This is very close to final official version, available on IEEExplore.

# Sparse Signal Recovery Using Iterative Proximal Projection

Fateme Ghayem, Mostafa Sadeghi, *Student Member, IEEE*, Massoud Babaie-Zadeh, *Senior Member, IEEE*, Saikat Chatterjee, *Member, IEEE*, Mikael Skoglund, *Senior Member, IEEE*, and Christian Jutten, *Fellow, IEEE*

***Abstract*—This paper is concerned with designing efficient algorithms for recovering sparse signals from noisy underdetermined measurements. More precisely, we consider minimization of a non-smooth and non-convex sparsity promoting function subject to an error constraint. To solve this problem, we use an alternating minimization penalty method, which ends up with an iterative proximal-projection approach. Furthermore, inspired by accelerated gradient schemes for solving convex problems, we equip the obtained algorithm with a so-called extrapolation step to boost its performance. Additionally, we prove its convergence to a critical point. Our extensive simulations on synthetic as well as real data verify that the proposed algorithm considerably outperforms some well-known and recently proposed algorithms.**

***Index Terms*—Sparse signal recovery, compressed sensing, SL0, proximal splitting algorithms, iterative sparsification-projection**

## I. Introduction

Over the past decade, the area of signal processing has been significantly affected by the notion of sparsity. The sparsity assumption on signal and image representations has been successfully utilized in a variety of applications, including image enhancement [1], blind source separation (BSS) [2], medical image reconstruction [3], and compressed sensing (CS) [4]–[6]. Specifically, CS aims to recover an unknown sparse signal, $\mathbf{x}^* \in \mathbb{R}^n$, from a set of underdetermined measurements $\mathbf{y} = \mathbf{A}\mathbf{x}^* \in \mathbb{R}^m$ ($m < n$), where $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the measurement (sensing) matrix. To achieve this goal, several sparse recovery problems have been formulated [7], [8], including $\ell_0$ (pseudo) norm[1] minimization [9], and basis pursuit denoising (BPDN) [10], which is based on minimizing $\ell_1$ norm.

In this paper, we focus on the following constrained problem to retrieve sparse signals:

$$\min_{\mathbf{x}} \; J(\mathbf{x}) \quad \text{s.t.} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon, \tag{1}$$

F. Ghayem, M. Sadeghi and M. Babaie-Zadeh are with the Electrical Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: fateme.ghayem@gmail.com; m.saadeghii@gmail.com; mbzadeh@yahoo.com).

S. Chatterjee and M. Skoglund are with the Communication Theory Lab, KTH, Royal Institute of Technology, Stockholm, 10044, Sweden (e-mail: sach@kth.se; skoglund@ee.kth.se)

C. Jutten is with the GIPSA-Lab, Department of Images and Signals, University of Grenoble and Institut Universitaire de France, France (e-mail: Christian.Jutten@inpg.fr).

This work has been supported by the Center for International Scientific Studies and Collaboration (CISSC), the European project ERC- 2012AdG-320684-CHESS, and Iran National Science Foundation (INSF).

[1]For a vector $\mathbf{x}$, its $\ell_0$ (pseudo) norm, denoted by $\|\mathbf{x}\|_0$, is defined as the number of non-zero entries of $\mathbf{x}$.

where, $J$ is a non-smooth sparsity promoting function like the $\ell_0$ norm, and $\epsilon$ is an error upper-bound. For $J(\mathbf{x}) = \|\mathbf{x}\|_1$, problem (1) is convex and, as such, it can be efficiently solved using algorithms with polynomial complexity [11]. Nevertheless, many studies have shown that using non-convex penalties, e.g., $\ell_p$ (pseudo) norms for $0 \leq p < 1$, leads to much better recovery performance than using convex functions like the $\ell_1$ norm. This has sparked significant research efforts toward designing better problems, mainly using non-convex sparsity promoting functions. Instances of such algorithms are proposed in [9], [12]–[14].

In this paper, we first propose an efficient solver for (1), when $J$ is non-convex and non-smooth. The proposed algorithm is based on penalty methods and proximal algorithms [15]. Then, motivated by existing accelerating techniques for gradient-based solvers of convex problems, we develop an improved version of our proposed algorithm, and establish its convergence. Finally, we discuss the connection of our proposed solver with some previous works. In particular, we show that our proposed algorithm is closely related to a recently introduced family of sparse recovery algorithms, called iterative sparsification-projection (ISP) [16], which also shares similarity with the family of iterative method with adaptive thresholding (IMAT) [17]–[20] algorithms. In fact, the overall framework of our algorithm is the same as the ISP algorithms, however, in contrast to [16], we directly derive the algorithm for solving (1) and prove its convergence to a critical point. Furthermore, we bring new insights into the smooth ISP algorithms and improve their performance.

The rest of the paper is organized as follows. Section II presents our main strategy for solving (1) and the resulting algorithm. This section is accompanied with discussing the connections of the proposed solver with some previous works. Then, Section III is devoted to simulation results.

Throughout the paper, we denote vector and matrix quantities by small and capital bold face characters, respectively. Domain of a function $f$ is represented as $\text{dom}_f$. For a set $\mathcal{C}$, we define the indicator function $\delta_{\mathcal{C}}(\mathbf{x})$ as

$$\delta_{\mathcal{C}}(\mathbf{x}) \triangleq \begin{cases} 0 & \mathbf{x} \in \mathcal{C} \\ \infty & \mathbf{x} \notin \mathcal{C} \end{cases}.$$

## II. Iterative Proximal Projection

In this section, we are going to directly solve (1) using an alternating minimization penalty method. Before proceeding into the details, we first provide a brief review of proximal

splitting algorithms [15], [21], which play an important role in developing our proposed algorithm.

### A. Background on proximal algorithms

Proximal (splitting) algorithms [15], [21]–[23] are efficient techniques for solving a broad class of convex, as well as non-convex minimization problems. In particular, the forward-backward splitting (FBS) [15] method targets minimization of composite objective functions expressed as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) + g(\mathbf{x}), \tag{2}$$

in which, $f:\ \mathbb{R}^n \mapsto \mathbb{R}$ is a smooth (possibly non-convex) gradient Lipschitz function, and $g:\ \mathbb{R}^n \mapsto (-\infty, +\infty]$ is a (usually) non-smooth (possibly non-convex) function. A key operator used in these algorithms is known as *proximal mapping* which is defined as follows:

**Definition 1** [15]**.** *The proximal mapping of a proper and lower semicontinuous function* $g:\ \mathrm{dom}_g \longrightarrow (-\infty, +\infty]$ *at* $\mathbf{x} \in \mathbb{R}^n$ *is defined as*

$$\mathrm{prox}_g(\mathbf{x}) = \underset{\mathbf{u}\in\mathrm{dom}_g}{\mathrm{argmin}} \; \left\{ \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + g(\mathbf{u}) \right\}. \tag{3}$$

Some important functions in CS have closed-form proximal mappings. For instance, the proximal mapping of $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_0$ is the hard-thresholding operator [1], $\mathcal{T}_\lambda^h(x)$, which returns $x$ if $|x| \geq \lambda$, and 0 otherwise. Furthermore, for $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ we get the soft-thresholding operator [1]: $\mathcal{T}_\lambda^s(x) \triangleq \mathrm{sign}(x)(|x|-\lambda)_+$, in which, $(x)_+ \triangleq \max(x, 0)$.

The beauty of the FBS method is that it handles $f$ and $g$ separately in each iteration, by taking one-step gradient descent of $f$ (forward step) and then evaluating the proximal mapping of $g$ at the resulting point (backward step). The FBS algorithm for solving (2) can be summarized as

$$\mathbf{x}_{k+1} = \mathrm{prox}_{\mu_k \cdot g}\Big(\mathbf{x}_k - \mu_k \nabla f(\mathbf{x}_k)\Big), \tag{4}$$

for a suitably chosen step-size $\mu_k > 0$.

The FBS scheme has been extensively utilized in compressed sensing for solving regularized sparse signal recovery problems. In view of (2), let $f(\mathbf{x}) = 1/2\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ and $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, where $\lambda > 0$ is a regularization parameter. Then, the FBS algorithm (4) leads to the well-known iterative shrinkage-thresholding algorithm (ISTA) [1], [24]. Also, $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_0$ ends up with another well-known family of CS algorithms, called iterative hard-thresholding (IHT) [9].

FBS assumes that one of the involved functions is differentiable with a Lipschitz continuous gradient. This requirement is restrictive in some applications. Another splitting method, known as backward-backward splitting [21], resolves this issue by proposing the following solver for (2) when both $f$ and $g$ are non-smooth:

$$\mathbf{x}_{k+1} = \mathrm{prox}_g\Big(\mathrm{prox}_{\mu_k \cdot f}(\mathbf{x}_k)\Big), \tag{5}$$

for some sequence $\{\mu_k\}$. While the above technique has received little attention in compressed sensing, in this paper, we make use of it for solving (1).

Proximal splitting algorithms are ideal choices for large-scale problems [15]. However, they usually exhibit slow convergence, needing many iterations to reach a desired solution. To remedy this issue, some acceleration schemes have been proposed that make these algorithms converge faster with virtually no excessive computational load. One such acceleration method is based on extrapolating the two most recent estimates in order to update the next one [25]. In this scheme, the forward step is computed at an extrapolated point between $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$, instead of $\mathbf{x}_k$. More precisely, considering problem (2), for a smooth $f$ with a Lipschitz continuous gradient, this accelerated scheme is described via

$$\begin{cases} \hat{\mathbf{x}}_k = \mathbf{x}_k + w \cdot (\mathbf{x}_k - \mathbf{x}_{k-1}) \\ \mathbf{x}_{k+1} = \mathrm{prox}_{\mu_k \cdot g}\Big(\hat{\mathbf{x}}_k - \mu_k \nabla f(\hat{\mathbf{x}}_k)\Big) \end{cases}, \tag{6}$$

where, $w \geq 0$ is a weighting constant.

The above technique has been utilized in some sparse recovery algorithms, the most well-known of which is the fast ISTA (FISTA) algorithm [26]. For this algorithm, and with a particular choice of $w$ which is iteration dependent, it has been proved that this simple acceleration scheme significantly improves the convergence rate of the plain version, i.e., ISTA [24], from $\mathcal{O}(1/k)$ to $\mathcal{O}(1/k^2)$.

### B. Proposed algorithm

Now, consider problem (1). We assume that $J$ is proximable, i.e., it has a known or easy-to-compute proximal mapping. Then, consider the following reformulation of (1)

$$\min_{\mathbf{x}} \; J(\mathbf{x}) + \delta_{\mathcal{A}_\epsilon}(\mathbf{x}), \tag{7}$$

where

$$\mathcal{A}_\epsilon \triangleq \{\mathbf{x} \in \mathbb{R}^n \mid \; \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon\}. \tag{8}$$

The cost function of (7) is non-smooth and non-convex, making it very challenging to solve. Nevertheless, it has the special structure of being the sum of two proximable functions. To make use of this structure, we consider the following equivalent form of (7)

$$\min_{\mathbf{x},\mathbf{z}} \; J(\mathbf{z}) + \delta_{\mathcal{A}_\epsilon}(\mathbf{x}) \quad \text{s.t.} \;\; \mathbf{z} = \mathbf{x}, \tag{9}$$

where, we have introduced an auxiliary variable, namely $\mathbf{z}$. Then, we solve this new problem using penalty methods [27], leading to

$$(P_\alpha): \quad \min_{\mathbf{x},\mathbf{z}} \; J(\mathbf{z}) + \delta_{\mathcal{A}_\epsilon}(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x} - \mathbf{z}\|_2^2, \tag{10}$$

in which, $\alpha > 0$ is a penalty parameter. Decreasing the value of $\alpha$ penalizes violation from the constraint $\mathbf{z} = \mathbf{x}$ more and more.

A widely-used strategy to solve problems of the form (10), where variables appear as disjoint blocks, is alternating minimization (also called block-coordinate descent). Utilizing this method, problem (10) is solved by iteratively optimizing over one variable while fixing the other one at its most recent value. This leads to the following iterations ($\forall k \geq 0$)

$$\begin{cases} \mathbf{z}_{k+1} = \mathrm{argmin}_{\mathbf{z}} \;\; \alpha J(\mathbf{z}) + \frac{1}{2}\|\mathbf{z} - \mathbf{x}_k\|_2^2 \\ \mathbf{x}_{k+1} = \mathrm{argmin}_{\mathbf{x}} \;\; \delta_{\mathcal{A}_\epsilon}(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{z}_{k+1}\|_2^2 \end{cases}, \tag{11}$$

which starts with some initial $\mathbf{x}_0$. Recalling the definition of proximal mapping in (3), we can rewrite (11) as

$$\begin{cases} \mathbf{z}_{k+1} = \text{prox}_{\alpha J}(\mathbf{x}_k) \\ \mathbf{x}_{k+1} = \text{prox}_{\delta_{\mathcal{A}_\epsilon}}(\mathbf{z}_{k+1}) \end{cases}, \tag{12}$$

or, in a more compact form, as

$$\mathbf{x}_{k+1} = \text{prox}_{\delta_{\mathcal{A}_\epsilon}}\Big(\text{prox}_{\alpha J}(\mathbf{x}_k)\Big) = \mathcal{P}_{\mathcal{A}_\epsilon}\Big(\text{prox}_{\alpha \cdot J}(\mathbf{x}_k)\Big) \tag{13}$$

in which, $\mathcal{P}_{\mathcal{A}_\epsilon}$ stands for the projection onto $\mathcal{A}_\epsilon$. As $\alpha$ determines the threshold by which the estimates are shrunk, choosing a very small value for it can lead to a quite slow convergence. To avoid this problem, we follow the same warm-starting approach used by penalty methods [27], and solve the problem $P_\alpha$ (10) for a decreasing sequence $\alpha_1, \alpha_2, \cdots$, started with a sufficiently large value. Each subproblem $P_{\alpha_j}$ (10) is solved by iteratively performing (13) until a stopping criterion is met, and then, its final solution is passed to the next subproblem, $P_{\alpha_{j+1}}$, as a starting point.

