**Journal of Scientific Computing manuscript No.**
(will be inserted by the editor)

# Fast Alternating Direction Multipliers Method by Generalized Krylov Subspaces

**Alessandro Buccini**

Received: date / Accepted: date

**Abstract** The Alternating Direction Multipliers Method (ADMM) is a very popular and powerful algorithm for the solution of many optimization problems. In the recent years it has been widely used for the solution of ill-posed inverse problems. However, one of its drawback is the possibly high computational cost, since at each iteration, it requires the solution of a large-scale least squares problem.

In this work we propose a computationally attractive implementation of ADMM, with particular attention to ill-posed inverse problems. We significantly decrease the computational cost by projecting the original large scale problem into a low-dimensional subspace by means of Generalized Krylov Subspaces (GKS). The dimension of the projection space is not an additional parameter of the method as it increases with the iterations. The construction of GKS allows for very fast computations, regardless of the increasing size of the problem. Several computed examples show the good performances of the proposed method.

**Keywords** ADMM · Ill-posed inverse problems · Generalized Krylov subspaces

**Mathematics Subject Classification (2000)** 65K10 · 65F22 · 65F10

A. Buccini
Dipartimento di Matematica e Informatica,
Università degli Studi di Cagliari,
Via Ospedale 72,
09124 Cagliari, Italy
E-mail: alessandro.buccini@unica.it

## 1 Introduction

In many applications of science and engineering we need to solve problems of the form

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \mu\mathfrak{R}(L\mathbf{x}), \tag{1}$$

where $\|\cdot\|_2$ denotes the Euclidean norm, $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ denotes some measured data, $\mathbf{x} \in \mathbb{R}^n$ is the unknown signal we are interested in recovering, $\mu > 0$ is a fixed parameter, $L \in \mathbb{R}^{s\times n}$, and $\mathfrak{R} : \mathbb{R}^s \to \mathbb{R}\cup\{\infty\}$ is closed, proper, and convex. We will assume that

$$\mathcal{N}(A) \cap \mathcal{N}(L) = \{\mathbf{0}\}, \tag{2}$$

where $\mathcal{N}(M)$ denotes the null space of the matrix $M$. Among the possible applications one of the most relevant for our purposes is the solution of ill-posed inverse problems. In this scenario $A$ is assumed to be severely ill-conditioned and may be rank deficient and the data $\mathbf{b}$ are usually corrupted by some errors; see, e.g., [26, 32, 33] for a discussion. In this case it is of interest to construct a regularization term such that $\mathfrak{R}(L\mathbf{x}_{\text{true}}) \approx 0$ and $\mathfrak{R}(L\mathbf{x}) \gg 0$ for $\mathbf{x}$ far from $\mathbf{x}_{\text{true}}$, where $\mathbf{x}_{\text{true}}$ denotes the exact solution of the problem. Many choices of regularization terms have been considered in the literature, like Total Variation [39], $\ell_p$ norms with $0 < p \leq 2$ [13–16, 22, 27, 36, 38], framelet/wavelet/Fourier transforms [11, 19–21, 24], etc.

We would like to mention that another application, which is extremely relevant in statistics, where this kind of problems arise is linear regression and variable selection. In this case the matrix $A$ contains the values of $n$ covariates obtained over $m$ observations, while the vector $\mathbf{b}$ collects the responses. The desired vector $\mathbf{x}$ represents the parameters on which the phenomenon analyzed depends and it is usually assumed to be sparse; see, e.g., [7, 8, 41].

Regardless of the application, once the minimization problem (1) is fixed, we need to select an appropriate algorithm to numerically solve it. One of the most popular methods for solving (1) is the Alternating Direction Multiplier Method (ADMM); see [1–3, 6] and references therein. This iterative algorithm allows to decouple the term $\|A\mathbf{x} - \mathbf{b}\|_2^2$ and the, possibly, non-linear term $\mathfrak{R}(L\mathbf{x})$; see Section 2 for more details. In particular, if the proximal operator of $\mathfrak{R}(\cdot)$ is known in closed form or can be cheaply computed, then the ADMM is extremely competitive. In this scenario, as we will show later, the main computational cost per iteration is the solution of a linear system of equations of the form

$$(A^TA + \rho L^TL)\mathbf{y} = \mathbf{g}, \tag{3}$$

where $\rho > 0$ is a fixed parameter. The linear system (3) is equivalent to the least squares problem

$$\min_{\mathbf{y}} \left\| \begin{bmatrix} A \\ \sqrt{\rho}L \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{g} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 .$$

The matrix $(A^TA + \rho L^TL)$ is obviously symmetric and positive definite under assumption (2). Moreover, if $\rho$ is large enough, it is well-conditioned as well.

Therefore, the solution of the system (3) can be easily computed using, for instance, the `cgls` method; see, e.g., [5]. Nevertheless, this approach may lead to a high overall computational cost.

Our purpose is to propose a very efficient implementation of the ADMM by projecting the problem into a fairly small Krylov-like subspace. In particular, we consider the so-called Generalized Krylov Subspaces (GKS). These spaces are constructed iteratively and are enlarged at each iteration by adding the normalized residual of the normal equation (3). Moreover, this procedure allows us to solve, at each iteration, the projected linear system only by computing a matrix-vector product with $A$ and $L$ and a matrix-vector product with $A^T$ and $L^T$. We would like to stress that it is not the aim of this paper to discuss how to construct the model (1), nor how to determine the parameter $\mu$. Therefore, we will assume that (1) is given a priori and that our task is merely to solve it.

To the best of our knowledge, Krylov methods have been coupled with ADMM only in [4, 42]. In [4] the authors only consider the case in which the operator $A$ can be expressed as the Kronecker product of two matrices and the regularization term is the TV norm. In this work we do not make any assumption on the structure of $A$ and we only require $\mathfrak{R}$ to be closed, proper, and convex. In [42] the authors first construct an appropriate Krylov subspace of fixed dimension and solve (1) in the smaller space. Our approach differs form the one in [42] in two ways: first we do not need to fix the dimension of the search subspace as it is constructed throughout the iterations of ADMM, second the space we use is not built using only $A$ and $\mathbf{b}$, but considering the whole iterations and $L$. The latter point is extremely relevant because, as we will show in the numerical examples, it allows us to obtain a higher degree of accuracy than the one obtained by the method in [42]. Finally, we would like to recall that in [38] GKS were coupled with an alternating minimization algorithm for the solution of $\ell_p - \ell_q$ minimization problem. Even though alternating minimization is not equivalent to ADMM these two methods are strictly related.

In this paper we do not consider acceleration techniques to improve the speed of convergence of ADMM like the ones proposed in [9, 29]. However, these techniques can be applied to the proposed method without substantially changing the results reported in this paper. In fact, we are concerned with lowering the computational cost of the single iteration of ADMM and not with speeding up its convergence.

This paper is structured as follows: Section 2 describes ADMM applied to (1), in Section 3 we describe our proposed method and derive some theoretical results, Section 4 reports selected numerical experiments, and we draw some conclusions and outline future work in Section 5.

## 2 Alternating Direction Multipliers Method

In this section we describe how to use the ADMM to solve (1). We recall that, if $\mathfrak{R}(\cdot)$ is closed, convex, and proper the iterates generated by ADMM converge to a solution of (1); see below.

First we reformulate (1) as the following constrained minimization problem

$$\min_{\mathbf{x},\mathbf{z}} \left\{ \frac{1}{2}\left\|A\mathbf{x}-\mathbf{b}\right\|_2^2 + \mu\mathfrak{R}(\mathbf{z});\ L\mathbf{x}=\mathbf{z} \right\}. \tag{4}$$

The Augmented Lagrangian function associated to (4) is

$$\mathcal{L}_\rho(\mathbf{x},\mathbf{z};\boldsymbol{\lambda}) = \frac{1}{2}\left\|A\mathbf{x}-\mathbf{b}\right\|_2^2 + \mu\mathfrak{R}(\mathbf{z}) + \boldsymbol{\lambda}^T(\mathbf{z}-L\mathbf{x}) + \frac{\rho}{2}\left\|\mathbf{z}-L\mathbf{x}\right\|_2^2,$$

where $\rho > 0$ is a user-defined parameter, $\boldsymbol{\lambda}$ is the Lagrangian multiplier, and the superscript $T$ denotes the transposition.

The ADMM is obtained by minimizing alternatively $\mathcal{L}_\rho$ with respect to the primal variables $\mathbf{x}$ and $\mathbf{z}$ and by updating the dual variable $\boldsymbol{\lambda}$; see Algorithm 1.

**Algorithm 1:** ADMM

```
1 $\mathbf{z}^{(0)} = \mathbf{0}$;
2 $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$;
3 for $k = 0, 1, \dots$ do
4     $\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \mathcal{L}_\rho\left(\mathbf{x}, \mathbf{z}^{(k)}; \boldsymbol{\lambda}^{(k)}\right)$;
5     $\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}} \mathcal{L}_\rho\left(\mathbf{x}^{(k+1)}, \mathbf{z}; \boldsymbol{\lambda}^{(k)}\right)$;
6     $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho\left(\mathbf{z}^{(k+1)} - L\mathbf{x}^{(k+1)}\right)$;
7     if $\frac{\left\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\right\|_2}{\left\|\mathbf{x}^{(k)}\right\|_2} < \tau$ then
8         exit;
```

Consider the computation of $\mathbf{x}^{(k+1)}$. We can rewrite it as

$$\begin{aligned}
\mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}} \mathcal{L}_\rho(\mathbf{x},\mathbf{z}^{(k)};\boldsymbol{\lambda}^{(k)}) \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\left\|A\mathbf{x}-\mathbf{b}\right\|_2^2 - \left(\boldsymbol{\lambda}^{(k)}\right)^T L\mathbf{x} + \frac{\rho}{2}\left\|\mathbf{z}^{(k)} - L\mathbf{x}\right\|_2^2 \\
&= \arg\min_{\mathbf{x}} \left\| \begin{bmatrix} A \\ \sqrt{\rho}L \end{bmatrix}\mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \sqrt{\rho}\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right) \end{bmatrix} \right\|_2^2,
\end{aligned} \tag{5}$$

i.e., $\mathbf{x}^{(k+1)}$ can be obtained by solving a least squares problem. We now move to the computation of $\mathbf{z}^{(k+1)}$.

$$\begin{aligned}
\mathbf{z}^{(k+1)} &= \arg\min_{\mathbf{z}} \mathcal{L}_\rho\left(\mathbf{x}^{(k+1)}, \mathbf{z}; \boldsymbol{\lambda}^{(k)}\right) \\
&= \arg\min_{\mathbf{z}} \mu\mathfrak{R}(\mathbf{z}) + (\boldsymbol{\lambda}^{(k)})^T\mathbf{z} + \frac{\rho}{2}\left\|\mathbf{z} - L\mathbf{x}^{(k+1)}\right\|_2^2 \\
&= \arg\min_{\mathbf{z}} \mu\mathfrak{R}(\mathbf{z}) + \frac{\rho}{2}\left\|\mathbf{z} - L\mathbf{x}^{(k+1)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right\|_2^2 \\
&= \arg\min_{\mathbf{z}} \frac{1}{2}\left\|\mathbf{z} - L\mathbf{x}^{(k+1)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right\|_2^2 + \frac{\mu}{\rho}\mathfrak{R}(\mathbf{z}),
\end{aligned}$$

i.e., $\mathbf{z}^{(k+1)}$ can be computed by the proximal operator of $\mathfrak{R}$.

