J. Appl. Numer. Optim. 2 (2020), No. 1, pp. 15-62
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.2.2020.1.03

SUPERIORIZATION VS. ACCELERATED CONVEX OPTIMIZATION: THE SUPERIORIZED / REGULARIZED LEAST-SQUARES CASE

YAIR CENSOR[1], STEFANIA PETRA[2,*], CHRISTOPH SCHNÖRR[3]

[1]*Department of Mathematics, University of Haifa, Haifa 3498838, Israel*
[2]*Mathematical Imaging Group, Heidelberg University, Heidelberg 69120, Germany*
[3]*Image and Pattern Analysis Group, Heidelberg University, Heidelberg 69120, Germany*

Abstract. We conduct a study and comparison of superiorization and optimization approaches for the reconstruction problem of superiorized / regularized least-squares solutions of underdetermined linear equations with nonnegativity variable bounds. Regarding superiorization, the state of the art is examined for this problem class, and a novel approach is proposed that employs proximal mappings and is structurally similar to the established forward-backward optimization approach. Regarding convex optimization, accelerated forward-backward splitting with inexact proximal maps is worked out and applied to both the natural splitting least-squares term / regularizer and to the reverse splitting regularizer / least-squares term. Our numerical findings suggest that superiorization can approach the solution of the optimization problem and leads to comparable results at significantly lower costs, after appropriate parameter tuning. On the other hand, applying accelerated forward-backward optimization to the reverse splitting slightly outperforms superiorization, which suggests that convex optimization can approach superiorization too, using a suitable problem splitting.

Keywords. Accelerated forward-backward iteration; Convex optimization; Inexact proximal mappings; Perturbation resilience; Superiorization.

CONTENTS

*Corresponding author.
E-mail addresses: yair@math.haifa.ac.il (Y. Censor), petra@math.uni-heidelberg.de (S. Petra), schnoerr@math.uni-heidelberg.de (C. Schnörr).

Received November 5, 2019; Accepted March 12, 2020.

1. Introduction

1.1. **Overview and Motivation.** The purpose of this work is to present a comparative study of superiorization and accelerated inexact convex optimization.

The motto of the *superiorization methodology (SM)* is to take an iterative algorithm – called the *basic algorithm* – which is known to converge to a point in a specified set and perturb its iterates such that the perturbed algorithm – called *the superiorized version of the basic algorithm* – will still converge to some point of the same set. Hence, these perturbations can be used to lower the value of a given exogenous *target function* without compromising convergence of the generated sequence of iterates to the specified set, which explains the adjective *superiorized*. In comparison to numerical iterative optimization methods, the SM strives to return a *superior* feasible point rather than an

optimal feasible point. On the other hand, the SM only requires minor modifications of existing code in order to perturb a basic algorithm. In addition, the SM does not depend on what target function is chosen for the application domain at hand. As a consequence, superiorization provides a flexible framework for applications that has attracted interest recently (see Subsection 1.3.1).

Convex optimization (CO) is an established mature field of research. For large-scale problems that we are mainly interested in, iterative proximal minimization are the methods of choice based on various problems splittings [4, Chapter 28], [26]. The most fundamental problem splitting is based on the decomposition

$$h\colon \mathbb{R}^n \to (-\infty, +\infty], \qquad h(x) = f(x) + g(x) \tag{1.1}$$

of a given convex objective function h into a *smooth* convex function f and a, possibly *non-smooth,* convex function g, respectively, where smooth means continuously differentiable with L_f-Lipschitz continuous gradient ∇f, whereas g is only required to be lower-semicontinuous. *Forward-backward splitting (FBS)* – also called *proximal gradient iteration* – iteratively computes for $k \in \{0, 1, \dots\}$ with arbitrary initial point $x_0 \in \mathbb{R}^n$

$$x_{k+1} = P_\alpha g\big(x_k - \alpha \nabla f(x_k)\big), \tag{1.2a}$$

$$= x_k - \alpha \psi_\alpha(x_k), \tag{1.2b}$$

with ψ_α given by

$$\psi_\alpha(x) = \nabla f(x) + \nabla e_\alpha g\big(x - \alpha \nabla f(x)\big), \qquad \alpha \in \Big(0, \frac{2}{L_f}\Big), \tag{1.2c}$$