To improve the performance of the algorithm, we utilize acceleration ideas for solving convex minimization problems. It should be noted that in the context of compressed sensing, most accelerated techniques have been mainly applied to convex sparsity regularized problems, like FISTA, as discussed in the previous subsection. In fact, little efforts have been done to accelerate non-convex sparse recovery solvers, especially those targeting constrained problems. It should also be noted that iterative algorithms for convex problems eventually converge to the global minimizer, and the principal role of the acceleration is to make this process faster. Nevertheless, for general non-convex problems, due to the presence of many local minima, the situation is complicated. Contrary to most works that have focused on composite smooth plus non-smooth objective functions, e.g., FISTA, here, we introduce a similar acceleration technique for backward-backward algorithms applied on objective functions where both components are non-smooth, like (7). This new version is stated as follows

$$\begin{cases} \tilde{\mathbf{x}}_k = \mathbf{x}_k + w \cdot (\mathbf{x}_k - \mathbf{x}_{k-1}) \\ \mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{A}_\epsilon}\Big(\text{prox}_{\alpha \cdot J}(\tilde{\mathbf{x}}_k)\Big) \end{cases}. \tag{14}$$

There exist some results on convergence of block-coordinate descent methods; see e.g., [28], [29]. However, they assume some conditions, such as block quasiconvexity of cost function [28], or convexity of non-smooth parts of the cost function [29], which do not apply here. As an alternative, in Appendix A, we derive an approximation of (14), summarized in Algorithm 1, and call it iterative proximal projection (IPP). In Appendix A, it is also shown that when the relaxation parameters $\mu_z$ and $\mu_x$ tend to 1, then the inner iterations of Algorithm 1, i.e., lines 4-9, approach to (14). Theorem 1 below establishes the convergence of IPP.

**Theorem 1.** *In Algorithm 1, assume that* $0 \leq w < \frac{1}{\max(\mu_x, \mu_z)} - 1$. *The sequence* $\Big\{\mathbf{u}_k \triangleq (\mathbf{x}_k, \mathbf{z}_k)\Big\}_{k=0}^{\infty}$ *generated by IPP for each value of* $\alpha$ *(the inner-loop iterations) converges to a critical point,* $\mathbf{u}^*$*, of the cost function defined in* (10)*. Furthermore, if the cost function satisfies the Kurdyka-Łojasiewicz (KL) property [22] with* $\psi(t) = c \cdot t^{1-\theta}$ *for some* $t > 0$ *and* $\theta \in [0, 1)$*, then:*

- *If* $\theta = 0$ *then the sequence* $\{\mathbf{u}_k\}_{k \geq 0}$ *converges in a finite number of steps.*
- *If* $\theta \in (0, 1/2]$ *then there exist* $d > 0$ *and* $\tau \in [0, 1)$ *such that* $\|\mathbf{u}_k - \mathbf{u}^*\|_2 \leq d \cdot \tau^k$.
- *If* $\theta \in (1/2, 1)$ *then there exist* $d > 0$ *such that* $\|\mathbf{u}_k - \mathbf{u}^*\|_2 \leq d \cdot k^{\frac{\theta-1}{2\theta-1}}$.

*Proof:* See Appendix A.

**Algorithm 1** Iterative Proximal Projection (IPP) for solving (1)

1: **Inputs:** $\mathbf{y}$, $\mathbf{A}$, $\epsilon$, $\alpha_i$, $\alpha_f$, $\tau$, $0 < c < 1$, $w$, $0 < \mu_x, \mu_z < 1$
2: **Initialization:** $k = 0$, $\mathbf{x}_0 = \mathbf{z}_0 = \mathbf{A}^\dagger \mathbf{y}$, $\alpha = \alpha_i$
3: **while** $\alpha > \alpha_f$ **do**
4: **while** $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2 > \tau$ **do**
5: $\tilde{\mathbf{x}}_k = \mathbf{x}_k + w \cdot (\mathbf{x}_k - \mathbf{x}_{k-1})$
6: $\mathbf{z}_{k+1} = \text{prox}_{\mu_z \cdot \alpha \cdot J}(\mathbf{z}_k + \mu_z(\tilde{\mathbf{x}}_k - \mathbf{z}_k))$
7: $\mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{A}_\epsilon}(\mathbf{x}_k + \mu_x(\mathbf{z}_{k+1} - \mathbf{x}_k))$
8: $k \to k + 1$
9: **end while**
10: $\alpha \leftarrow c \cdot \alpha$
11: **end while**
12: **Output:** $\mathbf{x}_k$

**Remark 1.** The above theorem proves the convergence of IPP for each particular value of $\alpha$. That is, the inner-loop (lines 4-9) of Algorithm 1. Nevertheless, this translates into proving the convergence of the whole algorithm. To elaborate, note that each inner-loop actually serves as an initializer for the next one. Therefore, we need to only focus on the last inner-loop which corresponds to the final value of the threshold, and treat the previous ones as only tools for producing a good initial point to begin the last inner-loop.

**Remark 2.** The above theorem (as well as Theorem 2 in Subsection II-C) requires the cost function to have the KL property. As mentioned in [22], [30], the KL property is satisfied by a broad class of functions, including $\ell_p$ (pseudo) norms for $p \geq 0$, real polynomial functions, and indicator function of a polyhedral set. It can be shown that the cost functions considered in this paper possess the KL property, as well.

**Remark 3.** Although proximal-based block coordinate descent algorithms with extrapolation have already been introduced and analyzed in previous works [23], [29], our proposed solver outlined in Algorithm 1 differs from them. The main difference lies in the fact that, here, we update $\mathbf{z}$ based on an extrapolation on $\mathbf{x}$, whereas the methods proposed in [23], [29] update each block using an extrapolation of its own block, but not of other blocks. In other words, based on the approaches in [23], [29], the line 5 of Algorithm 1 should be $\tilde{\mathbf{z}}_k = \mathbf{z}_k + w \cdot (\mathbf{z}_k - \mathbf{z}_{k-1})$. Furthermore, as will be shown in Appendix B, Algorithm 1 guarantees that the objective values in (10) are non-increasing, whereas this is not the case for the algorithms in [23], [29]. Moreover, if we apply the algorithms proposed in [23], [29]

to our problem stated in (10), then it can easily be shown that by letting $\mu_x, \mu_z \to 1$ we would recover the plain alternating minimization in (12); not the accelerated version (14). This is the main reason why we decided to update $\mathbf{z}$ using an extrapolation of the other block, i.e., $\mathbf{x}$. In fact, as derived in Appendix A, the proposed accelerated proximal scheme would reduce to (14) when $\mu_x, \mu_z \to 1$.

### C. Relation to prior art

*1) ISP algorithms:* The ISP algorithms are inspired by the smoothed $\ell_0$ (SL0) algorithm proposed in [13]. Let $f_\sigma(x) \triangleq 1-\exp(-x^2/\sigma^2)$. Then, it can be easily verified that as $\sigma \to 0$, the function $J_\sigma(\mathbf{x}) \triangleq \sum_{i=1}^n f_\sigma(x_i)$ approaches $\|\mathbf{x}\|_0$. In SL0, the $\ell_0$ norm is replaced with $J_\sigma$. The corresponding problem is

$$(P_\sigma): \quad \min_{\mathbf{x}} \; J_\sigma(\mathbf{x}) \quad \text{s.t.} \quad \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2 \leq \epsilon. \tag{15}$$

To avoid undesired local minima which are present when $\sigma$ is very small, SL0 uses the graduated non-convexity technique [31] by solving $P_\sigma$ for a decreasing sequence of $\sigma_1, \sigma_2, \cdots$, in which, the final solution of $P_{\sigma_j}$ is passed to $P_{\sigma_{j+1}}$ as a starting point. Thanks to this warm-starting scheme, a good estimate of the optimal solution of each $P_{\sigma_j}$ can easily be found. SL0 achieves this by performing a few gradient projection iterations. The gradient step can be decoupled over the entries of $\mathbf{x}$ as

$$x \leftarrow \mathcal{T}_\sigma^0(x) \triangleq x - \mu_\sigma \nabla f_\sigma(x), \tag{16}$$

in which, $\mu_\sigma = \mu_0 \cdot \sigma^2$ and $\mu_0 > 0$ is a constant that when $\mu_0 \in (0, 1/2]$ guarantees the convergence of the iterations [16]. It was shown in [16] that $\mathcal{T}_\sigma^0$ is actually a kind of shrinkage, and based on this fact, the general family of ISP algorithms was introduced. The ISP algorithms generalize the approach used in SL0 by replacing the gradient step with a thresholding (shrinkage) operation.

The ISP algorithms are much like the IPP algorithm outlined in Algorithm 1. The differences are as follows. First, the penalty parameter $\alpha$ in Algorithm 1 corresponds to the threshold of the ISP algorithms. Second, as opposed to IPP, the original ISP is not equipped with the extrapolation scheme, as used in line 5 of Algorithm 1. The most important difference, however, is concerning the derivation of the algorithms for non-smooth sparsity promoting functions. Whereas IPP has been derived as a direct solver of problem (1) with a non-smooth $J$, ISP has been developed in an indirect way. More precisely, the non-smooth variant of ISP is inspired by its smooth counterpart, e.g., ISP-SL0 (which uses the SL0 shrinkage), and gradient descent interpretation of proximal mapping for smooth functions [15]. Therefore, no convergence analysis has been established in [16] for non-smooth ISP. By contrast, here we prove the convergence of IPP.

Note that similar to IPP, the smooth ISP algorithm can also be equipped with an extrapolation step [23] to improve its performance. This improved version, which we call ImpISP, is summarized in Algorithm 2. In this algorithm, it is assumed that $J$ in (1) is smooth, with a smoothing parameter $\sigma$, like the one in the SL0 cost function (15), which when $\sigma \to 0$ yields a better approximation to the $\ell_0$ norm.

**Algorithm 2** Improved ISP (ImpISP) for solving (1)

1: **Inputs:** $\mathbf{y}$, $\mathbf{A}$, $\mu$, $\epsilon$, $\sigma_i$, $\sigma_f$, $\tau$, $0 < c < 1$, $w$
2: **Initialization:** $k = 0$, $\mathbf{x}_{-1} = \mathbf{0}$, $\mathbf{x}_0 = \mathbf{A}^\dagger \mathbf{y}$, $\sigma = \sigma_i$
3: **while** $\sigma > \sigma_f$ **do**
4:  **while** $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2 > \tau$ **do**
5:  $\tilde{\mathbf{x}}_k = \mathbf{x}_k + w \cdot (\mathbf{x}_k - \mathbf{x}_{k-1})$
6:  $\mathbf{x}_{k+1} = \mathcal{P}_{\mathcal{A}_\epsilon}(\tilde{\mathbf{x}}_k - \mu \cdot \nabla J(\tilde{\mathbf{x}}_k))$
7:  $k \leftarrow k + 1$
8:  **end while**
9:  $\sigma \leftarrow c \cdot \sigma$
10: **end while**
11: **Output:** $\mathbf{x}_k$

Convergence of the ImpISP algorithm can be proved using [23]. We have restated it in the following proposition.

**Proposition 1.** *Let $J$ in (7) be a smooth function, and let $L$ denote the Lipschitz constant of $\nabla J$. Furthermore, let $\mu > 0$ denote the gradient-descent step size, and define $\gamma \triangleq \mu \cdot L < 1$. Then, if*

$$w \leq \frac{\delta}{2}\left(\frac{1-\gamma}{1+\gamma}\right), \tag{17}$$

*for some $\delta < 1$, the sequence $\{\mathbf{x}_k\}_{k=0}^\infty$ generated by ImpISP, outlined in Algorithm 2, for each value of $\sigma$ (the inner-loop iterations) converges to a critical point, $\mathbf{x}^*$, of the cost function in (7). Furthermore, if the cost function satisfies the KL property [22] with $\psi(t) = c \cdot t^{1-\theta}$ for some $t > 0$ and $\theta \in [0, 1)$, then similar convergence rate results as in Theorem 1 hold here, too.*

*Proof:* The proof follows from [23].

