manuscript No.
(will be inserted by the editor)

A general framework for ADMM acceleration

**Alessandro Buccini · Pietro Dell'Acqua ·
Marco Donatelli**

Received: date / Accepted: date

Abstract The Alternating Direction Multipliers Method (ADMM) is a very popular algorithm for computing the solution of convex constrained minimization problems. Such problems are important from the application point of view, since they occur in many fields of science and engineering. ADMM is a powerful numerical tool, but unfortunately its main drawback is that it can exhibit slow convergence. Several approaches for its acceleration have been proposed in the literature and in this paper we present a new general framework devoted to this aim. In particular, we describe an algorithmic framework that makes possible the application of any acceleration step while still having the guarantee of convergence. This result is achieved thanks to a guard condition that ensures the monotonic decrease of the combined residual. The proposed strategy is applied to image deblurring problems. Several acceleration techniques are compared; to the best of our knowledge, some of them are investigated for the first time in connection with ADMM. Numerical results show that the proposed framework leads to a faster convergence with respect to other acceleration strategies recently introduced for ADMM.

Keywords Alternating Direction Multipliers Method (ADMM) · acceleration techniques

Alessandro Buccini
Department of Mathematics and Computer Science
University of Cagliari
Via Ospedale 72, 09124, Cagliari (Italy)
E-mail: alessandro.buccini@unica.it

Pietro Dell'Acqua
Dipartimento di Ingegneria, Scienze Informatiche e Matematica
Università degli Studi dell'Aquila
Via Vetoio, Loc. Coppito, 67010 L'Aquila (Italy)
E-mail: pietro.dellacqua@gmail.com

Marco Donatelli
Dipartimento di Scienza e Alta Tecnologia
Università degli Studi dell'Insubria
Via Valleggio 11, 22100 Como (Italy)
E-mail: marco.donatelli@uninsubria.it

1 Introduction

The task of minimizing a convex function $f(x)$ subject to some constraints is a well-known problem which arises in many fields of science and engineering. Consider the following minimization problem

$$\begin{aligned} &\min_{x,y} f(x) + g(y) \\ &\text{s.t. } Ax + By = c, \end{aligned} \tag{1}$$

where f and g are convex functions. A very popular method for computing an approximated solution of (1) is the Alternating Direction Multipliers Method (ADMM) [1].

The ADMM can be easily implemented, however, its main drawback is that it can exhibit slow convergence. Thus, several approaches have been proposed to accelerate it; see, e.g., [2,3]. In [2] the acceleration is obtained by an extrapolation strategy that involves a constant acceleration factor that, for ensuring convergence, has to be smaller than 1/3. In [3] the well-known FISTA approach [4,5] is employed to speed up the ADMM iterations and a restart condition is introduced to guarantee the convergence of the accelerated algorithm.

In this work, we would like to present a general framework for the acceleration of ADMM. In particular, we construct an algorithm that let us insert any acceleration step while ensuring convergence, thanks to the implementation of an appropriate guard condition. We would like to stress that our main goal is to develop a very general framework in which any acceleration strategy can be inserted and tested. Providing a theoretical analysis of the achieved order of convergence when implementing the after-mentioned acceleration strategies is out of the scope of this paper. On the other hand, the proposed framework ensures that the accelerated ADMM obtained still converges and that it is not slower than the non-accelerated version. As acceleration strategies, we adopt both the approaches proposed in [2, 3] as well as others developed in the context of image restoration. We consider, in particular, the automatic acceleration [6] and the ν acceleration [7], which is inspired by ν-method [8], that is an accelerated variant of Landweber method [9]. Moreover, we investigate the Simplified Topological ε-Algorithm proposed in [10]. Finally, we develop a new acceleration technique, that we call Geometric Series Acceleration, which proves to be very effective in our numerical results.

For our numerical experiments, we consider image deblurring with shift invariant blur as an example of (1). ADMM has been applied with success in several image deblurring algorithms, see, e.g., [11,12,13]. Image restoration represents an important application in which, especially when the noise affecting the data is small, ADMM can exhibit slow convergence. The observed image u can be modeled by

$$u = K\bar{x} + \delta,$$

where K is the blurring operator, i.e., a structured matrix (whose structure depends on the boundary conditions imposed in the discretization), $\bar{x}$ is the true object and δ is the noise, which is usually present in any detection process. The goal of image deblurring is to recover an approximation of $\bar{x}$ from u, knowing K.

We employ the Total Variation (TV) approach [14] and we consider the so called L2-TV model, that is a well-established model to restore blurred images corrupted by white Gaussian noise. However, other models like for instance $\ell^p - \ell^q$ [15] or framelets [16] can be considered as well. The L2-TV model is defined by the following minimization problem

$$\arg\min_x \frac{\mu}{2} \|Kx - u\|_2^2 + \|\nabla x\|_{2,1}. \tag{2}$$

Here ∇ denotes the discrete forward gradient operator (in both horizontal and vertical direction) defined by

$$(\nabla a)_{i,j} = (a^x_{i,j}, a^y_{i,j}) = (a_{i+1,j} - a_{i,j}, a_{i,j+1} - a_{i,j}),$$

whereas the norm $\|\cdot\|_{2,1}$ is defined by

$$\|a\|_{2,1} = \sum_{i,j} \sqrt{(a^x_{i,j})^2 + (a^y_{i,j})^2},$$

and $\mu > 0$ is the regularization parameter. For the sake of simplicity, we impose periodic boundary conditions (see, e.g., [17] for more information on boundary conditions), thus K is a Block Circulant with Circulant Blocks (BCCB) matrix and so all the computations involving K, even the inversion, can be performed efficiently by means of the Fast Fourier Transform (FFT).

We now reformulate (2) in form (1). Let $f(x) = \frac{\mu}{2} \|Kx - u\|_2^2$, $g(y) = \|y\|_{2,1}$, $A = \nabla$, $B = -I$, and $c = 0$, then we obtain the formulation

$$\begin{aligned} &\arg\min_{x,y} \frac{\mu}{2} \|Kx - u\|_2^2 + \|y\|_{2,1} \\ &\text{s.t. } \nabla x - y = 0. \end{aligned}$$

The total variation approach has the ability to preserve edges in the image due to the regularization properties of the gradient operator, while not amplifying the noise. The regularization parameter μ plays an important role, since it is used to control the relative weights of the data fitting term and regularization term. If it is too large, the regularized solution is under-smoothed and some noise may be reconstructed. On the other hand, if it is too small, the regularized solution does not fit the given data properly. Usually, μ is larger for problems with low levels of noise and smaller when the presence of noise is more significant. However, in this work we do not analyze criteria for choosing μ and we do not investigate the issues that the choice of such parameter entails, but we set it by trial and error, seeking a value close to the optimal one.

The present paper is structured as follows. In Section 2 we describe the general algorithmic framework, while different acceleration strategies that can be successfully employed for accelerating ADMM are presented in Section 3. In Section 4 we report results relative to some numerical experiments in image restoration, which show the efficacy of the proposed framework. Finally, in Section 5 we draw some conclusions.