where $P_\alpha g(\cdot)$ and $e_\alpha g$ denote the proximal mapping and the Moreau envelope with respect to g, respectively, as defined below by (1.8) and (1.9) [49, Definition 1.22]. The sequence $(x_k)_{k\geq 0}$ generated by (1.2a) is known to converge to an optimal point x^* minimizing the objective function h, provided the step-size is chosen in the interval $\alpha \in (0, \frac{2}{L_f})$ [4, Corollary 28.9].

Algorithm (1.2) bears a structural resemblance to the SM. The evaluation of the proximal map $P_\alpha g$ may be considered as a 'basic algorithm' that is perturbed in (1.2a) by the gradient ∇f steps without losing convergence. For example, if $g = \delta_C(x)$ is the indicator function (see Subsection 1.5 for the basic notation) of a closed convex feasibility set $C \subset \mathbb{R}^n$, then $P_\alpha g = \Pi_C$ is just the orthogonal projection onto C, i.e., evaluating the proximal map returns a feasible point. Regarding f as the target function then enables to interpret (1.2a) as a *perturbed* version of the basic algorithm that returns a feasible point that, not only is superior, but is rather optimal. CO provides theory that explains how this desirable asymptotic behavior of an iterative algorithm is achieved. In the case of the FBS (1.2), the perturbed proximal iteration (1.2a) can be interpreted as stepping along *descent directions* provided the *generalized gradient mapping* ψ_α (cf. (1.2b)), where the gradient is taken with respect to the composite objective function h (1.1), and 'generalized' indicates the evaluation of the gradient $\nabla e_\alpha g$ of the Moreau envelope of the nonsmooth function g (cf. (1.2c)).

This structural similarity between the SM and CO that is further discussed in Subsection 2.4, motivated us to conduct a comparative study based on a particular but representative problem instance of the form (1.1).

Throughout this paper, we indicate by '...' when the notion 'basic algorithm' is used to highlight the similarity of the SM and CO. Otherwise, *basic algorithm* without '...' refers to the technical term of the SM defined in Section 2.

1.2. **Contribution and Objectives.** We consider the problem

$$\min_{x\in\mathbb{R}^n} h(x), \qquad h(x) = \frac{1}{2}\|Ax-b\|^2 + \lambda R(x) \tag{1.3}$$

in order to conduct a comparative study of the SM and CO. Problem (1.3) is representative in that it comprises a data fitting term and a regularizing term, similar to countless variational formulations of inverse problems for data restoration and recovery, etc. The matrix $A \in \mathbb{R}^{m\times n}$ is assumed to have dimensions $m < n$ such that an affine subspace of points $x \in \mathbb{R}^n$ exists that minimize the first least-squares term on the right-hand side of (1.3). We refer to Subsection 2.1 for further details and assumptions regarding (1.3).

For the data and functions of (1.3), a superiorization approach will not try to solve the unconstrained penalized objective function h. Instead, it will amount to choosing a basic algorithm for solving the well-known least-squares problem, e.g., [5], and perturbing it by subgradients of R, which is regarded as a target function. By contrast, regarding CO through FBS, one naturally identifies in (1.1) the smooth component $f = \frac{1}{2}\|A\cdot -b\|^2$ and the nonsmooth component $g = R$, in order to apply iteration (1.2). Thus, from the perspective of the SM, the roles of the functions f and g are *interchanged*: the proximal map $P_{\alpha g}$ defines the basic algorithm of the CO approach, which is perturbed by the gradient ∇f of the least-squares term f.

Due to the opposing roles of the two terms of (1.3) in the superiorization and optimization approaches, respectively, we also consider the alternative case in which $f := R$ and $g := \frac{1}{2}\|A\cdot -b\|^2$, in order to make CO closer to the SM and our study more comprehensive. However, since FBS requires f to be continuously differentiable, this requires to approximate R by a C^1-function R_τ, which can be done in a standard way [1, Sect. 2.8], [24], whenever a regularizing function R is used that takes the form of a support function in the sense of convex analysis [49, Section 8.E] (see Section 2.1 for details). We point out that although the function R_τ is smooth in the mathematical sense, it is still a highly nonlinear function from the viewpoint of numerical optimization.

