{J}_{\boldsymbol{w}}$, $\nabla_{\boldsymbol{x}}R$, $\nabla_{\boldsymbol{x}}^{2}R$, $\nabla_{\boldsymbol{w}}S$ and $\nabla_{\boldsymbol{w}}^{2}S$ restricted to the inactive set via projection, respectively. The reduced problem (21) does not need to be solved to a high accuracy. For example in [10], a stopping tolerance of 10^{-1} is used to solve the reduced problem iteratively. After solving for $\delta \boldsymbol{x}_{\mathcal{I}}$, $\delta \boldsymbol{w}_{\mathcal{I}}$ can be computed by

$$\delta \boldsymbol{w}_{\mathcal{I}} = -\left(\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^{2}\hat{S}(\boldsymbol{w})\right)^{-1}\left(\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{x}}\delta \boldsymbol{x}_{\mathcal{I}} + \hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\boldsymbol{r} + \nabla_{\boldsymbol{w}}\hat{S}(\boldsymbol{w})\right). \tag{22}$$

For the active set, $\delta \boldsymbol{x}_{\mathcal{A}}$ and $\delta \boldsymbol{w}_{\mathcal{A}}$ are given by a scaled projected gradient descent step

$$\begin{bmatrix} \delta \boldsymbol{x}_{\mathcal{A}} \\ \delta \boldsymbol{w}_{\mathcal{A}} \end{bmatrix} = -\mu \begin{bmatrix} \tilde{\mathbf{J}}_{\boldsymbol{x}}^{\top}\boldsymbol{r} \\ \tilde{\mathbf{J}}_{\boldsymbol{w}}^{\top}\boldsymbol{r} \end{bmatrix} - \begin{bmatrix} \nabla_{\boldsymbol{x}}\tilde{R}\left(\boldsymbol{x} + \delta \boldsymbol{x}_{\mathcal{A}}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) \\ \nabla_{\boldsymbol{w}}\tilde{S}(\boldsymbol{w}) \end{bmatrix}, \tag{23}$$

where $\tilde{\mathbf{J}}_{\boldsymbol{x}}$, $\tilde{\mathbf{J}}_{\boldsymbol{w}}$, $\nabla_{\boldsymbol{x}}\tilde{R}$ and $\nabla_{\boldsymbol{w}}\tilde{S}$ represent the projection of $\mathbf{J}_{\boldsymbol{x}}$, $\mathbf{J}_{\boldsymbol{w}}$, $\nabla_{\boldsymbol{x}}R$ and $\nabla_{\boldsymbol{w}}S$ onto the active set, respectively.

Then $\delta \boldsymbol{x}$ and $\delta \boldsymbol{w}$ can be calculated as a scaled combination of $\delta \boldsymbol{x}_{\mathcal{A}}$, $\delta \boldsymbol{w}_{\mathcal{A}}$, $\delta \boldsymbol{x}_{\mathcal{I}}$ and $\delta \boldsymbol{w}_{\mathcal{I}}$ by

$$\begin{bmatrix} \delta \boldsymbol{x} \\ \delta \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \delta \boldsymbol{x}_{\mathcal{I}} \\ \delta \boldsymbol{w}_{\mathcal{I}} \end{bmatrix} + \gamma \begin{bmatrix} \delta \boldsymbol{x}_{\mathcal{A}} \\ \delta \boldsymbol{w}_{\mathcal{A}} \end{bmatrix}, \tag{24}$$

and the parameter γ is selected based on the recommendation in [8]

$$\gamma = \frac{\max\left(\|\delta \boldsymbol{x}_{\mathcal{I}}\|_{\infty}, \|\delta \boldsymbol{w}_{\mathcal{I}}\|_{\infty}\right)}{\max\left(\|\delta \boldsymbol{x}_{\mathcal{A}}\|_{\infty}, \|\delta \boldsymbol{w}_{\mathcal{A}}\|_{\infty}\right)}.$$

Finally, the projected Armijo line search is applied to find the solution. Here the modified Armijo condition is given by

$$\begin{aligned}
&\hat{\Phi}\left(\mathbf{P}_{\mathcal{C}_{\boldsymbol{x}}}\left(\boldsymbol{x} + \eta \delta \boldsymbol{x}\right), \mathbf{P}_{\mathcal{C}_{\boldsymbol{w}}}\left(\boldsymbol{w} + \eta \delta \boldsymbol{w}\right), \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) \\
&\qquad \leqslant \hat{\Phi}\left(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right) + c\eta \mathbf{Q}\left(\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right)\right)^{\top}\begin{bmatrix} \delta \boldsymbol{x} \\ \delta \boldsymbol{w} \end{bmatrix},
\end{aligned} \tag{25}$$

where $\mathbf{P}_{\mathcal{C}_{\boldsymbol{x}}}$ and $\mathbf{P}_{\mathcal{C}_{\boldsymbol{w}}}$ denotes the projections onto the feasible set for the image and parameters, respectively, $\mathbf{Q}\left(\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k}\right)\right)$ denotes the projected gradient, $0 < \eta \leqslant 1$ denotes the step size by backtracking, and we set $c = 10^{-4}$ as suggested in [15]. We point out that it is necessary to update the inactive set and the active set in each iteration because the projection

Algorithm 1. ADMM-LAP method for (7).

Input: $\left(\boldsymbol{y}^0, \boldsymbol{x}^0, \boldsymbol{w}^0, \boldsymbol{\lambda}^0\right) \in \mathbb{R}^{2n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^p \times \mathbb{R}^{2n^2}$, the penalty parameter $\beta > 0$.
Let $\{\epsilon_{k+1}\}_{k=0}^{\infty}$ be a sequence satisfying $\{\epsilon_{k+1}\}_{k=0}^{\infty} \subseteq [0, +\infty)$ and $\sum_{k=0}^{\infty} \epsilon_{k+1} < \infty$. Set $k = 0$.
Output: $\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k$.
Step 1 Compute $\boldsymbol{y}^{k+1}$ using (9).
Step 2 Find a minimizer of

$$\min_{\boldsymbol{x}\in\mathcal{C}_{\boldsymbol{x}}, \boldsymbol{w}\in\mathcal{C}_{\boldsymbol{w}}} \hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k).$$

Specifically,
1. Compute the step on the inactive set by the LAP method using (21) and (22).
2. Compute the step on the active set by (23) using the projected gradient descent method.
3. Combine the steps using (24).
4. Perform the projected Armijo line search satisfying (25) to update $\boldsymbol{x}^{k+1}$, $\boldsymbol{w}^{k+1}$.
5. Update active and inactive sets.

Step 3 if the residual $\begin{bmatrix} \eta_{\boldsymbol{x}}^{k+1} \\ \eta_{\boldsymbol{w}}^{k+1} \end{bmatrix} \coloneqq \nabla_{(\boldsymbol{x},\boldsymbol{w})} \hat{\Phi}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k)$ satisfies $\left\| \begin{bmatrix} \eta_{\boldsymbol{x}}^{k+1} \\ \eta_{\boldsymbol{w}}^{k+1} \end{bmatrix} \right\|_2 \leqslant \epsilon_{k+1}$, stop and go to step 4; else, repeat step 2.
Step 4 Compute $\boldsymbol{\lambda}^{k+1}$ using (11).
Step 5 If a termination criteria is met, stop; else, set $k \coloneqq k + 1$ and go to step 1.

does not prevent variables from leaving the active set and joining the inactive set. Hence, it makes sense that in algorithm 1, if the termination criterion is not met in step 3, we go back to step 2 and get a new solution.

3.3. ADMM-LAP method

The proposed ADMM-LAP method for solving myopic deconvolution problems with TV regularization is summarized in algorithm 1.

Notice that ADMM-LAP is an inexact ADMM, where the $\boldsymbol{y}$-subproblems are exactly solved while the $(\boldsymbol{x}, \boldsymbol{w})$-subproblems are inexactly solved. Inspired by the recent results in [14, 22], we prove that the ADMM-LAP algorithm converges subsequently to a stationary point.

Theorem 1. *Let $(\boldsymbol{y}^0, \boldsymbol{x}^0, \boldsymbol{w}^0, \boldsymbol{\lambda}^0)$ be any initial point and $\{(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\}$ be the sequence of iterates generated by algorithm 1. Then if $\beta > \frac{1}{a_1}\lambda_{\max}(A(\boldsymbol{w}^k)^{\mathrm{T}}A(\boldsymbol{w}^k))$ and β satisfies $\beta^2 a_1 - L\beta - 4a_2C_0 > 0$, where C_0, a_1, a_2, L are constants specified in lemmas 1 and 2, algorithm 1 converges subsequently, i.e., it generates a sequence that has a convergent subsequence, whose limit $(\boldsymbol{y}^*, \boldsymbol{x}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ is a stationary point of $\mathcal{L}_\beta$. That is, $0 \in \partial\mathcal{L}_\beta(\boldsymbol{y}^*, \boldsymbol{x}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}^*)$.*

To prove theorem 1, we define the following functions:

$$F(\boldsymbol{y}) = \sum_{i=1}^{n^2} \|\mathbf{y}_i\|_2,$$

$$G(\boldsymbol{x}, \boldsymbol{w}) = \frac{\mu}{2}\|A(\boldsymbol{w})\boldsymbol{x} - \boldsymbol{d}\|_2^2 + S(\boldsymbol{w}) + \delta_{\mathcal{C}_{\boldsymbol{x}}}(\boldsymbol{x}) + \delta_{\mathcal{C}_{\boldsymbol{w}}}(\boldsymbol{w}),$$

$$g(\boldsymbol{x}, \boldsymbol{w}) = \frac{\mu}{2}\|A(\boldsymbol{w})\boldsymbol{x} - \boldsymbol{d}\|_2^2 + S(\boldsymbol{w}).$$

Lemma 1. *The iterates of algorithm 1 satisfy:*

(*a*) $\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}) = -\sum_{i=1}^{n^2} D_i^{\mathrm{T}} \boldsymbol{\lambda}_i^{k+1} + \eta_{\boldsymbol{x}}^{k+1}$.

