Journal of Scientific Computing (2022) 93:67
https://doi.org/10.1007/s10915-022-02021-4

The Proximity Operator of the Log-Sum Penalty

Ashley Prater-Bennette[1] · **Lixin Shen**[2] · **Erin E. Tripp**[1]

Received: 17 February 2022 / Revised: 17 August 2022 / Accepted: 30 September 2022 /
Published online: 25 October 2022

Abstract
The log-sum penalty is often adopted as a replacement for the ℓ_0 pseudo-norm in compressive sensing and low-rank optimization. The proximity operator of the ℓ_0 penalty, i.e., the hard-thresholding operator, plays an essential role in applications; similarly, we require an efficient method for evaluating the proximity operator of the log-sum penalty. Due to the nonconvexity of this function, its proximity operator is commonly computed through the iteratively reweighted ℓ_1 method, which replaces the log-sum term with its first-order approximation. This paper reports that the proximity operator of the log-sum penalty actually has an explicit expression. With it, we show that the iteratively reweighted ℓ_1 solution disagrees with the true proximity operator in certain regions. As a by-product, the iteratively reweighted ℓ_1 solution is precisely characterized in terms of the chosen initialization. We also give the explicit form of the proximity operator for the composition of the log-sum penalty with the singular value function, as seen in low-rank applications. These results should be useful in the development of efficient and accurate algorithms for optimization problems involving the log-sum penalty. We present applications to solving compressive sensing problems and to mixed additive Gaussian white noise and impulse noise removal.

Keywords Proximity operator · Proximal mapping · Log-sum function · Iteratively reweighted ℓ_1 · Compressive sensing · Low-rank regularization

Lixin Shen and Erin E. Tripp have contributed equally to this work.

✉ Erin E. Tripp
erin.tripp.4@us.af.mil

Ashley Prater-Bennette
ashley.prater-bennette@us.af.mil

Lixin Shen
lshen03@syr.edu

[1] Information Directorate, Air Force Research Laboratory, Rome, NY, USA

[2] Department of Mathematics, Syracuse University, Syracuse, NY, USA

1 Introduction

The log-sum penalty function is defined as

$$f(x) := \sum_{i=1}^{n} \log\left(1 + \frac{|x_i|}{\epsilon}\right), \tag{1}$$

where $\epsilon > 0$ and $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. This function is commonly used to bridge the gap between the ℓ_0 and ℓ_1 norms in compressive sensing [6, 23] and as a nonconvex surrogate function of the matrix rank function in the low-rank regularization [4, 11, 12, 15]. The parameter ϵ controls the slope of the cusp at the origin. More specifically, as ϵ approaches 0, the slope increases, as illustrated in Fig. 1.

Iterative methods for solving compressive sensing problems typically require solving a subproblem of the form

$$\min\left\{\frac{1}{2\lambda}\|x - z\|^2 + \sum_{i=1}^{n} \log\left(1 + \frac{|x_i|}{\epsilon}\right) : x \in \mathbb{R}^n\right\}, \tag{P_1}$$

where $\lambda > 0$ is a regularization parameter. In the language of convex analysis, the solution to (P_1) is precisely the proximity operator of f with index λ at z (see, e.g., [2]). Due to the nonconvexity of the objective, this problem is difficult to solve directly; instead, an approximate solution is typically obtained through the iteratively reweighted ℓ_1 minimization method, which sequentially linearizes f around the current iterate and solves the linearized convex problem to obtain the next iterate [6]. It has been shown that the iteratively reweighted ℓ_1 algorithm is an optimal majorization-minimization approach to log-sum regularized optimization problems [21].

Similarly, an essential step in algorithms for low-rank optimization problems is solving

$$\min\left\{\frac{1}{2\lambda}\|X - Z\|_F^2 + \sum_{i=1}^{m\wedge n} \log\left(1 + \frac{\sigma_i(X)}{\epsilon}\right) : X \in \mathbb{R}^{m\times n}\right\}, \tag{P_2}$$

where $\sigma_i(X)$ is the ith singular value of X. Here, $m \wedge n := \min\{m, n\}$ and $\|\cdot\|_F$ denotes the Frobenius norm. A similar strategy in solving (P_1) is applied for solving (P_2). The solution

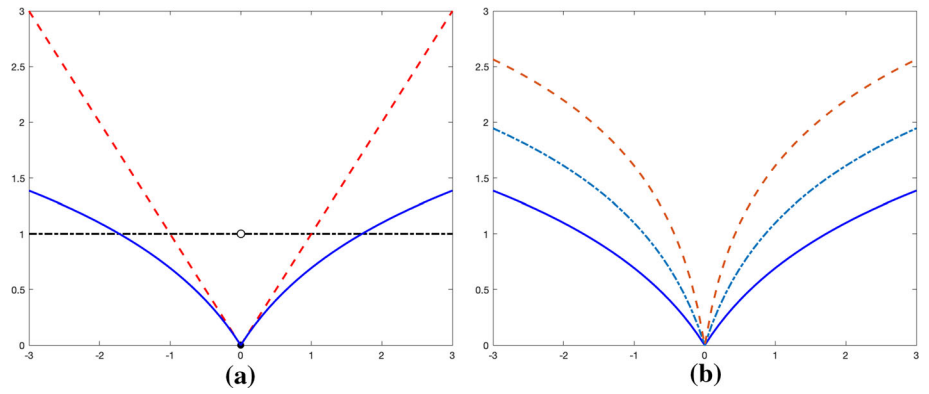

Fig. 1 The graphs of **a** ℓ_0 (dot-dashed), ℓ_1 (dashed), and the log-sum function (solid) and **b** the log-sum function with $\epsilon = 1$ (solid), 0.5 (dot-dashed), and 0.25 (dashed)

to (P_2) is the proximity operator of $f \circ \sigma$ with index λ at Z, where $\sigma : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\wedge n}$ gives all singular values of a matrix. The regularization term in (P_2) is the Log-Det heuristic used in [15]. In fact, if $m \le n$, then

$$\sum_{i=1}^{m\wedge n} \log\left(1 + \frac{\sigma_i(X)}{\epsilon}\right) = \log\det\left(I + \frac{1}{\epsilon}(XX^\top)^{1/2}\right).$$

If $m > n$, we simply replace $XX^\top$ by $X^\top X$ in the above equation.

Notice that the log-sum function is additively separable; that is,

$$f(x) = \sum_{i=1}^{n} g(x_i),$$

where $g : \mathbb{R} \to \mathbb{R}$ is defined by

$$g(w) = \log\left(1 + \frac{|w|}{\epsilon}\right). \tag{2}$$

As a result, the solutions to (P_1) and (P_2) can be given in terms of the proximity operator of g. Throughout this paper, f and g always refer to the functions given in (1) and (2), respectively.

The purpose of this paper is to show that there exist closed-form solutions to (P_1) and (P_2), and therefore time-consuming iterative procedures to approximate them can be avoided. These expressions do not appear in the existing literature to the best of our knowledge and should improve the efficiency and accuracy of algorithms in compressive sensing and low-rank minimization where the function f is used. A recent paper [27] attempts to find the proximity operator of f under the condition $\sqrt{\lambda} > \epsilon$. However, the results reported in [27] are inaccurate, as we show in Sect. 2.

We remark that since the objective functions in (P_1) and (P_2) are nonconvex, the sequences generated by the iterative scheme described above may not converge to a global solution of the corresponding optimization problem. In fact, we identify under what circumstances the iteratively reweighted algorithm for problem (P_1) does not produce an optimal solution.

The rest of the paper is outlined as follows: In the next section, we give an explicit expression of the proximity operator of g, followed by an explicit expression of the solution to (P_1). With this, we show in Sect. 4 that the iteratively reweighted ℓ_1 solution to (P_1) disagrees with the true proximity operator of the log-sum penalty in certain regions. These regions are completely determined by the chosen initial guess for the reweighted ℓ_1 algorithm. In Sect. 3, we give an explicit expression of solutions to (P_2). We take a closer look at the compressive sensing problem in Sect. 5 with numerical experiments comparing the performance of several algorithms. In Sect. 6, we present an application of the log-sum penalty in removing mixed additive Gaussian white noise and impulse noise in noisy images. Our conclusions are drawn in Sect. 7.

2 Solutions to Optimization Problem (P_1)

We begin in Sect. 2.1 by collecting some lemmas related to the proximity operator of g. In Sect. 2.2, we give the explicit expression of the proximity operator of g then use it to derive the proximity operator of f. The main results of this section are summarized in Propositions 1–2 and Theorem 3.

2.1 Fundamental Properties

The proximity operator of g at $z \in \mathbb{R}$ with index λ is defined by

$$\text{prox}_{\lambda g}(z) := \text{argmin}\left\{\frac{1}{2\lambda}(w-z)^2 + g(w) : w \in \mathbb{R}\right\}.$$

The proximity operator is a possibly set-valued mapping from $\mathbb{R} \to 2^{\mathbb{R}}$, the power set of $\mathbb{R}$. Because g, as defined by (2), is continuous and coercive, the set $\text{prox}_{\lambda g}(z)$ is not empty for any $\lambda > 0$ and $z \in \mathbb{R}$.

By definition, the elements of $\text{prox}_{\lambda g}$ are solutions of an optimization problem, and in order to characterize these solutions, we must understand the behavior of the objective function around its critical points. Given λ and z, define $q_{\lambda,z} : \mathbb{R} \to \mathbb{R}$ as follows

$$q_{\lambda,z}(x) = \frac{1}{2\lambda}(x-z)^2 + g(x).$$

Note that $q_{\lambda,z}$ is differentiable away from the origin with

$$\frac{d}{dx}q_{\lambda,z}(x) = \frac{1}{\lambda}(x-z) + \frac{1}{x + \epsilon\,\text{sgn}(x)}, \tag{3}$$

where $\text{sgn}(x)$ is the sign function. Clearly,

$$\text{prox}_{\lambda g}(z) = \arg\min\{q_{\lambda,z}(x) : x \in \mathbb{R}\}.$$

A straightforward consequence of this definition is that $\text{prox}_{\lambda g}$ is symmetric about the origin and shrinks points towards the origin.

Lemma 1 *(Symmetry and Shrinkage) Let z be nonzero. Then (i)* $\text{prox}_{\lambda g}(z) = -\text{prox}_{\lambda g}(-z)$, *and (ii)* $\text{prox}_{\lambda g}(z) \subseteq [0, z)$ *if z is positive and* $\text{prox}_{\lambda g}(z) \subseteq (z, 0]$ *if z is negative.*

Proof (i) This follows directly from the fact that $q_{\lambda,z}(x) = q_{\lambda,-z}(-x)$ for all $x \in \mathbb{R}$.

(ii) First assume $z > 0$. One can check that $q_{\lambda,z}(x) < q_{\lambda,z}(-x)$, for $x > 0$. Hence the elements in $\text{prox}_{\lambda g}(z)$ should be nonnegative. It can be verified from (3) that $q_{\lambda,z}(x)$ as a function of x is increasing on $[z, \infty)$, which implies that $\text{prox}_{\lambda g}(z) \subseteq [0, z]$. By using Taylor's expansion for expanding $q_{\lambda,z}(x)$ at z, one has

$$q_{\lambda,z}(x) = q_{\lambda,z}(z) + \frac{1}{z+\epsilon}(x-z) + \left(\frac{1}{2\lambda} - \frac{1}{2(z+\epsilon)^2}\right)(x-z)^2 + o(|x-z|^2).$$

From this expression, we see that $q_{\lambda,z}(x) < q_{\lambda,z}(z)$ when x is sufficiently close to z from below. We conclude that $\text{prox}_{\lambda g}(z) \subseteq [0, z)$.

The above discussion, along with (i), implies that $\text{prox}_{\lambda g}(z) \subseteq (z, 0]$ if z is negative. □

By item (i) of Lemma 1, it is sufficient to study the proximity operator $\text{prox}_{\lambda g}(z)$ for all non-negative z. Moreover, it follows immediately that for all $\lambda > 0$,

$$\text{prox}_{\lambda g}(0) = \{0\}. \tag{4}$$

More generally, the proximity operator depends on the structure of the function $q_{\lambda,z}$, which depends both on the parameters λ and ϵ as well as the value of z.

Lemma 2 *(Convexity of Objective) (i) If $\sqrt{\lambda} \le \epsilon$, then $q_{\lambda,z}$ is strictly convex on $(0, \infty)$. (ii) If $\sqrt{\lambda} > \epsilon$, then $q_{\lambda,z}$ is concave on $(0, \sqrt{\lambda} - \epsilon]$ and convex on $[\sqrt{\lambda} - \epsilon, \infty)$.*

Proof This follows immediately from the fact that $\frac{d^2}{dx^2}q_{\lambda,z}(x) = \frac{1}{\lambda} - \frac{1}{(x+\epsilon)^2}$ for $x \in (0, \infty)$. □

From Lemma 2, $q_{\lambda,z}$ has a unique minimizer for each z when $\sqrt{\lambda} \leq \epsilon$. In other words, $\text{prox}_{\lambda g}$ is single-valued in this case. To study $\text{prox}_{\lambda g}$ when $\sqrt{\lambda} > \epsilon$, we need the following two lemmas. Note from the proof that Lemma 3 is independent of the monotonicity of g and is instead an intrinsic quality of the proximity operator.

Lemma 3 *(Order-preservation of* prox*) Let* $0 \leq u < v$. *If* $\alpha \in \text{prox}_{\lambda g}(u)$ *and* $\beta \in \text{prox}_{\lambda g}(v)$, *then* $0 \leq \alpha \leq \beta$.

Proof By the definition of the proximity operator, one has $q_{\lambda,u}(\alpha) \leq q_{\lambda,u}(\beta)$ and $q_{\lambda,v}(\beta) \leq q_{\lambda,v}(\alpha)$. Then, $q_{\lambda,u}(\alpha) + q_{\lambda,v}(\beta) \leq q_{\lambda,u}(\beta) + q_{\lambda,v}(\alpha)$. After simplification, we have from the previous inequality that $(\alpha - \beta)(u - v) \geq 0$. Hence, $\alpha \leq \beta$. □

Lemma 4 *(Set-valued* prox*) If the set* $\text{prox}_{\lambda g}(z_*)$ *at some* $z_* > 0$ *contains zero and a positive number, then* $\text{prox}_{\lambda g}(z)$ *is a singleton for all* $|z| \neq z_*$. *In particular,* $\text{prox}_{\lambda g}(z) = \{0\}$ *for all* $|z| < z_*$ *and* $\text{prox}_{\lambda g}(z)$ *contains only one nonzero element for all* $|z| > z_*$.