*2) Successive Concave Sparsity Approximation:* A recent algorithm, called successive concave sparsity approximation (SCSA) [32], considers the following sparsity-inducing penalty:

$$f_\sigma^{\text{scsa}}(x) = \lambda \left\{ 1 - \exp\left(-\frac{|x|}{\sigma}\right) \right\}. \tag{18}$$

Similar to the SL0 function, the SCSA penalty has a smoothing parameter, i.e., $\sigma$. Moreover, as shown in [32], $f_\sigma^{\text{scsa}}$ provides a tighter approximation to the $\ell_0$ norm than the SL0 function. SCSA solves the following problem by employing the proximal algorithms:

$$\min_{\mathbf{x}} \; \left\{ \frac{1}{2}\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + f_\sigma^{\text{scsa}}(\mathbf{x}) \right\}. \tag{19}$$

A closed-form solution has been derived in [32] for the proximal mapping of $f_\sigma^{\text{scsa}}$. The overall procedure for solving (19) is based on the FBS framework stated in (4). Additionally, similar to the ISP and IPP algorithms, a decreasing sequence of $\sigma$ is considered, and for each value of this sequence a few iterations of (4) are performed. As shown in [32], starting with a sufficiently large value for $\sigma$, the shrinkage, i.e., the forward step of the FBS scheme, would be soft-thresholding, and as $\sigma$ decreases along the outer-loop iterations, the shrinkage moves toward hard-thresholding. This behavior is shown in Fig. 1. With this illustration in mind, it seems that for each particular value of $\sigma$, the SCSA penalty $f_\sigma^{\text{scsa}}$ approximates the $\ell_p$ norms

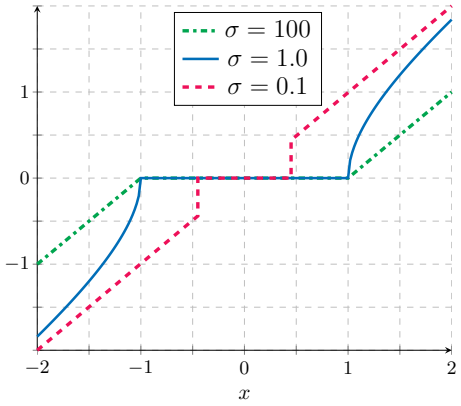



Fig. 1: Plots of the SCSA shrinkage for $\lambda = 1$ and different values of $\sigma$ [32].

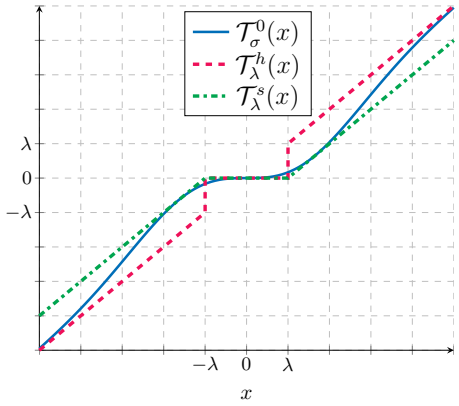



Fig. 2: Plots of the SL0 shrinkage (16) along with hard and soft thresholding functions (see Subsection II-A) with $\sigma = \lambda\sqrt{2e}$.

for $0 \leq p \leq 1$ [32]. For instance, $\sigma = 100$ and $\sigma = 0.1$ correspond to $p = 1$ and $p = 0$, respectively, while $\sigma = 1$ corresponds to some $0 < p < 1$.

There is a major difference between the SCSA algorithm and the ISP and IPP algorithms. The latter algorithms target an error-constrained sparse recovery problem. In contrast, SCSA solves a regularized problem, i.e., (19). Another noticeable difference is that whereas the overall shape of the shrinkage is fixed over the outer-loop iterations of ISP and IPP (lines 3-10 in Algorithm 2 and lines 3-11 in Algorithm 1), in SCSA the shrinkage gradually modifies its shape, from soft-thresholding to hard-thresholding, as $\sigma$ decreases.

### *D. An effective non-smooth function*

There are several types of functions which can be used as $J$ in (1), including $\ell_q$ (pseudo) norms for $0 < q < 1$ [33], smoothly clipped absolute deviation (SCAD) penalty [34], capped $\ell_1$ [35], and logarithmic penalty [36]. Among them, we empirically found that SCAD results in a much better performance. In this subsection, we discuss this penalty, and reveal some interesting facts about the SL0 shrinkage and its relation with SCAD.

*1) SL0 shrinkage versus hard/soft-thresholding:* In what follows, we show that $\mathcal{T}_\sigma^0$ (16) with $\mu_0 = 1/2$ is, in fact, an interpolation between hard and soft thresholdings and, as such, the SL0 shrinkage enjoys the properties of both. To illustrate this fact, the three thresholding functions are plotted in Fig. 2. The relation between $\lambda$ and $\sigma$ can be determined as follows. Computing the derivative of (16), the point on the positive orthant at which $\mathcal{T}_\sigma^0$ has a slope of 1 and is tangent point to $\mathcal{T}_\lambda^s$ can be easily found to be $\tilde{x} = \sigma/\sqrt{2}$. For this point, with $\mu_0 = 1/2$, we have

$$\mathcal{T}_\sigma^0(\tilde{x}) = \tilde{x} - \lambda,$$

yielding

$$\lambda = \frac{\sigma}{\sqrt{2e}}. \tag{20}$$

Now, it follows from Fig. 2 that for $|x| \leq 2\lambda$ or, equivalently, $|x| \leq \sigma\sqrt{2/e}$, the shrinkage function $\mathcal{T}_\sigma^0$ smoothly approximates soft-thresholding, while it gradually moves toward hard-thresholding as $|x|$ increases. Eventually, it coincides with hard-thresholding for $|x| \geq 5\lambda$ or $|x| \geq 5\sigma\sqrt{2e}$.

**Remark 4.** The justification presented in [16] for proving that SL0 behaves like hard thresholding is based on a property of proximal mappings that is only valid for a small enough $\sigma$. On the contrary, the above-mentioned hard-soft thresholding interpretation of the SL0 shrinkage is valid for any $\sigma > 0$.

Another interesting point about SL0 is that, whereas the set of objective values within each subproblem $P_{\sigma_j}$ (15) is decreasing with respect to the steepest descent iterations [16], we have non-decreasing objective values along $j$, provided that the subproblems are solved exactly. This is stated more precisely in the following lemma:

**Lemma 1.** *Let $\mathbf{x}_{\sigma_j}$ be the global minimizer of $P_{\sigma_j}$ (15). Then, the sequence of objective values $\left\{J_{\sigma_j}(\mathbf{x}_{\sigma_j})\right\}_{j=1}^{\infty}$ is non-decreasing.*

*Proof:* First, since $\mathbf{x}_{\sigma_j}$ is the minimizer of $P_{\sigma_j}$, we have

$$J_{\sigma_j}(\mathbf{x}_{\sigma_j}) \leq J_{\sigma_j}(\mathbf{x}_{\sigma_{j+1}}). \tag{21}$$

Then, since $\{\sigma_j\}_{j=1}^{\infty}$ is a decreasing series ($\sigma_{j+1} < \sigma_j$), from the relation $\exp(-x^2/\sigma_j^2) \geq \exp(-x^2/\sigma_{j+1}^2)$ it follows that $\forall \mathbf{x}: \ J_{\sigma_j}(\mathbf{x}) \leq J_{\sigma_{j+1}}(\mathbf{x})$, which in combination with (21) results in

$$J_{\sigma_j}(\mathbf{x}_{\sigma_j}) \leq J_{\sigma_{j+1}}(\mathbf{x}_{\sigma_{j+1}}).$$

■

*2) Smoothly clipped absolute deviation penalty (SCAD):* The SCAD penalty is a concave function whose proximal mapping is given by [34]

$$\mathcal{T}_{\lambda,a}^{scad}(x) \triangleq \begin{cases} \text{sign}(x)(|x|-\lambda)_+ & |x| \leq 2\lambda \\ \frac{(a-1)x - \text{sign}(x)a\lambda}{a-2} & 2\lambda < |x| \leq a\lambda \\ x & |x| > a\lambda \end{cases}. \tag{22}$$

As the above expression shows, the SCAD shrinkage corresponds to the soft-thresholding operator for small enough inputs, whereas for large enough ones it behaves like hard-thresholding and leaves the input intact. For moderate input values, SCAD corresponds to a linear function. The additional parameter "$a$" determines the slope of the transition from

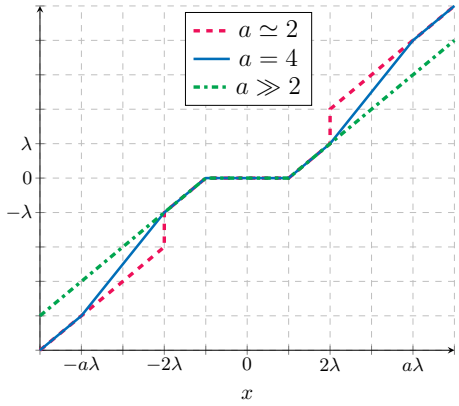



Fig. 3: Plots of the SCAD shrinkage, defined in (22), for three different values of $a$.

soft-thresholding to hard-thresholding: a value very close to 2 is equivalent to an abrupt transition, whereas as $a$ deviates from 2, the transition becomes smoother and smoother. When $a \to \infty$, SCAD becomes equivalent to soft-thresholding. This behavior is best illustrated in Fig. 3.

In the application of regression and variable selection, it has been shown [34] that, in contrast to hard and soft-thresholdings, the SCAD shrinkage simultaneously satisfies the three important properties of *unbiasedness*, *sparsity*, and *continuity* mentioned in [34]. The advantages of this shrinkage over hard and soft thresholdings have also been experimentally demonstrated in [34].

The behavior of SCAD resembles the SL0 shrinkage function. Actually, comparing Figs. 2 and 3 suggests that the SL0 shrinkage can be considered as a smooth approximation of SCAD for moderate values of $a$ (not very close to 2). Nevertheless, one important difference between these two shrinkage functions is the presence of the additional tuning parameter in SCAD, namely $a$. As will be demonstrated by our simulations, appropriate selection of this parameter may considerably boost the performance of IPP with SCAD penalty compared to ISP with the SL0 shrinkage.

## III. Simulations

### A. Simulation setup

In this section, the performance of the IPP and ImpISP algorithms in recovery of synthetically generated sparse and compressible signals, as well as real data is evaluated and compared with those of some well-known and recently proposed methods, including $\ell_q$ pseudo-norm minimization [12], [37], SCAD penalty-regularized minimization (simply referred to as SCAD) [38], expectation-maximization Gaussian-mixture approximate message passing (EM-GM-AMP) [39], generalized OMP (GOMP) [40], and SCSA-FIT, which is an accelerated version of SCSA [32]. We have used the MATLAB implementations of these algorithms provided by their authors.

The underlying signal was generated synthetically according to the model $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$. In this regard, we utilized a Bernoulli-Gaussian distribution to generate the sparse signal $\mathbf{x}$ of the length $n = 1000$, with different sparsity levels $s$, as the number of non-zero entries. Therefore, a larger $s$ indicates a lower sparsity condition, i.e., the signal is less sparse. In our experiments, we consider $s \leq 50$ as a highly sparse condition, $50 < s \leq 160$ as a moderate sparse situation, and $s > 160$ as a low sparse case. Moreover, the non-zero entries of $\mathbf{x}$ were produced from $\mathcal{N}(0, 1)$, with their locations being sampled uniformly at random. Furthermore, the entries of the measurement matrix $\mathbf{A}$ (of dimension $400 \times 1000$) were generated from the normal distribution $\mathcal{N}(0, 1)$. In our experiments, we considered both noiseless and noisy recovery, where for the noisy case a Gaussian noise vector generated from $\mathcal{N}(0, 0.01)$ was added to the measurements. A number of 300 Monte-Carlo simulations have been performed and the average results are reported.