(*b*) $\|\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k+1} - \sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k}\|_2 \leqslant C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^{k}\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^{k}\end{bmatrix}\right\|_2 + \|\eta_{\boldsymbol{x}}^{k+1}-\eta_{\boldsymbol{x}}^{k}\|_2$, *where* C_0 *is a constant.*

Proof. First, notice that for $(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})$ generated by ADMM, we have $\boldsymbol{x}^{k+1} = [x_1^{k+1}; x_2^{k+1}; \ldots; x_{n^2}^{k+1}] \in \mathcal{C}_{\boldsymbol{x}}$, $\boldsymbol{w}^{k+1} = [w_1^{k+1}; w_2^{k+1}; \ldots; w_p^{k+1}] \in \mathcal{C}_{\boldsymbol{w}}$. From the definition of the general subgradient, for all $\boldsymbol{v}_{\boldsymbol{x}} \in \partial\delta_{\mathcal{C}_{\boldsymbol{x}}}(\boldsymbol{x}^{k+1})$,

$$\delta_{\mathcal{C}_{\boldsymbol{x}}}(\boldsymbol{x}) - \delta_{\mathcal{C}_{\boldsymbol{x}}}(\boldsymbol{x}^{k+1}) \geqslant \boldsymbol{v}_{\boldsymbol{x}}\cdot(\boldsymbol{x}-\boldsymbol{x}^{k+1}), \quad \forall\, \boldsymbol{x}\in\mathbb{R}^{n^2}.$$

For any $\boldsymbol{x} = [x_1; x_2; \ldots; x_{n^2}]$, where $x_i \geqslant x_i^{k+1} \geqslant 0, i = 1,2,\ldots,n^2$, we have $\mathbf{0} \geqslant \boldsymbol{v}_{\boldsymbol{x}}\cdot(\boldsymbol{x}-\boldsymbol{x}^{k+1})$. For any $\boldsymbol{x} = [x_1; x_2; \ldots; x_{n^2}]$, where $0 \leqslant x_i \leqslant x_i^{k+1}$, $i = 1,2,\ldots,n^2$, we have $\mathbf{0} \leqslant \boldsymbol{v}_{\boldsymbol{x}}\cdot(\boldsymbol{x}-\boldsymbol{x}^{k+1})$. Thus, we have $\boldsymbol{v}_{\boldsymbol{x}} = \mathbf{0}$. Similarly, $\boldsymbol{v}_{\boldsymbol{w}} = \mathbf{0}$. By the first-order optimality condition at $(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})$,

$$\begin{bmatrix}\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})\\ \nabla_{\boldsymbol{w}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})\end{bmatrix} + \begin{bmatrix}\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k}\\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\sum_{i=1}^{n^2}\beta D_i^{\mathrm{T}}(\boldsymbol{y}_i^{k+1} - D_i\boldsymbol{x})\\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\eta_{\boldsymbol{x}}^{k+1}\\ \eta_{\boldsymbol{w}}^{k+1}\end{bmatrix} \in \partial\delta_{\mathcal{C}_{\boldsymbol{x}}\times\mathcal{C}_{\boldsymbol{w}}}(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1}).$$

Hence,

$$\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1}) + \sum_{i=1}^{n^2}\left(D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k} - \beta D_i^{\mathrm{T}}(\boldsymbol{y}_i^{k+1} - D_i\boldsymbol{x})\right) - \eta_{\boldsymbol{x}}^{k+1} = \mathbf{0}.$$

Then by the update of the multiplier $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^{k} - \beta\left(\boldsymbol{y}^{k+1} - D\boldsymbol{x}^{k+1}\right)$, we obtain

$$\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1}) = -\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k+1} + \eta_{\boldsymbol{x}}^{k+1}. \tag{26}$$

We make use of the decomposition

$$\begin{aligned}\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})\|_2 &\leqslant \|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k+1})\|_2\\ &\quad + \|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k+1}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1})\|_2.\end{aligned} \tag{27}$$

For the term $\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k+1})\|_2$, we have the estimate

$$\begin{aligned}&\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k},\boldsymbol{w}^{k+1})\|_2\\ &\quad= \|A(\boldsymbol{w}^{k})^{\mathrm{T}}A(\boldsymbol{w}^{k})\boldsymbol{x}^{k} - A(\boldsymbol{w}^{k})^{\mathrm{T}}\boldsymbol{d} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1})\boldsymbol{x}^{k} + A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\boldsymbol{d}\|_2\\ &\quad\leqslant \|A(\boldsymbol{w}^{k})^{\mathrm{T}}A(\boldsymbol{w}^{k})\boldsymbol{x}^{k} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k})\boldsymbol{x}^{k}\|_2\\ &\qquad+ \|A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k})\boldsymbol{x}^{k} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1})\boldsymbol{x}^{k}\|_2\\ &\qquad+ \|A(\boldsymbol{w}^{k})^{\mathrm{T}}\boldsymbol{d} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\boldsymbol{d}\|_2.\end{aligned} \tag{28}$$

Then, we estimate the terms in (28) separately with the two-weight case, i.e., $\boldsymbol{w} = [w; 1-w]$. Cases with $p \geqslant 3$ can be considered similarly, here we omit it. First,

$$\begin{aligned}
&\|A(\boldsymbol{w}^k)^{\mathrm{T}}A(\boldsymbol{w}^k)\boldsymbol{x}^k - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2 \\
&\quad\leqslant \|A(\boldsymbol{w}^k)^{\mathrm{T}} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\|_2 \cdot \|A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2 \\
&\quad\leqslant \|w^k A_1^{\mathrm{T}} + (1-w^k)A_2^{\mathrm{T}} - w^{k+1}A_1^{\mathrm{T}} - (1-w^{k+1})A_2^{\mathrm{T}}\|_2 \cdot \|A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2 \\
&\quad\leqslant \|A_1^{\mathrm{T}} - A_2^{\mathrm{T}}\|_2 \cdot \|A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2 \cdot \|w^k - w^{k+1}\|_2, \\
&\quad= \lambda_{\max}(A_1^{\mathrm{T}} - A_2^{\mathrm{T}}) \cdot \|A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2 \cdot \|w^k - w^{k+1}\|_2, \\
&\quad= c_1\|w^k - w^{k+1}\|_2, \qquad (29)
\end{aligned}$$

where $c_1 \coloneqq \lambda_{\max}(A_1^{\mathrm{T}} - A_2^{\mathrm{T}}) \cdot \|A(\boldsymbol{w}^k)\boldsymbol{x}^k\|_2$ is a constant. Similarly,

$$\begin{aligned}
&\|A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^k)\boldsymbol{x}^k - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1})\boldsymbol{x}^k\|_2 \\
&\quad\leqslant \lambda_{\max}(A_1 - A_2) \cdot \|A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\|_2 \cdot \|\boldsymbol{x}^k\|_2 \cdot \|w^k - w^{k+1}\|_2, \\
&\quad= c_2\|w^k - w^{k+1}\|_2, \qquad (30)
\end{aligned}$$

where $c_2 \coloneqq \lambda_{\max}(A_1 - A_2) \cdot \|A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\|_2 \cdot \|\boldsymbol{x}^k\|_2$ is a constant.

Moreover,

$$\begin{aligned}
\|A(\boldsymbol{w}^k)^{\mathrm{T}}\boldsymbol{d} - A(\boldsymbol{w}^{k+1})^{\mathrm{T}}\boldsymbol{d}\|_2 &\leqslant \|(A_1 - A_2)^{\mathrm{T}}\boldsymbol{d}\|_2 \cdot \|w^k - w^{k+1}\|_2 \\
&= c_3\|w^k - w^{k+1}\|_2, \qquad (31)
\end{aligned}$$

where $c_3 \coloneqq \|(A_1 - A_2)^{\mathrm{T}}\boldsymbol{d}\|_2$ is a constant.

For the term $\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^k, \boldsymbol{w}^{k+1}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1})\|_2$, we have

$$\begin{aligned}
\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^k, \boldsymbol{w}^{k+1}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1})\|_2 &= \mu\|A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1})(\boldsymbol{x}^k - \boldsymbol{x}^{k+1})\|_2 \\
&\leqslant \mu\lambda_{\max}(A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1})) \cdot \|\boldsymbol{x}^k - \boldsymbol{x}^{k+1}\|_2, \qquad (32)
\end{aligned}$$

Then we know from (26)–(32) that there exists a constant C_0 such that

$$\begin{aligned}
&\left\|\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k+1} - \sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^k\right\|_2 - \|\eta_{\boldsymbol{x}}^k - \eta_{\boldsymbol{x}}^{k+1}\|_2 \\
&\quad\leqslant \left\|-\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k+1} + \eta_{\boldsymbol{x}}^{k+1} + \sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^k - \eta_{\boldsymbol{x}}^k\right\|_2 \\
&\quad= \left\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^k, \boldsymbol{w}^k)\right\|_2 \\
&\quad\leqslant C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k\end{bmatrix}\right\|_2,
\end{aligned}$$

where $C_0 \coloneqq \sqrt{2}\max\{c_1 + c_2 + c_3, \mu\lambda_{\max}(A(\boldsymbol{w}^{k+1})^{\mathrm{T}}A(\boldsymbol{w}^{k+1}))\}$. Hence, we have

$$\left\|\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^{k+1} - \sum_{i=1}^{n^2} D_i^{\mathrm{T}}\boldsymbol{\lambda}_i^k\right\| \leqslant C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k\end{bmatrix}\right\|_2^2 + \|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2.$$

□

Lemma 2. *Let* $\{(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\}$ *be the sequence of iterates generated by algorithm* 1. *If* $\beta > \frac{1}{a_1}\lambda_{\max}(A(\boldsymbol{w}^k)^{\mathrm{T}}A(\boldsymbol{w}^k))$ *and* β *satisfies* $\beta^2 a_1 - L\beta - 4a_2C_0 > 0$, *where* a_1, a_2 *are constants depending on D, L denotes the Lipschitz constant of* $-g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^k)$, *then* $\{(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\}$ *satisfies:*

(*a*) $\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)$ *is lower bounded and there is a constant* $C_1 > 0$ *such that for all sufficiently large k, we have*

$$\begin{aligned}&\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k) - \mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})\\ &\quad\geqslant C_1\left(\sum_{i=1}^{n^2}\|\mathbf{y}_i^k - \mathbf{y}_i^{k+1}\|_2^2 + \left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2\right) - \frac{2}{\beta}\|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2^2.\end{aligned}$$

(*b*) $\{(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\}$ *is bounded.*

Proof. According to the optimality condition of the $\boldsymbol{y}$-subproblem, we define

$$d_i^{k+1} := \left(\boldsymbol{\lambda}^k\right)_i + \beta(\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^k) \in \partial\|\mathbf{y}_i^{k+1}\|_2.$$

We know from the definition of $\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)$ that

$$\begin{aligned}&\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k) - \mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\\ &= F(\boldsymbol{y}^k) - F(\boldsymbol{y}^{k+1}) - \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^k\right)_i^{\mathrm{T}}(\mathbf{y}_i^k - \mathbf{y}_i^{k+1}) + \frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k - D_i\boldsymbol{x}^k\|_2 - \frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^k\|_2\\ &= F(\boldsymbol{y}^k) - F(\boldsymbol{y}^{k+1}) - \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^k\right)_i^{\mathrm{T}}(\mathbf{y}_i^k - \mathbf{y}_i^{k+1}) + \frac{\beta}{2}\sum_{i=1}^{n^2}\left(\|\mathbf{y}_i^k - \mathbf{y}_i^{k+1}\|_2^2 + 2\langle\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^k, \mathbf{y}_i^k - \mathbf{y}_i^{k+1}\rangle\right)\\ &= \sum_{i=1}^{n^2}\left(\|\mathbf{y}_i^k\|_2 - \|\mathbf{y}_i^{k+1}\|_2 - \langle d_i^{k+1}, \mathbf{y}_i^k - \mathbf{y}_i^{k+1}\rangle\right) + \frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k - \mathbf{y}_i^{k+1}\|_2^2\\ &\geqslant \frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k - \mathbf{y}_i^{k+1}\|_2^2,\end{aligned}$$

where the second equality follows from the cosine rule: $\|b+c\|^2 - \|a+c\|^2 = \|b-a\|^2 + 2\langle a+c, b-a\rangle$ and the last inequality follows from the convexity of $\|\mathbf{y}\|_2$.