2 General framework

To fix our notation, we first describe the standard ADMM. Then we propose the general algorithmic framework we would like to construct and show that this accelerated ADMM converges.

2.1 Standard ADMM

Consider the minimization problem (1), since it is a constrained minimization problem, we can write the related augmented Lagrangian

$$\mathcal{L}(x,y,\lambda) = f(x) + g(y) - \langle \lambda, Ax + By - c \rangle + \frac{\omega}{2} \|c - Ax - By\|^2 ,$$

where $\omega > 0$ is a fixed constant and λ is the so-called Lagrangian multiplier.

The ADMM algorithm is obtained by combining an alternating minimization of $\mathcal{L}(x,y,\lambda)$ with respect to x and y with an update of the Lagrangian multiplier. The updating of the Lagrangian multiplier is obtained by solving

$$\lambda_{k+1} = \arg\max_{\lambda} \langle \lambda, c - Ax_{k+1} - By_{k+1} \rangle - \frac{1}{2\omega} \|\lambda - \lambda_k\|^2 ,$$

i.e. by looking for the λ which penalizes the most the distance between c and $Ax_{k+1} + By_{k+1}$ while being near to λ_k. We summarize the ADMM algorithm in Algorithm 1

Algorithm 1 (ADMM) *Consider the minimization problem* (1)*. Let y_0 and λ_0 be initial guesses for y and λ, respectively. Fix $\omega > 0$.*

```
for $k = 0, 1, 2, \dots$ do
    $x_{k+1} = \arg\min_x f(x) - \langle \lambda_k, Ax \rangle + \frac{\omega}{2} \|c - Ax - By_k\|^2$;
    $y_{k+1} = \arg\min_y g(y) - \langle \lambda_k, By \rangle + \frac{\omega}{2} \|c - Ax_{k+1} - By\|^2$;
    $\lambda_{k+1} = \lambda_k - \omega(Ax_{k+1} + By_{k+1} - c)$;
end
```

It is possible to prove that, denoting by (x^*, y^*) a solution of (1), it holds that

$$(x_k, y_k) \to (x^*, y^*) \quad \text{as } k \to \infty.$$

Let us now define the *combined residual* as

$$\gamma_k = \omega^{-1} \|\lambda_k - \lambda_{k-1}\|^2 + \omega \|B(y_k - y_{k-1})\|^2 , \qquad k > 0. \tag{3}$$

Following [3] we define two residuals

(i) *Primal Residual.* $r_k \|c - (Ax_k + By_k)\| \to 0$;
(ii) *Dual Residual.* $d_k = \left\|A^t B(y_k - y_{k-1})\right\|$.

In [3, Section 1.4] is shown that the limit point of the iterates generated by Algorithm 1 satisfy the KKT conditions for problem (1) if and only if $r_k \to 0$ and $d_k \to 0$ as $k \to 0$. From this it is trivial to see that

Lemma 1 *Let γ_k denotes the combined residual defined in* (3)*. Let x_k, y_k denotes the iterates generated by Algorithm 1. Assume that B is of full rank, then it holds that*

$$(x_k, y_k) \to (x^*, y^*) \iff \gamma_k \to 0 \quad \text{as } k \to \infty.$$

Proof The proof follows from the observation that

$$0 \le \left\|A^t B(y_k - y_{k-1})\right\| \le \|A\| \, \|B(y_k - y_{k-1})\| .$$

The following result is proved in [18, Theorem 4.1].

Lemma 2 *Let x_k, y_k, and λ_k be the iterations generated by Algorithm 1. Moreover, let γ_k be the combined residual defined in* (3)*. Then it holds*

$$\gamma_{k+1} \le \gamma_k.$$

2.2 Accelerated ADMM

Starting from the result in Lemma 2, we want to construct an accelerated version of ADMM such that the monotonic decrease of the combined residual is ensured. We obtain this monotonic convergence thanks to the introduction of a guard condition. Given the vectors associated to an accelerated iteration, if such condition is satisfied, the method proceeds to the next iteration, otherwise it replaces the vectors with those associated to an iteration of standard ADMM. Thus, for any accelerated version of ADMM, this ensures that the combined residual decreases in the way imposed by the guard condition or, in the worst case, as standard ADMM.

We highlight that this is not just a theoretical bound, useful for guaranteeing the convergence of the method. On the contrary, it can be exploited in practice and, once parameters of the guard condition have been suitably set, it allows to get fast convergent iterations, even when an acceleration directly applied to ADMM would give rise to semi-convergence or divergence phenomena. This will be evident in the numerical results reported in Section 4.

We now introduce the notation we are going to use in the following in order to provide a unified and simplified description of the accelerated ADMM. Let v be a vector that we would like to compute, we will denote by v_k the approximation of v computed at the k-th iteration of the algorithm at hand. Moreover, we will use the symbol $\bar{v}_k$ if the vector is computed by standard ADMM, while we will use the symbol $\hat{v}_k$ if the vector is computed by an acceleration technique.

Therefore, a step of a generic acceleration strategy (based on past ADMM iterations) will be denoted by

$$\hat{v}_{k+1} = \mathsf{acc}(\bar{v}_{k+1}, \bar{v}_k, \ldots). \tag{4}$$

Some acceleration strategies might also use $\hat{v}_k, \hat{v}_{k-1}, \ldots$ for computing $\hat{v}_{k+1}$, but for simplicity we prefer to employ also in such cases the notation introduced in (4).

We would like to stress that usually an acceleration step is based on $\{v_k\}$ instead of $\{\bar{v}_k\}$. However, similarly to [3], we consider the acceleration as defined in (4) since we have experimentally observed that this approach gives rise to a larger speed up in the convergence behaviour with respect to the classical one based on iterates $\{v_k\}$. However, it also entails a higher risk of instability and divergence phenomena, so it is necessary to introduce a control condition to avoid that these problems appear, compromising the performance of accelerated algorithms.

Note that computing $\hat{v}_{k+1}$ by equation (4) has usually a low computational effort since it involves only simple operations on vectors.

We first recall the Fast ADMM with restart presented in [3].