**Remark 5.** We noticed in our simulations that some algorithms, especially SCSA-FIT, are sensitive to the normalization of the measurement matrix. Actually, as will be shown in Subsection III-D, the performance of this algorithm deteriorates in the case of non-normalized[2] $\mathbf{A}$, which is more severe when the measurements are noisy. However, in some applications, the matrix $\mathbf{A}$ does not have normalized columns. For example, in CS this matrix is in fact the multiplication of a measurement matrix $\mathbf{\Phi}$ by a sparsifying transform $\mathbf{\Psi}$, i.e., $\mathbf{A} = \mathbf{\Phi}\mathbf{\Psi}$. Thus, the normalization of $\mathbf{A}$ cannot be guaranteed in this case. To address this issue and see the behaviors of the algorithms, we have considered three cases in our simulations. In the first case, the entries of $\mathbf{A}$ have been independently produced from $\mathcal{N}(0, 1)$, without normalizing the generated columns. In the second one, the columns of the generated sensing matrix $\mathbf{A}$ have been normalized before multiplication by $\mathbf{x}$ to produce the measurements. In the final case, a scaling is applied on the measurement matrix to normalize its columns, and the recovered sparse signal by each algorithm is scaled accordingly. More precisely, first note that

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{A}\mathbf{D}\mathbf{D}^{-1}\mathbf{x} + \mathbf{e} = \bar{\mathbf{A}}\bar{\mathbf{x}} + \mathbf{e} \qquad (23)$$

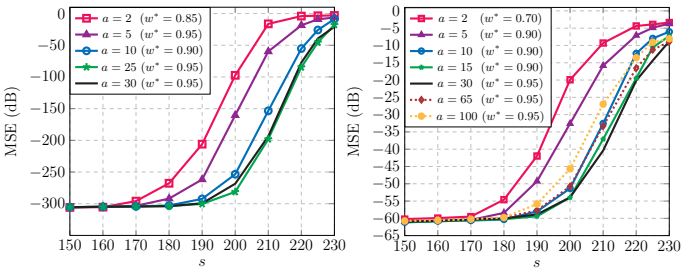
in which, $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \|\mathbf{a}_i\|_2^{-1}$, $\bar{\mathbf{A}} \triangleq \mathbf{A}\mathbf{D}$, and $\bar{\mathbf{x}} \triangleq \mathbf{D}^{-1}\mathbf{x}$. Then, $\bar{\mathbf{A}}$, which has normalized columns, is given to the algorithms. To evaluate the performance, let $\hat{\mathbf{x}}$ be an estimation of $\mathbf{x}$ returned by any of the algorithms. To take into account the normalization effect, we consider $\mathbf{D}\hat{\mathbf{x}}$ as the final estimate.

The parameters of each algorithm have been set as follows. For the IPP and ImpISP algorithms, we chose all the parameters the same as [16] except for the initial value of the penalty parameter ($\alpha$ in IPP and $\sigma$ in ImpISP) for which we experimentally found $3 \times \max(|\mathbf{A}^\dagger\mathbf{y}|)$ to be a good choice. Moreover, the number of outer-loop iterations has been considered 300 for these algorithms. We run the iteratively reweighted $\ell_q$ pseudo-norm minimization for $q = 0.5$. The parameters of SCSA-FIT and SCAD have been set the same as recommended in [32]. Finally, to choose the EMGMAMP and GOMP parameters, we utilized the values suggested in [39] and [40], respectively.

[2]Such behavior occurs for $\ell_1$ minimization, e.g., the LASSO algorithm [41], as well, because the algorithm tends to select those columns of $\mathbf{A}$ with larger $\ell_2$ norms.

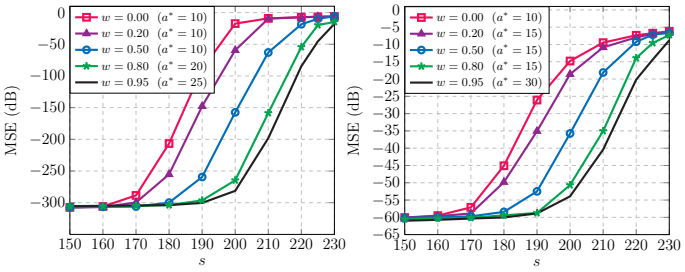
To evaluate and compare the performance of the algorithms, mean-squared error (MSE) and success rate (SR) have been used. MSE is computed as $20\log(\|\mathbf{x}_o - \hat{\mathbf{x}}\|_2/\|\mathbf{x}_o\|_2)$, where $\mathbf{x}_o$ and $\hat{\mathbf{x}}$ denote the original and the recovered sparse signals, respectively. SR is defined as the number of successful Monte-Carlo simulations divided by the total number of trials. A recovery was considered to be successful if MSE $\leq -60$ dB. Additionally, to roughly compare the computational complexities of the algorithms, we report their runtimes. Our simulations were performed on a macOS with a 3.2 GHz Intel core i5 CPU and 8 GB RAM.

The rest of this section consists of four parts. In Subsection III-B, appropriate values of "$a$" in the SCAD shrinkage (22) and "$w$" in Algorithms 1 and 2 are discussed through a set of simulations. Subsection III-C studies the performance of different shrinkages used in IPP and ImpISP, including SL0 shrinkage and hard and SCAD thresholdings. In addition, the effect of the extrapolation technique on the performance of these algorithms will be investigated in this subsection. Next, Subsection III-D compares the performance of IPP-SCAD (IPP with the SCAD penalty) with those of the previous algorithms mentioned in the beginning of this section. Moreover, the effect of normalization of the measurement matrix is studied in this subsection. Finally, in Subsection III-E and Subsection III-F, different algorithms are compared in recovery of compressible signals and in block-based compressed image recovery, respectively.



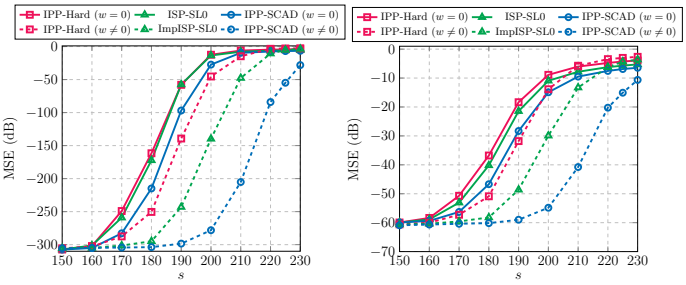

(a) Noise-free case (b) Noisy case

Fig. 4: Effect of the transition parameter "$a$" in IPP-SCAD with its associated extrapolation weight "$w^*$" on the final MSE for different sparsity levels, denoted by $s$.




(a) Noise-free case (b) Noisy case

Fig. 5: Effect of the extrapolation weight "$w$" in IPP-SCAD with its associated "$a^*$" parameter on the final MSE for different sparsity levels, denoted by $s$.




(a) Noise-free case (b) Noisy case

Fig. 6: Comparing the IPP and ISP algorithms with different shrinkages in tow modes: with extrapolation ($w = 0.95$, dashed curves) and without extrapolation ($w = 0$, solid curves).

### *B. Effects of IPP-SCAD parameters*

In this subsection, the effects of the two parameters $a$ and $w$ in IPP-SCAD are experimentally studied. Figures 4 and 5 depict the results in the noiseless and noisy cases, and for different values of $a$ and $w$. In what follows, these results are discussed in details.

In Fig. 4 the MSE values versus sparsity are plotted, where each curve corresponds to a particular value of $a$, with its associated best experimentally found extrapolation weight, i.e., $w^*$. In this figure, it is seen that, increasing $a$ up to around 25 remarkably improves the performance. Moreover, increasing $a$ beyond 25 does not improve the performance. For instance, in the noiseless case, IPP-SCAD had the best performance for $a = 25$, and increasing this parameter to $a = 30$ deteriorates the MSE. It is also noticeable that, appropriate values for $a$ differ in the noiseless and noisy cases. In fact, according to Fig. 4 part (a), $a = 25$ is the appropriate SCAD parameter in the noiseless case, whereas part (b) concludes that $a = 30$ is a better choice in the noisy case.

The effect of $w$ on the performance of IPP-SCAD is studied in Fig. 5. In this figure, the MSE curves versus sparsity for different extrapolation weights and their corresponding $a^*$ parameters that have been experimentally found, are plotted. As demonstrated by this figure, increasing the extrapolation weight significantly improves the performance. For instance, in the noiseless case, increasing the weight for $s = 200$ leads to a considerable improvement of about 250 dB in the MSE.

According to the descriptions in this subsection and considering the simulation results in Figs. 4 and 5, the best values of $a$ for the noiseless and noisy cases were found to be 25 and 30, respectively. Moreover, the corresponding best $w^*$ was set to 0.95 for both noiseless and noisy cases. Therefore, in the rest parts of our simulations we have chosen these values of $a$ and $w$ in the IPP-SCAD method.

### *C. Different shrinkage functions and convergence rate*

In this subsection, the impact of the non-smooth sparsity promoting function used in the IPP algorithm on its performance is investigated. In this regard, we have tested the IPP algorithm with the SCAD penalty (IPP-SCAD) and the

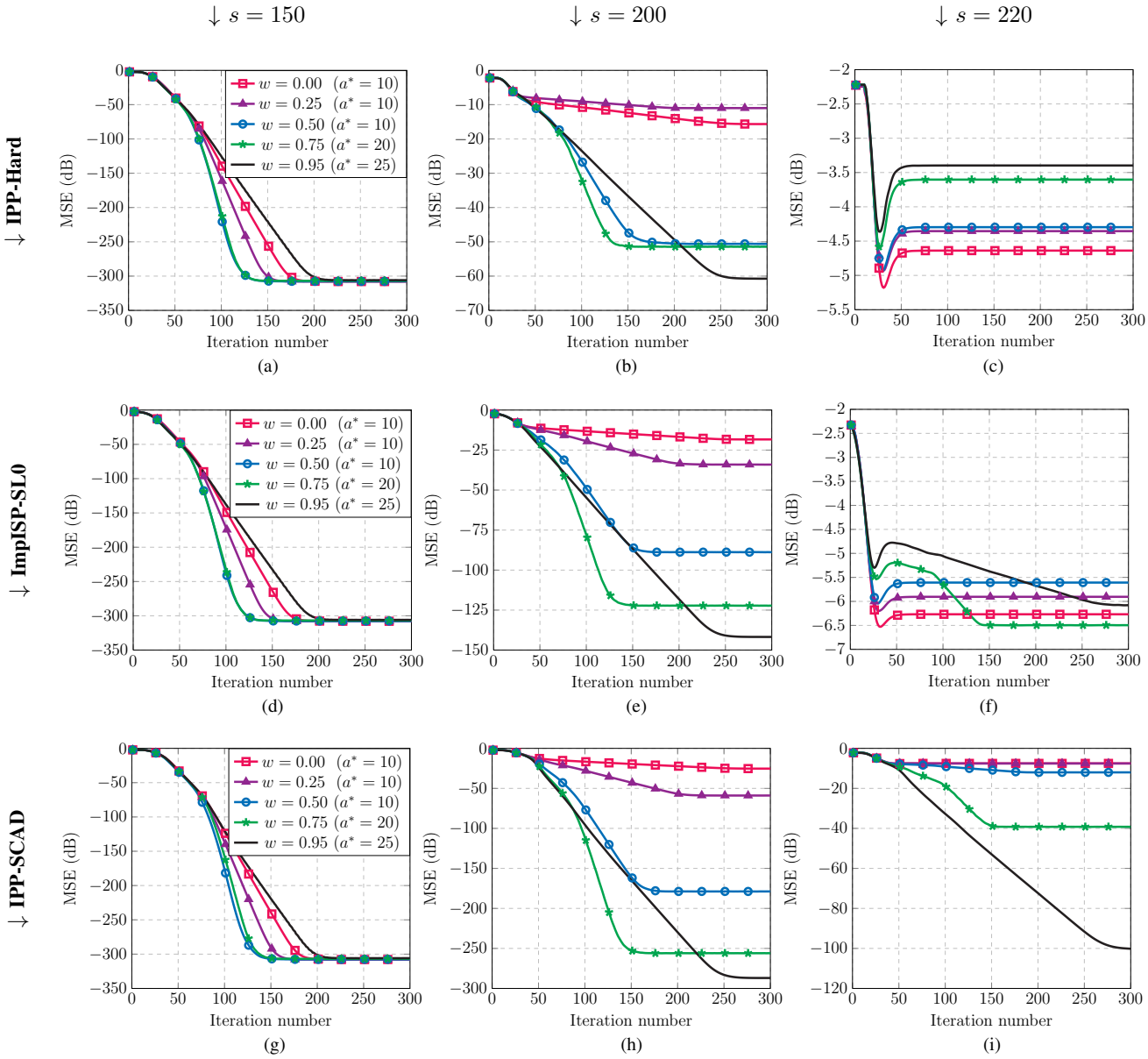



Fig. 7: Investigating the effect of different shrinkages and extrapolation weights in the convergence rate of IPP and ISP for different sparsity levels. Iteration number corresponds to the outer-loop iterations of Algorithms 1 and 2. Note also that the scales of the vertical axes are different.

$\ell_0$ norm (IPP-Hard). For comparison, the results of ISP-SL0 and its improved version are also included. The corresponding MSE curves of IPP-Hard, IPP-SCAD and ISP-SL0 are plotted in Fig. 6 (solid lines), for the two noiseless and noisy cases. Furthermore, to see the effect of the extrapolation scheme, the improved versions are also depicted in these figures (dashed lines). As illustrated in Fig. 6, IPP-SCAD significantly outperforms the IPP-Hard and ISP-SL0 methods. Moreover, comparing the dashed and solid curves, the outstanding effect of the extrapolation is clearly evident for both noiseless and noisy cases, which is more remarkable on the performance of IPP-SCAD. For instance, let us consider the low sparsity level of $s = 200$ in the noiseless case. It is observed in Fig. 6 (a) that for the extrapolation mode where $w \neq 0$, IPP-SCAD outperforms IPP-Hard and ImpISP-SL0 about 250 dB and 150 dB, respectively. Additionally, the SCAD shrinkage along with the extrapolation scheme signifies the very good performance of IPP in low sparsity ranges. For instance, comparing the results in part (a) for the extrapolation case, the IPP-SCAD method could recover the underlying signals with an MSE of $-200$ dB for the sparsity level of 200, whereas those of the plain versions (solid curves) are worse than $-50$ dB in the same sparsity level. Similar improvements in the performance of the algorithms are also observed in the noisy cases as depicted in part (b) of Fig. 6.