Assuming that $\mathfrak{R}$ is easily proxable, the main computational cost of Algorithm 1 is the solution of the least squares problem (5). In the following section we will provide an efficient way to compute $\mathbf{x}^{(k+1)}$.

We now report the main theoretical result on ADMM. For a proof and discussion see, e.g., [6]. We adapt the more general result in [6, Section 3.2] to our notation and considered case

**Theorem 1** *Assume that the extended-real-valued function $\mathfrak{R}$ is closed, proper, and convex. Moreover, assume that the unaugmented Lagrangian $\mathcal{L}_0$ has a saddle point, i.e., that there exists at least a point $(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*)$ such that the inequality*

$$\mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}) \leq \mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}^*) \leq \mathcal{L}_0(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda}^*)$$

*holds for all $\mathbf{x}$, $\mathbf{z}$, and $\boldsymbol{\lambda}$, then the iterates generated by Algorithm 1 satisfy the following*

*(i) Residual convergence:* $\lim_{k\to\infty} \left\|\mathbf{z}^{(k)} - L\mathbf{x}^{(k)}\right\|_2 = 0$, *i.e., the iterates approach feasibility;*

*(ii) Objective convergence:* $\lim_{k\to\infty} \frac{1}{2}\left\|A\mathbf{x}^{(k)} - \mathbf{b}\right\|_2^2 + \mu\mathfrak{R}(\mathbf{z}) = p^*$, *where $p^*$ is the minimum of* (4), *i.e., the objective function of the iterates approaches the optimal value;*

*(iii) Dual variable convergence:* $\lim_{k\to\infty} \boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^*$, *where $\boldsymbol{\lambda}^*$ is a dual optimal point.*

## 3 ADMM in Generalized Krylov Subspaces

We wish to propose a numerically cheap way of determining $\mathbf{x}^{(k+1)}$. We determine a solution in a fairly small subspace. Let $V_k \in \mathbb{R}^{n\times\hat{k}}$ with $\hat{k} \ll n$ have orthonormal columns and assume that these columns span the search space in which we wish to find $\mathbf{x}^{(k+1)}$. Therefore, $\mathbf{x}^{(k+1)}$ can be expressed as

$$\mathbf{x}^{(k+1)} = V_k\mathbf{y}^{(k+1)}.$$

Substituting this expression into (5) yields

$$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \left\| \begin{bmatrix} AV_k \\ \sqrt{\rho}LV_k \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{b} \\ \sqrt{\rho}\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right) \end{bmatrix} \right\|_2^2 . \tag{6}$$

Define the compact QR factorizations of $AV_k$ and $LV_k$ by

$$\begin{aligned} AV_k &= Q_A^{(k)} R_A^{(k)} \quad Q_A^{(k)} \in \mathbb{R}^{n\times\hat{k}},\ R_A^{(k)} \in \mathbb{R}^{\hat{k}\times\hat{k}}, \\ LV_k &= Q_L^{(k)} R_L^{(k)} \quad Q_L^{(k)} \in \mathbb{R}^{s\times\hat{k}},\ R_L^{(k)} \in \mathbb{R}^{\hat{k}\times\hat{k}}, \end{aligned}$$

where $Q_A^{(k)}$ and $Q_L^{(k)}$ have orthonormal columns and $R_A^{(k)}$ and $R_L^{(k)}$ are upper-triangular.

Substituting these factorizations in (6) we obtain

$$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \left\| \begin{bmatrix} R_A^{(k)} \\ \sqrt{\rho}R_L^{(k)} \end{bmatrix} \mathbf{y} - \begin{bmatrix} (Q_A^{(k)})^T\mathbf{b} \\ \sqrt{\rho}(Q_L^{(k)})^T\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right) \end{bmatrix} \right\|_2^2 .$$

Since $\hat{k} \ll n$ this least squares problem can be solved cheaply with direct methods; see, e.g., [25].

Once we have computed $\mathbf{y}^{(k+1)}$ we can enlarge the space by adding a new vector, namely the normalized residual of the normal equations associated to (5), i.e.,

$$(A^TA + \rho L^TL)\mathbf{x}^{(k+1)} = A^T\mathbf{b} + \rho L^T\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right),$$

defined by

$$\mathbf{r}^{(k+1)} = A^T(A\mathbf{x}^{(k+1)} - \mathbf{b}) + \rho L^T\left(L\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)} - \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right).$$

Therefore, the new basis vector is obtained by $\mathbf{v}^{(k+1)} = \mathbf{r}^{(k+1)}/\left\|\mathbf{r}^{(k+1)}\right\|_2$ and an orthonormal basis of the enlarged space is made up by the columns of $V_{k+1} = [V_k,\ \mathbf{v}^{(k+1)}]$. Note that the vector $\mathbf{r}^{(k+1)}$ is orthogonal to the space spanned by the columns of $V_k$. The search space constructed in this way is referred to as Generalized Krylov Subspace (GKS); see, e.g., [37].

We can compute and store the matrices $AV_{k+1}$ and $LV_{k+1}$ by

$$AV_{k+1} = \left[AV_k,\ A\mathbf{v}^{(k+1)}\right] \quad \text{and} \quad LV_{k+1} = \left[LV_k,\ L\mathbf{v}^{(k+1)}\right].$$

The QR factorizations of $AV_{k+1}$ and $LV_{k+1}$ can be easily computed from the ones of $AV_k$ and $LV_k$ as

$$\begin{aligned} AV_{k+1} &= [Q_A^{(k)}, \tilde{\mathbf{q}}_A] \begin{bmatrix} R_A^{(k)} & \mathbf{r}_A \\ \mathbf{0}^t & \tau_A \end{bmatrix}, \\ LV_{k+1} &= [Q_L^{(k)}, \tilde{\mathbf{q}}_L] \begin{bmatrix} R_L^{(k)} & \mathbf{r}_L \\ \mathbf{0}^t & \tau_L \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned}
\mathbf{r}_A &= \left(Q_A^{(k)}\right)^T (A\mathbf{v}^{(k+1)}), & \mathbf{q}_A &= A\mathbf{v}^{(k+1)} - Q_A^{(k)}\mathbf{r}_A,\\
\tau_A &= \|\mathbf{q}_A\|_2, & \tilde{\mathbf{q}}_A &= \mathbf{q}_A/\tau_A,\\
\mathbf{r}_L &= \left(Q_L^{(k)}\right)^T (L\mathbf{v}^{(k+1)}), & \mathbf{q}_L &= L\mathbf{v}^{(k+1)} - Q_L^{(k)}\mathbf{r}_L,\\
\tau_L &= \|\mathbf{q}_L\|_2, & \tilde{\mathbf{q}}_L &= \mathbf{q}_L/\tau_L;
\end{aligned}$$

see Daniel et al. [23] for details. Therefore, if we store the matrices $AV_k$ and $LV_k$, the computation of $\mathbf{x}^{(k+1)}$ requires only one matrix-vector product with $A$ and with $L$ and one with $A^T$ and $L^T$. Note that the computation of $\mathbf{r}^{(k+1)}$ can be rewritten as

$$\mathbf{r}^{(k+1)} = A^T\left(AV_k\mathbf{y}^{(k+1)} - \mathbf{b}\right) + \rho L^T\left(LV_k\mathbf{y}^{(k+1)} - \mathbf{z}^{(k)} - \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right).$$

We now show the convergence of Algorithm 2.

**Theorem 2** *Assume that the extended-real-valued function $\mathfrak{R}$ is closed, proper, and convex. Moreover, assume that the unaugmented Lagrangian $\mathcal{L}_0$ has a saddle point, i.e., that there exists at least a point $(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*)$ such that the inequality*

$$\mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}) \leq \mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}^*) \leq \mathcal{L}_0(\mathbf{x}, \mathbf{z}; \boldsymbol{\lambda}^*)$$

*holds for all $\mathbf{x}$, $\mathbf{z}$, and $\boldsymbol{\lambda}$, then the iterates generated by Algorithm 2 satisfy the following*

*(i) Residual convergence:* $\lim_{k\to\infty}\left\|\mathbf{z}^{(k)} - L\mathbf{x}^{(k)}\right\|_2 = 0$, *i.e., the iterates approach feasibility;*

*(ii) Objective convergence:* $\lim_{k\to\infty}\frac{1}{2}\left\|A\mathbf{x}^{(k)} - \mathbf{b}\right\|_2^2 + \mu\mathfrak{R}(\mathbf{z}) = p^*$, *where $p^*$ is the minimum of (4), i.e., the objective function of the iterates approaches the optimal value;*

*(iii) Dual variable convergence:* $\lim_{k\to\infty}\boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^*$, *where $\boldsymbol{\lambda}^*$ is a dual optimal point.*

*Proof* The proofs follows trivially from Theorem 1 as, after at most $n$ iterations, we have that the space spanned by the columns of $V_n$ is $\mathbb{R}^n$. Therefore, after at most $n$ iterations Algorithm 1 and Algorithm 2 coincide. □

The proof of Theorem 2 suggests that at least $n$ iterations are required to compute a good approximation of the limit point of the algorithm. However, as we will see in the numerical examples, this is not the case and few iterations are required to compute a good approximate solution.

We now can show our main theoretical result which sheds some light on the convergence properties of Algorithm 2 in the first $n$ iterations.