Thus, we actually study the following two variants of (1.3),

$$\min_{x\in\mathbb{R}^n} h_u(x), \qquad h_u(x) = \frac{1}{2}\|Ax-b\|^2 + \lambda R_\tau(x), \tag{1.4a}$$

$$\min_{x\in\mathbb{R}^n} h_c(x), \qquad h_c(x) = \frac{1}{2}\|Ax-b\|^2 + \lambda R_\tau(x) + \delta_{\mathbb{R}^n_+}(x), \tag{1.4b}$$

where (1.4b) additionally takes into account nonnegativity constraints $x \geq 0$ that are often important in applications and turn the objective function h_c ('c' for: constrained) into a truly nonsmooth version of the objective function h_u ('u' for: unconstrained).

The **goals and contributions** of this paper are the following.

(1) *Superiorization methodology (SM).*
   (a) Examine the state of the art regarding *basic algorithms* that are adequate for applying the SM to (1.4). This concerns, in particular, the *resilience* of possible basic algorithms with respect to perturbations that are supposed to lower the value of the target function. See Subsection 2.2 and Subsection 3.2 for specific definitions of *perturbation resilience* and Table 1 on page 28 for a list of basic algorithms that are examined in this paper.

(b) Study various strategies for generating suitable perturbations in view of the given target function (see again Table 1). Here we also contribute a novel strategy utilizing the proximal mapping, which further underlines the structural similarity between the SM and CO.

(2) *Convex optimization (CO).*
Apply state-of-the-art forward-backward splitting to (1.4), and take into account that the summands f, g of the general form (1.1) can be identified in two alternative ways with the terms of the specific objective functions (1.4), as discussed above. By "state of the art" we mean
   (a) the application of *accelerated* variants of FBS that achieve the currently best known convergences rates $h(x_k) - h(x^*) = \mathcal{O}(\frac{1}{k^2})$ for any convex objective function h in a class encompassing both objective functions h_u, h_c of (1.4);
   (b) to work out the *inexact* evaluation of the proximal map of the FBS-step (1.2a). This amounts to adopt an *inner* iterative loop for solving the convex optimization problem (1.8) that defines $P_\alpha g$, along with a termination criterion that allows for stopping early this inner iteration *without* compromising convergence of the outer FBS iteration.

   Clearly, both measures (a) and (b) aim at maximizing computational efficiency.

(3) *SM vs. CO.*
The studies of (1) and (2) above enable us to compare SM and CO and to discuss their common and different aspects. The variety of aspects seems comprehensive enough to draw conclusions that should more generally hold for other convex problems of the form (1.1). We also indicate further promising directions of research in view of the gap remaining between the SM and CO.

1.3. **Related Work.**

1.3.1. *Superiorization.* The *superiorization method (SM)* was born when the terms and notions "superiorization" and "perturbation resilience", in the present context, first appeared in the 2009 paper [32] which followed its 2007 forerunner [6]. The ideas have some of their roots in the 2006 and 2008 papers [8, 9]. All these culminated in Ran Davidi's 2010 PhD dissertation [31] and the many papers since then cited in [19].

Introductory and advanced materials about the SM, accompanied by relevant references, can be found in the recent papers [22, 20], in particular, [20, Section 2] contains the basics of the superiorization methodology in condensed form. A comprehensive overview of the state-of-the-art and current research on superiorization appears in our continuously updated online bibliography that currently contains 110 items [19]. Research works in this bibliography include a variety of reports ranging from new applications to new mathematical results on the foundations of superiorization. A special issue entitled: "Superiorization: Theory and Applications" of the journal Inverse Problems [21] contains several interesting papers on the theory and practice of SM, such as [14], [41], [50], to name but a few. Later papers continue research on perturbation resilience, which lies at the heart of the SM, see, e.g., [10]. Another recent work attempts at analysing the behavior of the SM via the concept of concentration of measure [22]. Efforts to understand the very good performance

of the SM in practice recently motivated Byrne [13] to notice and study similarities between some SM algorithms and optimization methods.