Moreover, we have

$$\begin{aligned}&\mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k) - \mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})\\ &= g(\boldsymbol{x}^k, \boldsymbol{w}^k) - g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}) - \sum_{i=1}^{n^2}\left((\boldsymbol{\lambda}^k)_i^{\mathrm{T}} - (\boldsymbol{\lambda}^{k+1})_i^{\mathrm{T}}\right)\mathbf{y}_i^{k+1} + \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^k\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^k\\ &\quad - \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^{k+1}\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^{k+1} + \sum_{i=1}^{n^2}\frac{\beta}{2}\|\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^k\|_2 + \sum_{i=1}^{n^2}\frac{\beta}{2}\|\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^{k+1}\|_2\\ &= g(\boldsymbol{x}^k, \boldsymbol{w}^k) - g(\boldsymbol{x}^k, \boldsymbol{w}^{k+1}) + g(\boldsymbol{x}^k, \boldsymbol{w}^{k+1}) - g(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}) - \sum_{i=1}^{n^2}\left((\boldsymbol{\lambda}^k)_i^{\mathrm{T}} - (\boldsymbol{\lambda}^{k+1})_i^{\mathrm{T}}\right)\mathbf{y}_i^{k+1}\\ &\quad + \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^k\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^k - \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^{k+1}\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^k + \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^{k+1}\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^k - \sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^{k+1}\right)_i^{\mathrm{T}}D_i\boldsymbol{x}^k\end{aligned}$$

$$+\frac{\beta}{2}\sum_{i=1}^{n^2}\left(\|-D_i\boldsymbol{x}^k+D_i\boldsymbol{x}^{k+1}\|_2+2\langle -D_i\boldsymbol{x}^{k+1}+\mathbf{y}_i^{k+1},-D_i\boldsymbol{x}^k+D_i\boldsymbol{x}^{k+1}\rangle\right)$$

$$\geqslant -\frac{L}{2}\|\boldsymbol{w}^{k+1}-\boldsymbol{w}^k\|_2^2+\frac{\beta}{2}\sum_{i=1}^{n^2}\left(\|-D_i\boldsymbol{x}^k+D_i\boldsymbol{x}^{k+1}\|_2+2\langle -D_i\boldsymbol{x}^{k+1}+\mathbf{y}_i^{k+1},-D_i\boldsymbol{x}^k+D_i\boldsymbol{x}^{k+1}\rangle\right)$$

$$+\sum_{i=1}^{n^2}\beta\langle \mathbf{y}_i^{k+1}-D_i\boldsymbol{x}^{k+1},-\mathbf{y}_i^{k+1}+D_i\boldsymbol{x}^{k+1}\rangle$$

$$=-\frac{L}{2}\|\boldsymbol{w}^{k+1}-\boldsymbol{w}^k\|_2^2-\frac{1}{\beta}\sum_{i=1}^{n^2}\|\left(\boldsymbol{\lambda}^{k+1}\right)_i-\left(\boldsymbol{\lambda}^k\right)_i\|_2^2+\frac{\beta}{2}\sum_{i=1}^{n^2}\|-D_i\boldsymbol{x}^k+D_i\boldsymbol{x}^{k+1}\|_2^2$$

$$\geqslant -\frac{L}{2}\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2-\frac{2a_2C_0}{\beta}\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2+\frac{\beta a_1}{2}\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2-\frac{2}{\beta}\|\eta_{\boldsymbol{x}}^k-\eta_{\boldsymbol{x}}^{k+1}\|_2^2$$

$$=C\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2-\frac{2}{\beta}\|\eta_{\boldsymbol{x}}^k-\eta_{\boldsymbol{x}}^{k+1}\|_2^2,$$

where $C:=-\frac{L}{2}-\frac{2a_2C_0}{\beta}+\frac{\beta a_1}{2}$ is a constant, a_1,a_2 are constants such that for $i=1,2,\ldots,n^2$, $\|D_i(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\|_2^2\geqslant a_1\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\|_2^2$ and $\|(\lambda^{k+1})_i-(\lambda^k)_i\|_2^2\leqslant a_2\|D_i^{\mathrm{T}}\left((\lambda^{k+1})_i-(\lambda^k)_i\right)\|_2^2$, respectively. The second equality follows from the cosine rule; the third inequality follows from the convexity of $g(\boldsymbol{x}^k,\boldsymbol{w}^{k+1})$ with respect to $\boldsymbol{x}$ and the Lipschitz differentiable property of $-g(\boldsymbol{x}^{k+1},\boldsymbol{w}^k)$ with respect to $\boldsymbol{w}$ [22], L is the Lipschitz constant; the fourth equality follows from the definition of the multiplier $\boldsymbol{\lambda}^{k+1}=\boldsymbol{\lambda}^k-\beta\left(\boldsymbol{y}^{k+1}-D\boldsymbol{x}^{k+1}\right)$; the fifth inequality follows from lemma 1. In order to ensure $C>0$, we require β satisfies $\beta^2a_1-L\beta-4a_2C_0>0$.

Then we know from two equalities above that

$$\mathcal{L}_\beta(\boldsymbol{y}^k,\boldsymbol{x}^k,\boldsymbol{w}^k,\boldsymbol{\lambda}^k)-\mathcal{L}_\beta(\boldsymbol{y}^{k+1},\boldsymbol{x}^{k+1},\boldsymbol{w}^{k+1},\boldsymbol{\lambda}^{k+1})$$
$$\geqslant C_1\left(\sum_{i=1}^{n^2}\|\mathbf{y}_i^k-\mathbf{y}_i^{k+1}\|_2^2+\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2^2\right)-\frac{2}{\beta}\|\eta_{\boldsymbol{x}}^k-\eta_{\boldsymbol{x}}^{k+1}\|_2^2. \tag{33}$$

This means for all sufficiently large k, $\mathcal{L}_\beta(\boldsymbol{y}^k,\boldsymbol{x}^k,\boldsymbol{w}^k,\boldsymbol{\lambda}^k)$ is nonincreasing. As $D_i\in\mathbb{R}^{2\times n^2}$ has full row rank, then there exists at one $\hat{\boldsymbol{x}}$ such that $D_i\hat{\boldsymbol{x}}=\mathbf{y}_i^k,\forall\ i=1,2,\ldots,n^2$. Thus, we arrive at

$$\mathcal{L}_\beta(\boldsymbol{y}^k,\boldsymbol{x}^k,\boldsymbol{w}^k,\boldsymbol{\lambda}^k)=\frac{\mu}{2}\|A(\boldsymbol{w}^k)\boldsymbol{x}^k-\boldsymbol{d}\|_2^2+\sum_{i=1}^{n^2}\|\mathbf{y}_i^k\|_2-\sum_{i=1}^{n^2}\left(\boldsymbol{\lambda}^k\right)_i^{\mathrm{T}}(\mathbf{y}_i^k-D_i\boldsymbol{x}^k)+\frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k-D_i\boldsymbol{x}^k\|_2^2$$

$$=\sum_{i=1}^{n^2}\|\mathbf{y}_i^k\|_2+\frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k-D_i\boldsymbol{x}^k\|_2^2+\frac{\mu}{2}\|A(\boldsymbol{w}^k)\boldsymbol{x}^k-\boldsymbol{d}\|_2^2-\sum_{i=1}^{n^2}\langle D_i^{\mathrm{T}}\left(\boldsymbol{\lambda}^k\right)_i,\hat{\boldsymbol{x}}-\boldsymbol{x}^k\rangle,$$

$$=\sum_{i=1}^{n^2}\|\mathbf{y}_i^k\|_2+\frac{\beta}{2}\sum_{i=1}^{n^2}\|\mathbf{y}_i^k-D_i\boldsymbol{x}^k\|_2^2+\frac{\mu}{2}\|A(\boldsymbol{w}^k)\hat{\boldsymbol{x}}^k-\boldsymbol{d}\|_2^2$$

$$=\sum_{i=1}^{n^2}\|\mathbf{y}_i^k\|_2+\frac{\mu}{2}\|A(\boldsymbol{w}^k)\hat{\boldsymbol{x}}^k-\boldsymbol{d}\|_2^2+\frac{\beta}{2}\sum_{i=1}^{n^2}\|D_i(\hat{\boldsymbol{x}}-\boldsymbol{x}^k)\|_2^2$$

$$\geqslant\sum_{i=1}^{n^2}\|\mathbf{y}_i^k\|_2+\frac{\mu}{2}\|A(\boldsymbol{w}^k)\hat{\boldsymbol{x}}^k-\boldsymbol{d}\|_2^2+\frac{\beta a_1}{2}\|\hat{\boldsymbol{x}}-\boldsymbol{x}^k\|_2^2>-\infty.$$

Since $\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)$ is lower bounded and $\sum_{i=1}^{n^2} \|\mathbf{y}_i^k\|_2 + \frac{\mu}{2}\|A(\boldsymbol{w}^k)\hat{\boldsymbol{x}}^k\|_2^2 + (\frac{1}{2}\beta a_1 - \frac{1}{2}\lambda_{\max}(A(\boldsymbol{w}^k)^{\mathrm{T}}A(\boldsymbol{w}^k)))\|\hat{\boldsymbol{x}} - \boldsymbol{x}^k\|_2^2$ is coercive over the feasible set $\Omega_{\mathrm{F}} := \{(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) : D_i\boldsymbol{x} = \mathbf{y}_i, i = 1, 2, \ldots, n^2\}$, we conclude that $\{\boldsymbol{y}^k\}$ and $\{(\boldsymbol{x}^k, \boldsymbol{w}^k)\}$ are bounded. Hence, by lemma 1, $\{\boldsymbol{\lambda}^k\}$ is bounded. □

Lemma 3. *Let* $\partial\mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}) = \left(\partial_{\boldsymbol{y}}\mathcal{L}_\beta, \nabla_{(\boldsymbol{x},\boldsymbol{w})}\mathcal{L}_\beta, \nabla_{\boldsymbol{\lambda}}\mathcal{L}_\beta\right)$. *Then there exists a constant* $\tilde{C} > 0$ *such that for all* $k \geqslant 1$, *for some* $p^{k+1} \in \partial\mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})$, *we have*

$$\|p^{k+1}\|_2 \leqslant \tilde{C}\left(\left\|\begin{bmatrix}\boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k\end{bmatrix}\right\|_2 + \|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2\right).$$