Now, we explore the effect of extrapolation on the convergence speed of the algorithms. To achieve this goal, we have run IPP-Hard, IPP-SCAD, and ISP-SL0 for recovery of sparse signals with different sparsity levels. We have also varied the extrapolation weight, $w$. Figure 7 shows the progress

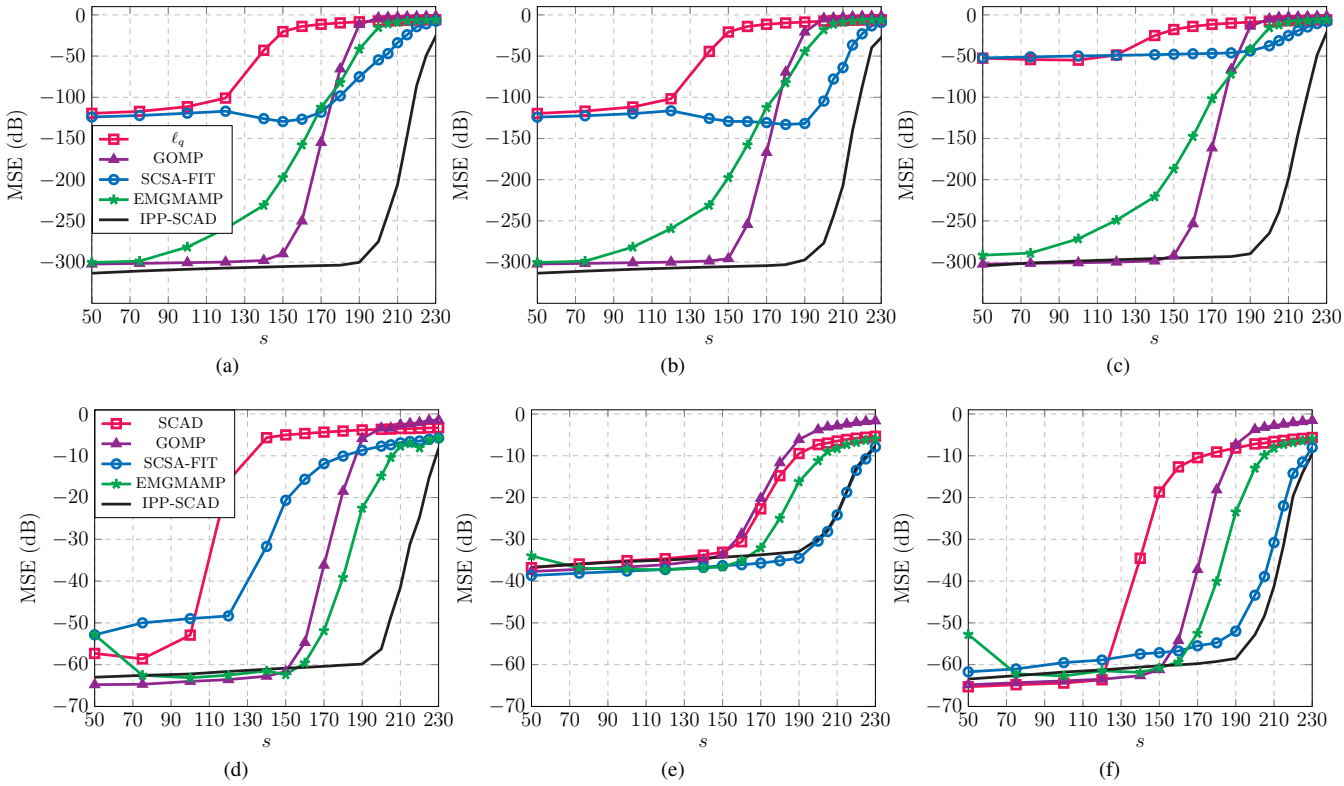



Fig. 8: Comparing the performance of IPP-SCAD with those of some state-of-the-art and recently proposed algorithms for different normalization modes of the measurement matrix. First column: without normalization, second column: with normalization, and the last column: the scaling scheme discussed in Subsection III-A. Moreover, two noiseless (top) and noisy (bottom) cases have been considered.

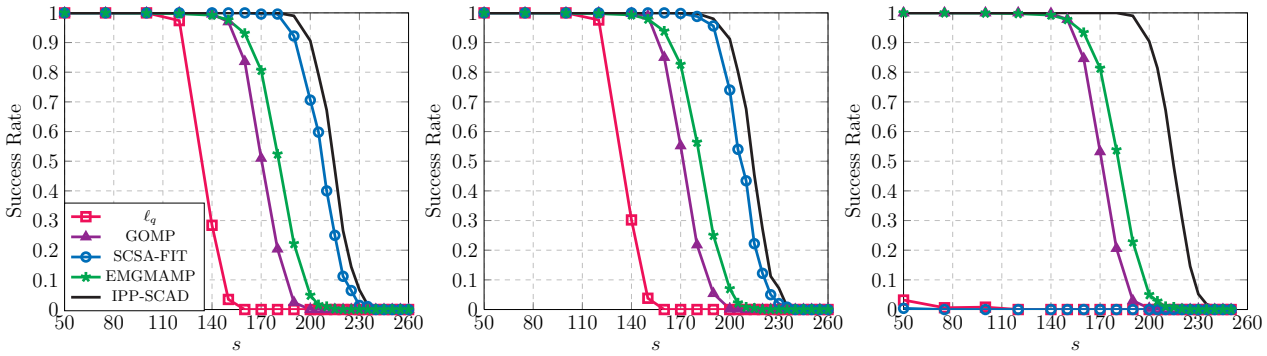
of the MSE values with respect to the iteration number of the algorithms for different extrapolation weights. As follows from this figure, using the extrapolation scheme with an appropriate weight leads to a considerable improvement in the convergence speed of all the algorithms. It is also noticeable that the best extrapolation weight for $s = 150$ was found to be 0.5, whereas $w = 0.95$ works much better for $s = 200$. This suggests that for harder problems, that is, when recovering less sparse signals, a larger extrapolation weight should be used. This figure also demonstrates the superiority of IPP-SCAD, especially for $s = 220$, over IPP-Hard and ImpISP-SL0 (note the different MSE scales in these figures).

As another observation, inspecting the results depicted in Fig. 7 for $s = 200$ and $s = 220$, especially those of IPP-SCAD, reveals an important advantage of using extrapolation. As clearly demonstrated by this figure, utilizing the extrapolation technique not only improves the convergence speed, but it also significantly boosts the performance. For instance, let us focus on the plots corresponding to $s = 200$. Inspecting the final MSEs of $w = 0$ and $w = 0.95$ indicates that whereas the plain versions of the algorithms have converged to a point far away from the underlying sparse signal, the improved versions ($w = 0.95$) have successfully recovered the unknown signal.

### *D. IPP-SCAD versus other algorithms*

This subsection compares the performance of IPP-SCAD with some state-of-the-art and recent algorithms mentioned in Section III, for both noiseless and noisy cases. To this aim, we considered the final MSEs reached by the algorithms, their success rates, and their corresponding runtimes. The results are illustrated in Figs. 8, 9, and Tables I and II.

From the aspect of the final MSEs, as Fig. 8 shows, in the noiseless case (top), and for all the three normalization cases, IPP-SCAD significantly outperforms the other algorithms, specifically $\ell_q$ minimization and SCSA-FIT. This is more noticeable when recovering less sparse signals. For instance, considering the very low sparsity level of $s = 210$ non-zero entries, IPP-SCAD is to recover the signal with an MSE of $s = -200$ dB, whereas the other methods have performed worse than $s = -50$ dB in this case. In Fig. 8, it is also noticeable that in the noiseless case, IPP-SCAD is robust against the normalization constraint. In contrast, the SCSA-FIT and the $\ell_q$ minimization methods are considerably sensitive to the normalization constraint. In addition, the same as the IPP-SCAD method, the GOMP and the EMGMAMP algorithms show a robust behavior against the normalization constraint. The most destructive influence of the normalization constraint appears when a noisy case is considered. As can be seen in Fig. 8 (bottom), comparing part (e) with (d) and (f) reveals that normalizing the columns of $\mathbf{A}$ deteriorates their




(a) (b) (c)

Fig. 9: Comparing the success rate of IPP-SCAD with those of some state-of-the-art and recently proposed algorithms for different normalization modes of the measurement matrix. (a): without normalization, (b): with normalization, and (c): using the scaling scheme discussed in Subsection III-A.

TABLE I: Averaged runtime (in second) of different algorithms, for some sparsity levels ($s$), and in the noiseless case. Each cell shows the averaged time (over 100 trials) that each algorithm takes to reach a specified MSE (dB). A dash sign indicates that the corresponding algorithm has not been able to reach that particular MSE. Blue color denotes the best result, whereas the red one denotes the second good result.

| Sparsity Level | $s=50$ | | | | $s=100$ | | | | $s=150$ | | | | $s=200$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm \ MSE*(dB) | -200 | -150 | -100 | -50 | -200 | -150 | -100 | -50 | -200 | -150 | -100 | -50 | -200 | -150 | -100 | -50 |
| IPP-SCAD($w=0$) | **0.19** | **0.17** | **0.15** | **0.10** | **0.20** | **0.18** | **0.16** | **0.11** | **0.24** | **0.21** | **0.17** | **0.11** | - | - | - | - |
| IPP-SCAD($w=0.95$) | **0.21** | **0.18** | **0.15** | **0.10** | **0.22** | **0.18** | **0.15** | **0.10** | **0.25** | **0.20** | **0.16** | **0.11** | **0.81** | **0.76** | **0.71** | **0.65** |
| GOMP | **0.06** | **0.06** | **0.06** | **0.06** | **0.06** | **0.06** | **0.06** | **0.06** | **0.27** | **0.27** | **0.27** | **0.27** | - | - | - | - |
| EMGMAMP | - | 0.47 | 0.39 | 0.34 | 3.06 | 0.47 | 0.34 | 0.29 | 3.88 | 0.66 | 0.61 | 0.55 | - | - | - | - |
| SCSA-FIT | - | - | - | 0.24 | - | - | - | 0.25 | - | - | - | 1.12 | - | - | - | - |

TABLE II: The same as Table I but for the noisy case.

| Sparsity Level | $s=50$ | | | | $s=140$ | | | | $s=160$ | | | | $s=200$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm \ MSE*(dB) | -60 | -50 | -40 | -30 | -60 | -50 | -40 | -30 | -60 | -50 | -40 | -30 | -60 | -50 | -40 | -30 |
| IPP-SCAD($w=0$) | 1.60 | 0.50 | 0.42 | 0.35 | 6.39 | 0.53 | 0.45 | 0.39 | 8.61 | 1.02 | 0.95 | 0.81 | - | - | - | - |
| IPP-SCAD($w=0.95$) | 0.72 | 0.49 | 0.40 | 0.34 | 3.42 | 0.53 | 0.45 | 0.38 | 4.33 | **0.53** | **0.45** | **0.38** | - | **1.22** | **1.15** | **1.09** |
| GOMP | **0.02** | **0.02** | **0.02** | **0.01** | **0.14** | **0.13** | **0.13** | **0.11** | **1.67** | 1.66 | 1.66 | 1.64 | - | - | - | - |
| EMGMAMP | 3.85 | 3.06 | 3.04 | 2.60 | **0.69** | **0.38** | 0.33 | 0.27 | **0.86** | **0.83** | 0.79 | 0.73 | - | - | - | - |
| SCSA-FIT | - | **0.03** | **0.01** | **0.01** | - | 0.54 | **0.04** | **0.04** | - | 3.39 | **0.08** | **0.07** | - | - | **2.29** | **2.29** |
| SCAD | **0.34** | 0.33 | 0.33 | **0.33** | 7.48 | 7.47 | 7.47 | 7.37 | - | - | - | - | - | - | - | - |

performance about 20 dB.

Now, let us compare the performance of different algorithms for the noiseless setting and in the sense of success rate (SR). As stated previously, we consider an algorithm to be successful in recovering a signal whenever MSE $\leq -60$ dB. According to Fig. 9, the proposed IPP-SCAD method demonstrates a better recovery performance than the other algorithms. Moreover, in parts (a) and (b) of this figure, it can be seen that in the two cases of with and without the normalization constraint on $\mathbf{A}$, the SCSA-FIT method has a similar performance. Nonetheless, according to part (c) of Fig. 9, when using the scaling strategy described in Subsection III-A, the SR of this algorithm significantly degrades. This behavior can also be observed in the $\ell_q$ minimization algorithm. By contrast, in the all parts of Fig. 9, the IPP-SCAD and EMGMAMP methods are robust against the normalization constraint on $\mathbf{A}$.