Firstly we recall that, for the standard ADMM, the quantity

$$h^{(k)} = \frac{1}{\rho}\left\|\boldsymbol{\lambda}^{(k)} - \boldsymbol{\lambda}^*\right\|_2^2 + \rho\left\|\mathbf{z}^{(k)} - \mathbf{z}^*\right\|_2^2$$

**Algorithm 2:** ADMM in GKS

```
$\mathbf{z}^{(0)} = \mathbf{0}$;
$\boldsymbol{\lambda}^{(0)} = \mathbf{0}$;
Construct a matrix $V_0 \in \mathbb{R}^{n\times k_0}$ such that $V_0^T V_0 = I$;
Compute and store $AV_0$ and $LV_0$;
Compute the compact QR factorizations $AV_0 = Q_A^{(0)} R_A^{(0)}$ and $LV_0 = Q_L^{(0)} R_L^{(0)}$;
for $k = 0, 1, \dots$ do
    $\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \left\| \begin{bmatrix} R_A^{(k)} \\ \sqrt{\rho} R_L^{(k)} \end{bmatrix} \mathbf{y} - \begin{bmatrix} (Q_A^{(k)})^T \mathbf{b} \\ \sqrt{\rho} (Q_L^{(k)})^T \left( \mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho} \right) \end{bmatrix} \right\|_2^2$;
    $\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}} \frac{1}{2} \left\| \mathbf{z} - LV_k \mathbf{y}^{(k+1)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho} \right\|_2^2 + \frac{\mu}{\rho} \mathfrak{R}(\mathbf{z})$;
    $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho(\mathbf{z}^{(k+1)} - LV_k \mathbf{y}^{(k+1)})$;
    if $\frac{\left\| \mathbf{y}^{(k+1)} - \mathbf{y}^{(k)} \right\|_2}{\left\| \mathbf{y}^{(k)} \right\|_2} < \tau$ then
        exit;
    $\mathbf{r}^{(k+1)} = A^T \left( AV_k \mathbf{y}^{(k+1)} - \right) + \rho L^T \left( LV_k \mathbf{y}^{(k+1)} - \mathbf{z}^{(k)} - \frac{\boldsymbol{\lambda}^{(k)}}{\rho} \right)$;
    $\mathbf{v}^{(k+1)} = \mathbf{r}^{(k+1)} / \left\| \mathbf{r}^{(k+1)} \right\|_2$;
    $V_{k+1} = \left[ V_k,\ \mathbf{v}^{(k+1)} \right]$;
    $AV_{k+1} = [AV_k,\ A\mathbf{v}^{(k+1)}]$;
    $LV_{k+1} = [LV_k,\ L\mathbf{v}^{(k+1)}]$;
    $\mathbf{r}_A = \left( Q_A^{(k)} \right)^T (A\mathbf{v}^{(k+1)})$;
    $\mathbf{q}_A = A\mathbf{v}^{(k+1)} - Q_A^{(k)} \mathbf{r}_A$;
    $\tau_A = \|\mathbf{q}_A\|_2$;
    $Q_A^{(k+1)} = [Q_A^{(k)},\ \tilde{\mathbf{q}}_A]$;
    $R_A^{(k+1)} = \begin{bmatrix} R_A^{(k)} & \mathbf{r}_A \\ \mathbf{0}^t & \tau_A \end{bmatrix}$;
    $\mathbf{r}_L = \left( Q_L^{(k)} \right)^T (L\mathbf{v}^{(k+1)})$;
    $\mathbf{q}_L = L\mathbf{v}^{(k+1)} - Q_L^{(k)} \mathbf{r}_L$;
    $\tau_L = \|\mathbf{q}_L\|_2$;
    $Q_L^{(k+1)} = [Q_L^{(k)},\ \tilde{\mathbf{q}}_L]$;
    $R_L^{(k+1)} = \begin{bmatrix} R_L^{(k)} & \mathbf{r}_L \\ \mathbf{0}^t & \tau_L \end{bmatrix}$;
$\mathbf{x}^* = V_k \mathbf{y}^{(k+1)}$;
```

is a Lyapunov function for the algorithm, i.e., a nonnegative quantity that decreases monotonically with the iterations; see, e.g., [6]. We wish to show that, even though $h^{(k)}$ is not a Lyapunov function for Algorithm 2, we can describe how it evolves in the first $n$ iterations.

We first need to reformulate our problem in an equivalent form. From Theorem 2 we know that Algorithm 2 converges. Therefore, after at most $n$ iterations, the GKS coincides with $\mathbb{R}^n$ and the algorithm has constructed an orthonormal basis for $\mathbb{R}^n$ denoted by $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$. Let $V = [\mathbf{v}_1, \dots, \mathbf{v}_n]$, we can reformulate (4) as

$$\min_{\mathbf{x},\mathbf{z}} \left\{ \frac{1}{2} \|AV\mathbf{x} - \mathbf{b}\|_2^2 + \mu\mathfrak{R}(\mathbf{z});\ LV\mathbf{x} = \mathbf{z} \right\}, \tag{7}$$

where, with a slight abuse of notation, we have denoted by $\mathbf{x}$ the coefficient of the vector $\mathbf{x}$ in (4) with respect to the basis $V$. Obviously the minimization problems (4) and (7) are equivalent and applying Algorithm 2 to either the first or the second one does not change the iterations. Moreover, we denote by $\mathcal{L}_\rho$ the augmented Lagrangian of (7) and by $(\mathbf{x}^*, \mathbf{z}^*, \boldsymbol{\lambda}^*)$ a saddle point of $\mathcal{L}_0$. We introduce some additional notation

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2} \|AV\mathbf{x} - \mathbf{b}\|_2^2, \\ g(\mathbf{z}) &= \mu\mathfrak{R}(\mathbf{z}), \\ p^{(k)} &= f\left(\mathbf{x}^{(k)}\right) + g\left(\mathbf{z}^{(k)}\right), \\ p^* &= f\left(\mathbf{x}^*\right) + g\left(\mathbf{z}^*\right), \\ \mathbf{r}^{(k)} &= \mathbf{z}^{(k)} - LV\mathbf{x}^{(k)}, \end{aligned}$$

where $\mathbf{x}^{(k)}$, $\mathbf{z}^{(k)}$, and $\boldsymbol{\lambda}^{(k)}$ denote the iterates computed by Algorithm 2 applied to (7). Note that the iterates of Algorithm 2 applied to (4) can be obtained by multiplying $\mathbf{x}^{(k)}$, $\mathbf{z}^{(k)}$, and $\boldsymbol{\lambda}^{(k)}$ by $V$. Finally, let $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$, then we define, for $k \leq n$

$$\mathbf{x}_k^* = [x_1^*, x_2^*, \ldots, x_k^*, 0, \ldots, 0]^T.$$

We are now in position to show our main result

**Theorem 3** *With the assumptions of Theorem 2 and the notation above we have, for $k \leq n$,*

$$h^{(k+1)} \leq h^{(k)} - \rho \left\|\mathbf{r}^{(k+1)}\right\|_2^2 - \rho \left\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right\|_2^2 + C \left\|\mathbf{x}^* - \mathbf{x}_k^*\right\|_2,$$

*where $C \geq 0$ is a constant independent of $k$ and $\mathbf{x}^{(k)}$.*

*Proof* The proof of this result is obtained by following the same steps of [6, Appendix A] with some substantial modifications.

By definition of $(\mathbf{x}^*, \mathbf{z}^* \boldsymbol{\lambda}^*)$ it holds

$$\mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}^*) \leq \mathcal{L}_0\left(\mathbf{x}^{(k+1)}, \mathbf{z}^{(k+1)}; \boldsymbol{\lambda}^*\right),$$

since $LV\mathbf{x}^* = \mathbf{z}^*$ it holds $\mathcal{L}_0(\mathbf{x}^*, \mathbf{z}^*; \boldsymbol{\lambda}^*) = p^*$, therefore,

$$p^* \leq p^{(k+1)} + (\boldsymbol{\lambda}^*)^T \left(\mathbf{z}^{(k+1)} - LV\mathbf{x}^{(k+1)}\right) = p^{(k+1)} + (\boldsymbol{\lambda}^*)^T \mathbf{r}^{(k+1)}.$$

Let $\iota_k$ be the indicator function of the linear subspace spanned by the first $k$ columns of the identity matrix, i.e.,

$$\iota_k(\mathbf{x}) = \begin{cases} 0 & \text{if } x_{k+1} = \ldots = x_n = 0, \\ \infty & \text{else.} \end{cases}$$

By construction we have that $\mathbf{x}^{(k+1)}$ minimizes $\mathcal{L}_\rho\left(\mathbf{x}, \mathbf{z}^{(k)}; \boldsymbol{\lambda}^{(k)}\right) + \iota_k(\mathbf{x})$. Since $f$ is differentiable and convex, we have

$$0 \in \nabla f\left(\mathbf{x}^{(k+1)}\right) + V^T L^T \boldsymbol{\lambda}^{(k)} + \rho V^T L^T\left(\mathbf{z}^{(k)} - LV\mathbf{x}^{(k+1)}\right) + \partial\iota_k\left(\mathbf{x}^{(k+1)}\right),$$

where $\partial\iota_k(\mathbf{x})$ denotes the subgradient of $\iota_k$ at $\mathbf{x}$. Using the fact that $\boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^{(k+1)} - \rho\mathbf{r}^{(k+1)}$ and rearranging the terms we obtain

$$0 \in \nabla f\left(\mathbf{x}^{(k+1)}\right) + V^T L^T\left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right) + \partial\iota_k\left(\mathbf{x}^{(k+1)}\right).$$

This implies that $\mathbf{x}^{(k+1)}$ minimizes

$$f(\mathbf{x}) + \left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right)^T LV\mathbf{x} + \iota_k(\mathbf{x}).$$

Observe that $\iota_k(\mathbf{x}_k^*) = 0$, therefore,

$$\begin{aligned} & f\left(\mathbf{x}^{(k+1)}\right) + \left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right)^T LV\mathbf{x}^{(k+1)} + \iota_k\left(\mathbf{x}^{(k+1)}\right) \\ & \leq f\left(\mathbf{x}_k^*\right) + \left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right)^T LV\mathbf{x}_k^* + \iota_k\left(\mathbf{x}_k^*\right) \\ & \quad + f\left(\mathbf{x}^*\right) - f\left(\mathbf{x}^*\right) + \left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right)^T LV(\mathbf{x}^* - \mathbf{x}^*) \end{aligned} \tag{8}$$

Similarly as above we can show that $\mathbf{z}^{(k+1)}$ minimizes $g(\mathbf{z}) + \left(\boldsymbol{\lambda}^{(k+1)}\right)^T\mathbf{z}$. Therefore,

$$g\left(\mathbf{z}^{(k+1)}\right) - \left(\boldsymbol{\lambda}^{(k+1)}\right)^T\mathbf{z}^{(k+1)} \leq g\left(\mathbf{z}^*\right) - \left(\boldsymbol{\lambda}^{(k+1)}\right)^T\mathbf{z}^* \tag{9}$$

Adding (8) and (9), rearranging the terms, and using the fact that $\mathbf{z}^* = LV\mathbf{x}^*$ we obtain

$$\begin{aligned} & f\left(\mathbf{x}^{(k+1)}\right) + g\left(\mathbf{z}^{(k+1)}\right) - \left(f\left(\mathbf{x}^*\right) + g\left(\mathbf{z}^*\right)\right) \\ & \leq -\left(\boldsymbol{\lambda}^{(k+1)}\right)^T\mathbf{r}^{(k+1)} + \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)^T\left(LV\mathbf{x}^* - LV\mathbf{x}^{(k+1)}\right) + \gamma_k \\ & = -\left(\boldsymbol{\lambda}^{(k+1)}\right)^T\mathbf{r}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)^T\left(-\mathbf{r}^{(k+1)} + \left(\mathbf{z}^{(k+1)} - \mathbf{z}^*\right)\right) + \gamma_k, \end{aligned}$$

where

$$\gamma_k = f\left(\mathbf{x}_k^*\right) - f\left(\mathbf{x}^*\right) + \left(\boldsymbol{\lambda}^{(k+1)} - \rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)\right)^T LV\left(\mathbf{x}_k^* - \mathbf{x}^*\right).$$

Summarizing we have obtained that

$$\begin{aligned} & p^* - p^{(k+1)} \le \left(\boldsymbol{\lambda}^*\right)^T \mathbf{r}^{(k+1)} \\ & p^{(k+1)} - p^* \le -\left(\boldsymbol{\lambda}^{(k+1)}\right)^T \mathbf{r}^{(k+1)} \\ & \qquad\qquad -\rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)^T \left(-\mathbf{r}^{(k+1)} + \left(\mathbf{z}^{(k+1)} - \mathbf{z}^*\right)\right) + \gamma_k \end{aligned} \tag{10}$$

Summing the inequalities in (10) and multiplying through by 2 we obtain

$$0 \ge 2\left(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}^{(k+1)}\right)\mathbf{r}^{(k+1)} - 2\rho\left(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right)^T \left(\mathbf{z}^* - \mathbf{z}^{(k+1)}\right) - 2\gamma_k$$

Proceeding as in the proof of the inequality A.1 in [6] we obtain that

$$h^{(k+1)} \le h^{(k)} - \rho\left\|\mathbf{r}^{(k+1)}\right\|_2^2 - \rho\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right\|_2^2 + 2\gamma_k.$$

Since the steps the inequality above are the same as in the proof of A.1 in [6] we do not report them here.