Proof. Because

$$\begin{aligned}\nabla_{\boldsymbol{\lambda}_i} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}) =& -(\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^{k+1}) \\ =& \frac{1}{\beta}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right),\end{aligned}$$

and

$$\nabla_{(\boldsymbol{x},\boldsymbol{w})} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}) = \begin{bmatrix}\sum_{i=1}^{n^2} D_i^{\mathrm{T}}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right) \\ \mathbf{0}\end{bmatrix},$$

we have

$$\begin{aligned}&\|\nabla_{\boldsymbol{\lambda}} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})\|_2 \\ &\quad= \frac{1}{\beta}\sum_{i=1}^{n^2}\|\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\|_2 \\ &\quad\leqslant \frac{1}{\beta\sqrt{a_1}}\sum_{i=1}^{n^2}\|D_i^{\mathrm{T}}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right)\|_2 \\ &\quad\leqslant \frac{1}{\beta\sqrt{a_1}}\left(C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k\end{bmatrix}\right\|_2 + \|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2\right)\end{aligned}$$

and

$$\|\nabla_{(\boldsymbol{x},\boldsymbol{w})} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})\|_2 \leqslant C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k\end{bmatrix}\right\|_2 + \|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2.$$

By the definition of the subgradient, we make use of the decomposition and obtain

$$\begin{aligned}\partial_{\boldsymbol{y}} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}) =& \partial\sum_{i=1}^{n^2}\left(\|\mathbf{y}_i^{k+1}\|_2 - \left(\boldsymbol{\lambda}^{k+1}\right)_i + \beta(\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^{k+1})\right) \\ =& \partial\sum_{i=1}^{n^2}\left(\|\mathbf{y}_i^{k+1}\|_2 - \left(\boldsymbol{\lambda}^k\right)_i + \beta(\mathbf{y}_i^{k+1} - D_i\boldsymbol{x}^k) - \left(\boldsymbol{\lambda}^{k+1}\right)_i\right. \\ &+ \left.\left(\boldsymbol{\lambda}^k\right)_i + \beta(D_i\boldsymbol{x}^k - D_i\boldsymbol{x}^{k+1})\right).\end{aligned}$$

Thus, according to the optimal condition

$$0 \in \partial \|\boldsymbol{y}_i^{k+1}\|_2 - \left(\boldsymbol{\lambda}^k\right)_i + \beta(D_i\boldsymbol{x}^k - D_i\boldsymbol{x}^{k+1}),$$

we have

$$-\sum_{i=1}^{n^2}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right) + \beta(D_i\boldsymbol{x}^k - D_i\boldsymbol{x}^{k+1}) \in \partial_{\boldsymbol{y}} L_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}).$$

Letting

$$p^{k+1} := (p_{\boldsymbol{y}}^{k+1},\ p_{(\boldsymbol{x},\boldsymbol{w})}^{k+1},\ p_{\boldsymbol{\lambda}}^{k+1}),$$

where

$$p_{\boldsymbol{y}}^{k+1} := -\sum_{i=1}^{n^2}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right) + \beta(D_i\boldsymbol{x}^k - D_i\boldsymbol{x}^{k+1}),$$

$$p_{(\boldsymbol{x},\boldsymbol{w})}^{k+1} := \sum_{i=1}^{n^2} D_i^{\mathrm{T}}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right),$$

$$p_{\boldsymbol{\lambda}}^{k+1} := \frac{1}{\beta}\sum_{i=1}^{n^2}\left(\left(\boldsymbol{\lambda}^{k+1}\right)_i - \left(\boldsymbol{\lambda}^k\right)_i\right).$$

Then we have

$$\begin{aligned}\|p^{k+1}\|_2 \leqslant & \frac{C_0}{\beta\sqrt{a_1}}\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2 + \frac{C_0}{\sqrt{a_1}}\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2 + C_0\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2 \\ & + \beta\sum_{i=1}^{n^2}\|D_i\|_2\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\|_2 + \left(\frac{C_0}{\beta\sqrt{a_1}} + \frac{C_0}{\sqrt{a_1}} + C_0\right)\|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2 \\ \leqslant & \tilde{C}\left(\left\|\begin{bmatrix}\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\\ \boldsymbol{w}^{k+1}-\boldsymbol{w}^k\end{bmatrix}\right\|_2 + \|\eta_{\boldsymbol{x}}^{k+1} - \eta_{\boldsymbol{x}}^k\|_2\right),\end{aligned}$$

where $\tilde{C} := C_0(1 + \frac{1}{\beta\sqrt{a_1}}\frac{1}{\sqrt{a_1}}) + \beta\sum_{i=1}^{n^2}\|D_i\|_2$ is a constant. □

Now we give the proof to theorem 1.

Proof of theorem 1. By lemma 2, the sequence $\{(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)\}$ is bounded, so there exists a convergent subsequence $\{(\boldsymbol{y}^{n_k}, \boldsymbol{x}^{n_k}, \boldsymbol{w}^{n_k}, \boldsymbol{\lambda}^{n_k})\}$, i.e., $\{(\boldsymbol{y}^{n_k}, \boldsymbol{x}^{n_k}, \boldsymbol{w}^{n_k}, \boldsymbol{\lambda}^{n_k})\}$ converges to $(\boldsymbol{y}^*, \boldsymbol{x}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ as $k \to \infty$. Notice that the residual is summable, then we know from $\|\eta_{\boldsymbol{x}}^k - \eta_{\boldsymbol{x}}^{k+1}\|_2 \leqslant \|\eta_{\boldsymbol{x}}^k\|_2 + \|\eta_{\boldsymbol{x}}^{k+1}\|_2$ that when $k \to \infty$, $\|\eta_{\boldsymbol{x}}^k - \eta_{\boldsymbol{x}}^{k+1}\|_2 \to 0$. Hence, for sufficiently large k, we have $L_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k)$ is nonincreasing and lower-bounded. We derive that when $k \to \infty$,

$$\mathcal{L}_\beta(\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k) - \mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1}) \to 0.$$

Then we know from the proof of lemma 2 that when $k \to \infty$,

$$\sum_{i=1}^{n^2} \|\mathbf{y}_i^k - \mathbf{y}_i^{k+1}\|_2^2 \to 0,$$

$$\left\| \begin{bmatrix} \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \\ \boldsymbol{w}^{k+1} - \boldsymbol{w}^k \end{bmatrix} \right\|_2^2 \to 0.$$

Then we know from lemma 3 that there exists $p^{k+1} \in \partial\mathcal{L}_\beta(\boldsymbol{y}^{k+1}, \boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{\lambda}^{k+1})$ such that when $k \to \infty$, $\|p^{k+1}\|_2 \to 0$. This leads to $\|p^{n_k}\|_2 \to 0$ as $k \to \infty$. Based on the definition of the general subgradient in [18], we obtain that $0 \in \partial\mathcal{L}_\beta(\boldsymbol{y}^*, \boldsymbol{x}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}^*)$, i.e., $(\boldsymbol{y}^*, \boldsymbol{x}^*, \boldsymbol{w}^*, \boldsymbol{\lambda}^*)$ is a stationary point. □

3.4. Complexity analysis

Now we consider the computational complexity of algorithm 1 for (7). In step 1, the main computational costs come from computing $D_i\boldsymbol{x}^k$ and $\frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i$. Since D_i denotes the first-order finite difference of $\boldsymbol{x}$ at the ith pixel, computing $D_i\boldsymbol{x}^k, i = 1, \ldots, n^2$ requires $2n^2$ flops. Computing the scalar-vector product $\frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i, i = 1, \ldots, n^2$ requires $2n^2$ flops. Finally, computing the two-norm of the vectors $D_i\boldsymbol{x}^k + \frac{1}{\beta}\left(\boldsymbol{\lambda}^k\right)_i \in \mathbb{R}^2, i = 1, \ldots, n^2$ needs $6n^2$ flops. Thus, step 1 costs $O\left(n^2\right)$.

In step 2, we first consider the computational cost of (21). Suppose we use the conjugate gradient method to solve for $\delta\mathbf{x}_{\mathcal{I}}$ and set the maximum iteration number to a small constant. Then the cost is determined by the cost of multiplying $\hat{\mathbf{J}}_{\boldsymbol{x}}^\top(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^\top)\hat{\mathbf{J}}_{\boldsymbol{x}} + \frac{1}{\mu}\nabla_{\boldsymbol{x}}^2\hat{R}\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)$ on a vector and computing $-\hat{\mathbf{J}}_{\boldsymbol{x}}^\top(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^\top)\boldsymbol{r}$ and $\hat{\mathbf{J}}_{\boldsymbol{x}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\nabla_{\boldsymbol{w}}\hat{S}(\boldsymbol{w})$. Since $\mathbf{J}_{\boldsymbol{x}} = A(\boldsymbol{w})^\top$ and each PSF matrix A_i allows the use of fast Fourier transforms (FFTs) [23, 24] to multiply it with a vector, the cost of multiplying $\hat{\mathbf{J}}_{\boldsymbol{x}}$ on a vector is $O\left(n^2 \log\ n\right)$. Similarly, the cost of multiplying $\hat{\mathbf{J}}_{\boldsymbol{w}}$ on a vector is also $O\left(n^2 \log\ n\right)$ by FFTs. So the cost of the matrix-vector product of $\hat{\mathbf{J}}_{\boldsymbol{x}}^\top(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^\top)\hat{\mathbf{J}}_{\boldsymbol{x}}$ and a vector is $O\left(n^2 \log\ n\right)$. Computing the matrix-vector product of $\nabla_{\boldsymbol{x}}^2\hat{R}\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)$ and a vector equals to computing the matrix-vector product of $\beta D^{\mathrm{T}}D$ and a vector, and it can be done in $O\left(n^2\right)$. Moreover, $-\hat{\mathbf{J}}_{\boldsymbol{x}}^\top(\boldsymbol{I} - \hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\hat{\mathbf{J}}_{\boldsymbol{w}}^\top)\boldsymbol{r}$ and $\hat{\mathbf{J}}_{\boldsymbol{x}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}}(\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{w}} + \nabla_{\boldsymbol{w}}^2\hat{S}(\boldsymbol{w}))^{-1}\nabla_{\boldsymbol{w}}\hat{S}(\boldsymbol{w})$ can be performed in $O\left(n^2 \log\ n\right)$ by FFTs. Thus the computational cost of (21) is $O\left(n^2 \log\ n\right)$. Next we consider the cost of (22). The main computational costs in (22) are from computing the matrix-vector product $\hat{\mathbf{J}}_{\boldsymbol{w}}^\top\hat{\mathbf{J}}_{\boldsymbol{x}}\delta\boldsymbol{x}_{\mathcal{I}}$ and $\hat{\mathbf{J}}_{\boldsymbol{w}}\boldsymbol{r}$. Since these matrix-vector products can also be accelerated by FFTs, the computational cost of (22) is $O\left(n^2 \log\ n\right)$. For (23), the main computational costs are from computing $\tilde{\mathbf{J}}_{\boldsymbol{x}}^\top\boldsymbol{r}$, $\tilde{\mathbf{J}}_{\boldsymbol{w}}^\top\boldsymbol{r}$. Similar to the analysis for (22), the costs of computing $\tilde{\mathbf{J}}_{\boldsymbol{x}}^\top\boldsymbol{r}$ and $\tilde{\mathbf{J}}_{\boldsymbol{w}}^\top\boldsymbol{r}$ are $O\left(n^2 \log\ n\right)$ due to FFTs. The scalar-vector product $-\mu\begin{bmatrix} \tilde{\mathbf{J}}_{\boldsymbol{x}}^\top\boldsymbol{r} \\ \tilde{\mathbf{J}}_{\boldsymbol{w}}^\top\boldsymbol{r} \end{bmatrix}$ can be done in $n^2 + p$ flops. In applications, we usually choose the image size to be $n \times n = 256 \times 256$ and the number of the unknown weights parameterizing the blurring matrix is usually chosen to be $p = 2, 3$ or some small values much smaller than n^2, so we always have $p \ll n^2$. Thus, the cost of (23) is $O\left(n^2 \log\ n\right)$. In (24), computing the scalar-vector multiplication $\gamma\begin{bmatrix} \delta\boldsymbol{x}_{\mathcal{A}} \\ \delta\boldsymbol{w}_{\mathcal{A}} \end{bmatrix}$ requires $n^2 + p$ flops and computing the summation of the vectors $\begin{bmatrix} \delta\boldsymbol{x}_{\mathcal{I}} \\ \delta\boldsymbol{w}_{\mathcal{I}} \end{bmatrix}$ and $\gamma\begin{bmatrix} \delta\boldsymbol{x}_{\mathcal{A}} \\ \delta\boldsymbol{w}_{\mathcal{A}} \end{bmatrix}$ also requires $n^2 + p$ flops. Hence, the cost of (24) is $O(n^2)$. The cost of (25) is mainly from forming $\nabla_{x,w}\hat{\Phi}\left(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)$ and computing