Algorithm 2 (Fast ADMM with restart, [3]) *Consider the minimization problem* (1). *Let* $\bar{y}_0 = \hat{y}_1$ *and* $\bar{\lambda}_0 = \hat{\lambda}_1$ *be initial guesses for* y *and* λ*, respectively. Fix* $\alpha_1 = 1$, $\omega > 0$, $\eta \in (0, 1)$ *and* $\bar{\gamma}_0 > 0$.

```
for $k = 0, 1, 2, \dots$ do
    $x_{k+1} = \arg\min_x f(x) - \left\langle \hat{\lambda}_{k+1}, Ax \right\rangle + \frac{\omega}{2} \|c - Ax - B\hat{y}_{k+1}\|^2$;
    $\bar{y}_{k+1} = \arg\min_y g(y) - \left\langle \hat{\lambda}_{k+1}, By \right\rangle + \frac{\omega}{2} \|c - Ax_{k+1} - By\|^2$;
    $\bar{\lambda}_{k+1} = \hat{\lambda}_{k+1} - \omega(Ax_{k+1} + B\bar{y}_{k+1} - c)$;
    $\bar{\gamma}_{k+1} = \omega^{-1} \left\|\bar{\lambda}_{k+1} - \hat{\lambda}_{k+1}\right\|^2 + \omega \|B(\bar{y}_{k+1} - \hat{y}_{k+1})\|^2$;
    % Begin of restart condition
    if $\bar{\gamma}_{k+1} < \eta\bar{\gamma}_k$ then
        % Begin of acceleration steps
        $\alpha_{k+2} = \dfrac{1 + \sqrt{1 + 4\alpha_{k+1}^2}}{2}$;
        $\hat{y}_{k+2} = \bar{y}_{k+1} + \dfrac{\alpha_{k+1} - 1}{\alpha_{k+2}}(\bar{y}_{k+1} - \bar{y}_k)$;
        $\hat{\lambda}_{k+2} = \bar{\lambda}_{k+1} + \dfrac{\alpha_{k+1} - 1}{\alpha_{k+2}}(\bar{\lambda}_{k+1} - \bar{\lambda}_k)$;
        % End of acceleration steps
    else
        $\alpha_{k+2} = 1$;
        $\hat{y}_{k+2} = \bar{y}_k$;
        $\hat{\lambda}_{k+2} = \bar{\lambda}_k$;
        $\bar{\gamma}_{k+1} \leftarrow \eta^{-1}\bar{\gamma}_k$;
    end
    % End of restart condition
    $y_{k+1} = \bar{y}_{k+1}$;
    $\lambda_{k+1} = \bar{\lambda}_{k+1}$;
    $\gamma_{k+1} = \omega^{-1} \|\lambda_{k+1} - \lambda_k\|^2 + \omega \|B(y_{k+1} - y_k)\|^2$;
end
```

In [3] it is proven that, thanks to the restart condition, the convergence to zero of the modified combined residual $\bar{\gamma}_{k+1}$ is guaranteed. Because it is desirable to restart the method as infrequently as possible, the authors recommend to set η close to 1, in particular $\eta = 0.999$.

Observe that, since x_{k+1} is computed directly from y_k and λ_k, no acceleration step is employed on x_{k+1}. Indeed, when x_{k+1} is computed, no information on the previous iterate x_k is retained, so any acceleration step would be ineffective.

Following a similar approach, we can formulate our accelerated ADMM algorithm with guard condition and then proving its convergence.

Algorithm 3 (Accelerated ADMM with guard condition) *Consider the minimization problem* (1). *Let* y_0 *and* λ_0 *be initial guesses for* y *and* λ*, respectively. Fix*

$\omega > 0$, $\eta \in (0,1)$ and $\gamma_0 > 0$.

```
for $k = 0, 1, 2, \ldots$ do
    $x_{k+1} = \arg\min_x f(x) - \langle \lambda_k, Ax \rangle + \frac{\omega}{2} \|c - Ax - By_k\|^2$;
    $\bar{y}_{k+1} = \arg\min_y g(y) - \langle \lambda_k, By \rangle + \frac{\omega}{2} \|c - Ax_{k+1} - By\|^2$;
    $\bar{\lambda}_{k+1} = \lambda_k - \omega(Ax_{k+1} + B\bar{y}_{k+1} - c)$;
    % Begin of acceleration steps
    $\hat{y}_{k+1} = acc(\bar{y}_{k+1}, \bar{y}_k, \ldots)$;
    $\hat{\lambda}_{k+1} = acc(\bar{\lambda}_{k+1}, \bar{\lambda}_k, \ldots)$;
    % End of acceleration steps
    $\gamma_{k+1} = \omega^{-1} \left\| \hat{\lambda}_{k+1} - \lambda_k \right\|^2 + \omega \|B(\hat{y}_{k+1} - y_k)\|^2$;
    % Begin of guard condition
    if $\gamma_{k+1} < \gamma_0 \eta^{k+1}$ then
        $y_{k+1} = \hat{y}_{k+1}$;
        $\lambda_{k+1} = \hat{\lambda}_{k+1}$;
    else
        $y_{k+1} = \bar{y}_{k+1}$;
        $\lambda_{k+1} = \bar{\lambda}_{k+1}$;
        $\gamma_{k+1} = \omega^{-1} \|\lambda_{k+1} - \lambda_k\|^2 + \omega \|B(y_{k+1} - y_k)\|^2$;
    end
    % End of guard condition
end
```

Both the algorithms are constructed with the aim of guaranteeing convergence of the method. However there are few important differences between them. Indeed, while in Algorithm 3 acceleration steps and guard condition are separated (first it is performed acceleration and then the condition is checked), in Algorithm 2 they are mixed together. Moreover, while in Algorithm 3 the guard condition is based on the combined residual, in Algorithm 2 the restart condition is based on a modified version of it. The consequence of these facts is that Algorithm 3 represents an improvement and a generalization of Algorithm 2, since it is able to employ any acceleration in its framework and to have a better control on the decrease of the combined residual.

We can now present our main theoretical result concerning the convergence of Algorithm 3.

Theorem 1 *Let x_k, y_k, and λ_k be the iterations generated by Algorithm 3, then the iterates converge in the sense that*

$$\gamma_k \to 0 \quad as \quad k \to \infty,$$

where γ_k denotes the combined residual defined in (3).

Proof This proof is inspired by the proof of [3, Theorem 3]. There are two possible cases: either the acceleration technique has been applied a finite number of times, or it has been applied an infinite number of times.

If we are in the first case, this means that, starting from a certain iteration, Algorithm 3 behaved like the standard ADMM algorithm and thus, by the standard ADMM theory, we know that $\gamma_k \to 0$ as $k \to \infty$.

On the other hand, if the acceleration technique has been applied for an infinite number of iterations it holds, in force of Lemma 2, that for all k

$$\gamma_k \leq \eta^{\hat{k}} \gamma_0,$$

where by $\eta^{\hat{k}}$ we denote the number of times that the acceleration technique has been applied before iteration k. Since $0 < \eta < 1$ we have that

$$0 \leq \gamma_k \leq \eta^{\hat{k}} \gamma_0 \to 0.$$

Therefore, the thesis holds true.

We would like to remark that Theorem 1 does not assure in general the algorithm convergence to a solution of (1). However, as we have seen in Lemma 1, in the non-accelerated case this type of convergence is strictly related to the residual and dual variable convergence. Moreover, numerical results reported in Section 4 show that the introduction of the acceleration does not affect the limit point of the algorithm, but only its convergence speed. Deeper convergence results may be obtained by analyzing the different acceleration techniques. However, for the sake of simplicity, here we report only this general result.