Proof By Lemma 1 and equation (4), we only need to consider $\text{prox}_{\lambda g}(z)$ for $z > 0$. Since $0 \in \text{prox}_{\lambda g}(z_*)$, then $\text{prox}_{\lambda g}(z) = \{0\}$ for all $0 \leq z < z_*$ by Lemma 3. Let $\alpha > 0$ be an element in $\text{prox}_{\lambda g}(z_*)$. Then, by Lemma 3 again, all elements in $\text{prox}_{\lambda g}(z)$ must greater than or equal to α for all $z > z_*$.

Suppose that $\text{prox}_{\lambda g}(z)$ for some $z > z_*$ has at least two nonzero elements, say β and γ. Then, one should have $q_{\lambda,z}(\beta) = q_{\lambda,z}(\gamma)$ and $\frac{d}{dx}q_{\lambda,z}(\beta) = \frac{d}{dx}q_{\lambda,z}(\gamma) = 0$. The intermediate value theorem implies that there exists another point between β and γ, say τ, at which $\frac{d}{dx}q_{\lambda,z}(\tau) = 0$. However, $\frac{d}{dx}q_{\lambda,z}(x) = \frac{1}{\lambda}(x - z) + \frac{1}{x+\epsilon}$ has at most two roots on $[0, \infty)$. We conclude that $\text{prox}_{\lambda g}(z)$ is a singleton for all $z > z_*$. This completes the proof. □

2.2 The Proximity Operators of *g* and *f*

We are now prepared to compute the proximity operators of g and f. As above, the problem is split between two cases: $\sqrt{\lambda} \leq \epsilon$ and $\sqrt{\lambda} > \epsilon$, i.e., the convex case and the nonconvex case.

For the rest of this section, we consider only $z > 0$. With item (ii) of Lemma 1, we therefore only need to investigate the behavior of $q_{\lambda,z}(x)$ for $x \geq 0$. In this case, the derivative of $q_{\lambda,z}$ can be rewritten as

$$\frac{d}{dx}q_{\lambda,z}(x) = \frac{(x - \frac{1}{2}(z - \epsilon))^2 + \lambda - \frac{1}{4}(z + \epsilon)^2}{\lambda(x + \epsilon)}. \tag{5}$$

Moreover, for $z \geq \max\{2\sqrt{\lambda} - \epsilon, 0\}$, the expression above can be factored as

$$\frac{d}{dx}q_{\lambda,z}(x) = \frac{1}{\lambda(x + \epsilon)}(x - r_1(z))(x - r_2(z)), \tag{6}$$

where

$$r_1(z) := \frac{1}{2}(z - \epsilon) - \sqrt{\frac{1}{4}(z + \epsilon)^2 - \lambda} \tag{7}$$

and

$$r_2(z) := \frac{1}{2}(z - \epsilon) + \sqrt{\frac{1}{4}(z + \epsilon)^2 - \lambda}. \tag{8}$$

First, a few remarks on the functions r_1 and r_2 which are central to the next lemmas. Clearly $r_2(z) > r_1(z)$ for all $z \in [2\sqrt{\lambda} - \epsilon, \infty)$, and $r_1(\frac{\lambda}{\epsilon}) = 0$, noting that

$$\frac{\lambda}{\epsilon} = (2\sqrt{\lambda} - \epsilon) + (\sqrt{\epsilon} - \sqrt{\lambda/\epsilon})^2 > 2\sqrt{\lambda} - \epsilon.$$

These functions are differentiable themselves, with

$$r_1'(z) = \frac{\sqrt{(z+\epsilon)^2 - 4\lambda} - (z+\epsilon)}{2\sqrt{(z+\epsilon)^2 - 4\lambda}} < 0$$

and

$$r_2'(z) = \frac{\sqrt{(z+\epsilon)^2 - 4\lambda} + (z+\epsilon)}{2\sqrt{(z+\epsilon)^2 - 4\lambda}} > 0.$$

That is, r_1 is strictly decreasing, while r_2 is strictly increasing.

Due to Lemmas 1 and 2, $\text{prox}_{\lambda g}$ is a single-valued operator when $\sqrt{\lambda} \leq \epsilon$. More precisely, we have the following result.

Proposition 1 *If $\sqrt{\lambda} \leq \epsilon$, then*

$$\text{prox}_{\lambda g}(z) = \begin{cases} \{0\}, & \text{if } |z| \leq \frac{\lambda}{\epsilon}; \\ \{\text{sgn}(z) r_2(|z|)\}, & \text{if } |z| > \frac{\lambda}{\epsilon}, \end{cases} \tag{9}$$

where r_2 is given in (8).

Proof From (3), $\frac{d}{dx} q_{\lambda,z}(x) > 0$ when $(x - z)(x + \epsilon) + \lambda > 0$. It can be verified directly that this holds for every x for $z \in [0, \frac{\lambda}{\epsilon}]$. That is, $q_{\lambda,z}$ is increasing, and therefore the minimum must occur at 0.

The factorization (6) is defined for $z \in [\frac{\lambda}{\epsilon}, \infty)$. By the discussion above, $r_1(z) < 0$ on this interval, so $\frac{d}{dx} q_{\lambda,z}(x)$ is negative for $x < r_2(z)$ and positive for $x > r_2(z)$. Thus, $x = r_2(z)$ is the minimizer. □

Recall from Lemma 4 that $\text{prox}_{\lambda g}$ is single-valued except possibly at $\pm z_*$ for some $z_* \in \mathbb{R}$. As we will see in the next result, the point z_* does exist and can be efficiently located when $\sqrt{\lambda} > \epsilon$. In this scenario, $\text{prox}_{\lambda g}$ is described in the following result.

Proposition 2 *If $\sqrt{\lambda} > \epsilon$, then for any given $z \in \mathbb{R}$*

$$\text{prox}_{\lambda g}(z) = \begin{cases} \{0\}, & \text{if } |z| < z_*; \\ \{0, \text{sgn}(z) r_2(z_*)\}, & \text{if } |z| = z_*; \\ \{\text{sgn}(z) r_2(|z|)\}, & \text{if } |z| > z_*, \end{cases} \tag{10}$$

z_* is the root of the function

$$r(z) := q_{\lambda,z}(r_2(z)) - q_{\lambda,z}(0) \tag{11}$$

on the interval $[2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon}]$ and $r_2(z)$ is given in (8).

Proof From equation (5), we may directly verify that $\frac{dq}{dx} > 0$ for $z \in (0, 2\sqrt{\lambda} - \epsilon)$. This implies that $x = 0$ is the unique minimizer for this range of z.

Next, we focus on the situation of $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon}]$. In this case, the factorization (6) holds. Furthermore, $r_1(2\sqrt{\lambda} - \epsilon) = r_2(2\sqrt{\lambda} - \epsilon) = \sqrt{\lambda} - \epsilon$. Because $r_1(z)$ is strictly decreasing, $x = r_1(z)$ implies that $x \leq \sqrt{\lambda} - \epsilon$. By Lemma 2, $\frac{d^2 q}{dx^2} < 0$ for $x < \sqrt{\lambda} - \epsilon$, implying that

$x = r_1(z)$ is a local maximum. By analogous reasoning, since $r_2(z)$ is increasing and q is convex on $(\sqrt{\lambda} - \epsilon, \infty)$, $x = r_2(z)$ must be a local minimum. Therefore the elements of $\text{prox}_{\lambda g}(z)$ must be 0, $r_2(z)$, or both.

To determine when the origin and/or r_2 are the minimizers of $q_{\lambda,z}$, we take a closer look at the function $r(z)$ in (11) for $2\sqrt{\lambda} - \epsilon < z \leq \frac{\lambda}{\epsilon}$. One can check directly that

$$r(2\sqrt{\lambda} - \epsilon) = -\log\left(\frac{\epsilon}{\sqrt{\lambda}}\right) + \left(\frac{\epsilon^2}{2\lambda} + \frac{2\epsilon}{\sqrt{\lambda}} - \frac{3}{2}\right) > 0$$

and

$$r\left(\frac{\lambda}{\epsilon}\right) = \frac{\epsilon^2}{2\lambda} + \log\left(\frac{\lambda}{\epsilon^2}\right) - \frac{\lambda}{2\epsilon^2} < 0$$

whenever $\sqrt{\lambda} > \epsilon$. Hence, the function r has at least one root $z_* \in (2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$. This means that $\{0, r_2(z_*)\} = \text{prox}_{\lambda g}(z_*)$. By Lemma 4, z_* is the only root of r on the interval $[2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon}]$. We conclude that $\text{prox}_{\lambda g}(z) = \{0\}$ when $z \in [2\sqrt{\lambda} - \epsilon, z_*)$ and $\text{prox}_{\lambda g}(z) = \{r_2(z)\}$ when $z \in (z_*, \frac{\lambda}{\epsilon}]$.

Finally, we consider $z \in (\frac{\lambda}{\epsilon}, \infty)$. Note that $r_1(z) < 0$ on this interval, so by Lemma 1, the proximity operator cannot be $r_1(z)$. By the previous discussion, we know that $r(z) < 0$ for z in this interval. Thus, $\text{prox}_{\lambda g}(z) = \{\text{sgn}(z) r_2(z)\}$. □

From the above proof, we know that z_* is the unique root of r on $[2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon}]$ and depends on parameters λ and ϵ only. Furthermore, from $r(2\sqrt{\lambda} - \epsilon) r(\frac{\lambda}{\epsilon}) < 0$, z_* can easily be found by the bisection method. We further remark that the z_* is simply given as $2\sqrt{\lambda} - \epsilon$ in [27], which clearly is incorrect.

Equation (10) for $\text{prox}_{\lambda g}(z)$ in the case of $\sqrt{\lambda} > \epsilon$ will reduce to Equation (10) when $\sqrt{\lambda} - \epsilon \to 0+$, since $z_* \in (2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$ should converge to $\sqrt{\lambda} = \frac{\lambda}{\epsilon}$.

Figure 2 displays the proximity operator $\text{prox}_{\lambda g}$ for two choices of (λ, ϵ). Figure 2a depicts $\text{prox}_{\lambda g}$ for $\sqrt{\lambda} \leq \epsilon$ with $(\lambda, \epsilon) = (2, 3)$ as in Proposition 1. Figure 2b depicts $\text{prox}_{\lambda g}$ for $\sqrt{\lambda} > \epsilon$ with $(\lambda, \epsilon) = (3, 1)$, corresponding to Proposition 2. In both situations, $\text{prox}_{\lambda g}(z) = \{0\}$ for z in a neighborhood of the origin, thus g is a sparsity promoting function as defined in [22].

Figure 3 illustrates the graph of $\text{prox}_{\lambda g}(z)$ for fixed λ and $z > 0$ with respect to varying ϵ (horizontal axis) in three situations. The symbol "○" represents $\epsilon = \sqrt{\lambda}$ while the symbol "x" represents $\epsilon = \frac{\lambda}{\epsilon}$. In the first situation, we assume that $\sqrt{\lambda} < \frac{\lambda}{z}$. If $\epsilon \in (0, \sqrt{\lambda})$, we have $z < \sqrt{\lambda} < 2\sqrt{\lambda} - \epsilon$, therefore, $\text{prox}_{\lambda g}(z) = 0$ Proposition 2; If $\epsilon \in [\sqrt{\lambda}, \frac{\lambda}{z})$, we know $\sqrt{\lambda} \leq \epsilon$ and $z < \frac{\lambda}{\epsilon}$, hence $\text{prox}_{\lambda g}(z) = 0$ by Proposition 1; if $\epsilon \in [\frac{\lambda}{z}, \infty)$, we get $\sqrt{\lambda} \leq \epsilon$ and $z \geq \frac{\lambda}{\epsilon}$, hence $\text{prox}_{\lambda g}(z) = r_2(z)$ from Proposition 1. As shown in Fig. 3a, the value of $\text{prox}_{\lambda g}(z)$ continuously changes with respect to the parameter ϵ. In the second situation, we assume that $\sqrt{\lambda} = \frac{\lambda}{z}$. Using the arguments in the first situation, we conclude that the value of $\text{prox}_{\lambda g}(z)$ continuously changes with respect to the parameter ϵ as displayed in Fig. 3b. The behavior of $\text{prox}_{\lambda g}(z)$ in the last situation of $\sqrt{\lambda} > \frac{\lambda}{z}$ is quite different from the previous ones. If $\epsilon \in (0, \frac{\lambda}{z})$, $\text{prox}_{\lambda g}(z)$ will jump at, say ϵ^*, in $(0, \frac{\lambda}{z})$; For all $\epsilon \in (\epsilon^*, \infty)$, we will have $\text{prox}_{\lambda g}(z) = r_2(z)$ due to Propositions 1 and 2.