the inner product $\left(\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x},\boldsymbol{w},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)\right)^{\top}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{w}\end{bmatrix}$. We know that $\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x},\boldsymbol{w},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)=\begin{bmatrix}A(\boldsymbol{w})^{\mathrm{T}}(A(\boldsymbol{w})\boldsymbol{x}-\boldsymbol{d})\\ \widehat{J}_{\boldsymbol{w}}^{\mathrm{T}}(A(\boldsymbol{w})\boldsymbol{x}-\boldsymbol{d})\end{bmatrix}$, so the cost of forming $\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x},\boldsymbol{w},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)$ is $O\left(n^2\log\ n\right)$ by FFTs, and computing the inner product $\left(\nabla_{\boldsymbol{x},\boldsymbol{w}}\hat{\Phi}\left(\boldsymbol{x},\boldsymbol{w},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)\right)^{\top}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{w}\end{bmatrix}$ costs $O\left(n^2\right)$. Thus the computational cost of step 2 is $O\left(n^2\log\ n\right)$.

In step 4, the computational cost is dominated by the cost of computing $D\boldsymbol{x}^{k+1}$ and forming $\beta\boldsymbol{y}^{k+1}$ and $\beta D\boldsymbol{x}^{k+1}$. It is easy to see that these operations cost $O\left(n^2\right)$. Therefore, the total computational complexity of algorithm 1 is $O\left(n^2\log\ n\right)$.

3.5. Comparison method

In this section, we briefly discuss a benchmark method ADMM-BCD of ADMM-LAP used in the numerical comparisons in section 4.

BCD in ADMM-BCD stands for the BCD method [15, 25], which is another popular approach for solving coupled optimization problems. The main idea of BCD is partitioning the optimization variables into a number of blocks, then minimize the objective function cyclically over each block while fixing the remaining blocks at their last updated values until convergence. For AO retinal image problems we consider in this paper, the variables can be naturally separated into two subsets, one for the image variable $\boldsymbol{x}$ and another for the parameters $\boldsymbol{w}$. For the tightly coupled $(\boldsymbol{x},\boldsymbol{w})$-subproblem

$$\min_{\boldsymbol{x}\in\mathcal{C}_{\boldsymbol{x}},\boldsymbol{w}\in\mathcal{C}_{\boldsymbol{w}}}\frac{\mu}{2}\|A\left(\boldsymbol{w}\right)\boldsymbol{x}-\boldsymbol{d}\|_2^2+S(\boldsymbol{w})+R\left(\boldsymbol{x},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right),$$

given the initial point $\left(\boldsymbol{x}^{k},\boldsymbol{w}^{k}\right)$, the iterative format of BCD is as follows

$$\boldsymbol{x}^{(l+1)}=\operatorname*{argmin}_{\boldsymbol{x}\in\mathcal{C}_{\boldsymbol{x}}}\frac{\mu}{2}\|A\left(\boldsymbol{w}^{(l)}\right)\boldsymbol{x}-\boldsymbol{d}\|_2^2+R\left(\boldsymbol{x},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right),\tag{34}$$

$$\boldsymbol{w}^{(l+1)}=\operatorname*{argmin}_{\boldsymbol{w}\in\mathcal{C}_{\boldsymbol{w}}}\frac{\mu}{2}\|A\left(\boldsymbol{w}\right)\boldsymbol{x}^{(l+1)}-\boldsymbol{d}\|_2^2+S(\boldsymbol{w}),\tag{35}$$

where $l\in\mathbb{N}$ denotes the lth iteration of BCD.

For the numerical experiments, we inexactly solve (34) and (35) by the projected Gauss–Newton method with bound constraints [8]. The search directions $\delta\boldsymbol{x}_{\mathcal{I}}$ at $\boldsymbol{x}^{(l)}$ can be computed as follows:

$$\begin{aligned}&\left(\mu\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{x}}+\nabla_{\boldsymbol{x}}^2\hat{R}\left(\boldsymbol{x}^{(l)},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)\right)\delta\boldsymbol{x}_{\mathcal{I}}\\&\quad=-\left(\mu\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top}\boldsymbol{r}(\boldsymbol{x}^{(l)},\boldsymbol{w}^{(l)})+\nabla_{\boldsymbol{x}}\hat{R}\left(\boldsymbol{x}^{(l)},\boldsymbol{y}^{k+1},\boldsymbol{\lambda}^{k}\right)\right),\end{aligned}\tag{36}$$

where $\hat{\mathbf{J}}_{\boldsymbol{x}}$, $\nabla_{\boldsymbol{x}}\hat{R}$ and $\nabla_{\boldsymbol{x}}^2\hat{R}$ represent $\mathbf{J}_{\boldsymbol{x}}$, $\nabla_{\boldsymbol{x}}R$ and $\nabla_{\boldsymbol{x}}^2R$ restricted to the inactive set via projection, respectively. Moreover, $\delta\boldsymbol{x}_{\mathcal{A}}$ is given by the projected gradient descent step and $\delta\boldsymbol{x}$ can be computed by a scaled combination [8]. Finally, the projected Armijo line search is applied to find the solution $\boldsymbol{x}^{(l+1)}$. Then the search directions $\delta\boldsymbol{w}_{\mathcal{I}}$ at $\boldsymbol{w}^{(l)}$ can be computed as follows:

$$\left(\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\hat{\mathbf{J}}_{\boldsymbol{w}}+\nabla_{\boldsymbol{w}}^2S(\boldsymbol{w})\right)\delta\boldsymbol{w}_{\mathcal{I}}=-\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}\boldsymbol{r}(\boldsymbol{x}^{(l+1)},\boldsymbol{w}^{(l)})+\nabla_{\boldsymbol{w}}S(\boldsymbol{w}),\tag{37}$$

Algorithm 2. ADMM-BCD method for (7).

Input: $\left(\boldsymbol{y}^0, \boldsymbol{x}^0, \boldsymbol{w}^0, \boldsymbol{\lambda}^0\right) \in \mathbb{R}^{2n^2} \times \mathbb{R}^{n^2} \times \mathbb{R}^p \times \mathbb{R}^{2n^2}$, the penalty parameter $\beta > 0$. Set $k = 0$.
Output: $\boldsymbol{y}^k, \boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{\lambda}^k$.
Step 1 Compute $\boldsymbol{y}^{k+1}$ using (9).
Step 2 Compute $\boldsymbol{x}^{k+1}$, $\boldsymbol{w}^{k+1}$ by iteratively solving (34) and (35).
Step 3 If a termination criteria is met, stop and go to step 4; else, repeat step 2.
Step 4 Compute $\boldsymbol{\lambda}^{k+1}$ using (11).
Step 5 If a termination criteria is met, stop; else, set $k \coloneqq k + 1$ and go to step 1.

where $\hat{\mathbf{J}}_w$, $\nabla_w \hat{S}(\boldsymbol{w})$ and $\nabla_w^2 \hat{S}(\boldsymbol{w})$ represent $\mathbf{J}_w$, $\nabla_w S(\boldsymbol{w})$ and $\nabla_w^2 S(\boldsymbol{w})$ restricted to the inactive set via projection, respectively. Similar to the above discussion, we can obtain $\delta \boldsymbol{w}_{\mathcal{A}}$ by projected gradient descent and obtain $\delta \boldsymbol{w}$ by a scaled combination. Finally, the solution $\boldsymbol{w}^{(l+1)}$ can be obtained by the projected Armijo line search.

For (36), due to the large dimension, the search direction can be inexactly computed by the conjugate gradients method. For (37), because the dimension is small, the search direction can be computed by a direct solver. Moreover, we note that BCD is fully decoupled while optimizing over one set of variables, and the optimization over another set of variables is neglected. This degrades the convergence for tightly coupled problems [15].

Similar to ADMM-LAP discussed in the previous section, ADMM-BCD uses ADMM as the outer optimization method to tackle the TV regularization and applies BCD to tackle the related $(\boldsymbol{x}, \boldsymbol{w})$-subproblems appearing in each ADMM iteration. The main operations of ADMM-BCD are summarized in algorithm 2.

The complexity analysis of algorithm 2 is presented as follows. First notice that step 1 and step 4 in algorithm 2 are identical to those in algorithm 1, thus these two steps cost $O\left(n^2\right)$.