In order to use in effective way the guard condition, it is necessary to suitably set the parameters γ_0 and η. In this respect, for the former we propose the following heuristic criterion: $\gamma_0 = \chi \dot{\gamma}_1$, where $\chi > 1$ is a constant (we take $\chi = 2$) and $\dot{\gamma}_1$ is the value of the combined residual after a step of standard ADMM. Regarding η, we set it equal to 0.85.

2.3 Restarted ADMM

With a slight modification, we can obtain a restarted version of Algorithm 3. In particular, we present θ-Restarted algorithm, which consists in applying acceleration every θ steps of standard ADMM.

Algorithm 4 (θ-Restarted ADMM with guard condition) *Consider the minimization problem* (1)*. Let y_0 and λ_0 be initial guesses for y and λ, respectively. Fix*

$\omega > 0$, $\eta \in (0,1)$ and $\gamma_0 > 0$.

```
for $k = 0,1,2,\dots$ do
    $x_{k+1} = \arg\min_x f(x) - \langle\lambda_k, Ax\rangle + \frac{\omega}{2}\|c - Ax - By_k\|^2$;
    $\bar{y}_{k+1} = \arg\min_y g(y) - \langle\lambda_k, By\rangle + \frac{\omega}{2}\|c - Ax_{k+1} - By\|^2$;
    $\bar{\lambda}_{k+1} = \lambda_k - \omega(Ax_{k+1} + B\bar{y}_{k+1} - c)$;
    if $k+1 \equiv 0 \mod \theta$ then
        % Begin of acceleration steps
        $\hat{y}_{k+1} = acc(\bar{y}_{k+1}, \bar{y}_k, \dots)$;
        $\hat{\lambda}_{k+1} = acc(\bar{\lambda}_{k+1}, \bar{\lambda}_k, \dots)$;
        % End of acceleration steps
        $\gamma_{k+1} = \omega^{-1}\left\|\hat{\lambda}_{k+1} - \lambda_k\right\|^2 + \omega\|B(\hat{y}_{k+1} - y_k)\|^2$;
        % Begin of guard condition
        if $\gamma_{k+1} < \gamma_0\eta^{k+1}$ then
            $y_{k+1} = \hat{y}_{k+1}$;
            $\lambda_{k+1} = \hat{\lambda}_{k+1}$;
        else
            $y_{k+1} = \bar{y}_{k+1}$;
            $\lambda_{k+1} = \bar{\lambda}_{k+1}$;
            $\gamma_{k+1} = \omega^{-1}\|\lambda_{k+1} - \lambda_k\|^2 + \omega\|B(y_{k+1} - y_k)\|^2$;
        end
        % End of guard condition
    else
        $y_{k+1} = \bar{y}_{k+1}$;
        $\lambda_{k+1} = \bar{\lambda}_{k+1}$;
        $\gamma_{k+1} = \omega^{-1}\|\lambda_{k+1} - \lambda_k\|^2 + \omega\|B(y_{k+1} - y_k)\|^2$;
    end
end
```

Clearly, if we set $\theta = 1$, we recover Algorithm 3. Moreover, theoretical results proven for such algorithm can be easily extended to Algorithm 4. We stress that this restarted approach can be successfully employed in particular with some acceleration techniques like Simplified Topological ε-Algorithm (STEA) [10], as illustrated in [19]. About the setting of guard condition, in case of Algorithm 4 we choose the same value for η ($\eta = 0.85$), but a larger value for χ ($\chi = 50$).

Once set the guard condition in a suitable way, the proposed framework can be applied in combination with any acceleration technique. In the next section, in order to give some meaningful examples, we briefly recall some acceleration strategies that occupy an important place in the literature.

3 Acceleration techniques

We now describe several strategies for speeding up sequences of vectors, which can be employed in the acceleration steps of Algorithms 3 and 4. Every acceleration technique obtains $\hat{v}_k$ by extrapolation from the previous iterates. Hence, most of them can be defined by this formula

$$\hat{v}_k = \bar{v}_k + \alpha_k(\bar{v}_k - \bar{v}_{k-1}) \tag{5}$$

and by specifying α_k. We remind that, according to the notation introduced before, v denotes either y or λ, while vector x is not directly involved in the acceleration process.

Stationary acceleration In [2] the most simple approach is employed, that is to pick in (5) a constant value $\alpha_k = \alpha$. In particular, the authors give a convergence proof for $\alpha < 1/3$.

Nesterov acceleration In [3] the following approach is utilized, defined by

$$\begin{aligned}\beta_k &= \frac{1+\sqrt{1+4\beta_{k-1}^2}}{2},\\ \alpha_k &= \frac{\beta_{k-1}-1}{\beta_k},\end{aligned}$$

with $\beta_0 = 1$, leading to Algorithm 2. About this acceleration, we recall that FISTA was developed in [4], while a similar algorithm was also developed by Nesterov in [5], and is essentially a fast version of Iterative Soft-Thresholding Algorithm (ISTA).

Automatic acceleration We consider also two popular acceleration techniques used to speed up iterative methods in the context of image restoration. The first is a form of vector extrapolation that predicts subsequent points based on previous points and is known as automatic acceleration [6], since it does not require the setting of any parameter. It is defined by

$$\begin{aligned}g_{k-1} &= \bar{v}_k - \hat{v}_{k-1},\\ g_{k-2} &= \bar{v}_{k-1} - \hat{v}_{k-2},\\ \alpha_k &= \frac{g_{k-1}^T g_{k-2}}{g_{k-2}^T g_{k-2}},\end{aligned}$$

where $\alpha_1 = 0$, $\alpha_2 = 0$ and $0 \leq \alpha_k \leq 1$, $\forall k$. As reported in [6], this acceleration can be applied when the restoration changes slowly between each iteration and also if the algorithm is insensitive to the changes introduced. Therefore, it can be used to accelerate several slow-converging methods, including ADMM analyzed in this work.

ν acceleration The second is ν acceleration, which does not involve (5), but is based on the formula

$$\hat{v}_k = \mu_k \bar{v}_k + (1-\mu_k)\bar{v}_{k-1} + \rho_k(\bar{v}_k - \bar{v}_{k-1}),$$

where ν is a user-defined constant, while μ_k and ρ_k are two parameters that depends on k and ν:

$$\begin{aligned}\mu_k &= 1 + \frac{(k-1)(2k-3)(2k+2\nu-1)}{(k+2\nu-1)(2k+4\nu-1)(2k+2\nu-3)},\\ \rho_k &= \frac{4(2k+2\nu-1)(k+\nu-1)}{(k+2\nu-1)(2k+4\nu-1)}.\end{aligned}$$

This strategy is inspired by the ν-method [8], that is a semi-iterative method able to speed up the classical Landweber method [9], and by ν acceleration presented in [7], which extends the same idea to statistical methods like Richardson-Lucy algorithm. We highlight that, while ν-method has a strong theoretical background, ν acceleration is simply an update formula without theoretical foundations. Nevertheless, thanks to the guard condition, we have convergence guarantee also for this heuristic acceleration. As remarked in [7], the choice of parameter ν in general is a non-trivial task. In our numerical experiments, we fix $\nu = 0.3$.