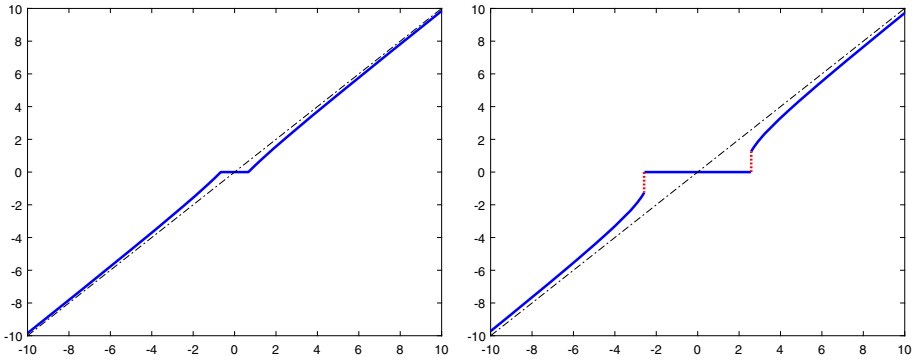

Fig. 2 The graphs (solid lines) of $\text{prox}_{\lambda g}$ for **a** $\sqrt{\lambda} \leq \epsilon$ with $(\lambda, \epsilon) = (2, 3)$ and **b** $\sqrt{\lambda} > \epsilon$ with $(\lambda, \epsilon) = (3, 1)$. The dotted lines are the graph of the identity mapping

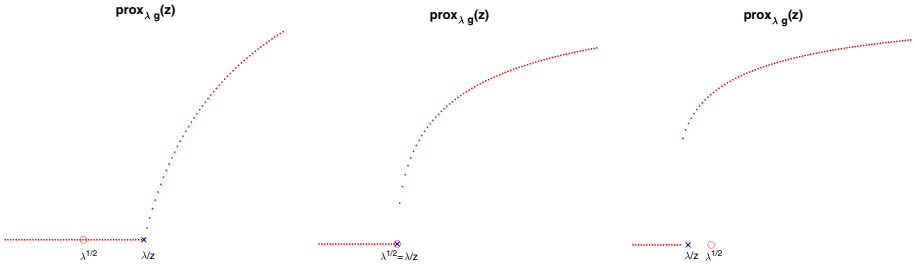

Fig. 3 The graphs (dotted lines) of $\text{prox}_{\lambda g}(z)$ with fixed λ and $z > 0$ with respect to varying ϵ (horizontal axis) in the situation of **a** $\sqrt{\lambda} < \frac{\lambda}{z}$; **b** $\sqrt{\lambda} = \frac{\lambda}{z}$; and **c** $\sqrt{\lambda} > \frac{\lambda}{z}$

Both Propositions 1 and 2 show that for large enough $|z|$, $r_2(|z|)$ the absolute value of the only element of $\text{prox}_{\lambda g}(z)$ has

$$r_2(|z|) \approx |z| - \frac{\lambda}{(|z| + \epsilon)} - \frac{\lambda^2}{(|z| + \epsilon)^3}$$

through Taylor's expansion the term $\sqrt{1 - \frac{4\lambda}{(|z|+\epsilon)^2}}$ in $\sqrt{\frac{1}{4}(|z| + \epsilon)^2 - \lambda} = \frac{1}{2}(|z| + \epsilon)\sqrt{1 - \frac{4\lambda}{(|z|+\epsilon)^2}}$. This indicates that the operator $\text{prox}_{\lambda g}$ is nearly unbiased for large values [14, 22], which supports the use of g in applications to replace the ℓ_0 norm. We are not aware of any existing work quantitatively explaining it in this way. Figure 2 further illustrates this claim.

We are ready to present the solution to problem (P_1).

Theorem 3 *For each $z \in \mathbb{R}^n$, $\epsilon > 0$, and $\lambda \leq 0$, and for f as defined by (1),*

$$\text{prox}_{\lambda f}(z) = \text{prox}_{\lambda g}(z_1) \times \cdots \times \text{prox}_{\lambda g}(z_n), \tag{12}$$

where $\text{prox}_{\lambda g}$ is given by Proposition 1 or 2. Moreover, if $x^\star \in \text{prox}_{\lambda f}(z)$, then $x_i^\star \in \text{prox}_{\lambda g}(z_i)$, $i = 1, 2, \ldots, n$.

Proof The results follow immediately from the relation (1) and the definition of proximity operator. Since $\text{prox}_{\lambda g}$ has an explicit expression, so does $\text{prox}_{\lambda f}$. □

3 Solutions to Optimization Problem (P_2)

In this section, we present the solution to log-sum penalized low-rank recovery problem.

Let $\mathcal{M}_{m,n}$ denote the Euclidean space of $m \times n$ real matrices, with inner product $\langle X, Y\rangle = \text{tr}X^\top Y$. The Frobenius norm is denoted by $\|\cdot\|_F$. For any matrix $X \in \mathcal{M}_{m,n}$, let X_{ij} denote its (i, j)-th entry. For any vector $x \in \mathbb{R}^{m\wedge n}$, let $\text{Diag}(x)$ denote the $m \times n$ matrix with $(\text{Diag}(x))_{ii} = x_i$ for all i, and $(\text{Diag}(x))_{ij} = 0$ for $i \neq j$. For any $X \in \mathcal{M}_{m,n}$, we define $\sigma(X) := (\sigma_1(X), \sigma_2(X), \ldots, \sigma_{m\wedge n}(X))^\top$, where

$$\sigma_1(X) \geq \sigma_2(X) \geq \ldots \geq \sigma_{m\wedge n}(X)$$

are the ordered singular values of X. Denote by $\mathcal{O}(X)$ the set of all pairs (U, V):

$$\mathcal{O}(X) := \left\{(U, V) \in \mathcal{M}_{m,m} \times \mathcal{M}_{n,n} : U^\top U = I, V^\top V = I, X = U\text{Diag}(\sigma(X))V^\top\right\}.$$

That is, for any pair $(U, V) \in \mathcal{O}(X)$, $U\text{Diag}(\sigma(X))V^\top$ is a singular value decomposition of X.

As it will be useful in the proof of Theorem 4, we explicitly describe how the order-preserving properties of $\text{prox}_{\lambda g}$ extend to $\text{prox}_{\lambda f}$ in the following lemma.

Lemma 5 *Define* $\mathbb{R}^n_\downarrow := \{x \in \mathbb{R}^n : x_1 \geq x_2 \geq \cdots \geq x_n \geq 0\}$. *For any* $\lambda > 0$ *and* $\epsilon > 0$, *if* $x \in \mathbb{R}^n_\downarrow$, *then* $\text{prox}_{\lambda f}(x) \subset \mathbb{R}^n_\downarrow$.

Proof This is a direct consequence of Lemma 3 and the fact that $\text{prox}_{\lambda f}$ applies $\text{prox}_{\lambda g}$ coordinatewise as in (12). □

In general, it may not be possible to compute the proximity operator of a composition of functions based on the proximity operators of the components. However, the following theorem tells us that we may compute $\text{prox}_{\lambda(f\circ\sigma)}(Z)$ from $\text{prox}_{\lambda f}(\sigma(Z))$. While revising this article, we became aware of a result giving this general form for select nonconvex functions composed with the singular value mapping [26].

Theorem 4 *For each* $Z \in \mathcal{M}_{m,n}$, $\epsilon > 0$, *and* $\lambda > 0$, *if* $X^\star \in \text{prox}_{\lambda f\circ\sigma}(Z)$, *then there exist a pair* $(U, V) \in \mathcal{O}(Z)$ *and a vector* $d \in \text{prox}_{\lambda f}(\sigma(Z))$ *such that*

$$X^\star = U\text{Diag}(d)V^\top. \tag{13}$$

Proof We show that $X^\star$ in (13) indeed is a solution to problem (P_2). Problem (P_2) can be equivalently reformulated as

$$\min_{d\in\mathbb{R}^{m\wedge n}_\downarrow}\left\{\min_{X\in\mathcal{M}_{m,n},\text{Diag}(\sigma(X))=d}\left\{\frac{1}{2\lambda}\|X - Z\|_F^2 + \sum_{i=1}^{m\wedge n}\log\left(1 + \frac{d_i}{\epsilon}\right)\right\}\right\}.$$

Note that

$$\begin{aligned}\|X - Z\|_F^2 &= \text{tr}(X^\top X) - 2\text{tr}(X^\top Z) + \text{tr}(Z^\top Z)\\ &= \sum_{i=1}^{m\wedge n} d_i^2 - 2\text{tr}(X^\top Z) + \sum_{i=1}^{m\wedge n}\sigma_i(Z)^2\end{aligned}$$

$$\geq \sum_{i=1}^{m\wedge n} d_i^2 - 2\sigma(Z)^\top d + \sum_{i=1}^{m\wedge n} \sigma_i(Z)^2.$$

The last inequality is due to von Neumann's trace inequality (see [19]). Equality holds when X admits the singular value decomposition $X = U\mathrm{Diag}(d)V^\top$, where $(U, V) \in \mathcal{O}(Z)$. Then the optimization problem reduces to

$$\min_{d\in\mathbb{R}^{m\wedge n}_{\downarrow}} \left\{ \sum_{i=1}^{m\wedge n} \left(\frac{1}{2\lambda}(d_i - \sigma_i(Z))^2 + \log\left(1 + \frac{d_i}{\epsilon}\right)\right)\right\}.$$

The objective function is completely separable and is minimized only when $d_i \in \mathrm{prox}_{\lambda g}(\sigma_i(Z))$. This is a feasible solution because $\sigma(Z) \in \mathbb{R}^{m\wedge n}_{\downarrow}$ implies $\mathrm{prox}_{\lambda f}(\sigma(Z)) \subset \mathbb{R}^{m\wedge n}_{\downarrow}$ by Lemma 5. This completes the proof. □

We remark that the main ideas in the above proof are from [9]. The result in Theorem 4 can be applied, for example in [4, 11, 12, 15], to avoid an inner loop for evaluating $\mathrm{prox}_{\lambda f\circ\sigma}$.

4 Analysis of Majorization-Minimization Algorithms for Computing $\mathbf{prox}_{\lambda g}$

An iterative algorithm in the fashion of classical majorization-minimization procedure was adopted in [12] to evaluate $\mathrm{prox}_{\lambda g}$ by generating and solving a sequence of convex optimization problems. At each iteration, the non-convex function g is approximated by means of a majorizing convex surrogate function. More precisely, if $x^{(k)}$ is the value of the current solution, the chosen convex surrogate function of g is $g(x^{(k)}) + \frac{|x|-|x^{(k)}|}{\epsilon+|x^{(k)}|}$, the first-order approximation of $g(x)$ at $x^{(k)}$. With it, the next iteration $x^{(k+1)}$ from given z and $x^{(k)}$ is

$$x^{(k+1)} = \mathrm{argmin}\left\{ \frac{1}{2\lambda}(x - z)^2 + g(x^{(k)}) + \frac{|x| - |x^{(k)}|}{\epsilon + |x^{(k)}|} : x \in \mathbb{R}\right\},$$

which can be written as

$$x^{(k+1)} = \mathrm{prox}_{\frac{\lambda}{\epsilon+|x^{(k)}|}|\cdot|}(z) = \begin{cases} 0, & |z| \leq \frac{\lambda}{\epsilon+|x^{(k)}|}; \\ \mathrm{sgn}(z)(|z| - \frac{\lambda}{\epsilon+|x^{(k)}|}), & |z| > \frac{\lambda}{\epsilon+|x^{(k)}|}. \end{cases} \tag{14}$$

By ignoring the terms which do not depend on x in the above optimization problem, the resulting expression, called reweighted ℓ_1, was used in [6] to approximate the ℓ_0 norm.

The sequence $\{x^{(k)}\}$ generated by the iterative majorization-minimization procedure (14) converges to a critical point of the objective function $\frac{1}{2\lambda}(\cdot - z)^2 + g(\cdot)$, see, e.g., [3]. We go one step further than the previous result and show that not only the sequence $\{x^{(k)}\}$ is always convergent, but also its limit depends on the initialization $x^{(0)}$ and the relationship of z with the parameters λ and ϵ. Lemmas 6 – 11 in the following describe the possible convergence behavior of (14). This is then compared to the true solution in Theorems 5 and 6. In particular, we identify the intervals where (14) will not achieve the true solution. These intervals are explicitly determined in terms of the initial guess $x^{(0)}$ and parameters λ and ϵ.

The following two theorems summarize our main results. The proofs of these results as well as relevant technical lemmas are given in Sect. 4.1.

Theorem 5 *For $\sqrt{\lambda} \leq \epsilon$, the iteratively reweighted algorithm will converge to the accurate solution to $\mathrm{prox}_{\lambda g}(z)$ for all $z \in \mathbb{R}$.*

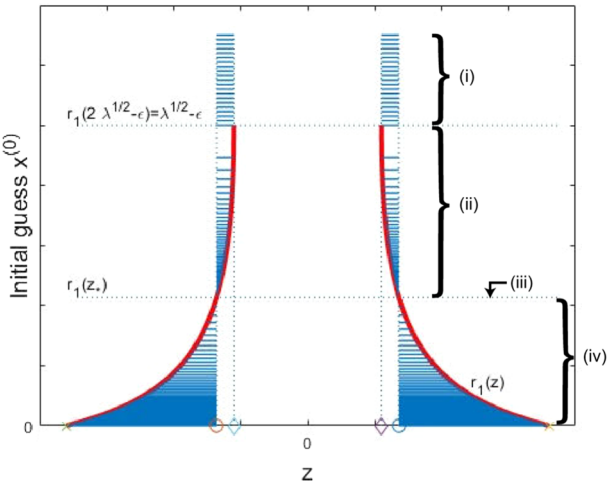

Fig. 4 Illustration of Theorem 6: the intervals on which the iteratively reweighted algorithm will fail. Here, '×', '∘', and '◇' denote $\pm\frac{\lambda}{\epsilon}$, $\pm z_*$, and $\pm(2\sqrt{\lambda}-\epsilon)$, respectively. The left and right red curves are the graphs of the function $r_1(-z)$ and $r_1(z)$, respectively

In the nonconvex setting, we see that the iteratively reweighted algorithm may not converge to the true value of $\text{prox}_{\lambda g}$. In this case, the regions where the algorithm fails depend on the root z_* (as defined by (11)) and the function r_1 (as defined by (7)). Note that when $\lambda > \sqrt{\epsilon}$, r_1 is a bijection from $[2\sqrt{\lambda}-\epsilon, \infty)$ to $(-\infty, \sqrt{\lambda}-\epsilon]$. Therefore, the function r_1^{-1} the inverse of r_1 exists and maps $(-\infty, \sqrt{\lambda}-\epsilon]$ to $[2\sqrt{\lambda}-\epsilon, \infty)$.