We now show that

$$2\gamma_k \le C\left\|\mathbf{x}^* - \mathbf{x}_k^*\right\|_2.$$

Since $k = 1, 2, \ldots, n$, we have that there exist two constants $C_1$ and $C_2$ such that, $\left\|\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)}\right\|_2 \le C_1$ and $\left\|\boldsymbol{\lambda}^{(k)}\right\|_2 \le C_2$ for all $k$. We have

$$\begin{aligned} 2\gamma_k &\le \left\|AV\mathbf{x}_k^* - \mathbf{b}\right\|_2^2 - \left\|AV\mathbf{x}_* - \mathbf{b}\right\|_2^2 + 2(C_1 + C_2)\left\|L\right\|_2\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &= \left\|AV\mathbf{x}_k^*\right\|_2^2 + 2\mathbf{b}^T\left(AV\mathbf{x}^* - AV\mathbf{x}_k^*\right) - \left\|AV\mathbf{x}_*\right\|_2^2 \\ &\quad + 2(C_1 + C_2)\left\|L\right\|_2\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &\le \left(AV\mathbf{x}_k^* + AV\mathbf{x}_*\right)^T\left(AV\mathbf{x}_k^* - AV\mathbf{x}_*\right) \\ &\quad + 2\left((C_1 + C_2)\left\|L\right\|_2 + 2\left\|\mathbf{b}\right\|_2\left\|A\right\|_2\right)\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &\le \left(\left\|A\right\|_2^2\left\|\mathbf{x}^* + \mathbf{x}_k^*\right\|_2 + 2\left((C_1 + C_2)\left\|L\right\|_2 + 2\left\|\mathbf{b}\right\|_2\left\|A\right\|_2\right)\right)\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &\le \left(\left\|A\right\|_2^2\left(\left\|\mathbf{x}^*\right\|_2 + \left\|\mathbf{x}_k^*\right\|_2\right) + 2\left((C_1 + C_2)\left\|L\right\|_2 + 2\left\|\mathbf{b}\right\|_2\left\|A\right\|_2\right)\right)\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &\overset{(a)}{\le} 2\left(\left\|A\right\|_2^2\left\|\mathbf{x}^*\right\|_2 + (C_1 + C_2)\left\|L\right\|_2 + 2\left\|\mathbf{b}\right\|_2\left\|A\right\|_2\right)\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 \\ &= C\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2, \end{aligned}$$

where the inequality $(a)$ follows from the fact that $\left\|\mathbf{x}_k^*\right\|_2 \le \left\|\mathbf{x}^*\right\|_2$. □

The result in Theorem 3 shows that, if $\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2$ is small enough, then we recover the same convergence property of standard ADMM. Since

$$\left\|\mathbf{x}_k^* - \mathbf{x}^*\right\|_2 = \left\|V\mathbf{x}_k^* - V\mathbf{x}^*\right\|_2,$$

we have that the quantity $\gamma_k$ is small for $k \ll n$ if the first $k$ basis vector constructed by Algorithm 2 well capture the behavior of the solution $\mathbf{x}^*$. This is usually true for $k$ relatively small in practical applications.

## 4 Numerical Examples

We now show the performances of our proposed method on some selected numerical examples. In particular, we consider three instances of image deblurring, two for black and white images and one with a color image, one of computer tomography, and one of seismic travel-time tomography; see, e.g., [35] for details on image deblurring, [17] for details on computer tomography, and [34] for details on seismic tomography. In all cases the matrix $A$ is severely ill-conditioned. Moreover, we consider one regression problem where the matrix $A$ is rectangular with more columns than rows and the exact solution is known to be sparse.

We will compare our method with the classical ADMM algorithm and the proposal in [42]. The algorithm in [4] coincides with Algorithm 2 if $A$ is the Kronecker product of two matrices, $L$ is a discretization of the gradient, and either $\mathfrak{R}(\cdot)$ is the isotropic on anisotropic Total Variation norm.

Moreover, when possible, we compare our proposal with the algorithm described in [38]. In this paper the authors consider the $\ell_p - \ell_q$ minimization in GKS. In particular, we consider the implementation described in [36]. The considered method is not strictly speaking a versions of ADMM, however, the method described in [38] can be seen equivalently as an alternating minimization method or a Majorization-Minimization (MM) method. We denote, following the notation in [36], this method by A-MM-GKS; see [36] for more details. The main cost per iteration for this method is a matrix-vector product with $A$, one with $A^T$, one with $L$, and one with $L^T$. However, A-MM-GKS algorithm requires, at each iteration, to compute a QR factorization of two skinny and tall matrices of increasing dimensions.

We consider different choices for $\mathfrak{R}(\mathbf{z})$ and two different choices of $L$: a discretization of the gradient, denoted by $L_G$, and a framelet operator, denoted by $L_F$.

Since we are dealing with two dimensional problems we define the following discretization of the gradient. Let $L_1$ be defined as follows

$$L_1 = \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ 1 & & & -1 \end{bmatrix},$$

we define $L_G$ by

$$L_G = \begin{bmatrix} L_1 \otimes I \\ I \otimes L_1 \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product.

Framelets are extensions of wavelets. They are defined as rectangular matrices $L_F \in \mathbb{R}^{s \times n}$ with $s \geq n$ such that $L_F^T L_F = I$. Tight frames have been used in many image restoration applications, including inpainting and deblurring [11,12,18,19,21]. The essential feature of tight frames is their redundancy. Due to the redundancy, the loss of some information can be tolerated. Usually images have a very sparse representation in the framelet domain. We will use tight frames determined by linear B-splines; see, e.g., [19].

Depending on the example, we will consider as $\mathfrak{R}(\mathbf{z})$ one or more of the following

– Isotropic TV, defined by

$$\|\mathbf{z}\|_{\text{I-TV}} = \sum_{i=1}^{n} \left\| \begin{bmatrix} ((L_1 \otimes I)\mathbf{z})_i \\ ((I \otimes L_1)\mathbf{z})_i \end{bmatrix} \right\|_2 ;$$

– Anisotropic TV, defined by

$$\|\mathbf{z}\|_{\text{A-TV}} = \|L_G \mathbf{z}\|_1;$$

– $\ell_1$ norm in the framelet domain, i.e.,

$$\|L_F \mathbf{z}\|_1;$$

– $\ell_1$ norm in the time domain, i.e.,

$$\|\mathbf{z}\|_1.$$

We recall here that the proximal operator of the $\ell_1$ norm is defined by

$$\arg\min_{\mathbf{z}} \frac{1}{2} \|\mathbf{y} - \mathbf{z}\|_2^2 + \mu\|\mathbf{z}\|_1 = S_\mu(\mathbf{y}),$$

where

$$S_\mu(\mathbf{y}) = \text{sign}(\mathbf{y})(|\mathbf{y}| - \mu)_+,$$

where all the operations are meant element-wise and $(y)_+ = \max\{y, 0\}$. Moreover, the proximal operator of the $\ell_2$ norm can be computed by

$$\arg\min_{\mathbf{z}} \frac{1}{2} \|\mathbf{y} - \mathbf{z}\|_2^2 + \mu\|\mathbf{z}\|_2 = \left(1 - \frac{\mu}{\max\{\|\mathbf{z}\|_2, \mu\}}\right) \mathbf{z}.$$

We now give some details on how we implemented the two ADMM we are comparing with. For Algorithm 1 the only thing that we need to discuss is the solution of the least squares problem (5). To solve this we use the CGLS algorithm. This is an effective choice as the matrix $\begin{bmatrix} A \\ \sqrt{\rho} L \end{bmatrix}$ is well conditioned and, therefore, this method converges really fast; see, e.g., [5] for a discussion.

The implementation of the method in [42] is very similar to the one of Algorithm 2. The main difference is that the search subspace is not enlarged at each iteration. In this case to determine the initial subspace we use $\ell$ steps of the Golub-Kahan bidiagonalization method with initial vector $\mathbf{b}$. After $\ell$ steps are performed we obtain the decompositions

$$AV_\ell = U_{\ell+1}^T B_{\ell+1,\ell} \quad \text{and} \quad A^T U_\ell = V_\ell B_{\ell,\ell}^T,$$

where $B_{\ell+1,\ell} \in \mathbb{R}^{\ell+1\times\ell}$ is lower bi-diagonal, $B_{\ell,\ell} \in \mathbb{R}^{\ell\times\ell}$ is the leading block of $B_{\ell+1,\ell}$, and $U_{\ell+1} = [\mathbf{u}_1, \ldots, \mathbf{u}_{\ell+1}] \in \mathbb{R}^{m\times\ell+1}$ and $V_\ell = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell] \in \mathbb{R}^{n\times\ell}$

have orthonormal columns and $\mathbf{u}_{\ell+1} = \mathbf{b}/\left\|\mathbf{b}\right\|_2$. The columns of $V_\ell$ span the Krylov subspace $\mathcal{K}_\ell(A^TA, A^T\mathbf{b})$, where

$$\mathcal{K}_\ell(A^TA, A^T\mathbf{b}) = \operatorname{span}\{A^T\mathbf{b}, A^TAA^T\mathbf{b}, (A^TA)^2A^T\mathbf{b}, \dots, (A^TA)^{\ell-1}A^T\mathbf{b}\};$$

see, e.g., [31] for more details on Golub-Kahan bidiagonalization.