In order to exploit the restarted approach, we present two further acceleration strategies. The first is the Simplified Topological ε-Algorithm (STEA) [10], while the second is a simple algorithm that we propose for the first time in this work and we call it Geometric Series Acceleration (GSA).

STEA Consider a sequence of scalars $\{s_n\}$ and a sequence of vectors $\{t_n\}$. In order to accelerate a scalar sequence, a popular tool is represented by Shanks transformation [20], whose implementation can be done recursively using the scalar ε-algorithm by Wynn [21]

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \quad n = 0, 1, \dots \\ \varepsilon_{0}^{(n)} = s_n, \quad n = 0, 1, \dots \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \left(\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)}\right)^{-1}, \quad k, n, = 0, 1, \dots . \end{cases} \tag{6}$$

The quantities in (6) with even lower indices are the extrapolated quantities, while those with odd lower indices are just intermediate quantities. If you have a sequence $s_0, s_1, \dots, s_m$ (with m even), the scalar ε-algorithm gives rise to a triangular scheme, called ε-array, whose vertex is $\varepsilon_m^{(0)}$, which is the final extrapolated value. In STEA, which has been developed from Topological ε-algorithm (TEA) [22], the scalar ε-algorithm is applied to the sequence $s_n = \langle w, t_n\rangle$, where w is an arbitrary vector. We underline that there are no general guidelines for this choice, which may hugely affect the performance of the algorithm; see [19]. Common choices, like random vector and vector of all ones [19], lead to bad numerical results for test problems reported in Section 4. Thus, in this paper, with the aim of improving the performance of STEA, we choose w (in a different way for each test) as one of the vectors belonging to the sequence $\{t_n\}$ taken into account. Several versions of STEA have been proposed [10], in particular here we consider STEA2-3 (as done in [19]), which is based on the following

$$\tilde{\varepsilon}_{2k+2}^{(n)} = \tilde{\varepsilon}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}} \left(\tilde{\varepsilon}_{2k}^{(n+2)} - \tilde{\varepsilon}_{2k}^{(n+1)}\right), \quad k, n = 0, 1, \dots,$$

with $\tilde{\varepsilon}_0^{(n)} = t_n, n = 0, 1, \dots$. Given a sequence $t_0, t_1, \dots, t_m$ (with m even), $\tilde{\varepsilon}_m^{(0)}$ is taken as the accelerated vector. In our case, considering the restarted algorithm, we have $\bar{v}_{k-(\theta-1)}, \dots, \bar{v}_k$ as sequence of vectors. However, we focus only on the last part of the sequence (i.e. $\bar{v}_{k-2}, \bar{v}_{k-1}, \bar{v}_k$), because we have experimentally observed that the algorithm provides better performance with this choice, then we take $\hat{v}_k = \tilde{\varepsilon}_2^{(0)}$.

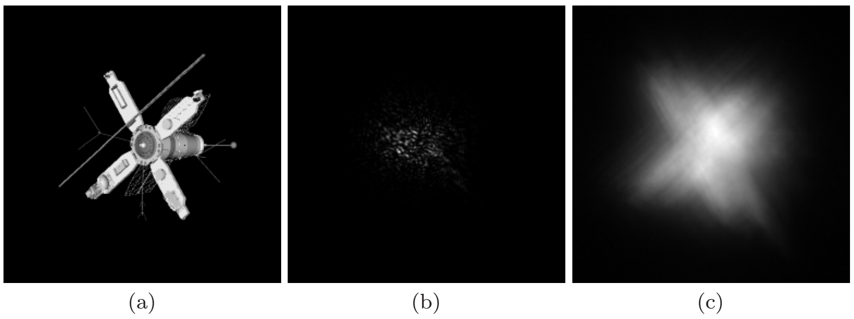

Fig. 1: Test 1: (a) true image (255×255 pixels), (b) point spread function, (c) blurred and noisy image.

GSA Finally, we introduce a new acceleration technique. Suppose that the standard method can be written as

$$\bar{v}_{k+1} = \bar{v}_k + \tau_k(\bar{v}_k - \bar{v}_{k-1}),$$

and that at iteration k we are near convergence, so we can assume that, for $j = 0, 1, 2, \ldots,\ 0 < \tau_{k+j} < 1$ and $\tau_k \approx \tau_{k+j+1}$. Thus, we can approximate the $\bar{v}_{k+\ell}$ iterate by

$$\bar{v}_{k+\ell} \approx \bar{v}_k + (\tau_k + \tau_k^2 + \ldots + \tau_k^\ell)(\bar{v}_k - \bar{v}_{k-1}).$$

Letting ℓ to infinity, we can approximate $\bar{v}_{k+\ell}$ by $\hat{v}_k$ obtaining the following formula for computing the accelerated iteration

$$\hat{v}_k = \bar{v}_k + \left(\frac{1}{1-\tau_k} - 1\right)(\bar{v}_k - \bar{v}_{k-1}). \tag{7}$$

We can rewrite (7) in form (5) as

$$\alpha_k = \kappa\frac{\tau_k}{1-\tau_k}, \quad \text{where } \tau_k = \frac{\|\bar{v}_k - \bar{v}_{k-1}\|}{\|\bar{v}_{k-1} - \bar{v}_{k-2}\|},$$

where we have added a correction factor κ (in this work we set $\kappa = 1.5$). Since the computation of α_k is based on the formula related to geometric series, we call this strategy Geometric Series Acceleration (GSA).

4 Numerical results

We report two image deblurring problems to illustrate the performance of acceleration strategies proposed in Algorithms 3 and 4. In Test 1 we consider the problem in Figure 1, involving an atmospheric blur and 1% of white Gaussian noise, while in Test 2 we consider the problem in Figure 2, involving a Gaussian blur and 0.5% of white Gaussian noise. In Figure 3, for each test, the best restoration computed by standard ADMM is reported.

For measuring the restoration quality, we use the Relative Restoration Error (RRE) defined by

$$\mathrm{RRE}(x) = \frac{\|x - \bar{x}\|_2}{\|\bar{x}\|_2}.$$

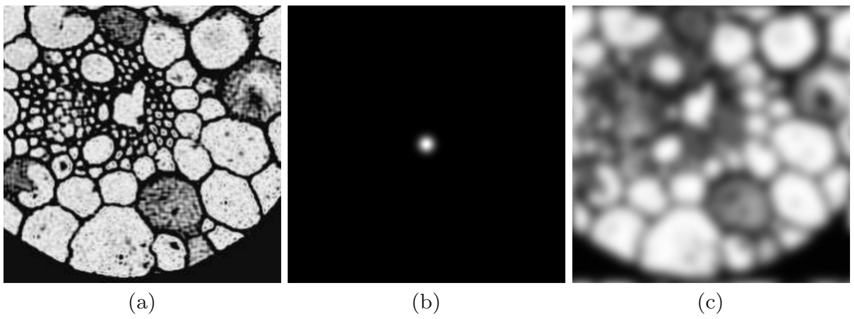

Fig. 2: Test 2: (a) true image (255×255 pixels), (b) point spread function, (c) blurred and noisy image.