There are two research directions in the general area of the SM. "Weak superiorization" assumes that the solution set is nonempty and uses "bounded perturbation resilience". In contrast, "strong superiorization" replaces asymptotic convergence to a solution point by ε-compatibility with a specified set and uses the notion of "strong perturbation resilience". The terms weak and strong superiorization, respectively, were proposed in in [29, Section 6] and [18].

The recent work [59, 39] applied the SM to the unconstrained linear least-squares problem. Specifically, the preconditioned conjugate gradient iteration (PCG) was used as the basic algorithm for solving the least-squares problem in the algebraic (fully-discretized) model of computed tomography (CT), and superiorized using the discretized total variation (TV) as a target function. The authors report that superiorized PCG compares favorably to a state of the art optimization algorithm FISTA [11], that minimizes (1.3) by accelerated FBS based on the choice $f = \frac{1}{2}\|A \cdot -b\|^2$ and $g = R$ in (1.1). The main contribution of [59] that we exploit in this paper, was to show perturbation resilience of the conjugate gradient (CG) method and hence to show that CG can be superiorized by *any* target function utilized as the criterion for superiorization.

1.3.2. *Convex Optimization.* Proximal iterations based on various operator splittings [26], while tolerating inexactness in terms of summable error sequences [53, 23], are well-known. More recent work has focused on inexactness criteria that can be checked computationally at each iteration. In particular, research has focused on extending the acceleration technique introduced by Nesterov [45] to such inexact proximal schemes. We refer to [47] and references therein for a recent comprehensive study and survey, including a novel accelerated inexact forward-backward scheme for minimizing finite convex objective functions. This scheme does not apply to (1.4b), however, due to the nonnegativity constraints.

We point out that problems of the form (1.4) have been extensively studied from the viewpoint of convex programming; see, e.g., [54, 12, 35, 30]. For example, FISTA [11] is well-known to be particularly efficient for convex problems with sparsity-enforcing ℓ_1 or TV regularizers, like R in (1.3), because proximal maps with respect to these regularizers, when iterated, can be carried out efficiently by shrinkage and TV-based denoising, respectively [35, 12].

Selecting the most efficient optimization approach is *not* the main objective of the present paper, however. Rather, by reversing the role of f and g in (1.1), we also consider FBS iterations that better mimic the structure of superiorization using R_τ as target function, even if this approach is *not* the most efficient one for optimizing (1.4).

1.4. **Organization.** Section 2 details our assumptions about problem (1.3) and the smoothing operation that turns R into R_τ so as to enable forward-backward splitting in two alternative ways (Subsection 2.1). Subsections 2.2 and 2.3 detail the superiorization approach and the convex optimization approach adopted in this paper, both from a top-level point of view, followed by a preliminary short discussion of common aspects and conceptual differences in Subsection 2.4. Subsections 2.2 and 2.3 provide templates for the concrete superiorization and optimization approaches worked out in Sections 3 and 4, respectively. Extensive numerical results are reported and discussed in Section 5. We conclude in Section 6.

1.5. **Preliminary Notions and Notation.** We set $[n] = \{1, 2, \ldots, n\}$ for any $n \in \mathbb{N}$. The Euclidean space is denoted by $\mathbb{R}^n$ and its nonnegative orthant by $\mathbb{R}^n_+$. The strictly positive real numbers are denoted by $\mathbb{R}_{++} = \{x \mid x > 0\}$. $\overline{\mathbb{R}} = [-\infty, +\infty]$ denotes the extended real line. For a function $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ the set $\operatorname{dom} f := \{x \in \mathbb{R}^n\colon f(x) < \infty\}$ denotes its effective domain.