In [42] the authors look for $\mathbf{x}^{(k+1)}$ of the form

$$\mathbf{x}^{(k+1)} = V_\ell \mathbf{y}^{(k+1)}.$$

Plugging this relation and the decompositions above in (5) yields

$$\mathbf{y}^{(k+1)} = \arg\min_{\mathbf{y}} \left\| \begin{bmatrix} B_{\ell+1,\ell} \\ \sqrt{\rho}R_L \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{e}_1 \left\|\mathbf{b}\right\|_2 \\ \sqrt{\rho}(Q_L)^T\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right) \end{bmatrix} \right\|_2^2,$$

where $LV_\ell = Q_LR_L$ is the QR factorization of $LV_\ell$. We summarize these computations in Algorithm 3.

**Algorithm 3:** ADMM in KS

```
1  $\mathbf{z}^{(0)} = \mathbf{0}$;
2  $\boldsymbol{\lambda}^{(0)} = \mathbf{0}$;
3  Run $\ell$ steps of the Golub-Kahan bidiagonalization process obtaining $U_{\ell+1}$, $V_\ell$, and
   $B_{\ell+1,\ell}$;
4  Compute $LV_\ell$;
5  Compute the compact QR factorization $LV_\ell = QR$;
6  for $k = 0, 1, \dots$ do
7      $\mathbf{y}^{(k+1)} \arg\min_{\mathbf{y}} \left\| \begin{bmatrix} B_{\ell+1,\ell} \\ \sqrt{\rho}R_L \end{bmatrix} \mathbf{y} - \begin{bmatrix} \mathbf{e}_1 \left\|\mathbf{b}\right\|_2 \\ \sqrt{\rho}Q_L^T\left(\mathbf{z}^{(k)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right) \end{bmatrix} \right\|_2^2$;
8      $\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}} \frac{1}{2}\left\|\mathbf{z} - LV_\ell\mathbf{y}^{(k+1)} + \frac{\boldsymbol{\lambda}^{(k)}}{\rho}\right\|_2^2 + \frac{\mu}{\rho}\mathfrak{R}(\mathbf{z})$;
9      $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \rho(\mathbf{z}^{(k+1)} - LV_\ell\mathbf{y}^{(k+1)})$;
10     if $\frac{\left\|\mathbf{y}^{(k+1)} - \mathbf{y}^{(k)}\right\|_2}{\left\|\mathbf{y}^{(k)}\right\|_2} < \tau$ then
11         exit;
12 $\mathbf{x}^* = V_\ell\mathbf{y}^{(k+1)}$;
```

Finally, we describe how we set the parameters in the considered algorithms. We set $\rho = 10^{-1}$ for the image deblurring example and $\rho = 1$ for the other examples, $\tau = 10^{-4}$ for all methods, and we set a maximum number of iterations of 500. In Algorithm 2 we set $V_0 = A^T\mathbf{b}/\left\|A^T\mathbf{b}\right\|_2$ and in Algorithm 3 we set $\ell = 100$. The selection of the parameter $\mu$ is out of the scope of this paper and we do not dwell on it, we fix a "reasonable" value by hand in each performed test. We would like to stress that the reduced computational cost of our proposal makes possible to run a-posteriori choice rules for the selection of the regularization parameter $\mu$, e.g., Cross Validation [40], Generalized Cross Validation [30], and L-curve [33], efficiently.

We compare the methods in terms of speed and accuracy. To assess the accuracy of the methods we use the Relative Restoration Error (RRE), defined by

$$\mathrm{RRE}(\mathbf{x}) = \frac{\|\mathbf{x} - \mathbf{x}_{\mathrm{true}}\|_2}{\|\mathbf{x}_{\mathrm{true}}\|_2},$$

and the Structure SIMilariry index (SSIM). The definition of the SSIM is involved and we do not report it here, we recall that this index is a statistical measure that asses how similar two images are to the human eye, its value is between 0 and 1, where a low SSIM implies that two images are perceived as very different while a high value implies that they perceived as similar.

All the computations were performed on Matlab 2021b with 15 significant digits, running on a laptop with an AMD Ryzen 7 5800HS CPU with 16GB of RAM.

*Cameraman.* We first consider an image deblurring problem. We blur the cameraman image in Figure 1(a) with the non-symmetric PSF in Figure 1(b). To the blurred image we add white Gaussian noise $\boldsymbol{\eta}$ such that $\|\boldsymbol{\eta}\|_2 = 0.01\,\|A\mathbf{x}_{\mathrm{true}}\|_2$, i.e., we add 1% of white Gaussian noise, obtaining the blurred and noisy image in Figure 1(c). To simulate realistic data we crop the boundaries of the image and, since the image is generic, we impose reflexive boundary conditions. We fix $\mu = 10^{-1}$ for all methods and considered regularization operators.

Note that, since the PSF is not separable, the method described in [4] cannot be applied.

We consider in this example $\mathfrak{R}$ to be either the anisotropic TV or the sparsity inducing term in the framelet domain and report the obtained results in Table 1. We can observe that our proposed method and the one in [42] are less computationally demanding than the standard ADMM. However, the proposed method is able to achieve a higher level of accuracy than Algorithm 3 and obtains basically the same solution as the standard ADMM. One could improve the accuracy of Algorithm 3 by significantly enlarging the Krylov subspace, however, this would increase the computational cost to the point that it would be higher than the one of Algorithm 2 without obtaining the same accuracy. To illustrate this we report in Table 2 the results obtained with Algorithm 3 with larger values of $\ell$, namely 200 and 300. We only consider $L = L_G$ as the results with $L = L_F$ are similar. We can observe that with larger $\ell$ the results very slightly improve, however, Algorithm 3 is still not able to recover the exact solution produced by standard ADMM and its cost becomes larger than the one of our proposed method. Moreover, we would like to stress that the dimension of the Krylov subspace is not a parameter in our method, while it is in Algorithm 3. We would like to observe that, in this case, our method looks for a solution in a 208-dimensional subspace, while Algorithm 3 looks for a solution in a subspace of dimension 300, nevertheless, our method is able to determine an approximation of the exact image that is more accurate. We can deduce that the GKS contains more information than

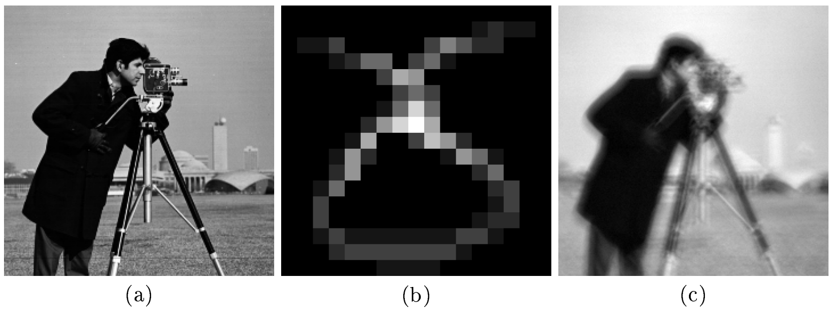

**Fig. 1** Cameraman test case: (a) true image ($239 \times 239$ pixels), (b) PSF ($17 \times 17$ pixels), (c) blurred and noisy image with 1% of white Gaussian noise.

the standard KS, this is due to the fact that our method incorporates into the construction of the search subspace the information collected throughout the iterations as well as the operator $L$. Finally, we compare our approach with the A-MM-GKS algorithm. We can observe that, the A-MM-GKS method provides a slighter more accurate reconstruction thant the other methods, however, the computational cost for $L = L_G$ is significantly higher than the one of Algorithm 2 even though less iterations are performed. This is due to the fact that, at each iteration, the computation of QR factorization of matrices of increasing dimension is required. Moreover, for $L = L_F$ the computational cost of A-MM-GKS becomes unbearable being almost 1000 times larger than the one of Algorithm 2.

We report the obtained reconstructions in Figure 2. From the visual inspection of these reconstructions we can confirm that the approximate solutions computed by Algorithm 2 are indistinguishable from the ones obtained using standard ADMM and that the ones obtained with Algorithm 3, although of high quality, are far from the exact ones.

*Satellite.* We now consider a different image deblurring example. We consider the satellite image in Figure 3(a) and we blur it with the Gaussian PSF in Figure 3(b). We add to the blurred image 10% of white Gaussian noise obtaining the blurred and noisy image in Figure 3(c). Similarly as in the previous case we do crop the boundaries to simulate realistic data. Since the image is an astronomical image we impose Dirichlet boundary conditions. In this case we fix $\mu = 10^{-1}$ as well. Similarly as before we consider $\mathfrak{R}$ to be either the anisotropic TV or the sparsity inducing term in the framelet domain, i.e., $\|L_F\mathbf{z}\|_1$.

Since the PSF is separable the method proposed in [4] coincides with Algorithm 2 when $\mathfrak{R}(z) = \|L_G\mathbf{z}\|_1$.

We report the computed results in Table 1. We can observe that the proposed method produces very similar reconstructions to the ones obtained by standard ADMM and that they are slightly more accurate, even though the difference is negligible. However, the computational cost of both Algorithms 2 and 3 is much lower than the one of standard ADMM. We can also see that, in this case for $L = L_G$ our proposed method is the fastest among

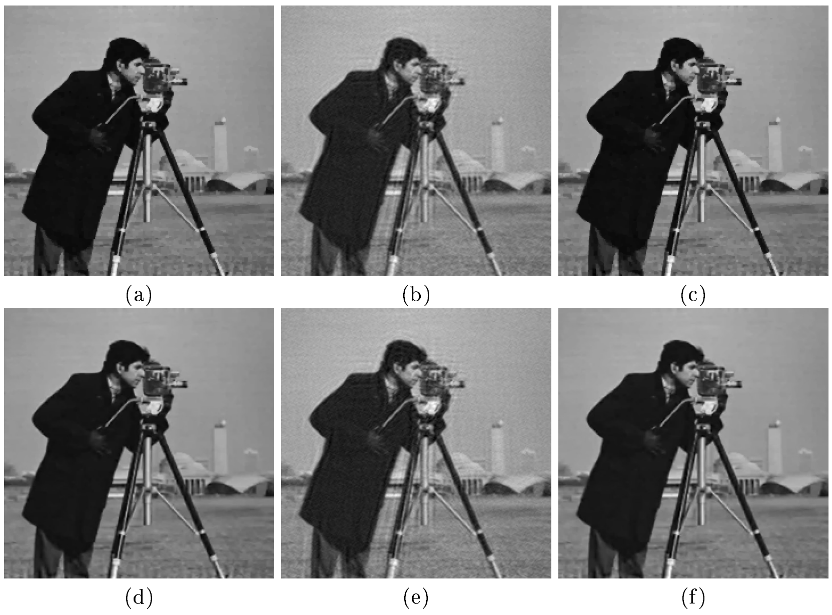

**Fig. 2** Cameraman test case, reconstructions: (a) Algorithm 1 with $L = L_G$, (b) Algorithm 3 with $L = L_G$, (c) Algorithm 2 with $L = L_G$, (d) Algorithm 1 with $L = L_F$, (e) Algorithm 3 with $L = L_F$, (f) Algorithm 2 with $L = L_F$.