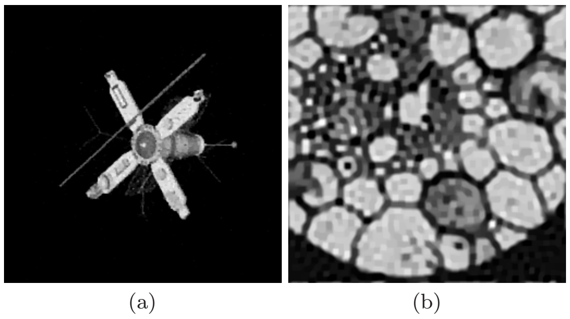

Fig. 3: Restoration computed by standard ADMM with the optimal parameter μ for Test 1 (a) and Test 2 (b).

We stop the iterations for all methods when two consecutive iterates are close enough, in particular, when

$$\frac{\|x_k - x_{k-1}\|_2}{\|x_{k-1}\|_2} \leq 5 \cdot 10^{-4}. \tag{8}$$

For the four algorithms in Section 2, numerical results relative to the best restoration (obtained by stopping the algorithm at iteration corresponding to the minimum RRE) and the restoration associated with the stopping criterion (8) are reported in Tables 1 and 2.

Note that all algorithms provide a comparable RRE, since it depends mainly on the restoration model, while they differ on how quickly they reach the minimum, or small enough, RRE. The latter property is crucial to evaluate the quality of an acceleration technique and the robustness of a method with respect to the stopping iteration.

We notice that the acceleration given by a fixed value of α provides good results for $\alpha = 0.3$, however, these results are further improved when $\alpha = 0.9$. We stress that without the guard condition it would not be possible to set $\alpha = 0.9$, since the convergence of the method has been proven only for $\alpha < 1/3$.

Comparing Nesterov acceleration employed in [3], i.e. Algorithm 2, and the same strategy utilized in our framework, i.e. Algorithm 3 with such acceleration,

	Method	Minimum RRE	IT	Stopping RRE	IT
	Algorithm 1	0.1376	113	0.1380	81
	Algorithm 2	0.1373	68	0.1374	53
Algorithm 3	Stationary (0.3) acc.	0.1376	88	0.1379	59
	Stationary (0.9) acc.	0.1370	38	0.1372	53
	Nesterov acc.	0.1371	35	0.1373	52
	Automatic acc.	0.1373	39	0.1374	52
	ν acc.	0.1370	34	0.1373	52
Algorithm 4	5-Rest. STEA	0.1376	105	0.1379	75
	10-Rest. STEA	0.1375	79	0.1377	57
	5-Rest. GSA	0.1375	73	0.1378	44
	10-Rest. GSA	0.1374	60	0.1375	45

Table 1: Test 1: Relative restoration error (RRE) and iteration number (IT) relative to the best restoration and the one associated with stopping criterion (8) for the four algorithms in Section 2 and the acceleration techniques in Section 3.

we can observe that the latter looks preferable in terms of iteration steps, since it reaches quicker a small RRE.

Automatic acceleration and ν acceleration have similar performance in Test 1, while the latter is particularly performing in Test 2, even if its performance strongly depends on the choice of ν.

Looking at the restarted approach, focusing on iteration steps, we can notice that, both considering best restorations and those associated with stopping criterion (8), GSA requires less iterations than STEA. Moreover, the setting of parameters involved in the acceleration technique appears to be critical for STEA. Even by setting the parameters by hand, looking for the optimal ones, the STEA approach may not be able to produce any meaningful acceleration; see the 5-Restarted version in Test 1. On the other hand, for different choices of the parameter θ, GSA yields always good numerical results, competitive with the group of accelerated versions of ADMM, and in particular exhibits stability in the last column of both Table 1 and Table 2.

Plots of RRE and of combined residual relative to the different techniques taken into account are shown in Figures 4-7 for Test 1 and in Figures 8-11 for Test 2. In order to appreciate the important role of the guard condition, it is worth to look at Figures 6, 7, 10, and 11, related to the combined residual, in which also the bound given by such condition is plotted. When a method tries to cross this line, possibly encountering divergence phenomena, the guard condition is able to prevent it, and in the worst case scenario the plot follows a trend similar to standard ADMM, as predicted by the theory. As illustrated by the figures, this does not hold for Fast ADMM with restart (Algorithm 2), since such strategy employs a restart condition that does not have a direct control on the combined residual. Moreover, from the visual inspection of the RRE plots in Figures 4, 5, 8, and 9, while Nesterov acceleration utilized in our framework gives rise to a stable convergence trend, Fast ADMM with restart has a more irregular behaviour.

	Method	Minimum RRE	IT	Stopping RRE	IT
	Algorithm 1	0.2369	85	0.2369	87
	Algorithm 2	0.2367	66	0.2367	58
Algorithm 3	Stationary (0.3) acc.	0.2369	67	0.2369	69
	Stationary (0.9) acc.	0.2364	28	0.2365	43
	Nesterov acc.	0.2367	32	0.2368	42
	Automatic acc.	0.2368	41	0.2368	50
	ν acc.	0.2364	25	0.2366	39
Algorithm 4	5-Rest. STEA	0.2362	51	0.2363	71
	10-Rest. STEA	0.2368	60	0.2368	64
	5-Rest. GSA	0.2367	35	0.2368	47
	10-Rest. GSA	0.2368	50	0.2368	53

Table 2: Test 2: Relative restoration error (RRE) and iteration number (IT) relative to the best restoration and the one associated with stopping criterion (8) for the four algorithms in Section 2 and the acceleration techniques in Section 3.

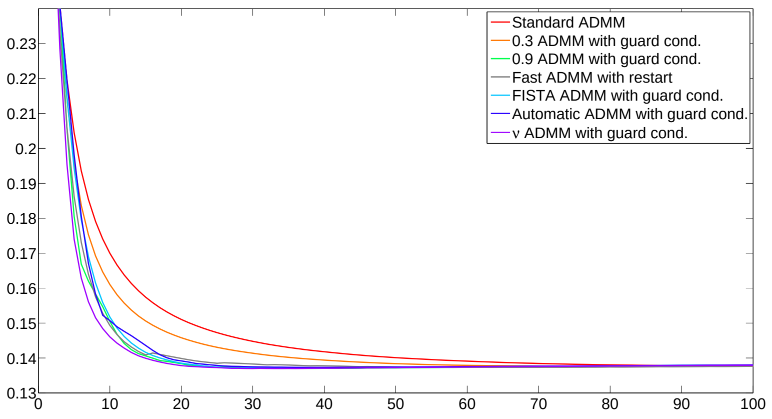

Fig. 4: Test 1: Plot of RRE relative to standard ADMM (Algorithm 1), Fast ADMM with restart (Algorithm 2) and accelerated versions of ADMM (Algorithm 3).