For step 2 in algorithm 2, we first analyze the computational cost of (34). To compute $\delta \boldsymbol{x}_{\mathcal{I}}$, we use the conjugate gradient method and set the maximum iteration number to a small constant. The main computational cost is from computing the multiplication of $\mu \hat{\mathbf{J}}_{\boldsymbol{x}}^{\top} \hat{\mathbf{J}}_{\boldsymbol{x}} + \nabla_{\boldsymbol{x}}^2 \hat{R}\left(\boldsymbol{x}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^k\right)$ and a vector, and computing $\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l)}, \boldsymbol{w}^{(l)})$. Similar to the complexity analysis in algorithm 1, this matrix-vector product can be done in $O\left(n^2 \log\ n\right)$ by FFTs. Then $\delta \boldsymbol{x}_{\mathcal{A}}$, $\delta \boldsymbol{x}$ can be computed in a similar way as discussed in algorithm 1and a projected Armijo line search is applied to find the solution. The cost of these steps is $O\left(n^2 \log\ n\right)$. Hence, the total computational cost of (34) is $O\left(n^2 \log\ n\right)$. As for the computational cost of (35), the main computational costs of computing $\delta \boldsymbol{w}_{\mathcal{I}}$ are from computing $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \hat{\mathbf{J}}_{\boldsymbol{w}}$ and $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l+1)}, \boldsymbol{w}^{(l)})$. We know from the definition (13) that $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top}$ is a $p \times n^2$ matrix, $p \ll n^2$, hence the cost of computing $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \hat{\mathbf{J}}_{\boldsymbol{w}}$ is $O\left(n^2\right)$. The matrix-vector multiplication $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l+1)}, \boldsymbol{w}^{(l)})$ can be done in $O\left(n^2 \log\ n\right)$ by FFTs. Moreover, the cost of computing $\delta \boldsymbol{w}_{\mathcal{A}}$, $\delta \boldsymbol{w}$ and applying the projected Armijo line search is $O\left(n^2 \log\ n\right)$. Therefore, the cost of (35) is $O\left(n^2 \log\ n\right)$. As a result, the total computational cost of algorithm 2 is $O\left(n^2 \log\ n\right)$.

As shown from numerical experiments from section 4, algorithm 2 takes longer time than that of algorithm 1. This is mainly because BCD requires computing the matrix-vector product $\hat{\mathbf{J}}_{\boldsymbol{x}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l)}, \boldsymbol{w}^{(l)})$ and $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l+1)}, \boldsymbol{w}^{(l)})$ in each iteration, while LAP only requires computing $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \boldsymbol{r}(\boldsymbol{x}^{(l)}, \boldsymbol{w}^{(l)})$. Hence, BCD takes more computational costs to compute the current residual value $\boldsymbol{r}$ and the matrix-vector multiplication $\hat{\mathbf{J}}_{\boldsymbol{w}}^{\top} \boldsymbol{r}$, which results in a larger number of FFTs. Moreover, the line search is applied twice in BCD to solve the search directions for both $\boldsymbol{x}$ and $\boldsymbol{w}$, while in LAP, only one line search is enough to obtain the search directions for both variables. Hence, algorithm 2 costs more. Moreover, it is important to point out that algorithm 1 takes the structure of the tightly coupled problem into consideration and converges faster.

4. Numerical experiments

In this section, we illustrate the numerical performance of the proposed ADMM-LAP method for the myopic deconvolution problems with TV regularization arising from the AO retinal image restoration. All our computational results are obtained by MATLAB R2018b running on a Macbook Air with Intel Core i5 CPU (1.4 GHz).

The following notations will be used throughout the section:

- BlurLevel: an indicator used to set the severity of the blur to one of the following: 'mild', 'medium' and 'severe';
- NoiseLevel $\triangleq \|\boldsymbol{e}\|_2/\|\boldsymbol{d}^*\|_2$: relative level of noise, where $\boldsymbol{e}$ denotes the noise vector of perturbations and $\boldsymbol{d}^*$ denotes the exact (noise free) data vector;
- #iter: the number of iterations performed by the algorithm;
- Rel. Err. $\boldsymbol{x}$: the relative error $\|\boldsymbol{x}-\boldsymbol{x}^*\|_2 \,/\, \|\boldsymbol{x}^*\|_2$, where $\boldsymbol{x}^*$ and $\boldsymbol{x}$ denote the true image and computed image, respectively;
- Rel. Err. $\boldsymbol{w}$: the relative error $\|\boldsymbol{w}-\boldsymbol{w}^*\|_2 \,/\, \|\boldsymbol{w}^*\|_2$, where $\boldsymbol{w}^*$ and $\boldsymbol{w}$ denote the true parameters and computed parameters, respectively;
- SNR $(\boldsymbol{x}) \triangleq 10 \cdot \log_{10}\frac{\|\overline{\boldsymbol{x}}-\tilde{\boldsymbol{x}}\|_2^2}{\|\overline{\boldsymbol{x}}-\boldsymbol{x}\|_2^2}$: signal-to-noise ratio (SNR), where $\overline{\boldsymbol{x}}$ is the original image, $\tilde{\boldsymbol{x}}$ is the mean intensity value of $\overline{\boldsymbol{x}}$ and $\boldsymbol{x}$ denotes the restored image. SNR $(\boldsymbol{x})$ measures the quality of restoration.

In all test problems, we set the regularization parameter $\mu = 5 \times 10^4$ by trial and errors such that the restored images had reasonable SNR and relative errors. The parameter ξ was set to $\xi = 100$ such that the obtained parameters satisfied $\sum_{j=1}^{p} w_j = 1$. The parameter β was chosen according to the assumptions in lemma 2. In order to guarantee the sequence $\{\epsilon_{k+1}\}_{k=0}^{\infty} \subseteq [0, +\infty)$ satisfies $\sum_{k=0}^{\infty} \epsilon_{k+1} < \infty$, we choose $\epsilon_{k+1} = \frac{1}{a(k+1)^2}$, where a is a constant. We used one uniform stopping criterion, that is

$$\frac{|\hat{\Phi}(\boldsymbol{x}^{k+1}, \boldsymbol{w}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{\lambda}^{k+1}) - \hat{\Phi}(\boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k)|}{|\hat{\Phi}(\boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k)|} < \epsilon,$$

where $\hat{\Phi}(\boldsymbol{x}^k, \boldsymbol{w}^k, \boldsymbol{y}^k, \boldsymbol{\lambda}^k)$ is the objective function value in the kth iteration and $\epsilon > 0$ is a given tolerance. In this paper, we set $\epsilon = 10^{-2}$. The maximum number of ADMM iterations was set to 50. We also report the numerical results obtained by running the ADMM-BCD method. We compare the relative error of the image $\boldsymbol{x}$ and the relative error of the parameter $\boldsymbol{w}$ estimated by both ADMM-LAP and ADMM-BCD as well as their computational time and SNR of the restored images.

Example 4.1. AO flood illumination retinal imaging is a popular technique which has been in use for more than a decade [2, 4]. AO retinal imaging technique gives us the opportunity to observe and study retinal structures at the cellular level, which is impossible to see directly in the living eye. Due to the poor contrast of the raw AO retinal images, interpretation of such images requires the myopic deconvolution. For this example, we consider the case $p = 2$. The global PSF used in the simulation is the sum of two PSFs with the first one being focused and the second one being defocused. Let A_1, A_2 denote the relative blurring marices. The test problem is a simulation generated from a 256×256 pixel portion of an actual AO retinal image.

We use the regularization toolbox IR Tools [7] to build up the test problem. *PRblurgauss* and *PRblurdefocus* are functions used to simulate spatially invariant Gaussian blur and spatially

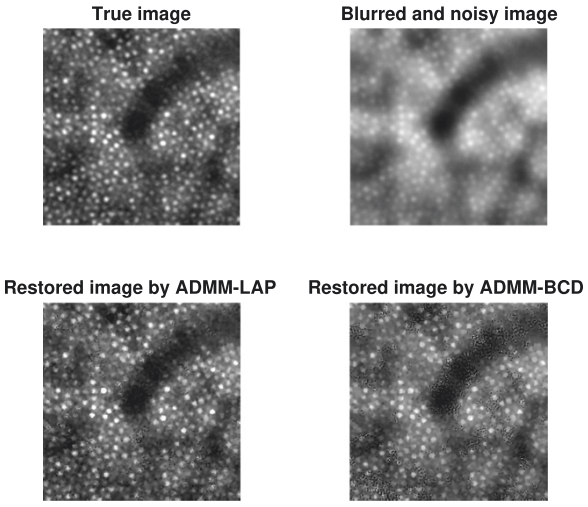

Figure 1. Restored images for the retinal image in example 4.1. The BlurLevel of *PRblurdefocus* is set to be 'medium'. The images from the top left to the bottom right show the true image, the blurred and noisy image, the restored image by ADMM-LAP and the restored image by ADMM-BCD.

invariant out-of-focus blurs, respectively. Specifically, *PRblurgauss* constructs a 256×256 PSF array P with entries

$$p_{ij} = \frac{1}{2\pi\sigma} \exp\left(-\frac{1}{2}\frac{(i-k)^2}{\sigma^2} + \frac{(j-\ell)^2}{\sigma^2}\right)$$

for a given value of σ, and *PRblurdefocus* constructs a 256×256 PSF array P with entries

$$p_{ij} = \begin{cases} 1/(\pi r^2), & \text{if } (i-k)^2 + (j-\ell)^2 \leqslant r^2, \\ 0, & \text{elsewhere} \end{cases}$$

for a given value of r.

In this example, three cases are considered. We fix one of the PSFs to be built up by *PRblurgauss* with 'mild' BlurLevel ($\sigma = 2$) and set the other as a combined PSF built up by *PRblurgauss* with 'mild' BlurLevel and *PRblurdefocus* with three different BlurLevels (i.e., 'mild' ($r = 7$), 'medium' ($r = 15$), 'severe' ($r = 31$)). In addition, function *PRnoise* is used to add Gaussian noise with NoiseLevel $= 0.01$. We set the true parameter $\boldsymbol{w}^* = [0.3; 0.7]$, choose the random image with the same size of the true image as the initial guess $\boldsymbol{x}^0$ and the initial guess $\boldsymbol{w}^0 = [0.5; 0.5]$.

We then test ADMM-LAP and ADMM-BCD for the two-weight case of (2) on this image. For the case with the BlurLevel of *PRblurdefocus* being set to be 'medium', the restored images by both methods are shown in figure 1. We can clearly see that the restored image obtained by ADMM-LAP is much sharper, hence has a much better contrast than the one obtained by ADMM-BCD. Moreover, the restored image is much closer to the true image.