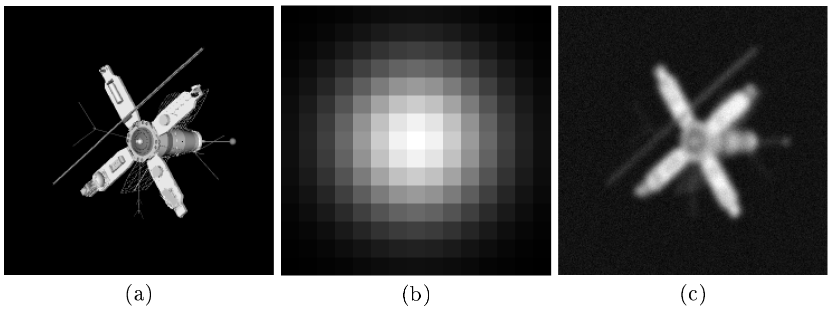

**Fig. 3** Satellite test case: (a) true image (241 × 241 pixels), (b) PSF (15 × 15 pixels), (c) blurred and noisy image with 10% of white Gaussian noise.

the three ADMMs, while for the case $L = L_F$ the computational cost of Algorithm 2 is higher than the one of Algorithm 3. Nevertheless, our proposed method is able to provide a more accurate reconstruction of the exact image. To achieve the same level of accuracy one would need to enlarge $\ell$ in Algorithm 3 to a point where this method would be no longer computationally attractive. Similarly as before the A-MM-GKS produces very similar reconstructions to the other methods in term of accuracy, however, the computational cost is significantly higher. These observations are confirmed by the visual inspection of the reconstructions in Figure 4.

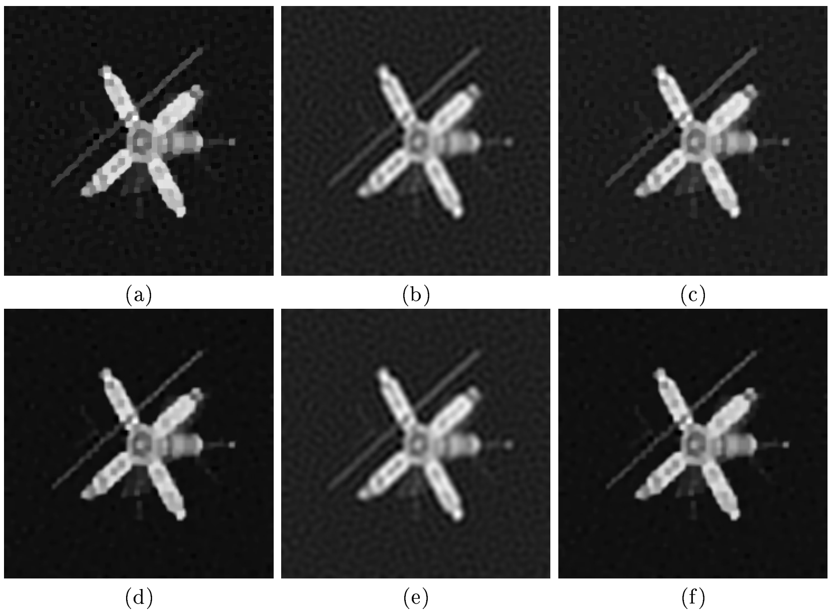

**Fig. 4** Satellite test case, reconstructions: (a) Algorithm 1 with $L = L_G$, (b) Algorithm 3 with $L = L_G$, (c) Algorithm 2 with $L = L_G$, (d) Algorithm 1 with $L = L_F$, (e) Algorithm 3 with $L = L_F$, (f) Algorithm 2 with $L = L_F$.

*Color image.* We consider now the deblurring of a color image. A color image in the RGB format can be seen as three two-dimensional images, one carrying the information on the red channel, one carrying the information of the green channel, and one carrying the information of the blue channel. We vectorize our image so that the three channels are stacked one above the other, i.e.,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_G \\ \mathbf{x}_B \end{bmatrix}.$$

Let $A_R$, $A_G$, and $A_B$ be three blurring matrices and $A_C \in \mathbb{R}^{3\times 3}$, our operator $A$ is of the form

$$A = (A_C \otimes I) \begin{bmatrix} A_R & & \\ & A_G & \\ & & A_B \end{bmatrix},$$

where $\otimes$ denotes the Kronecker product, $A_C \in \mathbb{R}^{3\times 3}$, and $I$ is the identity matrix, i.e., the blurred matrix is obtained by blurring each channel separately and then the three channels are mixed using the coefficient in $A_C$.

We consider the image in Fig. 5(a) and blur each channel with the same motion PSF in Fig. 5(b) and define

$$A_C = \frac{1}{10} \begin{bmatrix} 6 & 2 & 2 \\ 1 & 8 & 1 \\ 1 & 3 & 6 \end{bmatrix}.$$

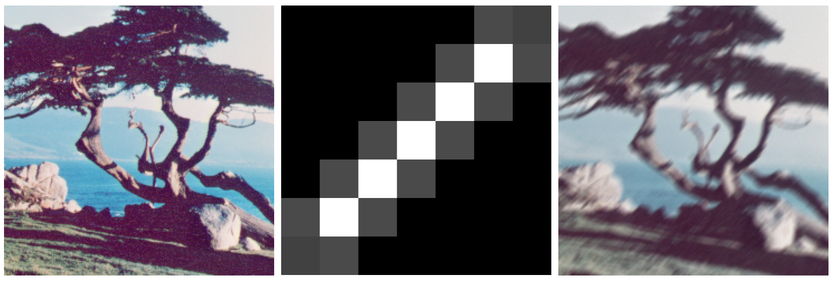

**Fig. 5** Color image test case: (a) true image (242 × 242 pixels), (b) PSF (7 × 7 pixels), (c) blurred and noisy image with 1% of white Gaussian noise.

We add 1% of white Gaussian noise and obtain the blurred and noisy image in Fig. 5(c). In this case we fix $\mu = 10^{-1}$. Let $\widetilde{L}_G$ and $\widetilde{L}_F$ denotes the regularization operator defined for two-dimensional images, in this case we set

$$L_G = \begin{bmatrix} \widetilde{L}_G & & \\ & \widetilde{L}_G & \\ & & \widetilde{L}_G \end{bmatrix} \quad \text{and} \quad L_F = \begin{bmatrix} \widetilde{L}_F & & \\ & \widetilde{L}_F & \\ & & \widetilde{L}_F \end{bmatrix}.$$

We would like to stress that, since $A$ is not a Kronecker product of two matrices, the method described in [4] cannot be used.

We can find the results obtained with the various methods in Table 1. We can observe that Algorithm 1 and Algorithm 2 produce very similar reconstructions for $L = L_F$ and our proposal outperforms the standard ADMM for $L = L_G$. In both cases the computational cost of our method is lower, especially if the cost-per-iteration is considered. Algorithm 3 is outperformed by our method in terms of accuracy, even though the computational cost is lower, even if the difference is not very pronounced.

Finally, we can observe that, for $L = L_G$, the A-MM-GKS required almost 1 hour of computational time to converge, while for $L = L_F$ the method had to be stopped as it did not converge even after 2 hours of computations.

We report some computed solution if Figure 6. We can observe that for $L = L_G$ the reconstruction computed by Algorithm 2 is significantly better than the ones obtained with the other two algorithms. Even though it is not shown here, the approximate solution computed by A-MM-GKS is extremely similar to the one obtained by our proposal, however, as we noted above, the computational cost for computing it is too high. Finally, we can observe that for both choices of $L$ the reconstruction obtained in Algorithm 3 is affected by some ringing effect.

*Tomography.* We consider now a tomography example. We consider the Shep-Logan phantom in Figure 7(a) and shine it with 181 rays at 36 equispaced angles between 0 and $\pi$. We add 1% of white Gaussian noise to the computed sinogram, obtaining the one in Figure 7(b). These computations were

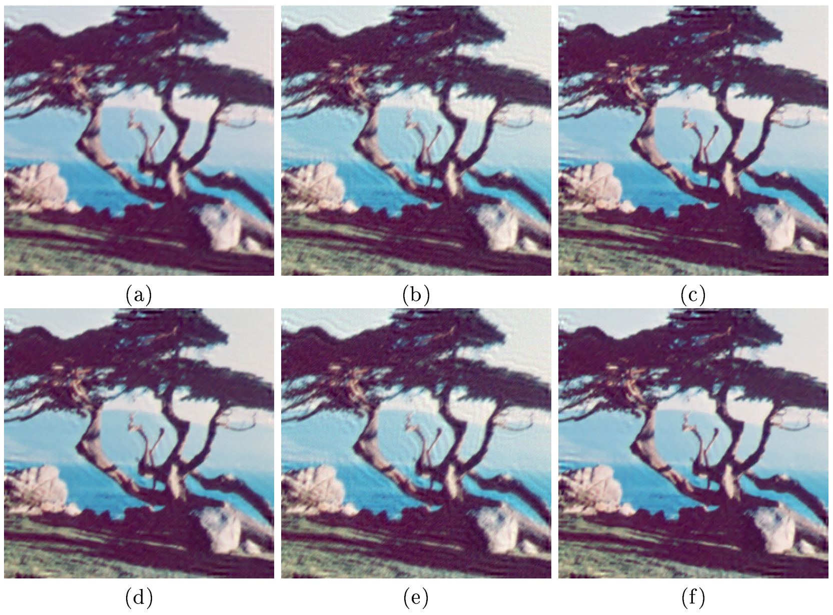

**Fig. 6** Color image test case, reconstructions: (a) Algorithm 1 with $L = L_G$, (b) Algorithm 3 with $L = L_G$, (c) Algorithm 2 with $L = L_G$, (d) Algorithm 1 with $L = L_F$, (e) Algorithm 3 with $L = L_F$, (f) Algorithm 2 with $L = L_F$.

performed using the IRtools toolbox [28]. We fix $\mu = 2$. Like in the first two examples we consider $\mathfrak{R}$ to be either the anisotropic TV or the sparsity inducing term in the framelet domain.

Note that, since $A$ is not a Kronecker product of two matrices the method proposed in [4] cannot be applied.