Finally, Table I and Table II roughly compare the computational complexities of the algorithms and their convergence speeds in noiseless and noisy cases, respectively. In these tables, each cell shows the averaged CPU time that the corresponding algorithm takes to achieve a particular MSE between the sparse estimate and the true signal. Also, different sparsity and MSE levels have been considered. A dash sign in these tables means that the respective algorithm has failed in reaching the specified MSE. According to part $c$ in Figure 8, since the $\ell_q$ algorithm does not have a good performance in compared with the other algorithms, we have not reported its runtimes. Examining these results, we arrive at several conclusions. First, whereas other algorithms often fail in less sparse regimes, our algorithm is successful in satisfying the MSE criteria. Second, GOMP demonstrates a very good performance in highly sparse scenarios. However, it performs worse than the proposed algorithm for less sparse signals. For the noiseless case, it is seen that GOMP is much faster than IPP-SCAD. Nevertheless, in moderate sparsity conditions, e.g., $s = 150$, IPP-SCAD outperforms GOMP. Also, for $s = 200$ as a low sparse condition, Table II shows that GOMP cannot converge, whereas IPP-SCAD still performs well. Moreover, this table

highlights the importance of the extrapolation weight, i.e., $w$. In the case of extrapolation ($w = 0.95$), IPP-SCAD performs quite well for the low sparsity conditions, whereas this is not the case for $w = 0$. Next, according to Table II, it is seen that SCSA-FIT has a very low runtime, outperforming IPP-SCAD. However, in contrast to IPP-SCAD, SCSA-FIT cannot reach a good MSE, i.e., $-60$ dB even in the case of very sparse signals. Comparing the proposed algorithm with EMGMAMP, in some cases of the moderate level of sparsity, i.e., $s = 140$, it is seen that EMGMAMP is faster than IPP-SCAD. But, for $s = 160$, IPP-SCAD is faster than EMGMAMP, and then for $s = 200$, EMGMAMP cannot converge anymore, whereas IPP-SCAD can still recover up to $-50$ dB. To sum up, it can be inferred from Table II that for the highly sparse condition ($s = 50$) and a moderate sparsity situation ($s = 140$), GOMP is faster than IPP-SCAD. Also, SCSA-FIT and SCAD are faster than IPP-SCAD, but cannot perform as well as IPP-SCAD in terms of MSE. However, for low sparsity conditions ($s = 200$), IPP-SCAD is, in general, the best algorithm.

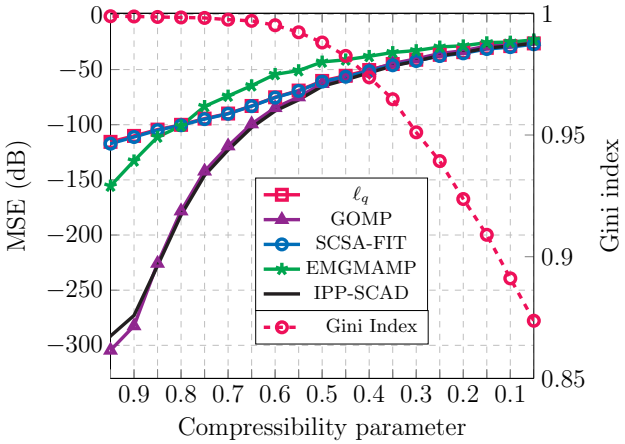



Fig. 10: Final MSEs (dB) of different algorithms in recovery of compressible signals from their underdetermined measurements for various degrees of compressibility, $1-q$. The signal length and the total number of measurements are $n = 1000$ and $m = 400$, respectively.

### *E. Compressible signals*

In this subsection, we consider recovery of "compressible signals" from underdetermined measurements. A compressible signal is not necessarily $s$-sparse in the sense that it has $s$ exactly zero entries, but the sorted magnitudes of its coefficients exhibit an exponential decay [42]. We have followed the experiment of [16] and compared the performance of the algorithms in recovery of compressible signals of length $n = 1000$ from their $m = 400$ linear measurements. The compressible signals were generated from the generalized Pareto distribution (GPD) [42]. The probability density function (pdf) of GPD is as follows

$$P(x; q, \lambda) = \frac{q}{2\lambda}(1 + \frac{|x|}{\lambda})^{-(q+1)}. \tag{24}$$

As shown in [42], for a signal $\mathbf{x} \in \mathbb{R}^n$ generated from GPD, its sorted coefficients denoted by $\{\bar{x}_i\}$ obey $|\bar{x}_i| \lesssim \lambda \cdot (n/i)^{-1/q}$.

In addition, as a robust measure of sparsity of the generated compressible signals, similar to [16], we have used the Gini index [43]. Gini index yields values between 0 and 1, with 0 corresponding to the least sparse signal comprising equal energy entries, and 1 for the most sparse signal with all of its energy concentrated in only one entry.

The final MSEs of the recovered signals versus the compressibility parameter are illustrated in Fig. 10. According to this figure, for higher values of the compressibility parameter, for which the signals are more sparse, IPP-SCAD and GOMP perform significantly better than the other methods. Furthermore, as the compressibility parameter decreases, which is equivalent to decreasing the sparsity, the performance of the all algorithms deteriorate and converge to the same value.

### *F. Block-based compressed image recovery*

Here, we investigate and compare the performance of the algorithms in a block-based compressed image recovery problem (see, e.g., [44]). The setup for this simulation is as follows. Let $\mathbf{X}$ denote the matrix of pixel intensities of a target image. We extracted all $8 \times 8$ blocks of $\mathbf{X}$ with $50\%$ overlap. Let us arrange the $i$th block in a 64 dimensional vector denoted by $\mathbf{x}_i$. Then, we took $m$ random measurements of each $\mathbf{x}_i$ via a Gaussian matrix $\mathbf{\Phi} \in \mathbb{R}^{m\times 64}$ whose entries are drawn from $\mathcal{N}(0, 1)$. Denoting the measurement vector as $\mathbf{y}_i$, we thus have $\mathbf{y}_i = \mathbf{\Phi}\mathbf{x}_i$. Then, the task is to reconstruct the original image $\mathbf{X}$ using only the $\mathbf{y}_i$ vectors. This can be done by first recovering the original blocks $\mathbf{x}_i$'s from $\mathbf{y}_i$'s, and then computing an estimate of $\mathbf{X}$ by averaging the reconstructed blocks. Using compressed sensing, the sparsity of image blocks in a sparsifying basis, here, the $64\times64$ discrete cosine transform (DCT) denoted as $\mathbf{\Psi}$, is utilized to estimate each $\mathbf{x}_i$. More precisely, if $\mathbf{s}_i$ denotes the sparsest solution of $\mathbf{y}_i = \mathbf{\Phi}\mathbf{\Psi}\mathbf{s}_i$, then $\hat{\mathbf{x}}_i = \mathbf{\Psi}\mathbf{s}_i$ would be an estimate of $\mathbf{x}_i$.

To evaluate the competing sparse recovery algorithms, we applied them on this task. We used three benchmark test images: House, Barbara, and Monarch, which are shown in Fig 11. Furthermore, as a measure of performance, we used peak signal to noise ratio (PSNR) between the original images and their reconstructed ones. The averaged PSNRs (over 10 trials) for different undersampling ratios, $\delta = m/64$ (rounded to the nearest integer), are reported in Table III. To see the effect of the extrapolation in IPP, we have included the results of IPP with $w = 0$ as well as $w = 0.85$, denoted respectively, by IPP($w = 0$) and IPP($w = 0.85$). Moreover, Fig. 11 provides a visual comparison of the reconstructed images for $\delta = 0.4$ using IPP (with $w = 0$ and $w = 0.85$) and SCSA (which is the best algorithm among the others). Inspecting Table III and Fig. 11 reveals that IPP with $w = 0.85$ achieves the best performance. Furthermore, it performs much better than its plain version, i.e., IPP with $w = 0$.

## IV. Conclusions and Future Works

In this paper, we considered recovery of sparse signals from underdetermined measurements through minimizing a non-smooth and non-convex sparsity promoting function. We developed an efficient solver for this problem, called IPP,

TABLE III: Comparison of different algorithms for block-based compressed image recovery, and for different undersamling ratios, denoted by $\delta$. The values are PSNRs in dB. Blue color denotes the best result, whereas the red one denotes the second good result.

| | $\delta = 0.1$ | | | $\delta = 0.2$ | | | $\delta = 0.3$ | | | $\delta = 0.4$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | House | Barbara | Monarch | House | Barbara | Monarch | House | Barbara | Monarch | House | Barbara | Monarch |
| $\ell_q$ | 25.13 | 22.99 | 19.87 | 28.56 | 25.30 | 22.71 | 31.17 | 27.38 | 24.97 | 34.15 | 30.17 | 27.65 |
| GOMP | 25.00 | 22.38 | 19.03 | 26.68 | 23.94 | 21.23 | 30.85 | 27.14 | 24.43 | 33.59 | 29.69 | 27.10 |
| SCSA | 25.11 | 22.96 | 19.88 | **28.62** | 25.26 | 22.64 | **31.35** | 27.37 | **24.99** | **34.49** | 30.23 | **27.79** |
| EMGMAMP | 25.02 | 22.94 | 19.69 | 27.96 | 25.06 | 22.22 | 30.91 | 27.15 | 24.65 | 33.64 | 29.72 | 27.06 |
| IPP ($w = 0$) | **25.31** | **23.08** | **20.46** | 27.72 | **25.36** | **22.95** | 31.11 | **27.61** | 24.73 | 33.56 | **30.41** | 27.70 |
| IPP ($w = 0.85$) | **25.57** | **23.38** | **20.67** | **28.65** | **25.63** | **23.45** | **32.50** | **28.17** | **25.55** | **35.01** | **30.92** | **28.47** |

Fig. 11: Visual comparison of reconstructed images from their compressed blocks with undersampling ratio $\delta = 0.4$. From left to right, each column corresponds to: original image, and reconstructed images using SCSA, IPP($w = 0$), and IPP($w = 0.85$), respectively.

which uses penalty method along with proximal ideas. The resulting algorithm was then shown to be closely related to a recently proposed family of algorithms, called ISP. Although the two algorithms share similar structures, ISP has only been analyzed for smooth sparsity inducing functions. IPP uses a simple extrapolation technique which can greatly improve its performance, as demonstrated by our simulations. In addition, the particular non-smooth sparsity promoting function has a determining impact on the ability of IPP in successful recovery of sparse signals. We experimentally showed that

using the SCAD penalty leads to a much better performance compared with other non-smooth functions like the $\ell_0$ pseudo norm. We also equipped the smooth ISP algorithms with the extrapolation technique and illustrated its profound effect through different simulations. The convergence of IPP to a critical point was also established. Overall, our simulations confirmed that IPP has a better performance compared with some well-known or recent methods.

There are some future research directions concerning our algorithm. One direction is to theoretically examine why the proposed algorithm leads to a better recovery performance in terms of MSE. This could be done by, e.g., deriving an explicit convergence rate for our algorithm. As another interesting future work, one could examine the local minima properties of the proposed problem, and understand why the particular extrapolation scheme leads to a considerable performance improvement.

## Appendix A
## Derivation of Algorithm 1

Here, we are going to derive the update formulas of $\mathbf{x}$ and $\mathbf{z}$ outlined in Algorithm 1. As mentioned in Subsection II-B, this algorithm is an approximate solver of (7) based on proximal gradient method [15]. To this end, consider the following reformulation of (10):

$$\min_{\mathbf{z},\mathbf{x}} \; r_1(\mathbf{z}) + r_2(\mathbf{x}) + Q(\mathbf{x},\mathbf{z}), \tag{25}$$

in which $Q(\mathbf{x},\mathbf{z}) \triangleq \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2$, $r_1(\mathbf{z}) \triangleq \alpha J(\mathbf{z})$ and $r_2(\mathbf{x}) \triangleq \delta_{A_e}(\mathbf{x})$. As described in (11), the alternating minimization for solving this problem would be as follows

$$\begin{cases} \mathbf{z}_{k+1} = \operatorname{argmin}_{\mathbf{z}} \; r_1(\mathbf{z}) + Q(\mathbf{x}_k,\mathbf{z}) \\ \mathbf{x}_{k+1} = \operatorname{argmin}_{\mathbf{x}} \; r_2(\mathbf{x}) + Q(\mathbf{x},\mathbf{z}_{k+1}) \end{cases}. \tag{26}$$

Instead of performing the above iterations, we follow the proximal gradient approach, and propose to update $\mathbf{z}$ and $\mathbf{x}$ by replacing $Q(\mathbf{x},\mathbf{z})$ with its quadratic approximation. Doing so, we would have

$$\mathbf{z}_{k+1} = \operatorname*{argmin}_{\mathbf{z}} \; r_1(\mathbf{z}) + \langle \nabla_z Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k), \mathbf{z}-\mathbf{z}_k\rangle + \frac{1}{2\mu_z}\|\mathbf{z}-\mathbf{z}_k\|_2^2 \tag{27}$$

and

$$\mathbf{x}_{k+1} = \operatorname*{argmin}_{\mathbf{x}} \; r_2(\mathbf{x}) + \langle \nabla_x Q(\mathbf{x}_k,\mathbf{z}_{k+1}), \mathbf{x}-\mathbf{x}_k\rangle + \frac{1}{2\mu_x}\|\mathbf{x}-\mathbf{x}_k\|_2^2, \tag{28}$$

where, $\tilde{\mathbf{x}}_k = \mathbf{x}_k + w\cdot(\mathbf{x}_k-\mathbf{x}_{k-1})$, and $0<\mu_x,\mu_z<1$ are some step-sizes.