Theorem 6 *For $\sqrt{\lambda} > \epsilon$ and an initial point $x^{(0)}$, the following statements hold for the iteratively reweighted algorithm.*

(i) *If $x^{(0)} \geq \sqrt{\lambda}-\epsilon$, the iteratively reweighted algorithm will not converge to* $\text{prox}_{\lambda g}(z)$ *for z in $(-z_*, -2\sqrt{\lambda}+\epsilon] \cup [2\sqrt{\lambda}-\epsilon, z_*)$.*

(ii) *If $\sqrt{\lambda}-\epsilon > x^{(0)} > r_1(z_*)$, the iteratively reweighted algorithm will not converge to* $\text{prox}_{\lambda g}(z)$ *for z in $(-z_*, -r_1^{-1}(x^{(0)})] \cup [r_1^{-1}(x^{(0)}), z_*)$.*

(iii) *If $x^{(0)} = r_1(z_*)$, the iteratively reweighted algorithm will not converge to* $\text{prox}_{\lambda g}(z)$ *for z in $\{-z_*, z_*\}$.*

(iv) *If $r_1(z_*) > x^{(0)} \geq 0$, the iteratively reweighted algorithm will not converge to* $\text{prox}_{\lambda g}(z)$ *for z in $[-r_1^{-1}(x^{(0)}), -z_*) \cup (z_*, r_1^{-1}(x^{(0)})]$.*

The results in Theorem 6 can be visualized through Fig. 4. For each initial point $x^{(0)}$, the intervals for which $\text{prox}_{\lambda g}(z)$ disagrees with the reweighted ℓ_1 solution are represented by the solid horizontal lines. For example, the top segment of the figure corresponds to item (i) of the theorem.

4.1 Proofs of Theorems 5 and 6

We begin with several technical lemmas describing the convergence of the iteratively reweighted algorithm. The limit points of this algorithm can then be compared to $\text{prox}_{\lambda g}$ directly.

Lemma 6 *Let the sequence $\{x^{(k)}\}$ be generated by the iterative scheme (14) for a given $z > 0$ and an initial guess $x^{(0)} \geq 0$. Suppose that $x^{(k)} > 0$ for all $k \geq 1$. Then the sequence $\{x^{(k)}\}$ is increasing (resp. decreasing) if $x^{(1)} > x^{(0)}$ (resp. $x^{(1)} < x^{(0)}$); this sequence is constant if $x^{(1)} = x^{(0)}$.*

Proof Since $x^{(k)} > 0$ for all $k \geq 1$, one has from the iterative scheme (14) that $x^{(k+1)} = z - \frac{\lambda}{\epsilon + x^{(k)}}$ for all $k \geq 0$. Therefore,

$$x^{(k+1)} - x^{(k)} = \frac{\lambda(x^{(k)} - x^{(k-1)})}{(\epsilon + x^{(k)})(\epsilon + x^{(k-1)})} = \frac{\lambda^k (x^{(1)} - x^{(0)})}{\Pi_{i=1}^{k}((\epsilon + x^{(i)})(\epsilon + x^{(i-1)}))}.$$

All statements immediately follow from the above equation. □

Lemma 7 *Let the sequence $\{x^{(k)}\}$ be generated by the iterative scheme (14) for a given $z \geq 0$ and an initial guess $x^{(0)} \geq 0$. If there exists $k_0 \geq 0$, such that $x^{(k_0)} = 0$, then the sequence $\{x^{(k)}\}$ converges to 0 if $z \leq \frac{\lambda}{\epsilon}$ and to $r_2(z)$ if $z > \frac{\lambda}{\epsilon}$.*

Proof Without loss of generality, let us assume $k_0 = 0$, i.e, $x^{(0)} = 0$. We have

$$x^{(1)} = \begin{cases} 0, & \text{if} z \leq \frac{\lambda}{\epsilon}; \\ z - \frac{\lambda}{\epsilon}, & \text{if} z > \frac{\lambda}{\epsilon}. \end{cases}$$

Obviously, if $z \leq \frac{\lambda}{\epsilon}$, $x^{(k)} = 0$ for all $k \geq 0$, that is, $\{x^{(k)}\}$ converges to 0. If $z > \frac{\lambda}{\epsilon}$, then $x^{(1)} > x^{(0)} = 0$, yielding $x^{(k+1)} = z - \frac{\lambda}{\epsilon + x^{(k)}} > 0$ for all $k \geq 0$. So $\{x^{(k)}\}$ is increasing by Lemma 6 and converges, say to a positive number $x^{(\infty)}$, due to $0 \leq x^{(k)} < z$ for all $k \geq 0$. We have $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$. So $x^{(\infty)}$ must be $r_2(z)$ for $z > \frac{\lambda}{\epsilon}$. □

The next identity is useful in the following discussion. For given $x^{(0)} \geq 0$ and $z > 0$, if $x^{(1)} = z - \frac{\lambda}{\epsilon + x^{(0)}} > 0$, then

$$x^{(1)} - x^{(0)} = \begin{cases} -\frac{1}{\epsilon + x^{(0)}}((x^{(0)} - \frac{1}{2}(z - \epsilon))^2 + (\lambda - \frac{1}{4}(z + \epsilon)^2)), & \text{if} z < 2\sqrt{\lambda} - \epsilon; \\ -\frac{1}{\epsilon + x^{(0)}}(x^{(0)} - r_1(z))(x^{(0)} - r_2(z)), & \text{if} z \geq 2\sqrt{\lambda} - \epsilon, \end{cases} \tag{15}$$

where $r_1(z)$ and $r_2(z)$ are given in (7) and (8), respectively.

Lemma 8 *Let the sequence $\{x^{(k)}\}$ be generated by the iterative scheme (14) for a given $z \geq \frac{\lambda}{\epsilon}$ and an initial guess $x^{(0)} > 0$. Then, $x^{(k)} > 0$ for all $k \geq 0$ and the sequence $\{x^{(k)}\}$ converges to $r_2(z)$.*

Proof For $z \geq \frac{\lambda}{\epsilon}$ and $x^{(0)} > 0$, we know $x^{(1)} = z - \frac{\lambda}{\epsilon + x^{(0)}} > 0$. As a consequence, it also implies $x^{(k)} > 0$ for all $k \geq 0$. To show the convergence of the sequence $\{x^{(k)}\}$, we compare the values of $x^{(0)}$ and $x^{(1)}$ from (15) in order to infer the monotonicity of the sequence based on Lemma 6. To this end, our discussion is conducted for two cases: (i) $z > \frac{\lambda}{\epsilon}$ or $z = \frac{\lambda}{\epsilon}$ and $\sqrt{\lambda} > \epsilon$; and (ii) $z = \frac{\lambda}{\epsilon}$ and $\sqrt{\lambda} \leq \epsilon$. The following facts are useful: $r_1(z) < 0 < r_2(z)$ for all $z > \frac{\lambda}{\epsilon}$, and

$$r_1\left(\frac{\lambda}{\epsilon}\right) = \begin{cases} \frac{\lambda}{\epsilon} - \epsilon, & \text{if} \sqrt{\lambda} \leq \epsilon; \\ 0, & \text{if} \sqrt{\lambda} > \epsilon \end{cases} \quad \text{and} \quad r_2\left(\frac{\lambda}{\epsilon}\right) = \begin{cases} 0, & \text{if} \sqrt{\lambda} \leq \epsilon; \\ \frac{\lambda}{\epsilon} - \epsilon, & \text{if} \sqrt{\lambda} > \epsilon. \end{cases}$$

Case (i): $z > \frac{\lambda}{\epsilon}$ or $z = \frac{\lambda}{\epsilon}$ and $\sqrt{\lambda} > \epsilon$. Hence, $r_2(z) > 0$ and $r_2(z)$ is the only positive solution of $x = z - \frac{\lambda}{\epsilon + x}$.

- If $x^{(0)} \in (0, r_2(z))$, one has $x^{(1)} > x^{(0)}$ from (15). We can conclude that $x^{(k+1)} > x^{(k)}$ for all $k \geq 0$ and the sequence $\{x^{(k)}\}$ converges, say to $x^{(\infty)}$, which is a positive number satisfying $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$. So $x^{(\infty)} = r_2(z)$.
- If $x^{(0)} = r_2(z) > 0$, then $x^{(k+1)} = r_2(z)$ for all $k \geq 0$. Hence, the limit of the sequence $\{x^{(k)}\}$ is $r_2(z)$.
- If $x^{(0)} \in (r_2(z), \infty)$, then $x^{(1)} < x^{(0)}$ from (15). We conclude $x^{(k+1)} < x^{(k)}$ for all $k \geq 0$ and the sequence $\{x^{(k)}\}$ converges, say to $x^{(\infty)} \geq 0$, satisfying $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$. Hence, $x^{(\infty)}$ must be $r_2(z)$.

Case (ii): $z = \frac{\lambda}{\epsilon}$ and $\sqrt{\lambda} \leq \epsilon$. If $\sqrt{\lambda} \leq \epsilon$, then $r_2(z) = 0$, so $x^{(1)} < x^{(0)}$ from (15). Then $x^{(k+1)} < x^{(k)}$ for all $k \geq 0$ and the sequence $\{x^{(k)}\}$ converges, say to $x^{(\infty)} \geq 0$, satisfying $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$. Hence, $x^{(\infty)}$ must be $r_2(z) = 0$.

From the discussion above, we know that the sequence $\{x^{(k)}\}$ converges to $r_2(z)$. □

It can be concluded from Lemmas 7 and 8 that the sequence $\{x^{(k)}\}$ always converges to $r_2(z)$ for $z > \frac{\lambda}{\epsilon}$ regardless of the initial guess $x^{(0)}$. The next lemma shows that the sequence $\{x^{(k)}\}$ always converges to 0 for all $z \in (0, 2\sqrt{\lambda} - \epsilon)$ if $2\sqrt{\lambda} - \epsilon > 0$, independent of the initial guess $x^{(0)}$.

Lemma 9 *Suppose* $2\sqrt{\lambda} - \epsilon > 0$. *Let the sequence* $\{x^{(k)}\}$ *be generated by the iterative scheme* (14) *for a given* $z \in (0, 2\sqrt{\lambda} - \epsilon)$ *and an initial guess* $x^{(0)} \geq 0$. *Then the sequence* $\{x^{(k)}\}$ *converges to* 0.

Proof Notice that $2\sqrt{\lambda} - \epsilon \leq \frac{\lambda}{\epsilon}$ for all positive λ and ϵ. If there exists $k_0 \geq 0$ such that $x^{(k_0)} = 0$, by Lemma 7, the sequence $\{x^{(k)}\}$ converges to 0.

Now, assume that all elements $x^{(k)}$ are positive. By (15), we have $x^{(1)} < x^{(0)}$ for all $z \in (0, 2\sqrt{\lambda} - \epsilon)$ and an initial guess $x^{(0)} > 0$. By Lemma 7, the sequence $\{x^{(k)}\}$ is decreasing and convergent. Suppose that $\lim_{k \to \infty} x^{(k)} = x^{(\infty)} \geq 0$. Then, $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$ due to all $x^{(k)} > 0$, but it is impossible for $z \in (0, 2\sqrt{\lambda} - \epsilon)$. We conclude that the sequence $\{x^{(k)}\}$ converges to 0. □

The next two lemmas deal with the convergence of the sequence $\{x^{(k)}\}$ for $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$.

Lemma 10 *Suppose parameters* λ *and* ϵ *satisfying conditions* $2\sqrt{\lambda} - \epsilon > 0$ *and* $\sqrt{\lambda} \leq \epsilon$. *Let the sequence* $\{x^{(k)}\}$ *be generated by the iterative scheme* (14) *for a given* $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$ *and an initial guess* $x^{(0)} \geq 0$. *Then the sequence* $\{x^{(k)}\}$ *converges to* 0.

Proof If there exists $k_0 \geq 0$ such that $x^{(k_0)} = 0$, by Lemma 7, the sequence $\{x^{(k)}\}$ converges to 0.

Now, assume that all elements $x^{(k)}$ are positive. Since $r_1(z) < 0$ and $r_2(z) < 0$ for $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$, we know from (15) that $x^{(1)} < x^{(0)}$ if $x^{(0)} \in (0, \infty)$. Then, the sequence $\{x^{(k)}\}$ is decreasing by Lemma 6, and therefore convergent. Suppose that $\lim_{k \to \infty} x^{(k)} = x^{(\infty)} \geq 0$. One has $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$. Therefore, $x^{(\infty)}$ should be $r_1(z)$ or $r_2(z)$. However, it is impossible due to both $r_1(z)$ and $r_2(z)$ are negative for $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$. □

Lemma 11 *Suppose* $\sqrt{\lambda} > \epsilon$. *Let the sequence* $\{x^{(k)}\}$ *be generated by the iterative scheme* (14) *for a given* $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$ *and an initial guess* $x^{(0)} \geq 0$. *Then the following statements hold*

(i) *If* $x^{(0)} \in (0, r_1(z))$*, then the sequence* $\{x^{(k)}\}$ *converges to* 0*; If* $x^{(0)} = r_1(z)$*, then the sequence* $\{x^{(k)}\}$ *converges to* $r_1(z)$.
(ii) *If* $x^{(0)} \in (r_1(z), \infty)$*, then the sequence* $\{x^{(k)}\}$ *converges to* $r_2(z)$.

Proof First, we show that

$$\frac{\lambda}{z} - \epsilon < r_1(z) \tag{16}$$

holds for all $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$. Actually, by the definition of $r_1(z)$ in (7) and through some manipulations, inequality (16) is equivalent to $\sqrt{(z+\epsilon)^2 - 4\lambda} < (z+\epsilon) - \frac{2\lambda}{z}$. Since the expression $(z+\epsilon) - \frac{2\lambda}{z}$ is positive for $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$, squaring the previous inequality followed by some simplifications yields $z < \frac{\lambda}{\epsilon}$, which is obviously true.

(i) If $x^{(0)} \le \frac{\lambda}{z} - \epsilon$, then $x^{(1)} = 0$ from (14). By Lemma 7, the sequence $\{x^{(k)}\}$ converges to 0. If $x^{(0)} > \frac{\lambda}{z} - \epsilon$, then $0 < x^{(1)} \le x^{(0)}$ from (15). Using a similar argument above, if there exists $k_0 \ge 0$ such that $x^{(k_0))} = 0$, by Lemma 7, the sequence $\{x^{(k)}\}$ converges to 0. Now, assume that all elements $x^{(k)}$ are positive. Then the sequence $\{x^{(k)}\}$ is decreasing and convergent by Lemma 6. Suppose that $\lim_{k\to\infty} x^{(k)} = x^{(\infty)} \ge 0$. Then, $x^{(\infty)}$ must be strictly less than $r_1(z)$ and $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$, which, however, contradict to each other. Hence, the sequence $\{x^{(k)}\}$ converges to 0.

If $x^{(0)} = r_1(z)$, so $x^{(0)} > \frac{\lambda}{z} - \epsilon$ which implies that $x^{(1)} = x^{(0)}$. In this case, $\{x^{(k)}\}$ is a constant sequence and its limit is $r_1(z)$.