The obtained results are reported in Table 1. We can observe that, similarly to the previous case, the two Krylov methods are sensibly cheaper than the standard method and that Algorithm 3 is the fastest among the two. However, the approximate solutions obtained by Algorithm 3 are of poor quality when compared with the ones obtained by standard ADMM and our proposed method that are basically identical for both choices of $L$. Moreover, we can observe that, differently from the examples above, the A-MM-GKS is computationally competitive, as it converges in few iterations. However, the accuracy of the computed solution is usually lower than the one obtained with Algorithms 1 and 2. We would like here to stress that a more careful tuning of $\mu$ for $L = L_F$ could have produced better results with all the considered method, however, the main point of this work is to show that our algorithmic proposal can produce very similar accuracy of the computed solution compared to standard ADMM with a much lower computational cost. Therefore, we do not dwell further on the choice of $\mu$. We report the computed solutions in Figure 8. From the visual inspection of these reconstructions we can see

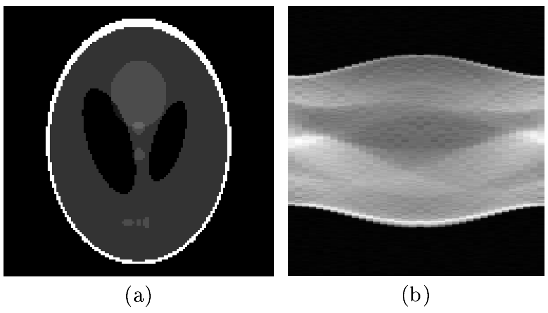

**Fig. 7** Tomography test case: (a) true image (128 × 128 pixels), (b) sinogram (36 angles and 181 rays) with 1% of white Gaussian noise.

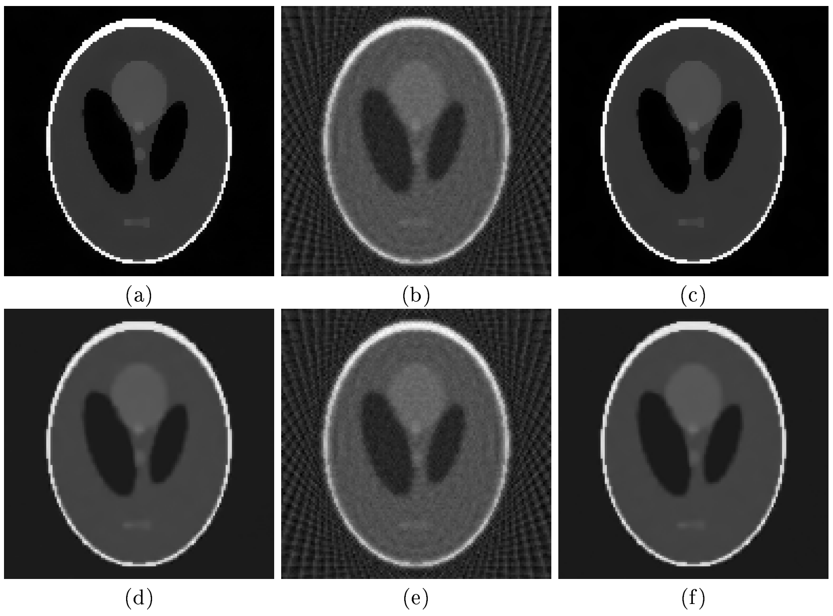

**Fig. 8** Tomography test case, reconstructions: (a) Algorithm 1 with $L = L_G$, (b) Algorithm 3 with $L = L_G$, (c) Algorithm 2 with $L = L_G$, (d) Algorithm 1 with $L = L_F$, (e) Algorithm 3 with $L = L_F$, (f) Algorithm 2 with $L = L_F$.

that the reconstructions obtained with ADMM and our proposed method are extremely accurate, especially for $L = L_G$.

*Seismic Tomography.* We consider now a seismic tomography problem. The matrix $A \in \mathbb{R}^{180000\times 90000}$ is extremely ill-conditioned and the data is corrupted by 0.5% of white Gaussian noise. As the problem is extremely large we reduce the maximum number of iteration to 200. We set $\mu = 10$. We report the exact solution and the measured data in Fig. 9. The computations are made using the IRtools toolbox [28].

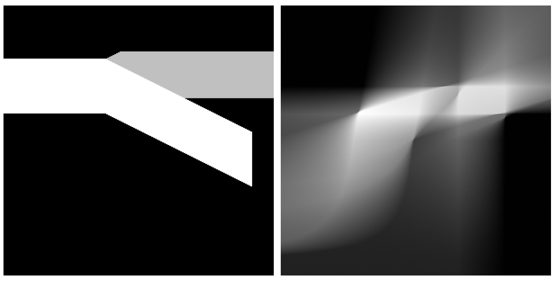

**Fig. 9** Seismic tomography test case: (a) true image ($300 \times 300$ pixels), (b) sinogram ($600 \times 300$) with 0.5% of white Gaussian noise.

In this case we consider the two TV terms for $\mathfrak{R}$, isotropic and anisotropic. For the isotropic TV note that the A-MM-GKS cannot be applied.

We report the obtained results in Table 1. We can observe that, similarly as above the projection the the Krylov subspaces greatly improves the computational cost of the methods. However, even though our proposed method is more expensive than the one in [42] it is able to provide more accurate approximate solutions with a reasonable computational effort. Note that, for the isotropic TV, the A-MM-GKS algorithm provides a slightly more accurate reconstruction than our proposal, however, the computational cost is extremely higher and does not justify, in our opinion, the slight improvement in quality. Some of the obtained reconstructions are reported in Fig. 10.

*Linear Regression.* For our last example we create a matrix $A \in \mathbb{R}^{1000\times 5000}$ where each entry is a realization of a Gaussian random variable with 0 mean and variance equal to 1. The obtained matrix is well-conditioned, in particular, its conditioning number is smaller than 3. We generate a sparse exact solution $\mathbf{x}_{\text{true}}$ where only 10 randomly selected entries are nonvanishing. These entries are random integers between 1 and 15. We generate the data by adding 5% of white Gaussian noise to $A\mathbf{x}_{\text{true}}$. We fix $\mu = 100$. As the signal is sparse in the canonical domain we only consider $\mathfrak{R}(\mathbf{z}) = \|\mathbf{z}\|_1$.

We report the computed results in Table 1, note that, in this case, the SSIM cannot be computed. We can observe that both the Krylov-based methods and the A-MM-GKS are sensibly faster than the standard ADMM. However, the one proposed in [42] fails to identify a meaningful approximation of the exact solution, while our proposed method determines a very accurate approximate solution. On the other hand, while the A-MMM-GKS is able to identify an accurate reconstruction of the exact solution its computational cost is higher than the one of our proposal, especially if the cost-per-iteration is taken into account.

**Table 1** Comparison of the results obtained in all the examples with the considered methods: Relative Restoration Error (RRE), Structure SIMilarity index (SSIM), CPU time in seconds, number of iterations performed.

| Example | $\mathfrak{R}(\mathbf{z})$ | Method | RRE | SSIM | CPU time | Iter. |
|---|---|---|---|---|---|---|
| Cameraman | $\|L_G\mathbf{z}\|_1$ | ADMM | 0.06822 | 0.87888 | 25.27 | 139 |
| | | ADMM in KS | 0.097358 | 0.75951 | 3.6344 | 40 |
| | | ADMM in GKS | 0.067995 | 0.8787 | 9.07 | 207 |
| | | A-MM-GKS | 0.066796 | 0.88053 | 33.603 | 129 |
| | $\|L_F\mathbf{z}\|_1$ | ADMM | 0.070249 | 0.88257 | 13.74 | 68 |
| | | ADMM in KS | 0.096965 | 0.75913 | 5.3126 | 25 |
| | | ADMM in GKS | 0.072066 | 0.88084 | 5.4576 | 94 |
| | | A-MM-GKS | 0.07125 | 0.87062 | 3409.5 | 500 |
| Satellite | $\|L_G\mathbf{z}\|_1$ | ADMM | 0.25355 | 0.63011 | 76.531 | 500 |
| | | ADMM in KS | 0.27394 | 0.50771 | 3.431 | 18 |
| | | ADMM in GKS | 0.25134 | 0.63118 | 1.9967 | 83 |
| | | A-MM-GKS | 0.2689 | 0.6133 | 196.58 | 283 |
| | $\|L_F\mathbf{z}\|_1$ | ADMM | 0.23635 | 0.78978 | 37.642 | 183 |
| | | ADMM in KS | 0.2743 | 0.55217 | 5.6708 | 19 |
| | | ADMM in GKS | 0.23618 | 0.78985 | 28.107 | 251 |
| | | A-MM-GKS | 0.23766 | 0.72616 | 51.689 | 82 |
| Color image | $\|L_G\mathbf{z}\|_1$ | ADMM | 0.084949 | 0.89375 | 42.539 | 102 |
| | | ADMM in KS | 0.08435 | 0.8854 | 12.982 | 130 |
| | | ADMM in GKS | 0.07122 | 0.91908 | 25.021 | 228 |
| | | A-MM-GKS | 0.071845 | 0.92022 | 2904.2 | 500 |
| | $\|L_F\mathbf{z}\|_1$ | ADMM | 0.076914 | 0.91347 | 96.815 | 131 |
| | | ADMM in KS | 0.08305 | 0.88645 | 26.064 | 96 |
| | | ADMM in GKS | 0.071594 | 0.92041 | 49.676 | 174 |
| | | A-MM-GKS | – | – | – | |
| Tomography | $\|L_G\mathbf{z}\|_1$ | ADMM | 0.059347 | 0.98108 | 22.324 | 500 |
| | | ADMM in KS | 0.37327 | 0.46252 | 1.023 | 78 |
| | | ADMM in GKS | 0.060076 | 0.98561 | 9.2513 | 444 |
| | | A-MM-GKS | 0.073317 | 0.97059 | 2.6686 | 53 |
| | $\|L_F\mathbf{z}\|_1$ | ADMM | 0.18735 | 0.95966 | 25.627 | 481 |
| | | ADMM in KS | 0.37384 | 0.46517 | 2.7472 | 150 |
| | | ADMM in GKS | 0.18519 | 0.96245 | 28.517 | 500 |
| | | A-MM-GKS | 0.2124 | 0.90675 | 4.195 | 46 |
| Seismic Tomography | $\|\mathbf{z}\|_{\text{I-TV}}$ | ADMM | 0.1161 | 0.48202 | 128.06 | 69 |
| | | ADMM in KS | 0.11357 | 0.45713 | 16.621 | 130 |
| | | ADMM in GKS | 0.058921 | 0.67843 | 107.91 | 500 |
| | $\|L_G\mathbf{z}\|_1$ | ADMM | 0.11585 | 0.48237 | 130 | 70 |
| | | ADMM in KS | 0.11427 | 0.45704 | 16.82 | 146 |
| | | ADMM in GKS | 0.055258 | 0.67215 | 108.97 | 500 |
| | | A-MM-GKS | 0.030537 | 0.88749 | 387.62 | 251 |
| Linear Regression | $\|\mathbf{z}\|_1$ | ADMM | 0.054968 | – | 7.106 | 147 |
| | | ADMM in KS | 0.89135 | – | 1.0226 | 327 |
| | | ADMM in GKS | 0.055126 | – | 0.9977 | 172 |
| | | A-MM-GKS | 0.052736 | – | 1.8939 | 44 |