The plots of the relative errors of the restored image $\boldsymbol{x}$ and the estimated parameters $\boldsymbol{w}$ against iteration can be seen in figure 2. It is easy to see that ADMM-LAP can reach lower

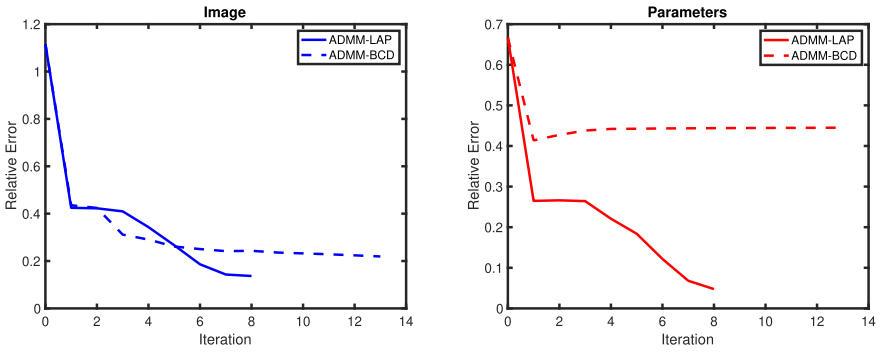

Figure 2. Relative errors of the restored image $\boldsymbol{x}$ (left) and the parameters $\boldsymbol{w}$ (right) for the myopic deconvolution problem in example 4.1 with 'medium' BlurLevel of *PRblurdefocus*.

Table 1. Numerical results of ADMM-LAP and ADMM-BCD for the AO retinal image in example 4.1. The columns from left to right give the BlurLevel, the method name, the stopping iteration, the relative error of the restored image, the relative error of the solution parameters, the computational time in seconds and SNR of the restored images.

BlurLevel	Method	#iter	Rel. Err. $\boldsymbol{x}$	Rel. Err. $\boldsymbol{w}$	Time/s	SNR($\boldsymbol{x}$)
'Mild'	ADMM-LAP	6	$\mathbf{1.48 \times 10^{-1}}$	$\mathbf{2.63 \times 10^{-2}}$	**50.23**	**9.57**
	ADMM-BCD	11	1.88×10^{-1}	1.11×10^{-1}	154.53	7.48
'Medium'	ADMM-LAP	8	$\mathbf{1.37 \times 10^{-1}}$	$\mathbf{4.76 \times 10^{-2}}$	**56.42**	**10.26**
	ADMM-BCD	13	2.19×10^{-1}	4.45×10^{-1}	145.09	6.16
'Severe'	ADMM-LAP	12	$\mathbf{1.17 \times 10^{-1}}$	$\mathbf{4.39 \times 10^{-2}}$	**69.10**	**11.60**
	ADMM-BCD	12	3.04×10^{-1}	9.13×10^{-1}	122.22	3.32

relative errors than ADMM-BCD for both the restored image and the obtained parameters in this test.

In table 1, we report the relative error of the restored image $\boldsymbol{x}$, the relative error of the parameters $\boldsymbol{w}$, the computational time and SNR of the restored images for both methods for all three BlurLevels. As can be seen from the fourth and fifth columns of table 1, ADMM-LAP can obtain more accurate restored images and more accurate parameters than ADMM-BCD on all three cases. The sixth column shows that ADMM-LAP is faster than ADMM-BCD, this is mainly because BCD takes a larger number of FFTs to compute the current residual value $\boldsymbol{r}$ and the matrix-vector multiplication $\hat{\mathbf{J}}_w^\top \boldsymbol{r}$, and the line search is applied twice to solve the search directions for $\boldsymbol{x}$ and $\boldsymbol{w}$. Moreover, we can see from the seventh column that the quality of restored images obtained by ADMM-LAP is much better than those obtained by ADMM-BCD. We point out that it is very difficult to analyze the difference of the relative errors of $\boldsymbol{x}$ and $\boldsymbol{w}$ between the test problems, especially because of the nonlinear relationship between $\boldsymbol{x}$ and $\boldsymbol{w}$. We note that even for a linear problem, the quality of the error in $\boldsymbol{x}$ will depend not only on how ill-conditioned the matrix is, but on the distribution of singular values (e.g., is there a well-defined gap between large and small singular values, or do they decay smoothly). Our main purpose for this table of results is to compare the two methods (ADMM-LAP and

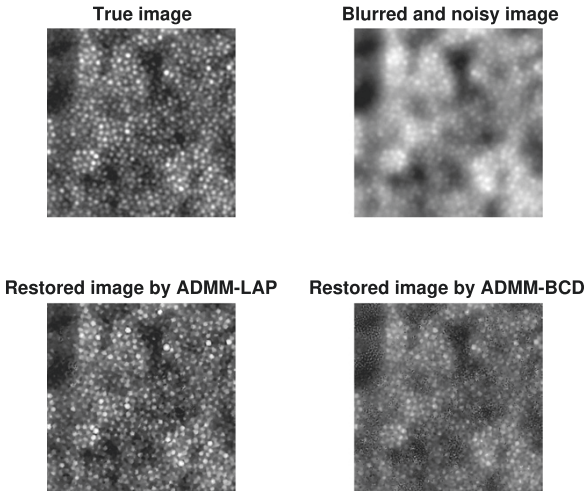

Figure 3. Restored images for the retinal image in example 4.2. The BlurLevel of *PRblurdefocus* is set to be 'medium'. The images from the top left to the bottom right show the true image, the blurred and noisy image, the restored image by ADMM-LAP and the restored image by ADMM-BCD.

ADMM-BCD) for each test problem, and to see that ADMM-LAP consistently outperforms ADMM-BCD even if the test problems change.

Example 4.2. For this example, we consider the test problem from another simulation using a different 256×256 portion of an actual AO retinal image with two blurring matrices. The size of the image is 256×256. We use *PRblurgauss* with 'mild' BlurLevel, *PRblurdefocus* with three different BlurLevels (i.e., 'mild', 'medium', 'severe') in IR Tools to build up a combined PSF, the other PSF is built up by *PRblurgauss* with 'mild' BlurLevel only. In addition, function *PRnoise* is used to add Gaussian noise with NoiseLevel $= 0.01$. We set the true parameter $\boldsymbol{w}^* = [0.3; 0.7]$, choose the random image with the same size of the true image as the initial guess $\boldsymbol{x}^0$ and set the initial guess $\boldsymbol{w}^0 = [w_1^0; w_2^0] \in \mathbb{R}^2$, where w_1^0, w_2^0 are random constants from $[0, 1]$ satisfying $w_1^0 + w_2^0 = 1$.

For the case with the BlurLevel of *PRblurdefocus* being set to be 'medium', the restored images by both ADMM-LAP and ADMM-BCD method are shown in figure 3. Like example 4.1, it is easy to see that the restored image obtained by ADMM-LAP is much sharper, and is much closer to the true image than the one obtained by ADMM-BCD.

We plot the relative errors of the restored image $\boldsymbol{x}$ and the estimated parameters $\boldsymbol{w}$ against iteration in figure 4. We can also see that ADMM-LAP reaches lower relative errors of both the restored image and the obtained parameters, hence it tends to recover more accurate restored images and parameters.

Table 2 shows the relative error of the restored image $\boldsymbol{x}$, the relative error of the parameters $\boldsymbol{w}$, the computational time and SNR of the restored images associated with three cases for both methods. Similar to example 4.1, ADMM-LAP still outperforms ADMM-BCD in terms of the quality of the restored images and the accuracy of the obtained parameters by large margin. It can be clearly seen from the sixth column of table 2 that ADMM-LAP is faster than ADMM-BCD. This is mainly because ADMM-BCD takes a larger number of FFTs and the line search

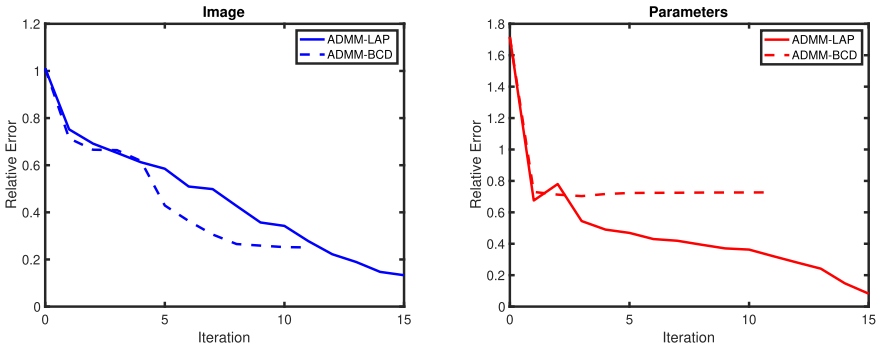

Figure 4. Relative errors for both the restored image $\boldsymbol{x}$ (left) and the parameters $\boldsymbol{w}$ (right) for the myopic deconvolution problem in example 4.2 with 'medium' BlurLevel of *PRblurdefocus*.

Table 2. Numerical results of ADMM-LAP and ADMM-BCD for the retinal image in example 4.2. The columns from left to right give the BlurLevel, the method name, the stopping iteration, the relative error of the restored image, the relative error of the solution parameters, the computational time in seconds and SNR of the restored images.

BlurLevel	Method	#iter	Rel. Err. $\boldsymbol{x}$	Rel. Err. $\boldsymbol{w}$	Time/s	SNR($\boldsymbol{x}$)
'Mild'	ADMM-LAP	11	$\mathbf{1.90 \times 10^{-1}}$	$\mathbf{1.01 \times 10^{-1}}$	**64.76**	**6.52**
	ADMM-BCD	14	2.19×10^{-1}	2.74×10^{-1}	152.13	5.31
'Medium'	ADMM-LAP	15	$\mathbf{1.33 \times 10^{-1}}$	$\mathbf{8.23 \times 10^{-2}}$	**88.12**	**9.63**
	ADMM-BCD	11	2.51×10^{-1}	7.27×10^{-1}	119.20	4.12
'Severe'	ADMM-LAP	5	$\mathbf{1.62 \times 10^{-1}}$	7.77×10^{-1}	**40.99**	**7.97**
	ADMM-BCD	5	2.83×10^{-1}	$\mathbf{3.76 \times 10^{-1}}$	76.84	3.08

is applied twice to compute the search directions, which is its computational bottleneck. The seventh column clearly shows that ADMM-LAP obtains restored images with a much better quality. Above all, we can see that ADMM-LAP is much more efficient and accurate for this test example.

Example 4.3. In this example, we consider the case where three blurring matrices are provided, i.e., $p = 3$. The test problem is generated from the same AO retinal image used in example 4.1. The size of the image is 256×256. The first PSF is built up by *PRblurgauss* with 'mild' BlurLevel. The second PSF is a combined PSF built up by *PRblurgauss* with 'mild' BlurLevel and *PRblurdefocus* with 'medium' BlurLevel. The third PSF is a combined PSF built up by *PRblurgauss* with 'mild' BlurLevel and *PRblurdefocus* with two different BlurLevels (i.e., 'mild' and 'severe'). In addition, function *PRnoise* is used to add Gaussian noise with noiselevel $= 0.01$. The true parameter vector $\boldsymbol{w}^* = [w_1^*; w_2^*; w_3^*] \in \mathbb{R}^3$ is set to be a random vector that satisfies $w_i^* \geqslant 0$ and $\sum_{i=1}^3 w_i^* = 1$. We choose the random image with the same size of the true image as the initial guess $\boldsymbol{x}^0$ and set the initial guess $\boldsymbol{w}^0 = [\frac{1}{3}; \frac{1}{3}; \frac{1}{3}]$.