(ii) If $x^{(0)} \in (r_1(z), \infty)$, we have $x^{(0)} > \frac{\lambda}{z} - \epsilon$ by (16). From (15), we have $x^{(1)} \ge x^{(0)}$ if $x^{(0)} \in (r_1(z), r_2(z)]$. So the sequence $\{x^{(k)}\}$ is increasing and must converge to $r_2(z)$.

From (15), we have $x^{(1)} < x^{(0)}$ if $x^{(0)} \in (r_2(z), \infty)$. Further, we can show that $x^{(1)} > r_2(z)$. Indeed, from $x^{(1)} - r_2(z) = z - \frac{\lambda}{\epsilon + x^{(0)}} - r_2(z)$ and the definition of $r_2(z)$, we have after some simplification

$$x^{(1)} - r_2(z) = \frac{2(x^{(0)} - r_2(z))}{(z+\epsilon) + \sqrt{(z+\epsilon)^2 - 4\lambda}} > 0$$

holds for $z \in [2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon})$ and $x^{(0)} \ge r_2(z)$. Hence, the sequence $\{x^{(k)}\}$ is decreasing and $\lim_{k\to\infty} x^{(k)} = x^{(\infty)} \ge r_2(z)$. We further have $x^{(\infty)} = z - \frac{\lambda}{\epsilon + x^{(\infty)}}$ which implies $x^{(\infty)} = r_2(z)$. □

Proof of Theorem 5 The result follows directly from Proposition 1 and Lemmas 7–10. □

Proof of Theorem 6 By Proposition 1, Lemmas 7, and 9, the iteratively reweighted algorithm provides the accurate solution to $\text{prox}_{\lambda g}(z)$ when $|z| > \frac{\lambda}{\epsilon}$ or $|z| < 2\sqrt{\lambda} - \epsilon$. The rest of the proof will focus on the situation for $z \in [2\sqrt{\lambda} - \epsilon), \frac{\lambda}{\epsilon}]$ due to Lemma 1.

(i) We know that $\sqrt{\lambda} - \epsilon = r_1(2\sqrt{\lambda} - \epsilon) > r_1(z)$ for all $z \in (2\sqrt{\lambda} - \epsilon, \frac{\lambda}{\epsilon}]$. From Lemmas 8 and 11, the limit of the sequence generated by the algorithm is $r_2(z)$ for $z \in [2\sqrt{\lambda} - \epsilon), \frac{\lambda}{\epsilon}]$. Hence, the limit does not match to the true solution $\text{prox}_{\lambda g}(z)$ when $z \in (-z_*, -2\sqrt{\lambda} + \epsilon] \cup [2\sqrt{\lambda} - \epsilon, z_*)$.
(ii) Notice that $r_1(z)$ is strictly decreasing on $[2\sqrt{\lambda} - \epsilon), \frac{\lambda}{\epsilon}]$. From $\sqrt{\lambda} - \epsilon > x^{(0)} > r_1(z_*)$, we have $2\sqrt{\lambda} - \epsilon < r_1^{-1}(x^{(0)}) < z_*$; $x^{(0)} < r_1(z)$ for all $z \in [2\sqrt{\lambda} - \epsilon, r_1^{-1}(x^{(0)}))$; and $x^{(0)} > r_1(z)$ for all $z \in (r_1^{-1}(x^{(0)}), \frac{\lambda}{\epsilon}]$. Accordingly, by Lemma 11, the limit $x^{(\infty)}$ of the

sequence generated by the algorithm is

$$x^{(\infty)} = \begin{cases} 0, & \text{if} z \in [2\sqrt{\lambda} - \epsilon, r_1^{-1}(x^{(0)})); \\ x^{(0)}, & \text{if} z = r_1^{-1}(x^{(0)}); \\ r_2(z), & \text{if} z \in (r_1^{-1}(x^{(0)}), \frac{\lambda}{\epsilon}]. \end{cases} \tag{17}$$

Hence, the limit does not match to the true solution $\text{prox}_{\lambda g}(z)$ when $z \in (-z_*, -r_1^{-1}(x^{(0)})] \cup [r_1^{-1}(x^{(0)}), z_*)$.

(iii) It is directly from Lemma 11 and the fact of $0 < r_1(z_*) < r_2(z_*)$.

(iv) Using similar arguments in item (ii), from $r_1(z_*) > x^{(0)} \geq 0$, we have $z_* < r_1^{-1}(x^{(0)}) \leq \frac{\lambda}{\epsilon}$; $x^{(0)} < r_1(z)$ for all $z \in [2\sqrt{\lambda} - \epsilon, r_1^{-1}(x^{(0)}))$; and $x^{(0)} > r_1(z)$ for all $z \in (r_1^{-1}(x^{(0)}), \frac{\lambda}{\epsilon}]$. Accordingly, by Lemma 11, for $r_1(z_*) > x^{(0)} > 0$ the limit $x^{(\infty)}$ of the sequence generated by the algorithm is given in (17). Hence, the limit does not match to the true solution $\text{prox}_{\lambda g}(z)$ when $z \in [-r_1^{-1}(x^{(0)}), z_*) \cup (z_*, r_1^{-1}(x^{(0)})]$. This statement is also true for $x^{(0)} = 0$ by Lemma 6.

□

5 An Application to Compressive Sensing

This section is devoted to showing the optimization algorithms and their numerical performance of the log-sum penalization for compressive sensing. Compressive sensing provides a method to reconstruct a sparse signal $x \in \mathbb{R}^n$ from linear measurements

$$b = Ax + \text{noise}, \tag{18}$$

where A is a given $m \times n$ measurement with $m < n$ and $b \in \mathbb{R}^m$ is the measurement vector acquired. It was shown in [5] that under the sparsity assumption, the signal can be exactly reconstructed from the given measurements and the chance of its being wrong is infinitesimally small.

The basic principle in compressive sensing is that a sparse or compressible signal can be reconstructed from a small number of measurements, measured through appropriate linear combinations of signal values, via an optimization approach. The optimization model for reconstruction of a sparse signal from model (18) is

$$\min \left\{ \frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu} S(x) : x \in \mathbb{R}^n \right\}, \tag{19}$$

where S is a sparsity promoting function, such as, the ℓ_p norm ($0 \leq p \leq 1$) [5, 7, 8] and the difference of the ℓ_1 and ℓ_2 norm [29]. In this section, we particularly choose S being the log-sum function f given in (1), problem (19) becomes

$$\min \left\{ \frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu} f(x) : x \in \mathbb{R}^n \right\}. \tag{20}$$

The solutions to problem (20) always exist since the objective function of the problem is coercive.

In the next subsection, we will present four different methods for problem (20) based on the structures of its objective function.

5.1 Algorithms

The first method for solving problem (20), described in Algorithm 1, is based on the forward-backward splitting approach in [1].

Algorithm 1 The forward-backward splitting algorithm for problem (20).

Input: Let $x^{(0)} \in \mathbb{R}^n$ and $0 < L < \frac{1}{\|A^\top A\|_F}$.

for $k = 0, 1, \ldots$ **do**

$$x^{(k+1)} \in \text{prox}_{\frac{1}{\mu L} f}\left(x^{(k)} - \frac{1}{L} A^\top (Ax^{(k)} - b)\right)$$

end for

Output: $x^{(\infty)}$

Notice that the objective function of problem (20) is nonnegative and coercive. Then the sequence $\{x^{(k)}\}$ in the algorithm is bounded and converges to some critical point of problem (20) ([1, Theorem 5.1]), that is, towards a point $x^{(\infty)}$ that satisfies

$$(A^\top A x^{(\infty)})_i + \frac{\text{sgn}(x_i^{(\infty)})}{|x_i^{(\infty)}| + \epsilon} = (A^\top b)_i$$

for i such that $x_i^{(\infty)} \neq 0$. Moreover, the sequence $\{x^{(k)}\}$ has a finite length, i.e. $\sum_{k=1}^{\infty} \|x^{(k+1)} - x^{(k)}\|_2 < \infty$. In the algorithm, the proximity operator $\text{prox}_{\frac{1}{\mu L} f}$ is available from Proposition 1 if $\frac{1}{\mu L} \leq \epsilon^2$ or Proposition 2 if $\frac{1}{\mu L} > \epsilon^2$.

By exploring properties of the log-sum function f, we are able to rewrite the objective function $\frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu} f(x)$ of problem (20) as the sum of two or three terms with special properties. The rest three algorithms are developed based on new formulations of the objective function. The second method is for

$$\min \left\{ \underbrace{\frac{1}{2}\|Ax - b\|^2 - \frac{\rho}{2}\|x\|^2}_{P(x)} + \underbrace{\frac{1}{\mu} f(x) + \frac{\rho}{2}\|x\|^2}_{Q(x)} : x \in \mathbb{R}^n \right\}, \tag{21}$$

where the positive parameter ρ will be determined in the following lemma.

Lemma 12 *Let the functions P and Q be given in (21). Then,*

(i) Q is convex when $\rho \geq \frac{1}{\mu\epsilon^2}$.

(ii) The gradient of P is L-smooth with $L = \max\{\|A\|^2 - \rho, \rho\}$.

By Lemma 12, when $\rho \geq \frac{1}{\mu\epsilon^2}$, the objective function of (21) is the composition of a differentiable function P and a convex function Q. Hence, the iPiano (Inertial Proximal Algorithm for Nonconvex Optimization) algorithm developed in [20] can be adopted for solving (21). The whole procedure is given in Algorithm 2.

For the sequence $\{x^{(k)}\}$ from Algorithm 2, it was shown that there exists a converging subsequence, and any limit point of $\{x^{(k)}\}$ is a critical point of problem (21).

Algorithm 2 The algorithm iPiano for problem (21)

Input: Given $\rho \geq \frac{1}{\mu\epsilon^2}$; choose $L = \max\{\|A\|^2 - \rho, \rho\}$, $\beta \in [0, 1)$, $\alpha < \frac{2(1-\beta)}{L}$; choose $x^{(0)} = x^{(1)} \in \mathbb{R}^n$.
for $k = 1, 2, \ldots$ **do**

$$x^{(k+1)} = \text{prox}_{\alpha Q}(x^{(k)} - \alpha(A^\top(Ax^{(k)} - b) - \rho x^{(k)}) + \beta(x^{(k)} - x^{(k-1)}))$$

end for
Output: $x^{(\infty)}$

By a simple manipulation, we know that

$$\text{prox}_{\alpha Q}(\cdot) = \text{prox}_{\frac{\alpha}{\mu(1+\alpha\rho)} f}\left(\frac{1}{1+\alpha\rho}\cdot\right) \tag{22}$$

Since $\rho \geq \frac{1}{\mu\epsilon^2}$, we know that

$$\frac{\alpha}{\mu(1+\alpha\rho)} < \epsilon^2.$$

Hence, $\text{prox}_{\alpha Q}$ in (22) is available from Proposition 1.

The third method is for

$$\min\left\{\frac{1}{2}\|Ax - b\|^2 + \underbrace{\frac{1}{\mu}f(x) + \frac{\rho}{2}\|x\|^2}_{Q(x)} - \underbrace{\frac{\rho}{2}\|x\|^2}_{P(x)} : x \in \mathbb{R}^n\right\}. \tag{23}$$

The objective function of problem (23) is the sum of $\frac{1}{2}\|Ax-b\|^2$ the smooth convex function with a Lipschitz continuous gradient, and the difference of two convex functions $Q(x) = \frac{1}{\mu}f(x) + \frac{\rho}{2}\|x\|^2$ and $P(x) = \frac{\rho}{2}\|x\|^2$. For such a kind of objective function, the proximal difference-of-convex algorithm with extrapolation approach proposed in [25] can be adopted for solving problem (23). It is described as follows.

Algorithm 3 The proximal difference-of-convex algorithm with extrapolation for problem (23)

Input: Given $\rho \geq \frac{1}{\mu\epsilon^2}$; choose $\alpha = 1/\|A\|^2$, $\{\beta_k\} \subset [0, 1)$ with $\sup_k \beta_k < 1$; set $x^{(-1)} = x^{(0)} \in \mathbb{R}^n$.
for $k = 1, 2, \ldots$ **do**

$$y^{(k)} = x^{(k)} + \beta_k(x^{(k)} - x^{(k-1)})$$
$$x^{(k+1)} = \text{prox}_{\alpha Q}(y^{(k)} - \alpha(A^\top(Ay^{(k)} - b) - \rho x^{(k)}))$$

end for
Output: $x^{(\infty)}$

Let $\{x^{(k)}\}$ be a sequence generated by Algorithm 3 for solving problem (23). Then, this sequence is bounded and any accumulation point of $\{x^{(k)}\}$ is a stationary point of $\frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu}f(x)$, see [25, Theorem 4.1].

The fourth method, also based on the proximal difference-of-convex algorithm with extrapolation approach, was proposed in [25]. To this end, we rewrite problem (20) as follows

$$\min \left\{ \frac{1}{2}\|Ax-b\|^2 + \underbrace{\frac{1}{\mu\epsilon}\|x\|_1}_{Q(x)} - \underbrace{\left(\frac{1}{\mu\epsilon}\|x\|_1 - \frac{1}{\mu}f(x)\right)}_{P(x)} : x \in \mathbb{R}^n \right\}, \tag{24}$$

In (24), the function P is differentiable and convex. Problem (24) is solved as described in Algorithm 4. Similar to the conclusion made for Algorithm 3, the sequence $\{x^{(k)}\}$ is bounded and any of its accumulation point is a stationary point of $\frac{1}{2}\|Ax-b\|^2 + \frac{1}{\mu}f(x)$. We remark that $\text{prox}_{\alpha Q}$ in Algorithm 4 is the well-known soft-thresholding operator with threshold $\frac{\alpha}{\mu\epsilon}$.

Algorithm 4 The proximal difference-of-convex algorithm with extrapolation for problem (24)

Input: Given $\rho \geq \frac{1}{\mu\epsilon^2}$; choose $\alpha = 1/\|A\|^2$, $\{\beta_k\} \subset [0,1)$ with $\sup_k \beta_k < 1$; set $x^{(-1)} = x^{(0)} \in \mathbb{R}^n$.

for $k = 1, 2, \ldots$ **do**

$$y^{(k)} = x^{(k)} + \beta_k (x^{(k)} - x^{(k-1)})$$
$$x^{(k+1)} = \text{prox}_{\alpha Q}\left(y^{(k)} - \alpha (A^\top (Ay^{(k)} - b) - \frac{1}{\mu\epsilon}\frac{x^{(k)}}{|x^{(k)}| + \epsilon}) \right)$$

end for

Output: $x^{(\infty)}$

5.2 Numerical Simulations

We now demonstrate the performance of the four algorithms developed in the previous subsection for compressive sampling reconstruction in terms of efficacy and accuracy. At the end of this section, we also provide an accelerated proximal point algorithm to improve the performance of Algorithm 1.