In the remaining of this section, we will prove that if $\mu_x,\mu_z \to 1$, then the above update formula approach those in Algorithm 1 or (14). To proceed, note that (27) and (28) can be equivalently written as

$$\begin{cases} \mathbf{z}_{k+1} = \operatorname{argmin}_{\mathbf{z}} \; \mu_z r_1(\mathbf{z}) + \frac{1}{2}\|\mathbf{z}-\hat{\mathbf{z}}_k\|_2^2 = \operatorname{Prox}_{\mu_z r_1}(\hat{\mathbf{z}}_k) \\ \mathbf{x}_{k+1} = \operatorname{argmin}_{\mathbf{x}} \; \mu_x r_2(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}_k\|_2^2 = \operatorname{Prox}_{\mu_x r_2}(\hat{\mathbf{x}}_k) \end{cases} \tag{29}$$

where $\hat{\mathbf{z}}_k \triangleq \mathbf{z}_k - \mu_z\nabla_z Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k)$ and $\hat{\mathbf{x}}_k \triangleq \mathbf{x}_k - \mu_x\nabla_x Q(\mathbf{x}_k,\mathbf{z}_{k+1})$. Replacing $\nabla_{\mathbf{z}} Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k) = -(\tilde{\mathbf{x}}_k-\mathbf{z}_k)$ and $\nabla_{\mathbf{x}} Q(\mathbf{x}_k,\mathbf{z}_{k+1}) = (\mathbf{x}_k-\mathbf{z}_{k+1})$, we end up with the following equations for $\hat{\mathbf{z}}_k$ and $\hat{\mathbf{x}}_k$:

$$\begin{cases} \hat{\mathbf{z}}_k = (1-\mu_z)\mathbf{z}_k + \mu_z\tilde{\mathbf{x}}_k \\ \hat{\mathbf{x}}_k = (1-\mu_x)\mathbf{x}_k + \mu_x\mathbf{z}_{k+1} \end{cases}. \tag{30}$$

Now, it is clear that if $\mu_x,\mu_z \to 1$, then $\hat{\mathbf{z}}_k \to \tilde{\mathbf{x}}_k$ and $\hat{\mathbf{x}}_k \to \mathbf{z}_{k+1}$. Consequently, we would have

$$\begin{cases} \tilde{\mathbf{x}}_k = \mathbf{x}_k + w\cdot(\mathbf{x}_k-\mathbf{x}_{k-1}) \\ \mathbf{z}_{k+1} = \operatorname{Prox}_{r_1}(\tilde{\mathbf{x}}_k) \\ \mathbf{x}_{k+1} = \operatorname{Prox}_{r_2}(\mathbf{z}_{k+1}) \end{cases}, \tag{31}$$

which is actually equivalent to (14). In our simulations, we observed that by setting $\mu_z$ and $\mu_x$ to a value close to 1, say $0.99$, we can obtain a very good approximation to (14).

## Appendix B
## Proof of Theorem 1

The proof of Theorem 1 follows similar steps as in [22]. However, due to the fact that unlike [22], here, we use an extrapolation step on $\mathbf{x}$, there are some differences with the proof in [22]. Furthermore, the proof uses some definitions of classical optimization theory, including the definitions of domain of a function, proper and lower-semicontinuous function, subdifferential, and Lipchitz continuity, which can be found in [45].

To start, first note that since $\mathbf{z}_{k+1}$ is the minimizer of (27), we can write:

$$r_1(\mathbf{z}_{k+1}) + \langle \nabla_z Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k), \mathbf{z}_{k+1}-\mathbf{z}_k\rangle + \frac{1}{2\mu_z}\|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2^2 \;\le r_1(\mathbf{z}_k). \tag{32}$$

On the other hand, according to the descent lemma (see, e.g., [22, Lemma 1]), we have the following inequality:

$$Q(\mathbf{x}_k,\mathbf{z}_{k+1}) \le Q(\mathbf{x}_k,\mathbf{z}_k) + \langle \nabla_z Q(\mathbf{x}_k,\mathbf{z}_k), \mathbf{z}_{k+1}-\mathbf{z}_k\rangle + \frac{1}{2}\|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2^2. \tag{33}$$

Adding both sides of (32) and (33), results in

$$\begin{aligned} r_1(\mathbf{z}_{k+1}) + Q(\mathbf{x}_k,\mathbf{z}_{k+1}) \le r_1(\mathbf{z}_k) + Q(\mathbf{x}_k,\mathbf{z}_k) + \\ \langle \nabla_z Q(\mathbf{x}_k,\mathbf{z}_k) - \nabla_z Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k), \mathbf{z}_{k+1}-\mathbf{z}_k\rangle + \\ (\frac{1}{2}-\frac{1}{2\mu_z})\|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2^2. \end{aligned} \tag{34}$$

Now, we apply the Cauchy–Schwarz inequality to (34) as follows:

$$\begin{aligned} r_1(\mathbf{z}_{k+1}) + Q(\mathbf{x}_k,\mathbf{z}_{k+1}) \le r_1(\mathbf{z}_k) + Q(\mathbf{x}_k,\mathbf{z}_k) \\ + \|\nabla_z Q(\mathbf{x}_k,\mathbf{z}_k) - \nabla_z Q(\tilde{\mathbf{x}}_k,\mathbf{z}_k)\|_2 \cdot \|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2 \\ + (\frac{1}{2}-\frac{1}{2\mu_z})\|\mathbf{z}_{k+1}-\mathbf{z}_k\|_2^2. \end{aligned} \tag{35}$$

Simplifying the above inequality by replacing $\nabla_z Q(\mathbf{x}_k, \mathbf{z}_k) - \nabla_z Q(\tilde{\mathbf{x}}_k, \mathbf{z}_k)$ with $w \cdot (\mathbf{x}_k - \mathbf{x}_{k-1})$, and then using Young's inequality [46]

$$a \cdot b \leq \frac{a^2}{2\epsilon} + \frac{\epsilon b^2}{2}, \quad \forall a, b \geq 0, \epsilon > 0 \tag{36}$$

with $\epsilon = 1$, we obtain

$$\begin{aligned} &r_1(\mathbf{z}_{k+1}) + Q(\mathbf{x}_k, \mathbf{z}_{k+1}) \leq r_1(\mathbf{z}_k) + Q(\mathbf{x}_k, \mathbf{z}_k) \\ &+ w \cdot \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2 \cdot \|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2 + (\frac{1}{2} - \frac{1}{2\mu_z})\|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2 \\ &\leq r_1(\mathbf{z}_k) + Q(\mathbf{x}_k, \mathbf{z}_k) + \frac{w}{2} \cdot \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 \\ &+ \frac{w}{2} \cdot \|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2 + (\frac{1}{2} - \frac{1}{2\mu_z})\|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2, \end{aligned} \tag{37}$$

which is simplified to

$$\begin{aligned} &r_1(\mathbf{z}_{k+1}) + Q(\mathbf{x}_k, \mathbf{z}_{k+1}) \leq r_1(\mathbf{z}_k) + Q(\mathbf{x}_k, \mathbf{z}_k) + \\ &\frac{w}{2} \cdot \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2 + (\frac{w}{2} + \frac{1}{2} - \frac{1}{2\mu_z}) \cdot \|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2. \end{aligned} \tag{38}$$

Following a similar procedure for $\mathbf{x}$, we have

$$\begin{aligned} &r_2(\mathbf{x}_{k+1}) + \langle \nabla_x Q(\mathbf{x}_k, \mathbf{z}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k \rangle \\ &+ \frac{1}{2\mu_x}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 \leq r_2(\mathbf{x}_k). \end{aligned} \tag{39}$$

Afterwards, using the descent lemma we obtain

$$\begin{aligned} &Q(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}) \leq Q(\mathbf{x}_k, \mathbf{z}_{k+1}) + \\ &\langle \nabla_x Q(\mathbf{x}_k, \mathbf{z}_{k+1}), \mathbf{x}_{k+1} - \mathbf{x}_k \rangle + \frac{1}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2. \end{aligned} \tag{40}$$

Adding both sides of (39) and (40) results in

$$\begin{aligned} &r_2(\mathbf{x}_{k+1}) + Q(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}) \leq (\frac{1}{2} - \frac{1}{2\mu_x})\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 \\ &+ r_2(\mathbf{x}_k) + Q(\mathbf{x}_k, \mathbf{z}_{k+1}). \end{aligned} \tag{41}$$

Finally, defining $F(\mathbf{x}, \mathbf{z}) \triangleq r_1(\mathbf{z}) + r_2(\mathbf{x}) + Q(\mathbf{x}, \mathbf{z})$, and adding both sides of (38) and (41), we end up with

$$\begin{aligned} &\gamma_x \cdot \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 + \gamma_z \cdot \|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2 \leq \\ &F(\mathbf{x}_k, \mathbf{z}_k) - F(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}), \end{aligned} \tag{42}$$

where

$$\gamma_x = (\frac{1}{2\mu_x} - \frac{w}{2} - \frac{1}{2}), \quad \gamma_z = (\frac{1}{2\mu_z} - \frac{w}{2} - \frac{1}{2}). \tag{43}$$

Due to the assumption on $w$, i.e., $w \leq \frac{1}{\max(\mu_x, \mu_z)} - 1$, we have $\gamma_x \geq 0$ and $\gamma_z \geq 0$. Then, writing (42) for all $k \geq 0$ and adding them up leads to

$$\begin{aligned} &\sum_{k=0}^{\infty} \gamma_x \cdot \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 + \gamma_z \cdot \|\mathbf{z}_{k+1} - \mathbf{z}_k\|_2^2 \leq \\ &F(\mathbf{x}_0, \mathbf{z}_0) - F(\mathbf{x}_\infty, \mathbf{z}_\infty). \end{aligned} \tag{44}$$

On the other hand, (42) implies that the set of objective values $\{F(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ is non-increasing. Furthermore, since $F(\mathbf{x}, \mathbf{z}) \geq 0$, the sequence $\{F(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ converges to a finite value, i.e., $F(\mathbf{x}_\infty, \mathbf{z}_\infty)$. So, the right-hand side of (44) is bounded and non-negative. Therefore, we have

$$\mathbf{x}_{k+1} \to \mathbf{x}_k, \quad \mathbf{z}_{k+1} \to \mathbf{z}_k, \tag{45}$$

as $k \to \infty$. In the following lemma, we prove that the sequence $\{(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ is bounded, and thus it contains a converging subsequence.

**Lemma 2.** *The sequence $\{(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ generated by (29) is bounded.*

*Proof:* Since $\mathbf{x}_{k+1} = \text{Prox}_{\mu_x r_2}(\hat{\mathbf{x}}_k) = \mathcal{P}_{\mathcal{A}_\epsilon}(\hat{\mathbf{x}}_k)$ and $\mathcal{A}_\epsilon$ is a bounded set, we conclude that $\mathbf{x}_k$ remains bounded. Therefore, there exists a constant $R > 0$ such that $\forall k, i$: $|x_k^i| \leq R$ with $x_k^i$ denoting the $i$th entry of $\mathbf{x}_k$. On the other hand, from $\mathbf{z}_{k+1} = \text{Prox}_{\mu_z r_1}(\hat{\mathbf{z}}_k)$, we can write

$$\forall i, k: \ |z_{k+1}^i| \leq |\hat{z}_k^i|. \tag{46}$$

This inequality is due to the fact that for the non-smooth $r_1$ functions considered in this paper, i.e., $\ell_0$ function and SCAD penalty, we have $\forall x$ : $|\text{Prox}_{\mu_z r_1}(x)| \leq |x|$. Moreover, from (30), we have $\hat{z}_k^i = (1 - \mu_z) z_k^i + \mu_z \tilde{x}_k^i$, where $\tilde{x}_k^i = x_k^i + w(x_k^i - x_{k-1}^i)$. So, (46) results in

$$\begin{aligned} |z_{k+1}^i| &\leq (1 - \mu_z)|z_k^i| + \mu_z|\tilde{x}_k^i| \\ &\leq (1 - \mu_z)|z_k^i| + \mu_z \cdot (2w + 1) \cdot R. \end{aligned} \tag{47}$$

By successive application of the above inequality, we obtain

$$|z_{k+1}^i| \leq (1 - \mu_z)^{k+1}|z_0^i| + \mu_z(2w + 1)R\sum_{t=0}^{k}(1 - \mu_z)^t. \tag{48}$$