For both tests, if we look at the plots of different methods in the initial iterations, we can notice that the fastest method is Algorithm 3 with ν acceleration. Regarding restarted strategies employed in Algorithm 4, 5-Restarted STEA converges quickly for Test 2, but behaves very similarly to standard ADMM for Test 1; 10-Restarted STEA has satisfactory performance in both tests, but by comparing it with 10-Restarted GSA ADMM we can appreciate that GSA produces better plots. Overall, by looking at the numerical test presented here, we can say that GSA has better performance than STEA.

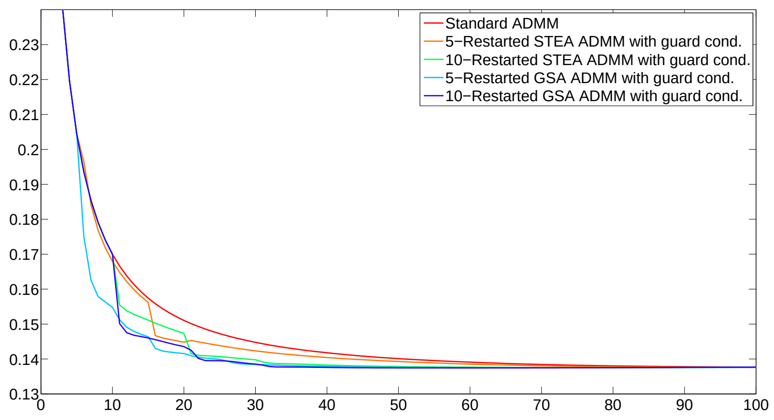

Fig. 5: Test 1: Plot of RRE relative to standard ADMM (Algorithm 1) and restarted versions of ADMM (Algorithm 4).

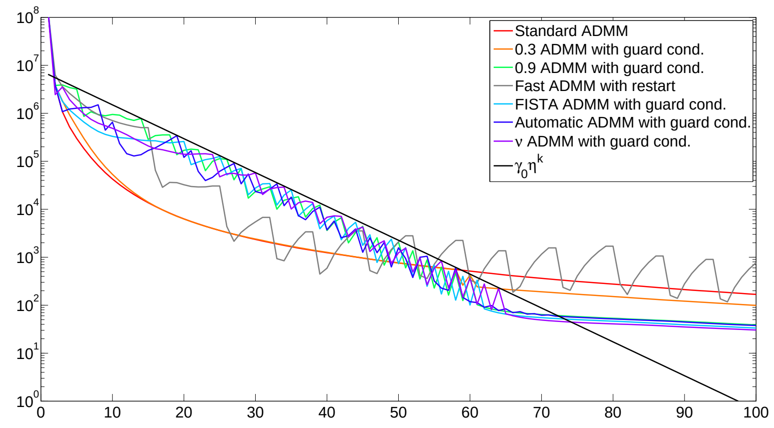

Fig. 6: Test 1: Plot of the combined residual relative to standard ADMM (Algorithm 1), Fast ADMM with restart (Algorithm 2) and accelerated versions of ADMM (Algorithm 3). The black line is related to the guard condition employed in Algorithm 3.

5 Conclusions

In this work we have presented a general framework for acceleration of ADMM algorithm. In particular, we have described an algorithm in which it is possible to insert any acceleration step while having the guarantee of convergence, thanks to a guard condition. Moreover, we have presented two new acceleration techniques,

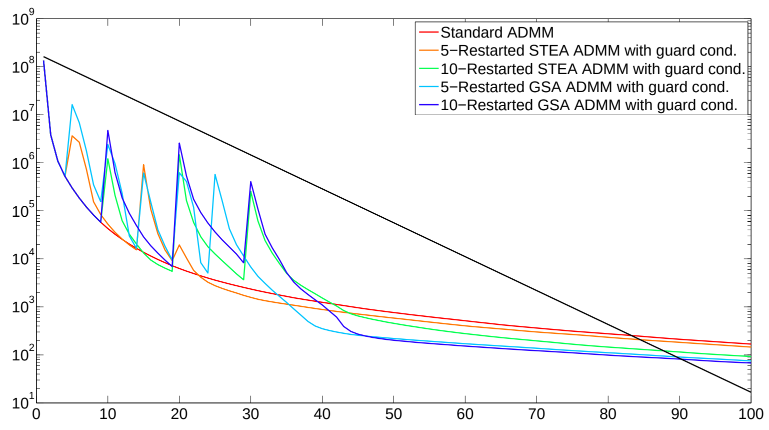

Fig. 7: Test 1: Plot of the combined residual relative to to standard ADMM (Algorithm 1) and restarted versions of ADMM (Algorithm 4). The black line is related to the guard condition employed in Algorithm 4

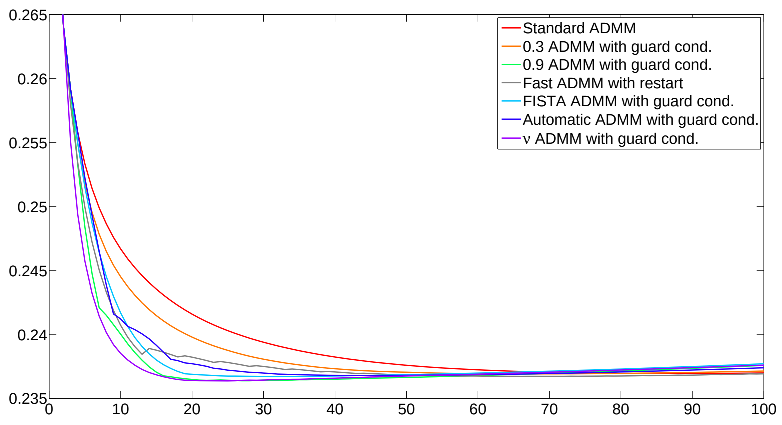

Fig. 8: Test 2: Plot of RRE relative to standard ADMM (Algorithm 1), Fast ADMM with restart (Algorithm 2) and accelerated versions of ADMM (Algorithm 3).

namely ν acceleration, which shows fast convergence for Algorithm 3, and GSA, which shows a stable behaviour for Algorithm 4.

Numerical results relative to image restoration have shown that several acceleration strategies can be employed and this framework leads to an improvement with respect to the state of the art. Finally, we would like to stress that, while the only application taken into account here is image deblurring with the L2-TV

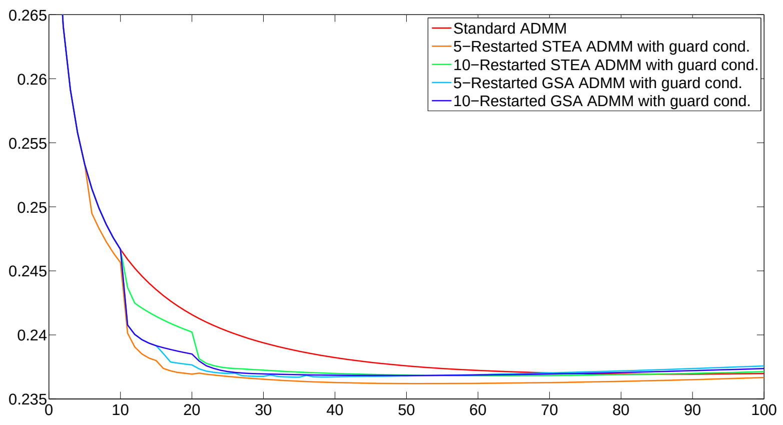

Fig. 9: Test 2: Plot of RRE relative to standard ADMM (Algorithm 1) and restarted versions of ADMM (Algorithm 4).