Through this section, all random $m \times n$ matrices A and length-n, s-sparse vectors x are generated based on the following assumption: entries of A and x on their support are i.i.d. Gaussian random variables with zero mean and unit variances. The locations of the nonzero entries (i.e., the support) of x are randomly permuted. We then generate the observation vector b by (18). We obtain the reconstruction $x^{(\infty)}$ from b by using the above four algorithms.

Three metrics, the number of iterations, the CPU time consumed, and the value of objective function, are used to evaluate the efficiency of the algorithms. The smaller the values of these metrics are the better the performed algorithm will be. Another three metrics, the relative ℓ_2 error, the number of missing nonzero coefficients, and the number of misidentified nonzero coefficients, are used to evaluate the quality of the reconstruction. More precisely, the relative ℓ_2 error is $\frac{\|x^{(\infty)}-x\|_2}{\|x\|_2}$, the number of missing nonzero coefficients refers to the number of nonzero coefficients that an algorithm "misses," i.e., determines to be zero; the number of misidentified nonzero coefficients refers to the number of nonzero coefficients that are "misidentified," i.e., coefficients that are determined to be nonzero when they should be zero. The last two metrics measure how well each algorithm finds the signal support, meaning the

locations of the nonzero coefficients. The smaller the values of these metrics are the better the reconstructed signals will be. The efficiency and accuracy of all test algorithms are measured by the average of values of these six metrics over 30 trials.

In our numerical experiments below, three configurations for the n dimension of the signal, m the number of measurements, and s the sparsity of the vector x, are considered, namely $(m, n, s) = (360i, 1280i, 40i)$ for $i = 1, 2, 3$. Each test algorithm is terminated when

$$\frac{\|x^{(k+1)} - x^{(k)}\|_2}{\max\{\|x^{(k)}\|_2, 1\}} < 10^{-5}. \tag{25}$$

The maximum number of iteration is set to be MAX = 10000.

We first look at the performance of all algorithms in noise-free case, i.e., $b = Ax$ in (18). For $\epsilon = 0.1$ in (1) and various values of the regularization parameter μ, Table 1 reports the efficiency of the test algorithms in terms of the metrics of the number of iterations, the CPU time consumed, and the value of objective function while Table 2 reports the accuracy of the test algorithms in terms of the metrics of the relative ℓ_2 error, the number of missing nonzero coefficients, and the number of misidentified nonzero coefficients. Similarly, the results for $\epsilon = 0.5$ with various values of the regularization parameter μ are reported in Tables 3 and 4. From these tables, we clearly see that Algorithms 3 and 4 are comparable and are better than Algorithms 1 and 2 in terms of all metrics.

Using publicly available code for the iteratively reweighted ℓ_1 (RWℓ_1) algorithm, Table 5 reports the performance of RWℓ_1 under the same settings in the above noise-free case. Note, however, that the RWℓ_1 algorithm is designed for the constrained problem, which results in slightly higher accuracy in the noise free case. Algorithm 4 with the log-sum penalty achieves comparable accuracy but is more efficient.

Next, for the noisy case, we set the observation b in (18) as $b = Ax + 0.01\xi$, where $\xi \in \mathbb{R}^m$ is a random vector with i.i.d. standard Gaussian entries. In our simulation, we set $\epsilon = 0.5$ and μ being 1000 and 2000. Similarly as we did in the noise-free case, the numerical results are reported in Tables 6 and 7 for various metrics. Again, we have the same conclusion that Algorithms 3 and 4 are comparable and are better than Algorithms 1 and 2 in terms of all metrics.

To improve the performance of Algorithm 1, we adapt an accelerated proximal gradient method in [17] for solving problem (20). The resulting procedure is described in Algorithm 5. Under the same settings in the above for noise-free and noise cases, the numerical results from Algorithm 5 are reported in Tables 1 and 4. We conclude that Algorithm 5 performs better than Algorithm 1.

We plot in Fig. 5 the true signal and the estimated signals obtained by Algorithms 1–5 for solving (18) with $\epsilon = 0.5$ and $\mu = 2000$ on a random instance $(m, n, s) = (720, 2560, 80)$. The true signal x is represented by circles while the estimated signals obtained by Algorithms 1–4 are marked by pluses. To generate the results in Fig. 5, the numbers of iterations used by Algorithms 1–5 are respectively 20000 (the maximum allowed number of iterations), 4464, 501, 501, and 8295 while the CPU times consumed are respectively 11.47, 3.16, 0.36, 0.30, and 18.60 s. The relative ℓ_2 errors of the recovered signals from Algorithms 1–5 are 0.6255, 0.0325, 0.0320, 0.0320, and 0.0321, respectively; the numbers of missing nonzero coefficients of the recovered signals from Algorithms 1–5 are 10, 1, 1, 1, and 1, respectively; and the number of misidentified nonzero coefficients of the recovered signals from Algorithms 1–5 are 1164, 546, 524, 519, and 564, respectively. Correspondingly, the values of the objective function at the recovered signals are 0.0897, 0.0377, 0.0377, 0.0377, and 0,0377. We conclude that the estimated signals obtained by Algorithms 2–5 are close to the true signal, but not the one by Algorithm 1. We conclude that Algorithm 5 performs better than

Table 1 Noise-free case: efficiency of algorithms for solving problem (20) with $\epsilon = 0.1$ and various values of μ

μ	Problem size (n, m, s)	Iterations					CPU times					Objective values				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
	(1280,360,40)	3307	226	137	187	392	0.30	0.02	0.01	0.01	0.13	1.0179	1.0178	1.0178	1.0178	1.0178
10^2	(2560,720,80)	4461	224	134	189	385	3.78	0.19	0.12	0.16	0.59	2.0057	2.0053	2.0053	2.0053	2.0053
	(5120,1440,120)	6087	221	133	192	382	27.92	1.01	0.61	0.89	3.42	3.9907	3.9892	3.9892	3.9892	3.9892
	(1280,360,40)	MAX	1533	286	291	3000	1.07	0.16	0.03	0.02	0.97	0.4249	0.1024	0.1024	0.1024	0.1025
10^3	(2560,720,80)	MAX	1492	286	287	2975	8.95	1.29	0.25	0.24	7.25	0.9459	0.2020	0.2020	0.2020	0.2025
	(5120,1440,120)	MAX	1452	284	284	2839	45.44	6.44	1.27	1.27	29.48	2.0751	0.4021	0.4021	0.4021	0.4021
	(1280,360,40)	MAX	9988	1147	1145	MAX	1.36	1.19	0.15	0.10	3.93	0.0799	0.0303	0.0103	0.0103	0.0443
10^4	(2560,720,80)	MAX	MAX	1132	1109	MAX	9.65	8.98	1.04	0.91	24.54	0.1653	0.0595	0.0202	0.0202	0.0870
	(5120,1440,120)	MAX	MAX	1126	1078	MAX	45.43	44.45	5.03	4.70	114.32	0.3406	0.1154	0.0402	0.0402	0.1700

Table 2 Noise-free case: accuracy of algorithms for solving problem (20) with $\epsilon = 0.1$ and various values of μ

μ	Problem size (n, m, s)	The ℓ_2 error					# missing nonzero coeffs					# misidentified nonzero coeffs				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
10^2	(1280,360,40)	0.01114	0.01058	0.01056	0.01056	0.01061	1.33	1.37	1.37	1.37	1.37	0	0	0	0	0
	(2560,720,80)	0.01196	0.01100	0.01098	0.01098	0.01106	3.23	3.30	3.30	3.30	3.3	0	0	0	0	0
	(5120,1440,120)	0.01335	0.01194	0.01193	0.01193	0.01199	7.07	7.33	7.33	7.33	7.27	0	0	0	0	0
10^3	(1280,360,40)	0.54998	0.00103	0.00101	0.00102	0.00106	4.87	0.13	0.13	0.13	0.13	374	0	0	0	0
	(2560,720,80)	0.62072	0.00111	0.00111	0.00111	0.00115	12.6	0.63	0.63	0.63	0.72	843	0	0	0	0
	(5120,1440,120)	0.67295	0.00117	0.00117	0.00116	0.00121	25.2	0,70	0.70	0.70	0.70	1916	0	0	0	0
10^4	(1280,360,40)	0.80438	0.31876	0.00011	0.00012	0.56865	3.03	2.03	0.03	0.03	4.5	885	350	0	0	426
	(2560,720,80)	0.81601	0.31515	0.00013	0.00012	0.56658	4.70	4.13	0.10	0.10	10.47	1911	727	0	0	845
	(5120,1440,120)	0.82453	0.30606	0.00012	0.00013	0.56166	8.17	7.60	0.05	0.03	30.30	4071	1444	0	0	1687

Table 3 Noise-free case: efficiency of algorithms for solving problem (20) with $\epsilon = 0.5$ and various values of μ

μ	Problem size (n, m, s)	Iterations					CPU times					Objective values				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
	(1280,360,40)	7637	449	235	241	853	0.75	0.05	0.02	0.02	0.23	0.5155	0.5154	0.5154	0.5154	0.5153
10^2	(2560,720,80)	9810	440	239	246	835	8.58	0.39	0.21	0.20	1.84	1.1222	1.0114	1.0114	1.0114	1.0114
	(5120,1440,120)	MAX	435	244	249	826	44.71	1.93	1.08	1.11	7.76	3.5011	2.0090	2.0090	2.0090	2.0090
	(1280,360,40)	MAX	3932	442	439	7856	1.19	0.43	0.05	0.03	2.24	0.1987	0.0517	0.0517	0.0517	0.0518
10^3	(2560,720,80)	MAX	3839	427	424	7679	9.58	3.50	0.40	0.36	19.25	0.4232	0.1014	0.1014	0.1014	0.1014
	(5120,1440,120)	MAX	3785	423	417	7570	45.57	16.98	1.89	1.8	80.63	0.8914	0.2014	0.2014	0.2014	0.2014
	(1280,360,40)	347	MAX	2600	2598	MAX	0.07	1.44	0.37	0.26	4.03	0.0297	0.0165	0.0052	0.0052	0.0203
10^4	(2560,720,80)	436	MAX	2547	2546	MAX	0.48	8.96	2.28	2.02	22.70	0.0584	0.0323	0.0101	0.0101	0.0398
	(5120,1440,120)	567	MAX	2504	2497	MAX	2.93	44.76	11.14	10.75	113.85	0.1152	0.0631	0.0201	0.0201	0.0778

Table 4 Noise-free case: accuracy of algorithms for solving problem (20) with $\epsilon = 0.5$ and various values of μ

μ	Problem size (n, m, s)	The ℓ_2 error					# missing nonzero coeffs					# misidentified nonzero coeffs				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
	(1280,360,40)	0.00528	0.00449	0.00449	0.00448	0.00453	0.33	0.30	0.30	0.30	0.30	0	0	0	0	0
10^2	(2560,720,80)	0.03981	0.00463	0.00463	0.00462	0.00468	1.43	0.77	0.77	0.77	0.77	160	0	0	0	0
	(5120,1440,120)	0.24610	0.00482	0.00481	0.00480	0.00487	8.10	1.27	1.27	1.27	1.27	1163	0	0	0	0
	(1280,360,40)	0.70331	0.00042	0.00045	0.00045	0.00123	3.97	0.07	0.07	0.07	0.07	688	0	0	0	8.5
10^3	(2560,720,80)	0.73986	0.00043	0.00047	0.00047	0.00054	6.97	0.20	0.20	0.20	0.20	1542	0	0	0	0
	(5120,1440,120)	0.76743	0.00045	0.00048	0.00048	0.00056	13.00	0.10	0.10	0.10	0.10	3424	0	0	0	0
	(1280,360,40)	0.84702	0.60203	0.00008	0.00008	0.71211	0.03	3.97	0	0	3.8	1237	555	0.20	0.20	710
10^4	(2560,720,80)	0.84767	0.59994	0.00007	0.00007	0.03977	0	8.58	0.03	0.03	7.40	2476	1112	0.97	0.50	1416
	(5120,1440,120)	0.84786	0.59702	0.00007	0.00007	0.70956	0	16.60	0	0	16.80	4953	2206	0.03	0.00	2820

Table 5 Noise-free case: iteratively Reweighted ℓ_1

Problem size	The ℓ_2 error	CPU times	# missing nonzero coeffs	# misidentified nonzero coeffs
(1280,360,40)	5.48949E-07	0.34	0	0
(2560,720,80)	7.08896E-07	1.98	0	0
(5120,1440,120)	9.40293E-07	17.77	0	0

Algorithm 5 The accelerated proximal gradient method for problem (20).

Input: Let $z^{(1)} = x^{(1)} = x^{(0)} \in \mathbb{R}^n$, $t_1 = 1, t_0 = 0, \alpha_x < 1/\|A\|^2, \alpha_y < 1/\|A\|^2$, and $F(x)$ is the objective function of problem (20).

for $k = 1, 2, \ldots$ **do**

$$
\begin{aligned}
y^{(k)} &= x^{(k)} + \frac{t_{k-1}}{t_k}(z^{(k)} - x^{(k)}) + \frac{t_{k-1}-1}{t_k}(x^{(k)} - x^{(k-1)}) \\
z^{(k+1)} &\in \operatorname{prox}_{\frac{\alpha_y}{\mu} f}\left(y^{(k)} - \alpha_y A^\top (Ay^{(k)} - b)\right) \\
v^{(k+1)} &\in \operatorname{prox}_{\frac{\alpha_x}{\mu} f}\left(x^{(k)} - \alpha_x A^\top (Ax^{(k)} - b)\right) \\
t_{k+1} &= \frac{\sqrt{4t_k^2 + 1} + 1}{2} \\
x^{(k+1)} &= \begin{cases} z^{(k+1)} & \text{if } F(z^{(k+1)}) < F(v^{(k+1)}); \\ v^{(k+1)}, & \text{otherwise.} \end{cases}
\end{aligned}
$$

end for

Output: $x^{(\infty)}$

Algorithm 1. Although Algorithm 5 is still slower than both Algorithm 3 and Algorithm 4, they are comparable in terms of other metrics.