The restored images by both ADMM-LAP and ADMM-BCD method are shown in figure 5. Like examples 4.1 and 4.2, we can also see that ADMM-LAP reaches a much sharper image,

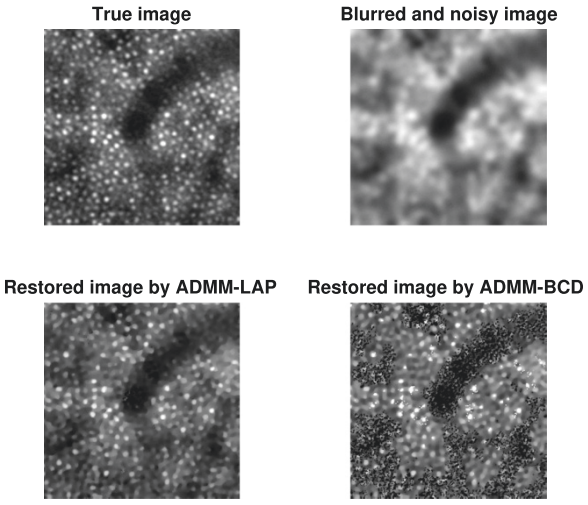

Figure 5. Restored images for the retinal image in example 4.3. The BlurLevel of *PRblurdefocus* in the third PSF is set to be 'mild'. The images from the top left to the bottom right show the true image, the blurred and noisy image, the restored image by ADMM-LAP and the restored image by ADMM-BCD.

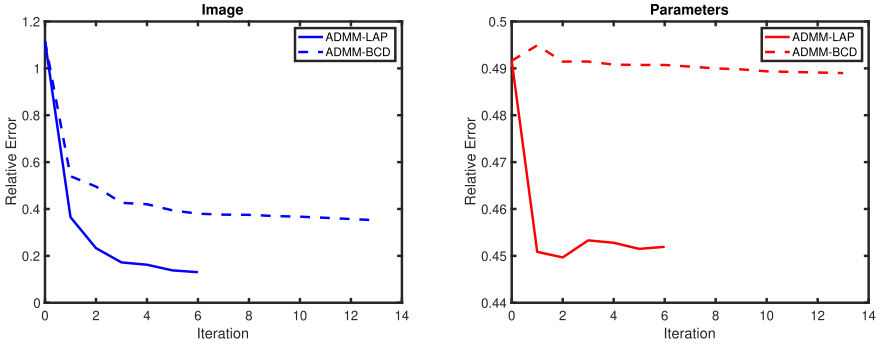

Figure 6. Relative errors of the restored image $\boldsymbol{x}$ (left) and the parameters $\boldsymbol{w}$ (right) for the myopic deconvolution problem in example 4.3 with 'mild' BlurLevel of *PRblurdefocus* in the third PSF.

it tends to recover an image that is much closer to the true image than the one obtained by ADMM-BCD.

We plot the relative errors of the restored image $\boldsymbol{x}$ and the estimated parameters $\boldsymbol{w}$ against iteration in figure 6. We can also see that ADMM-LAP achieve lower relative errors of both the restored image and the obtained parameters.

In table 3, we present the relative error of the restored image $\boldsymbol{x}$, the relative error of the parameters $\boldsymbol{w}$, the computational time and SNR of the restored images associated with two cases for both methods. We can also see from the sixth column that ADMM-LAP has evident

Table 3. Numerical results of ADMM-LAP and ADMM-BCD for the retinal image in example 4.3. The columns from left to right give the BlurLevel, the method name, the stopping iteration, the relative error of the restored image, the relative error of the solution parameters, the computational time in seconds and SNR of the restored images.

BlurLevel	Method	#iter	Rel. Err. $\boldsymbol{x}$	Rel. Err. $\boldsymbol{w}$	Time/s	SNR($\boldsymbol{x}$)
'Mild'	ADMM-LAP	6	$\mathbf{1.30 \times 10^{-1}}$	$\mathbf{4.52 \times 10^{-1}}$	**134.03**	**10.68**
	ADMM-BCD	13	3.52×10^{-1}	4.89×10^{-1}	267.21	2.05
'Severe'	ADMM-LAP	11	$\mathbf{1.92 \times 10^{-1}}$	$\mathbf{1.79 \times 10^{-1}}$	**156.05**	**7.33**
	ADMM-BCD	10	2.18×10^{-1}	3.27×10^{-1}	186.31	6.19

advantage over ADMM-BCD in the computational time. Moreover, the seventh column shows that the quality of restored images obtained by ADMM-LAP is much better than those obtained by ADMM-BCD.

5. Conclusion

In this paper, we propose a new efficient ADMM-LAP method for solving large scale ill-posed inverse problems, and more specifically myopic deconvolution problems with TV regularization arising from the AO retinal image restoration. Specifically, ADMM is applied to tackle the nondifferentiable and nonlinear TV regularization term first, then LAP is applied to tackle tightly coupled $(\boldsymbol{x}, \boldsymbol{w})$-subproblems appearing within each ADMM iteration. The convergence results of ADMM-LAP are presented. Moreover, the efficiency of the proposed algorithm is demonstrated with a theoretical computational complexity analysis. In our numerical experiments we show that the proposed ADMM-LAP method is superior to ADMM-BCD method in terms of both the accuracy and the efficiency. In our future work, we plan to exploit appropriate preconditioning techniques to further reduce the iteration number and the iteration time.

Acknowledgments

JN acknowledges support from the US National Science Foundation under grant DMS-1819042 and the National Institutes of Health under grant 1R13EB028700-01. The authors also thank Dr John M Nickerson, Prof. of Ophthalmology at Emory University for helpful discussions and for providing data used for numerical experiments in this paper.

ORCID iDs

James G Nagy https://orcid.org/0000-0002-0451-3853
Yuanzhe Xi https://orcid.org/0000-0003-0361-0931

References

[1] Beck A and Teboulle M 2009 A fast iterative shrinkage-thresholding algorithm for linear inverse problems *SIAM J. Imag. Sci.* **2** 183–202
[2] Blanco L and Mugnier L M 2011 Marginal blind deconvolution of adaptive optics retinal images *Opt. Express* **19** 23227–39
[3] Blanco L, Mugnier L M and Glanc M 2011 Myopic deconvolution of adaptive optics retina images *Spie Bios.* (International Society for Optics and Photonics)

[4] Blanco L, Mugnier L M, Montmerle A M and Paques M 2014 Registration and restoration of adaptive-optics corrected retinal images *2014 Int. Workshop on Computational Intelligence for Multimedia Understanding* (*IWCIM*) *IEEE)* 1–5
[5] Boyd S, Parikh N, Chu E, Peleato B and Eckstein J 2010 Distributed optimization and statistical learning via the alternating direction method of multipliers *Found. Trends Mach. Learn.* **3** 1–122
[6] Chen Z, Nagy J G, Xi Y and Yu B 2019 Structured FISTA for image restoration *Numer. Lin. Algebra Appl.* **27** e2278
[7] Gazzola S, Hansen P C and Nagy J G 2019 IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems *Numer. Algorithms* **81** 773–811
[8] Haber E 2014 *Computational Methods in Geophysical Electromagnetics* (Philadelphia, PA: SIAM)
[9] Hansen P, Nagy J G and O'Leary D P 2006 *Deblurring Images* (Philadelphia, PA: SIAM)
[10] Herring J L, Nagy J G and Ruthotto L 2018 LAP: a linearize and project method for solving inverse problems with coupled variables *Sampl. Theory Signal Image Process.* **17** 127–51
[11] Hu Y, Nagy J G, Zhang J and Andersen M S 2019 Nonlinear optimization for mixed attenuation polyenergetic image reconstruction *Inverse Problems* **35** 064004
[12] Kelley C T 1999 *Iterative Methods for Optimization* (Philadelphia, PA: SIAM)
[13] Lai R and Osher S 2014 A splitting method for orthogonality constrained problems *J. Sci. Comput.* **58** 431–49
[14] Mei J-J, Dong Y, Huang T-Z and Yin W 2018 Cauchy noise removal by nonconvex ADMM with convergence guarantees *J. Sci. Comput.* **74** 743–66
[15] Nocedal J and Wright S 1999 *Numerical Optimization* (Berlin: Springer)
[16] Rudin L I, Osher S and Fatemi E 1992 Nonlinear total variation based noise removal algorithms *Phys.* D **60** 259–68
[17] Rudin L and Osher S 1994 Total variation based image restoration with free local constraints *Proc. of the 1st IEEE Int. Conf. on Image Processing* vol 1 31–5
[18] Rockafellar R T and Wets R J B 2009 *Variational Analysis* (Berlin: Springer)
[19] Tao M, Yang J and He B 2009 Alternating direction algorithms for total variation deconvolution in image reconstruction *Tech. Rep. TR0918* (Department of Mathematics, Nanjing University)
[20] Tikhonov A N and Arsenin V Y 1977 *Solution of Ill-Posed Problems* (Washington, DC: V H Winston and Sons)
[21] Wang Y, Yang J, Yin W and Zhang Y 2008 A new alternating minimization algorithm for total variation image reconstruction *SIAM J. Imag. Sci.* **1** 248–72
[22] Wang Y, Yin W and Zeng J 2019 Global convergence of ADMM in nonconvex nonsmooth optimization *J. Sci. Comput.* **78** 29–63
[23] Xi Y, Xia J, Cauley S and Balakrishnan V 2014 Superfast and stable structured solvers for Toeplitz least squares via randomized sampling *SIAM J. Matrix Anal. Appl.* **35** 44–72
[24] Xia J, Xi Y and Gu M 2012 A superfast structured solver for Toeplitz linear systems via randomized sampling *SIAM J. Matrix Anal. Appl.* **33** 837–58
[25] Xu Y and Yin W 2013 A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion *SIAM J. Imag. Sci.* **6** 1758–89
[26] Xu Y, Yin W, Wen Z and Zhang Y 2012 An alternating direction algorithm for matrix completion with nonnegative factors *Front. Math. China* **7** 365–84
[27] Yang L, Pong T K and Chen X 2012 Alternating direction method of multipliers for nonconvex background/foreground extraction *SIAM J. Imag. Sci.* **10** 74–110
[28] Zhang J and Nagy J G 2018 An alternating direction method of multipliers for the solution of matrix equations arising in inverse problems *Numer. Lin. Algebra Appl.* **25** e2123