Then, considering the initialization of Algorithm 1, we have $z_0^i = x_0^i = \mathbf{A}^\dagger \mathbf{y} = \mathcal{P}_{\mathcal{A}_\epsilon}(\mathbf{0})$. So, $|z_0^i| \leq R$. Using this and after some simplifications, (48) leads to

$$|z_{k+1}^i| \leq R + 2wR - 2wR(1 - \mu_z)^{k+1} \leq R + 2wR, \tag{49}$$

where we have used the fact that $0 < \mu_z < 1$. This concludes the proof. ∎

The next step is to show that the sequence $\{(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ approaches the set of critical points [22]. To this aim, we first note that from (27), we have the following optimality condition for $\mathbf{z}_{k+1}$:

$$\mathbf{0} \in \partial r_1(\mathbf{z}_{k+1}) + \nabla_z Q(\tilde{\mathbf{x}}_k, \mathbf{z}_k) + \frac{1}{\mu_z}(\mathbf{z}_{k+1} - \mathbf{z}_k), \tag{50}$$

where, $\partial r_1$ denotes the subdifferential of $r_1$. From (50), we can write:

$$A_z^k \in \partial_z F(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}), \tag{51}$$

where

$$A_z^k \triangleq (\mathbf{x}_k - \mathbf{x}_{k+1}) + w(\mathbf{x}_k - \mathbf{x}_{k-1}) + (1 - \frac{1}{\mu_z})(\mathbf{z}_{k+1} - \mathbf{z}_k). \tag{52}$$

Similarly, we can obtain the following result:

$$A_x^k \in \partial_x F(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}) \tag{53}$$

where,

$$A_x^k \triangleq (1 - \frac{1}{\mu_z})(\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{54}$$

We then have $(A_z^k, A_x^k) \in \partial F(\mathbf{x}_{k+1}, \mathbf{z}_{k+1})$ [22]. Now, because of (45), we have

$$A_z^k, A_x^k \to \mathbf{0} \tag{55}$$

as $k \to \infty$. Let $(\mathbf{x}^*, \mathbf{z}^*)$ be a limit point of $\{(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$. Then, noting the continuity of $\nabla_z Q$ and $\nabla_x Q$, and lower semi-continuity of $r_1$ and $r_2$, and by following similar line of arguments as in [22, Lemma 5 and Theorem 1], we reach to the conclusion that $(\mathbf{x}^*, \mathbf{z}^*)$ is a critical point of $F$, i.e., $\mathbf{0} \in \partial F(\mathbf{x}^*, \mathbf{z}^*)$ [45], and the sequence $\{(\mathbf{x}_k, \mathbf{z}_k)\}_{k=0}^{\infty}$ globally converges to $(\mathbf{x}^*, \mathbf{z}^*)$.

The proof of the convergence rate analysis remains the same as in [47, Theorem 5].

## REFERENCES


[1] M. Elad, *Sparse and Redundant Representations*, Springer, 2010.

[2] P. Comon and C. Jutten, Eds., *Handbook of Blind Source Separation*, Elsevier, 2010.

[3] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Proc. Magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[4] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Proc. Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[5] Y. Eldar and G. Kutyniok, Eds., *Compressed Sensing: Theory and Applications*, Cambridge University Press, 2012.

[6] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2013.

[7] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.

[8] M. F. Duarte and Y.C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Trans. on Signal Proc.*, vol. 59, no. 9, pp. 4053–4085, 2011.

[9] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[10] S. S. Chen, D. D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, pp. 129–159, 2001.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[12] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008.

[13] H. Mohimani, M. Babaie-Zadeh, and Ch. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed $\ell^0$ norm," *IEEE Trans. on Signal Processing*, vol. 57, pp. 289–301, 2009.

[14] H. S. Mousavi, V. Monga, and T. D. Tran, "Iterative convex refinement for sparse recovery," *IEEE Signal Proc. Letters*, vol. 22, no. 11, pp. 1903–1907, 2015.

[15] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2014.

[16] M. Sadeghi and M. Babaie-Zadeh, "Iterative sparsification-projection: Fast and robust sparse signal approximation," *IEEE Trans. on Signal Proc.*, vol. 64, no. 21, pp. 5536–5548, 2016.

[17] R. Eghbali, A. Kazerooni, A. Rashidinejad, and F. Marvasti, "Iterative method with adaptive thresholding for sparse signal reconstruction," in *International Workshop on Sampling Theory and Applications (SAMPTA)*, 2011.

[18] M. Azghani and F. Marvasti, "Iterative methods for random sampling and compressed sensing recovery," *Proceedings of the 10th International Conference on Sampling Theory and Applications (SAMPTA)*, 2013.

[19] M. Azghani, P. Kosmas, and F. Marvasti, "Microwave medical imaging based on sparsity and an iterative method with adaptive thresholding," *IEEE Trans. on Medical Imaging*, vol. 34, no. 2, pp. 357–365, 2015.

[20] F. Marvasti and M. Boloursaz, "Wideband analog to digital conversion by random or level crossing sampling," US patent, no. 9729160, Aug. 2017.

[21] P. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pp. 185–212, 2011.

[22] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[23] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, pp. 1–35, 2017.

[24] I. Daubechies, M. Defrise, and C. De-Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[25] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87 of *Applied Optimization*, Kluwer Academic Publishers, Boston, MA, 2004.

[26] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[27] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.

[28] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[29] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

[30] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gaussseidel methods," *Mathematical Programming*, vol. 137, no. 1, pp. 91–129, 2013.

[31] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, Cambridge, 1987.

[32] M. Malek-Mohammadi, A. Koochakzadeh, M. Babaie-Zadeh, M. Jansson, and C. R. Rojas, "Successive concave sparsity approximation for compressed sensing," *IEEE Trans. on Signal Proc.*, vol. 64, no. 21, pp. 5657–5671, 2016.

[33] L. Zheng, A. Maleki, X. Wang, and T. Long, "Does $\ell_p$-minimization outperform $\ell_1$-minimization?," 2015, http://arxiv.org/abs/1501.03704.

[34] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of American Statistical Association*, vol. 96, pp. 1348–1360, 2001.

[35] T. Zhang, "Analysis of multi-stage convex relaxation for sparse regularization," *Journal of Machine Learning Research*, vol. 11, pp. 1081–1107, 2010.

[36] J. Friedman, "Fast sparse regression and classification," *International Journal of Forecasting*, vol. 28, no. 3, pp. 722–738, 2012.

[37] S. Foucart and M. Lai, "Sparsest solutions of underdetermined linear systems via $\ell_q$-minimization for $0 < q \leq 1$," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 395–407, 2009.

[38] G. Gasso, A. Rakotomamonjy, and S. Canu, "Recovering sparse signals with a certain family of nonconvex penalties and dc programming," *IEEE Trans. on Signal Process.*, vol. 57, no. 12, pp. 4686–4698, 2009.

[39] J. P. Vila and P. Schniter, "Expectation-maximization gaussian-mixture approximate message passing," *IEEE Trans. on Signal Proc.*, vol. 61, no. 19, pp. 4658–4672, 2013.

[40] J. Wang, S. Kwon, and B. Shim, "Generalized orthogonal matching pursuit," *IEEE Trans. on Signal Proc.*, vol. 60, no. 12, pp. 6202–6216, 2012.

[41] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal. Statist. Soc B.*, vol. 58, no. 1, pp. 267–288, 1996.

[42] V. Cevher, "Learning with compressible priors," in *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, 2009.

[43] N. Hurley and S. Rickard, "Comparing measures of sparsity," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4723–4741, 2009.

[44] J. Zhang, C. Zhao, D. Zhao, and W. Gao, "Image compressive sensing recovery using adaptively learned sparsifying basis via l0 minimization," *Signal Processing*, vol. 103, pp. 114–126, 2014.

[45] R. Tyrrell Rockafellar and R. J-B Wets, *Variational Analysis*, Springer, 1998.

[46] W. H. Young, "On classes of summable functions and their fourier series," *Proceedings of the Royal Society A*, vol. 87, no. 594, pp. 225–229, 1912.

[47] H. Attouch and J. Bolte, "On the convergence of the proximal algorithm for nonsmooth functions involving analytic features," *Math. Program.*, vol. 116, pp. 5–16, 2009.

**Fateme Ghayem** received her B.Sc. degree in electrical engineering (communications), from EE department, Shiraz University, Shiraz, Iran (2013). She also received her M.Sc. degree in biomedical engineering from EE department, Sharif University of Technology, Tehran, Iran (2015). After finishing her M.Sc. study, she spent two years (2015-17) as an internship student at DSP-lab, Sharif University of Technology, which was in collaboration with GIPSA-lab, Grenoble, France. Currently, Fateme is a Ph.D. student at GIPSA-lab, University Grenoble Alpes, Grenoble, France. Her research is mainly focused on optimization, machine learning, sparse representation, dictionary learning, multi-modality and blind source separation.

**Mostafa Sadeghi** received the B.Sc. degree from Ferdowsi University of Mashhad, Mashhad, Iran, in 2010, and the M.Sc. degree from Sharif University of Technology, Tehran, Iran, in 2012, both in electrical engineering (communication systems). He is currently working toward the Ph.D. degree in the electrical engineering department, Sharif University of Technology. His main research areas are sparse signal processing, dictionary learning for sparse representation, machine learning, and deep neural networks.

**Masoud Babaie-Zadeh** (M04-SM09) received the B.S. degree in electrical engineering from Isfahan University of Technology, Isfahan, Iran in 1994, and the M.S degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1996, and the Ph.D degree in Signal Processing from Institute National Polytechnique of Grenoble (INPG), Grenoble, France, in 2002. He received the best Ph.D. thesis award of INPG for his Ph.D. dissertation. Since 2003, he has been a faculty member of the Electrical Engineering Department of Sharif University of Technology, Tehran, IRAN, in which, he is currently a full professor. His main research areas are Blind Source Separation (BSS) and Sparsity-aware Signal Processing.

**Saikat Chatterjee** is an assistant professor and docent in the Dept of Information Science and Engineering, KTH-Royal Institute of Technology, Sweden. He received Ph.D. degree from Indian Institute of Science, India. He has published more than 90 papers in international journals and conferences. He was a co-author of the paper that won the best student paper award at ICASSP 2010. His current research interests are signal processing, machine learning, speech and audio processing, and computational biology.

**Mikael Skoglund** (S'93-M'97-SM'04) received the Ph.D. degree in 1997 from Chalmers University of Technology, Sweden. In 1997, he joined the Royal Institute of Technology (KTH), Stockholm, Sweden, where he was appointed to the Chair in Communication Theory in 2003. At KTH, he heads the Department of Information Science and Engineering.

Dr. Skoglund has worked on problems in source-channel coding, coding and transmission for wireless communications, Shannon theory, information and control, and statistical signal processing. He has authored and co-authored more than 138 journal and some 330 conference papers.

Dr. Skoglund has served on numerous technical program committees for IEEE sponsored conferences. During 2003–08 he was an associate editor with the IEEE Transactions on Communications and during 2008–12 he was on the editorial board for the IEEE Transactions on Information Theory.

**Christian Jutten** (AM 92-M 03-SM 06-F 08) received Ph.D. and Doctor es Sciences degrees in signal processing from Grenoble Institute of Technology (GIT), France, in 1981 and 1987, respectively. From 1982, he was an Associate Professor at GIT, before being Full Professor at University Joseph Fourier of Grenoble, in 1989. Since 80s, his research interests have been machine learning and source separation, including theory (separability, source separation in nonlinear mixtures, sparsity, multimodality) and applications (brain and hyperspectral imaging, chemical sensor array, speech). He is author or coauthor of more than 100 papers in international journals, 4 books, 27 keynote plenary talks and about 225 communications in international conferences.

He has been visiting professor at Swiss Federal Polytechnic Institute (Lausanne, Switzerland, 1989), at Riken labs (Japan, 1996) and at Campinas University (Brazil, 2010). He was director or deputy director of his lab from 1993 to 2010, especially head of the signal processing department (120 people) and deputy director of GIPSA-lab (300 people) from 2007 to 2010. He was a scientific advisor for signal and images processing at the French Ministry of Research (19961998) and for the French National Research Center (20032006). From May 2012 to September 2014, he was deputy director at the Institute for Information Sciences (INS2I) at French National Center of Research (CNRS) in charge of signal and image processing.

Christian Jutten was organizer or program chair of many international conferences, especially of the 1st International Conference on Blind Signal Separation and Independent Component Analysis in 1999 (ICA99). He has been a member of a few IEEE Technical Committees, and currently in SP Theory and Methods of the IEEE Signal Processing society. He received best paper awards of EURASIP (1992) and of IEEE GRSS (2012), and Medal Blondel (1997) from the French Electrical Engineering society for his contributions in source separation and independent component analysis. He was elevated as IEEE fellow (2008), EURASIP fellow (2013) and as a Senior Member of Institut Universitaire de France since 2008. He is the recipient of a 2012 ERC Advanced Grant for the project Challenges in Extraction and Separation of Sources (CHESS). In 2016, he was awarded one Grand Prix of the French Acadmie des Sciences.