5.3 Comparing Nonconvex Penalties for Sparse Recovery

In this subsection, we provide simulations to compare the log-sum penalty with two popular nonconvex penalties: the minimax concave penalty (MCP) [30] and the $\ell_1 - \ell_2$ penalty [13] for sparse recovery.

The optimization model with the MCP penalty for reconstruction of a sparse signal from model (18) is

$$
\min\left\{\frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu}(\|x\|_1 - \operatorname{env}_\tau(\|\cdot\|_1)(x)) : x \in \mathbb{R}^n\right\}, \tag{26}
$$

where μ is a regularization parameter and $\operatorname{env}_\tau(\|\cdot\|_1)$ is the Moreau envelope of the ℓ_1 norm with index α. A detailed discussion on the MCP penalty can be found in [22, 30]. Since the $\operatorname{env}_\tau(\|\cdot\|_1)$ is differentiable and its gradient equals to $\frac{1}{\tau}(I - \operatorname{prox}_{\tau\|\cdot\|_1})$, we can adopt the proximal difference-of-convex algorithm with extrapolation approach in [25] for solving problem (26). The resulting algorithm is referred to as MCP-DCA for simplicity.

The optimization model with $\ell_1 - \ell_2$ penalty for reconstruction of a sparse signal from model (18) is

$$
\min\left\{\frac{1}{2}\|Ax - b\|^2 + \frac{1}{\mu}(\|x\|_1 - \|x\|_2) : x \in \mathbb{R}^n\right\}, \tag{27}
$$

where μ is a regularization parameter. Various efficient algorithms for the optimization problem (27) have been developed in [18]. The accelerated forward-backward splitting (AFBS) is chosen here. We refer to this algorithm applied to problem (27) as $(\ell_1 - \ell_2)$-AFBS for clarity. For the optimization problem (20) regularized by the log-sum penalty, we use Algorithm 4 to solve the problem.

In our numerical experiments, we set the observation b in (18) as $b = Ax + 0.03\xi$, where $\xi \in \mathbb{R}^m$ is a random vector with i.i.d. standard Gaussian entries. In our simulations, we set

Table 6 Noise case: efficiency of algorithms for solving problem (20) with $\epsilon = 0.5$ and various values of μ

μ	Problem size (n, m, s)	Iterations					CPU times					Objective values				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
10^3	(1280,360,40)	MAX	2513	501	501	8520	0.99	0.23	0.05	0.04	2.57	0.0880	0.0386	0.0386	0.0386	0.0557
	(2560,720,80)	MAX	2480	501	501	8380	10.12	2.44	0.49	0.45	16.81	0.1927	0.0758	0.0758	0.0758	0.1091
	(5120,1440,120)	MAX	2468	501	501	8274	46.57	11.19	2.33	2.29	91.03	0.4152	0.1505	0.1505	0.1505	0.2169
$2 \cdot 10^3$	(1280,360,40)	MAX	2513	501	501	MAX	1.76	0.74	0.08	0.07	4.61	0.0544	0.0194	0.0194	0.0194	0.0523
	(2560,720,80)	MAX	2480	501	501	MAX	12.87	5.63	0.64	0.58	23.19	0.1156	0.0382	0.0381	0.0381	0.1013
	(5120,1440,120)	MAX	2468	501	501	MAX	45.73	20.53	2.23	2.19	114.52	0.2430	0.0758	0.0757	0.0757	0.1970

Table 7 Noise case: accuracy of algorithms for solving problem (20) with $\epsilon = 0.5$ and various values of μ

μ	Problem size (n, m, s)	The ℓ_2 error					# missing nonzero coeffs					# misidentified nonzero coeffs				
		A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5	A.1	A.2	A.3	A.4	A.5
10^3	(1280,360,40)	0.5584	0.0281	0.0281	0.0281	0.0114	3.60	0.53	0.50	0.50	0.27	531	279	266	266	299
	(2560,720,80)	0.6296	0.0284	0.0284	0.0284	0.0139	7.93	1.20	1.13	1.13	0.57	1205	553	529	529	587
	(5120,1440,120)	0.6839	0.0289	0.0289	0.0289	0.0141	16.80	1.77	1.87	1.87	1.17	2724	1107	1058	1058	1181
$2 \cdot 10^3$	(1280,360,40)	0.6876	0.0301	0.0300	0.0300	0.3046	4.03	0.50	0.53	0.53	1.70	690	312	294	295	416
	(2560,720,80)	0.7284	0.0303	0.0302	0.0302	0.2980	6.57	1.20	1.13	1.17	3.23	1547	623	587	587	827
	(5120,1440,120)	0.7591	0.0309	0.0307	0.0307	0.2923	13.23	1.73	1.70	1.70	6.40	3430	1245	1162	1162	1657

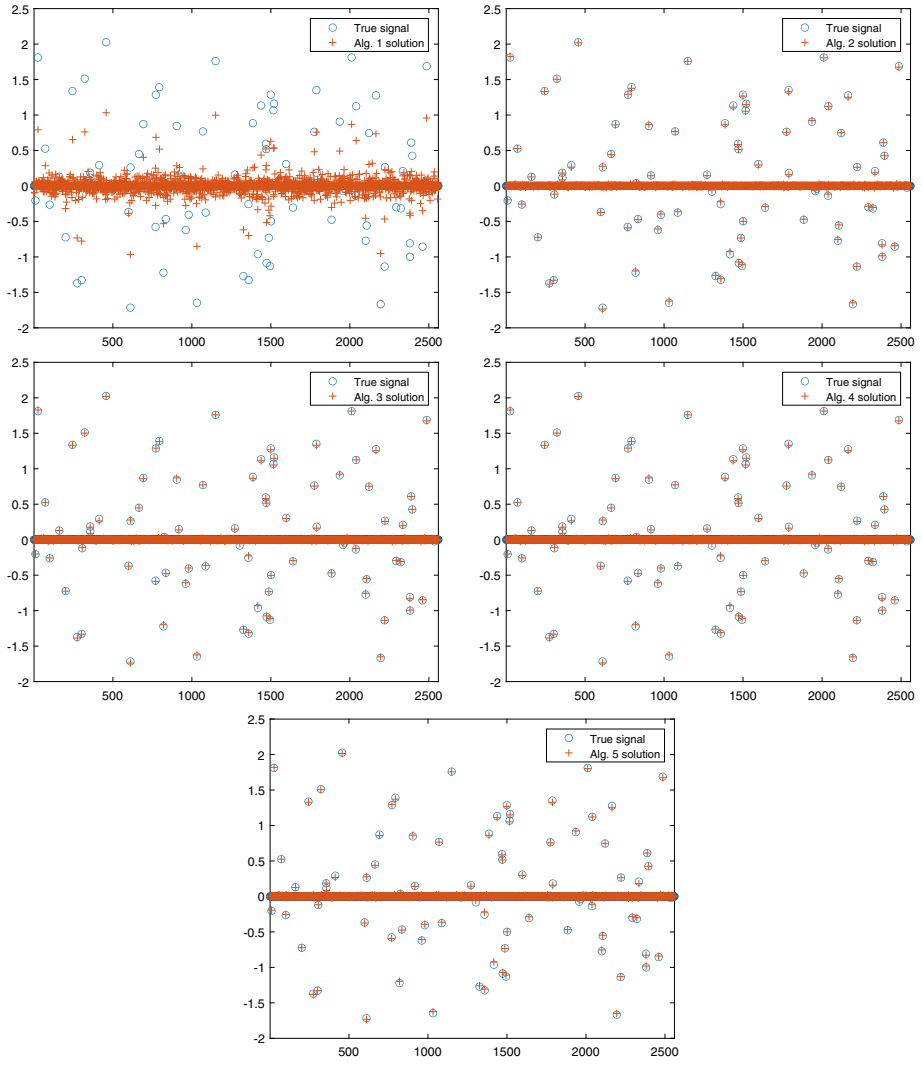

Fig. 5 The true solution and the solution obtained by solving (18) with $\epsilon = 0.5$ and $\mu = 2000$ through Algorithms 1–5

$\epsilon = 0.5$ for the log-sum function and μ being 10, 100, and 1000 for both problems (20), (26), and (27). For the MCP, we set $\tau = 0.5$ in our simulation. For the AFBS algorithm, the initial estimate is an ℓ_1 optimization problem as suggested in [18]. These algorithms are terminated when either inequality (25) is satisfied or the number of iteration meets 10000.

Table 8 reports the numerical results, averaged over 50 realizations, from Algorithm 4, MCP-DCA and $(\ell_1 - \ell_2)$-AFBS under various scenarios. Associated with each algorithm, there are five columns representing the number of iterations, the CPU time consumed, the relative ℓ_2 error, the number of missing nonzero coefficients, and the number of misidentified nonzero coefficients, respectively. The rows of Table 8 are separated into three blocks by horizontal lines, corresponding the configurations (m, n, s) being (360, 1280, 40),

(720, 2560, 80), and (1440, 5120, 160), respectively. Each such block consists of three rows which show numerical results corresponding to the regularization parameter μ being 10, 100, and 1000. From this table, we can conclude that overall the optimization model (20) with Algorithm 4 can efficiently produce recovery results that are comparable to MCP and AFBS with $\ell_1 - \ell_2$ penalty in terms of the metrics of the CPU time consumed, the relative ℓ_2 error, the number of missing nonzero coefficients, and the number of misidentified nonzero coefficients.

6 An Application to Low-Rank Regularization for Image Denoising

In this section, we present an application of the log-sum penalty in removing mixed additive Gaussian white noise and impulse noise in noisy images. Mathematically, a noisy image in the presence of additive Gaussian white noise and impulse noise can be formulated as

$$y = \mathcal{N}_{imp}(x + z), \tag{28}$$

where x is the true image in $\mathbb{R}^{n_0 \times m_0}$, y is the observed image, z is the additive Gaussian white noise, $\mathcal{N}_{imp} : \mathbb{R}^{n_0 \times m_0} \to \mathbb{R}^{n_0 \times m_0}$ represents impulse noise which assigns some pixels of $x+z$ in its dynamic range $[0, 255]$. Those pixels are called the outliers in y and the probability of outliers is called the noise level of impulse noise in y. Assume that the original image x in model (28) is indexed by $\Omega = \{1, 2, \ldots, n_0\} \times \{1, 2, \ldots, m_0\}$ and the outlier candidate set $\mathcal{Z}$ is

$$\mathcal{Z} = \{(i, j) \in \Omega : y_{ij} \neq x_{ij} + z_{ij}\}.$$

The level of impulse noise refers to the ratio of the cardinalities of $\mathcal{Z}$ and Ω. Recently, approaches based on low-rank regularization have been proposed, for example, in [16, 28]. In particular, since the log-sum penalty was adopted in [28] to promote low-rankness of a patch formed from similar patches, we will demonstrate the usefulness of the results developed in this paper.

Let us first briefly review the NLR-TP (nonlocal low-rank regularized two-phase) approach developed in [28]. The NLR-TP has two basic phases: (i) detecting locations of outlier candidates and (ii) solving an optimization problem with a content-driven fidelity term and a nonlocal low-rank regularization term. Phase-I can be achieved by many existing impulse noise removals, such as the adaptive median filter and the adaptive center-weighted median filter. Phase-II is to approximate the ideal image x from the outlier-free data on $\Omega \setminus \mathcal{Z}$. To this end, the counterpart of model (28) in patch format will be formed. For an image x of size $n_0 \times m_0$, a square patch of size $\sqrt{n} \times \sqrt{n}$ centered at position (i, j) is denoted by x_ℓ, where $\ell = (j-1)n_0 + i$. Due to non-local self-similarity of natural images, we are able to find patches $\{x_{\ell_k}\}_{k=1}^{m-1}$ by a block-matching algorithm [10] that are similar to x_ℓ. With these patches, we form an $n \times m$ matrix X_ℓ, called the ℓth patch matrix of the image x, as follows:

$$X_\ell := \left[x_\ell \; x_{\ell_1} \; \cdots \; x_{\ell_{m-1}}\right],$$

where x_ℓ and x_{ℓ_k} are the vectorization of themselves. It is reasonable to assume that X_ℓ is low-rank.

Now, let $X \in \mathbb{R}^{n \times m}$ be a patch matrix of the image x. We form accordingly the matrices Y and Z from the corresponding locations of y and z in model (28), respectively. Then,

$$Y = \mathcal{N}_{imp}(X + Z), \tag{29}$$

Table 8 Numerical results from Algorithm 4, MCP-DCA and $(\ell_1 - \ell_2)$-AFBS in terms of the number of iterations, the CPU time consumed, the relative ℓ_2 error, the number of missing nonzero coefficients, and the number of misidentified nonzero coefficients

(n, m, s)	μ	Algorithm 4					MCP-DCA					$(\ell_1 - \ell_2)$-AFBS				
(1280,360,40)	10	147	0.01	0.048	3.5	0	174	0.01	0.021	1.6	1.03	33	0.14	0.055	1.7	17.3
	100	261	0.02	0.032	0.8	169.9	420	0.03	0.037	0.7	223	107	0.35	0.046	0.9	253.0
	1000	625	0.07	0.046	0.7	302.3	1059	0.07	0.045	0.6	330	243	0.95	0.055	0.8	322.4
(2560,720,80)	10	150	0.08	0.049	6.9	0	175	0.07	0.022	3.4	1.7	29	0.65	0.059	3.7	39.7
	100	258	0.16	0.033	1.7	339.9	425	0.15	0.038	1.7	446	124	2.15	0.048	1.7	507.5
	1000	586	0.37	0.046	1.5	608.0	1051	0.40	0.045	1.6	649	3	1.77	0.058	1.7	637.7
(5120,1440,120)	10	151	0.52	0.050	14.6	0	177	0.51	0.023	7.4	3.8	25	2.94	0.061	7.6	83.4
	100	252	0.91	0.034	2.9	678	419	1.21	0.039	3.1	895	129	9.06	0.050	3	1021.3
	1000	561	2.09	0.047	2.7	1224	1001	3.09	0.047	3.0	1301	3	8.15	0.059	3.0	1271.5

$\mathcal{N}_{imp} : \mathbb{R}^{n\times m} \to \mathbb{R}^{n\times m}$ represents impulse noise which assigns some pixels of $X + Z$ in its dynamic range [0, 255]. Here, we did not distinguish the operator $\mathcal{N}_{imp}$ used in between (28) and (29). Moreover, we also use Ω and $\mathcal{Z}$ to represent the index set of X and the outlier candidate set, respectively. With these preparations, the optimization problem in the second phase of the NLR-TP is

$$\min\left\{J(X) = \left\|\varphi_\eta(D\odot(X-Y))\right\|_1 + \frac{1}{\mu}f(\sigma(X)) : X\in\mathbb{R}^{n\times m}\right\}, \tag{30}$$

where the positive parameter μ balances the importance between the content-dependent fidelity term $\left\|\varphi_\eta(D\odot(X-Y))\right\|_1$ and the low-rank regularization term $f(\sigma(X))$. Here, $\varphi_\eta(t) := \frac{\eta t^2}{\eta+|t|}$ with the parameter η determining the shape of φ_η, and $\varphi_\eta(D\odot(X-Y))$ is viewed as an $n\times m$ matrix whose (i,j)th entry is $\varphi_\eta(D_{ij}(X_{ij}-Y_{ij}))$. Since the fidelity term has the Lipschitz continuous gradient with Lipschitz constant 2 and the regularization term is concave, it was proposed in [28] to approximate the fidelity term by a quadrature form and the regularization term by its first-order expansion, therefore, leading a double-loop iterative scheme for solving problem (30).