**Table 2** Cameraman test case: Results obtained with Algorithm 3 with $\ell \in \{200, 300\}$ and $L = L_G$.

| $\ell$ | RRE | SSIM | CPU time | Iter. |
|---|---|---|---|---|
| 200 | 0.097344 | 0.75947 | 17.2 | 40 |
| 300 | 0.097343 | 0.75948 | 35.6 | 40 |

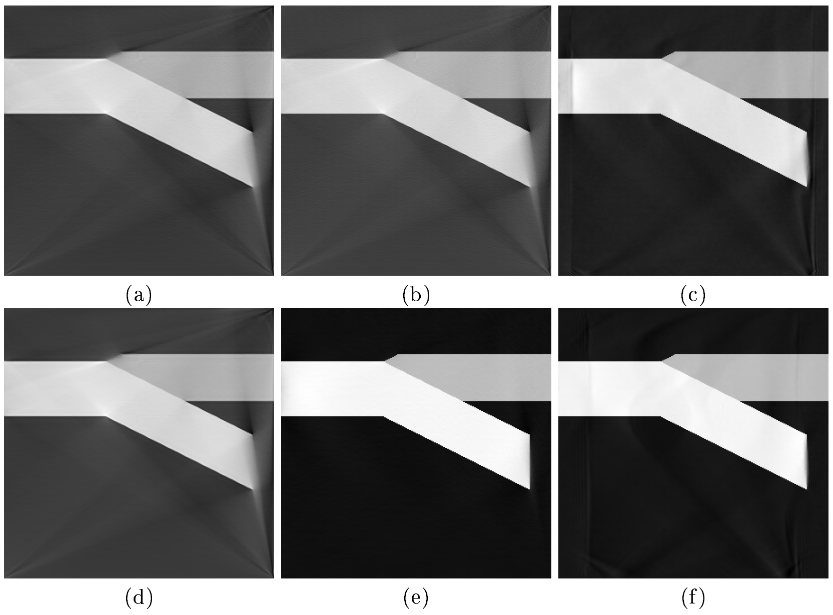

**Fig. 10** Seismic tomography test case, reconstructions: (a) Algorithm 1 with $\mathfrak{R}(\mathbf{z}) = \|\mathbf{z}\|_{\text{I}-\text{TV}}$, (b) Algorithm 3 with $\mathfrak{R}(\mathbf{z}) = \|\mathbf{z}\|_{\text{I}-\text{TV}}$, (c) Algorithm 2 with $\mathfrak{R}(\mathbf{z}) = \|\mathbf{z}\|_{\text{I}-\text{TV}}$, (d) Algorithm 1 with $\mathfrak{R}(\mathbf{z}) = \|L_G\mathbf{z}\|_1$, (e) A-MM-GKS with $\mathfrak{R}(\mathbf{z}) = \|L_G\mathbf{z}\|_1$, (f) Algorithm 2 with $\mathfrak{R}(\mathbf{z}) = \|L_G\mathbf{z}\|_1$.

## 5 Conclusions

In this paper we have introduced a numerically attractive implementation of the ADMM for the solution of ill-posed inverse problems. The proposed method combined ADMM with GKS to project the, possibly very large problem, into a subspace of fairly small dimension. Moreover, our method did not introduce any additional parameter as the dimension of the projection subspace is automatically determined by the algorithm itself. Matters of future research include the application of this method to non-linear problems and to bi-linear functions; see, e.g., [10] and computationally efficient strategy for the determination of the parameter $\mu$.

## Acknowledgments

The author would like to thank Prof. Marco Donatelli, Prof. Lothar Reichel, and the two anonimous referees for the precious comments that greatly improved the quality of this paper. The author is a member of the GNCS-INdAM. and his research is partially supported by the Regione Autonoma della Sardegna research project “Algorithms and Models for Imaging Science [AMIS]” (RASSR57257, intervento finanziato con risorse FSC 2014-2020 - Patto per lo Sviluppo della Regione Sardegna).

## References

1. Almeida, M.S., Figueiredo, M.: Deconvolving images with unknown boundaries using the alternating direction method of multipliers. IEEE Transactions on Image processing **22**(8), 3074–3086 (2013)
2. Almeida, M.S., Figueiredo, M.A.: Blind image deblurring with unknown boundaries using the alternating direction method of multipliers. In: 2013 IEEE International Conference on Image Processing, pp. 586–590. IEEE (2013)
3. Almeida, M.S., Figueiredo, M.A.: Frame-based image deblurring with unknown boundary conditions using the alternating direction method of multipliers. In: 2013 IEEE International Conference on Image Processing, pp. 582–585. IEEE (2013)
4. Bentbib, A.H., El Guide, M., Jbilou, K.: A generalized matrix Krylov subspace method for TV regularization. Journal of Computational and Applied Mathematics **373**, 112,405 (2020)
5. Björck, Å.: Numerical methods for least squares problems. Siam (1996)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
7. Buccini, A., De la Cruz Cabrera, O., Donatelli, M., Martinelli, A., Reichel, L.: Large-scale regression with non-convex loss and penalty. Applied Numerical Mathematics **157**, 590–601 (2020)
8. Buccini, A., De la Cruz Cabrera, O., Koukouvinos, C., Mitrouli, M., Reichel, L.: Variable selection in saturated and supersaturated designs via-minimization. Communications in Statistics-Simulation and Computation pp. 1–22 (2021)
9. Buccini, A., Dell'Acqua, P., Donatelli, M.: A general framework for ADMM acceleration. Numerical Algorithms **85**, 829–848 (2020)
10. Buccini, A., Donatelli, M., Ramlau, R.: A semiblind regularization algorithm for inverse problems with application to image deblurring. SIAM Journal on Scientific Computing **40**(1), A452–A483 (2018)
11. Buccini, A., Park, Y., Reichel, L.: Numerical aspects of the nonstationary modified linearized Bregman algorithm. Applied Mathematics and Computation **337**, 386–398 (2018)
12. Buccini, A., Pasha, M., Reichel, L.: Linearized Krylov subspace Bregman iteration with nonnegativity constraint. Numerical Algorithms **In press**, 1–24 (2020)
13. Buccini, A., Pasha, M., Reichel, L.: Modulus-based iterative methods for constrained $\ell_p - \ell_q$ minimization. Inverse Problems **36**(8), 084,001 (2020)
14. Buccini, A., Reichel, L.: An $\ell^2 - \ell^q$ regularization method for large discrete ill-posed problems. Journal of Scientific Computing **78**(3), 1526–1549 (2019)
15. Buccini, A., Reichel, L.: An $\ell_p - \ell_q$ minimization method with cross-validation for the restoration of impulse noise contaminated images. Journal of Computational and Applied Mathematics p. 112824 (2020)
16. Buccini, A., Reichel, L.: Generalized cross validation for $\ell^p - \ell^q$ minimization. Numerical Algorithms **In press**, 1–22 (2021)
17. Buzug, T.M.: Computed tomography. In: R. Kramme, K.P. Hoffmann, R.S. Pozos (eds.) Springer Handbook of Medical Technology, pp. 311–342. Springer, Berlin, Heidelberg (2011)
18. Cai, J.F., Chan, R.H., Shen, L., Shen, Z.: Simultaneously inpainting in image and transformed domains. Numerische Mathematik **112**(4), 509–533 (2009)
19. Cai, J.F., Osher, S., Shen, Z.: Linearized Bregman iterations for compressed sensing. Mathematics of computation **78**(267), 1515–1536 (2009)
20. Cai, J.F., Osher, S., Shen, Z.: Split Bregman methods and frame based image restoration. Multiscale modeling & simulation **8**(2), 337–369 (2010)
21. Cai, J.F., Osher, S., Shen, Z.: Split Bregman methods and frame based image restoration. Multiscale modeling & simulation **8**(2), 337–369 (2010)
22. Chan, R.H., Liang, H.X.: Half-quadratic algorithm for $\ell_p - \ell_q$ problems with applications to TV-$\ell_1$ image restoration and compressive sensing. In: Efficient algorithms for global optimization methods in computer vision, pp. 78–103. Springer (2014)

23. Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W.: Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. Mathematics of Computation **30**(136), 772–795 (1976)
24. Donatelli, M., Huckle, T., Mazza, M., Sesana, D.: Image deblurring by sparsity constraint on the fourier coefficients. Numerical Algorithms **72**(2), 341–361 (2016)
25. Eldén, L.: Algorithms for the regularization of ill-conditioned least squares problems. BIT Numerical Mathematics **17**(2), 134–145 (1977)
26. Engl, H.W., Hanke, M., Neubauer, A.: Regularization of inverse problems, vol. 375. Springer Science & Business Media (1996)
27. Estatico, C., Gratton, S., Lenti, F., Titley-Peloquin, D.: A conjugate gradient like method for p-norm minimization in functional spaces. Numerische Mathematik **137**(4), 895–922 (2017)
28. Gazzola, S., Hansen, P.C., Nagy, J.G.: IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems. Numerical Algorithms **81**(3), 773–811 (2019)
29. Goldstein, T., O'Donoghue, B., Setzer, S., Baraniuk, R.: Fast alternating direction optimization methods. SIAM Journal on Imaging Sciences **7**(3), 1588–1623 (2014)
30. Golub, G.H., Heath, M., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics **21**(2), 215–223 (1979)
31. Golub, G.H., Van Loan, C.F.: Matrix computations, vol. 3. JHU press (2013)
32. Hanke, M., Hansen, P.C.: Regularization methods for large-scale problems. Surv. Math. Ind **3**(4), 253–315 (1993)
33. Hansen, P.C.: Rank Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion. SIAM (1998)
34. Hansen, P.C., Jørgensen, J.S.: Air tools ii: algebraic iterative reconstruction methods, improved implementation. Numerical Algorithms **79**(1), 107–137 (2018)
35. Hansen, P.C., Nagy, J.G., O'Leary, D.P.: Deblurring Images: Matrices, Spectra, and Filtering. SIAM, Philadelphia (2006)
36. Huang, G., Lanza, A., Morigi, S., Reichel, L., Sgallari, F.: Majorization–minimization generalized Krylov subspace methods for $\ell_p - \ell_q$ optimization applied to image restoration. BIT Numerical Mathematics **57**(2), 351–378 (2017)
37. Lampe, J., Reichel, L., Voss, H.: Large-scale Tikhonov regularization via reduction by orthogonal projection. Linear algebra and its applications **436**(8), 2845–2865 (2012)
38. Lanza, A., Morigi, S., Reichel, L., Sgallari, F.: A generalized Krylov subspace method for $\ell_p - \ell_q$ minimization. SIAM Journal on Scientific Computing **37**(5), S30–S50 (2015)
39. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D: nonlinear phenomena **60**(1-4), 259–268 (1992)
40. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the royal statistical society: Series B (Methodological) **36**(2), 111–133 (1974)
41. Weisberg, S.: Applied linear regression, vol. 528. John Wiley & Sons (2005)
42. Yun, J.D., Yang, S.: ADMM in Krylov subspace and its application to total variation restoration of spatially variant blur. SIAM Journal on Imaging Sciences **10**(2), 484–507 (2017)