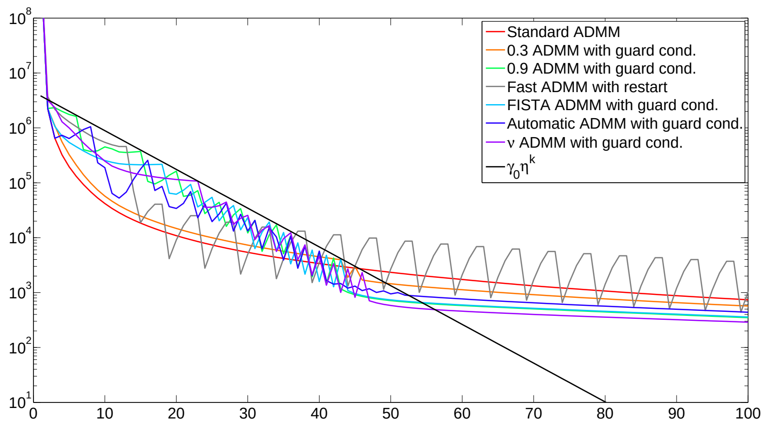

Fig. 10: Test 2: Plot of the combined residual relative to standard ADMM (Algorithm 1), Fast ADMM with restart (Algorithm 2) and accelerated versions of ADMM (Algorithm 3). The black line is related to the guard condition employed in Algorithm 3

model, the proposed approach can be considered to any model and application and it is completely general.

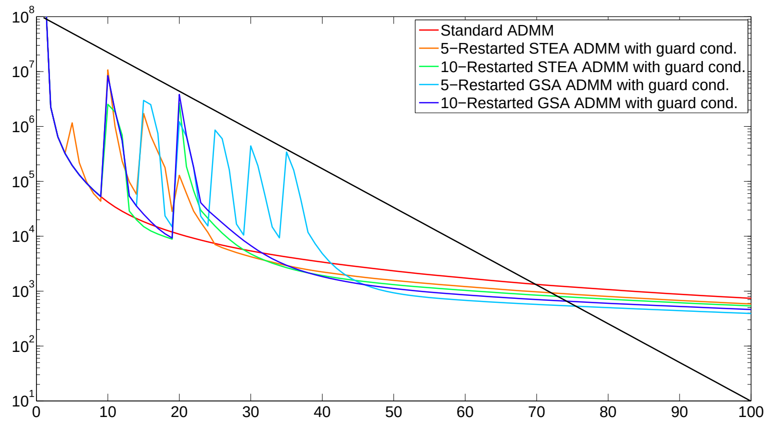

Fig. 11: Test 2: Plot of the combined residual relative to standard ADMM (Algorithm 1) and restarted versions of ADMM (Algorithm 4). The black line is related to the guard condition employed in Algorithm 4

Acknowledgments

The work of the authors is partially founded by the group GNCS of INdAM.

References

1. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine Learning 3 (1) (2011) 1–122.
2. C. Chen, R. H. Chan, S. Ma, J. Yang, Inertial proximal admm for linearly constrained separable convex optimization, SIAM Journal on Imaging Sciences 8 (4) (2015) 2239–2267.
3. T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, Fast alternating direction optimization methods, SIAM Journal on Imaging Sciences 7 (3) (2014) 1588–1623.
4. A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Img. Sci. 2 (2009) 183—-202.
5. Y. Nesterov, Introductory lectures on convex optimization: a basic course, Kluwer Academic Publishers, 2004.
6. D. S. C. Biggs, M. Andrews, Acceleration of iterative image restoration algorithms, Appl. Opt. 36 (1997) 1766–1775.
7. P. Dell'Acqua, ν acceleration of statistical iterative methods for image restoration, Signal, Image and Video Processing 10 (2016) 927–934.
8. M. Hanke, Accelerated landweber iterations for the solutions of ill-posed equations, Numer. Math. 60 (1991) 341–373.
9. L. Landweber, An iteration formula for fredholm integral equations of the first kind, American Journal of Mathematics 73 (1951) 615–624.
10. C. Brezinski, M. Redivo-Zaglia, The simplified topological ε-algorithms for accelerating sequences in a vector space, SIAM J. Sci. Comput. 36 (2014) A2227–A2247.
11. R. H. Chan, M. Tao, X. Yuan, Constrained total variation deblurring models and fast algorithms based on alternating direction method of multipliers, SIAM Journal on imaging Sciences 6 (1) (2013) 680–697.

12. A. Lanza, S. Morigi, M. Pragliola, F. Sgallari, Space-variant tv regularization for image restoration, in: European Congress on Computational Methods in Applied Sciences and Engineering, Springer, 2017, pp. 160–169.
13. M. S. Almeida, M. Figueiredo, Deconvolving images with unknown boundaries using the alternating direction method of multipliers, IEEE Transactions on Image processing 22 (8) (2013) 3074–3086.
14. L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 259–268.
15. A. Lanza, S. Morigi, F. Sgallari, Convex image denoising via non-convex regularization with parameter selection, Journal of Mathematical Imaging and Vision 56 (2) (2016) 195–220.
16. M. S. Almeida, M. A. Figueiredo, Frame-based image deblurring with unknown boundary conditions using the alternating direction method of multipliers, in: Image Processing (ICIP), 2013 20th IEEE International Conference on, IEEE, 2013, pp. 582–585.
17. P. C. Hansen, J. G. Nagy, D. P. O'leary, Deblurring images: matrices, spectra, and filtering, Vol. 3, Siam, Philadelphia, 2006.
18. B. He, X. Yuan, On non-ergodic convergence rate of douglas–rachford alternating direction method of multipliers, Numerische Mathematik 130 (3) (2015) 567–577.
19. S. Gazzola, A. Karapiperi, Image reconstruction and restoration using the simplified ε-algorithm, Applied Mathematics and Computation 274 (2016) 539–555.
20. D. Shanks, Nonlinear transformations of divergent and slowly convergent sequences, J. Math. Phys. 34 (1955) 1–42.
21. P. Wynn, On a device for computing the $e_m(S_n)$ transformation, MTAC 10 (1956) 91–96.
22. C. Brezinski, Généralisations de la transformation de Shanks, de la table de Padé et de l'ε-algorithme, Calcolo 12 (1975) 317–360.