We now propose a new approach for solving problem (30) based on the discussion in the previous sections. By viewing the fidelity term $\left\|\varphi_\eta(D\odot(X-Y))\right\|_1$ and the regularization term $f(\sigma(X))$ in (30) as the fidelity term $\frac{1}{2}\|Ax-b\|^2$ and regularization $f(x)$ in (20) respectively, an algorithm, which mimics Algorithm 1, can be developed for problem (30) as described in

Algorithm 6 The forward-backward splitting algorithm for problem (30).

Input: Let $X^{(0)}\in\mathbb{R}^{n\times m}$ and $0<L<2$.
for $k=0,1,\ldots$ **do**

$$X^{(k+1)} \in \text{prox}_{\frac{1}{\mu L}f\circ\sigma}\left(X^{(k)} - \frac{1}{L}D\odot\varphi_\eta'(D\odot(X^{(k)}-Y))\right),$$

end for
Output: $X^{(\infty)}$

Now, we demonstrate the performance of Algorithm 6 for mixed noise removal. Two test images of "Barbara" and "House" are shown in Fig. 6a1 and b1, respectively. The images Fig. 6a2, b2 are corrupted images from Fig. 6a1, b1 by the salt-and-pepper noise used with noise level 50 and the Gaussian with standard variation 10, respectively. We use the parameters $\epsilon = 10^{-2}$ and $\mu = 10^{-4}$. The denoised images by the NLR-TP are displayed in Fig. 6a3, b3 while the denoised images by Algorithm 6 are displayed in Fig. 6a4, b4. The quantitative qualities of the denoised images can be measured by three metrics, namely, the peak signal-to-noise ratio (PSNR), the structural similarity (SSIM) [24] and the feature similarity (FSIM) [31]. The PSNR mainly measures the intensity similarity between an reconstructed image and its reference image while the other two mainly measure the perceptual image quality of an reconstructed image. Normally, the higher PSNR, SSIM, and FSIM scores are, the better the quality of the reconstructed images is. The values of PSNR, SSIM, and FSIM for the image in Fig. 6a3 are 31.92, 0.9563, and 0,9705 while those values for the image in Fig. 6a4 are 32.83, 0.9629, and 0.9742. The values of PSNR, SSIM, and FSIM for the image in Fig. 6b3 are 36.58, 0.9430, and 0.9704 while those values for the image in Fig. 6b4 are 37.31, 0.9410,

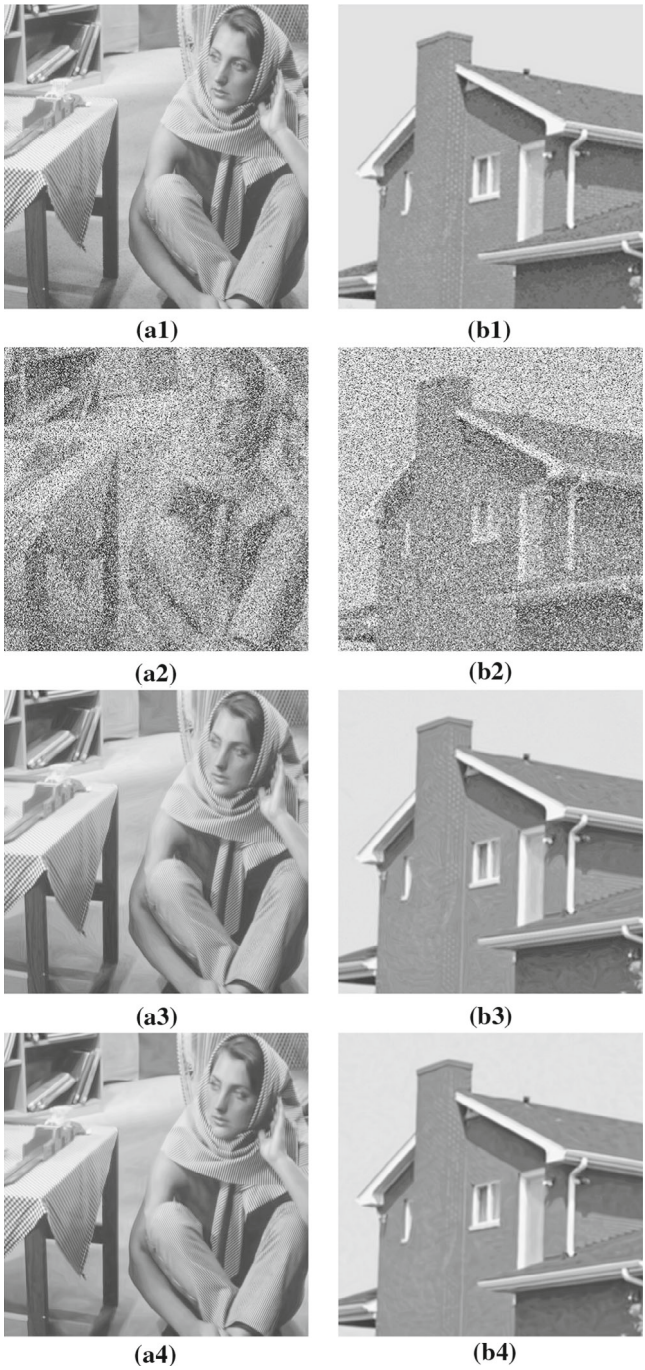

Fig. 6 Salt-and-Pepper noise with noise level 50 plus Gaussian noise with $\sigma = 10$. The images from top to bottom are the original images, noisy images, the denoised images by NLR-TP, and the denoised images by proposed algorithm for the images of "Barbara" (left column) and "House" (right column)

and 0.9677. Clearly, one can observe the improvement provided by Algorithm 6. We plan to do more comprehensive study on optimization problem (30) for image processing

7 Conclusions

We presented the explicit expressions of the proximity operators of the log-sum penalty and its composition with the singular value function. In the existing literature, these proximity operators were computed through iteratively reweighted ℓ_1 methods that are inefficient and may sometimes give inaccurate results, as analyzed in Theorem 6 and demonstrated in Fig. 4. By applying the results from this paper, one can avoid using iterative approaches to compute the proximity operator of the log-sum penalty, and can prevent inaccurate solutions from sub-optimal initial values. Moreover, we have characterized the behavior of the proximity operator for the log-sum penalty, and further justified its use as a nonconvex surrogate in ℓ_0 and ℓ_1 norm minimization problems. Several algorithms are provided for the log-sum regularized compressed sensing problem (20) and their performance is compared. We demonstrated numerically the log-sum regularized compressed sensing problem (20), the MCP regularized compressed sensing problem (26) and the $\ell_1 - \ell_2$ regularized compressed sensing problem (27). In addition, we provided an application of the log-sum function to low-rank regularization for image denoising.

Acknowledgements This work was funded in part by Air Force Office of Scientific Research (AFOSR) grant 21RICOR035. The work of L. Shen was supported in part by the National Science Foundation under grant DMS-1913039, 2020 U.S. Air Force Summer Faculty Fellowship Program, and the 2020 Air Force Visiting Faculty Research Program funded through AFOSR grant 18RICOR029. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Air Force Research Laboratory. Cleared for public release 08 Jan 2021: Case number AFRL-2021-0024.

Funding The authors have not disclosed any funding.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Declarations

Code Availability The code generated for numerical experiments in this article is available from the corresponding author on reasonable request.

Competing interests The authors have not disclosed any competing interests.

References

1. Attouch, H., Bolte, J., Svaiter, B.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. Math. Program. Ser. A **137**, 91–129 (2013)
2. Bauschke, H.L., Combettes, P.L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces. AMS Books in Mathematics. Springer, New York (2011)
3. Bolte, J., Pauwels, E.: Majorization-minimization procedures and convergence of SQP methods for semi-algebraic and tame Programs. Math. Oper. Res. **41**, 442–465 (2016)
4. Cai, J.-F., Choi, J.K., Li, J., Wei, K.: Image restoration: structured low rank matrix framework for piecewise smooth functions and beyond. Appl. Comput. Harmon. Anal. **56**, 26–60 (2022)

5. Candes, E., Tao, T.: Near optimal signal recovery from random projections: universal encoding strategies? IEEE Trans. Inf. Theory **52**, 5406–5425 (2006)
6. Candes, E., Wakin, M.B., Boyd, S.: Enhancing sparsity by reweighted ℓ^1 minimization. J. Fourier Anal. Appl. **14**, 877–905 (2008)
7. Chartrand, R.: Exact reconstruction of sparse signals via nonconvex minimization. Signal Process. Lett. IEEE **14**, 707–710 (2007)
8. Chen, F., Shen, L., Suter, B.W.: Computing the proximity operator of the ℓ_p norm with $0 < p < 1$. IET Signal Proc. **10**, 557–565 (2016)
9. Chen, K., Dong, H., Chan, K.-S.: Reduced rank regression via adaptive nuclear norm penalization. Biometrika **100**, 901–920 (2013)
10. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3d transform-domain collaborative filtering. IEEE Trans. Image Process. **16**, 2080–2095 (2007)
11. Deng, Y., Dai, Q., Liu, R., Zhang, Z., Hu, S.: Low-rank structure learning via nonconvex heuristic recovery. IEEE Trans. Neural Netw. Learn. Syst. **24**, 383–396 (2013)
12. Dong, W., Shi, G., Li, X., Ma, Y., Huang, F.: Compressive sensing via nonlocal low-rank regularization. IEEE Trans. Image Process. **23**, 3618–3632 (2014)
13. Esser, E., Lou, Y., Xin, J.: A method for finding structured sparse solutions to nonnegative least squares problems with applications. SIAM J. Imaging Sci. **6**, 2010–2046 (2013)
14. Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. J. Am. Stat. Assoc. **96**, 1348–1360 (2001)
15. Fazel, M., Hindi, H., Boyd, S.: Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices, vol. 3 of Proceedings of American Control Conference, pp. 2156–2162 (2003)
16. Huang, T., Dong, W., Xie, X., Shi, G., Bai, X.: Mixed noise removal via Laplacian scale mixture modeling and nonlocal low-rank approximation. IEEE Trans. Image Process. **26**, 3171–3186 (2017)
17. Li, H., Lin, Z.: Accelerated proximal gradient methods for nonconvex programming. In: Advances in Neural Information Processing Systems, pp. 379–387, pp. 3171–3186 (2015)
18. Lou, Y., Yan, M.: Fast L1–L2 minimization via a proximal operator. J. Sci. Comput. **74**(2), 767–785 (2018)
19. Mirsky, L.: A trace inequality of John von Neumann. Monatshefte fur Mathematik **79**, 303–306 (1975)
20. Ochs, P., Chen, Y., Brox, T., Pock, T.: iPiano: Inertial proximal algorithm for nonconvex optimization. SIAM J. Imaging Sci. **7**, 1388–1419 (2014)
21. Ochs, P., Dosovitskiy, A., Brox, T., Pock, T.: On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision. SIAM J. Imaging Sci. **8**, 331–372 (2015)
22. Shen, L., Suter, B.W., Tripp, E.E.: Structured sparsity promoting functions. J. Optim. Theory Appl. **183**, 386–421 (2019)
23. Shen, Y., Fang, J., Li, H.: Exact reconstruction analysis of log-sum minimization for compressed sensing. IEEE Signal Process. Lett. **20**, 1223–1226 (2013)
24. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Trans. Image Process. **13**, 600–612 (2004)
25. Wen, B., Chen, X., Pong, T.K.: A proximal difference-of-convex algorithm with extrapolation. Comput. Optim. Appl. **69**, 297–324 (2018)
26. Wen, F., Chu, L., Liu, P., Qiu, R.C.: A Survey on nonconvex regularization-based sparse and low-rank recovery in signal processing, statistics, and machine learning. IEEE Access **6**, 69883–69906 (2018)
27. Xia, L.-Y., Wang, Y.-W., Meng, D.-Y., Yao, X.-J., Chai, H., Liang, Y.: Descriptor selection via Log-Sum regularization for the biological activities of chemical structure. Int. J. Mol. Sci. **19**, 30 (2017)
28. Xu, C., Liu, X., Zheng, J., Shen, L., Jiang, Q., Lu, J.: Nonlocal low-rank regularized two-phase approach for mixed noise removal. Inverse Prob. **37**, 085001 (2021)
29. Yin, P., Lou, Y., He, Q., Xin, J.: Minimization of ℓ_{1-2} for compressed sensing. SIAM J. Sci. Comput. **37**, A536–A563 (2015)
30. Zhang, C.-H.: Nearly unbiased variable selection under minimax concave penalty. Ann. Stat. **38**, 894–942 (2010)
31. Zhang, L., Zhang, L., Mou, X., Zhang, D.: FSIM: a feature similarity index for image quality assessment. IEEE Trans. Image Process. **20**, 2378–2386 (2011)