We denote by K^* the polar cone of a cone $K \subseteq \mathbb{R}^n$, where K is either $\mathbb{R}^n$ or $\mathbb{R}^n_+$ in the following. For a closed convex set $C \subset \mathbb{R}^n$, $N_C(x)$ denotes the normal cone at $x \in C$ and δ_C is the indicator function of C, i.e., $\delta_C(x) = 0$ if $x \in C$, and $\delta_C(x) = +\infty$, otherwise. The orthogonal projection onto C is denoted by Π_C. In the specific case $C := K := \mathbb{R}^n_+$, we simply write $x_+ = \Pi_K(x)$ and, similarly, $x_- = \Pi_{K^*}(x) = x - x_+$. The identity matrix in $\mathbb{R}^{n\times n}$ is denoted by I_n or by I if the dimension is clear from the context. The Euclidean vector norm and inner product are denoted by $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$, respectively. For two vectors $x, y \in \mathbb{R}^n$ we write $x \perp y$ whenever they are orthogonal. For a matrix $A \in \mathbb{R}^{m\times n}$, $\|A\|_2$ denotes its spectral norm, $A^\top$ the transpose of A and $\|A\|$ the Frobenius norm induced by the inner product $\langle A, B\rangle = \operatorname{tr}(A^\top B)$, where $\operatorname{tr}(\cdot)$ returns the trace of a square matrix as argument. The least-squares solution of minimal Euclidean norm is denoted by x_{LS}.

We assume that images are discretized on n grid points in a two dimensional domain in $\mathbb{R}^2$. Using the one-dimensional discrete derivative operator

$$\partial_d : \mathbb{R}^d \to \mathbb{R}^d, \quad \partial_d = \begin{cases} -1, & i = j < d, \\ +1, & j = i + 1, \\ 0, & \text{otherwise}, \end{cases} \tag{1.5}$$

along each spatial direction, with $d \in \{M, N\}$, we define the discrete gradient matrix of an $M \times N$ discrete image

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} = \begin{pmatrix} \partial_M \otimes I_N \\ I_M \otimes \partial_N \end{pmatrix} \in \mathbb{R}^{2n\times n}, \tag{1.6}$$

where $\otimes$ stands for the Kronecker product and I_M, I_N are identity matrices of appropriate dimensions.

The following classes of convex functions are relevant to our investigation.

$$\mathcal{F}_c := \{f\colon \mathbb{R}^n \to \overline{\mathbb{R}} \mid f \text{ is convex, proper and lower semicontinuous}\}, \tag{1.7a}$$

$$\mathcal{F}^1_c(L) := \{f \in \mathcal{F}_c \mid f \in C^1 \text{ and } \nabla f \text{ is } L\text{-Lipschitz-continuous}\}, \tag{1.7b}$$

$$\mathcal{F}^1_c(L, \mu) := \{f \in \mathcal{F}^1_c(L) \mid f \text{ is } \mu\text{-strongly monotone convex}\}. \tag{1.7c}$$

We use subscripts L_f, μ_f to specify the corresponding concrete function f. f^* denotes the Legendre-Fenchel conjugate of $f \in \mathcal{F}_c$. It is well-known that $f^* \in \mathcal{F}_c$ if and only if $f \in \mathcal{F}_c$.

Given $f \in \mathcal{F}_c$ and $\alpha > 0$, the *proximal mapping* of the point $x \in \mathbb{R}^n$ is defined by

$$P_\alpha f(x) := \arg\min_{y\in\mathbb{R}^n} \Big\{ f(y) + \frac{1}{2\alpha}\|y - x\|^2 \Big\} \in \operatorname{dom} f, \tag{1.8}$$

whereas the *Moreau envelope* is the function defined by

$$e_\alpha f(x) := \inf_{y\in\mathbb{R}^n} \Big\{ f(y) + \frac{1}{2\alpha}\|y - x\|^2 \Big\}. \tag{1.9}$$

This function is continuously differentiable with gradient [49, Theorem 2.26]

$$\nabla e_\alpha f(x) = \frac{1}{\alpha}\big(x - P_\alpha f(x)\big). \tag{1.10}$$

2. Superiorization Versus Convex Optimization

2.1. **Problem Formulation.** We further specify problems (1.4). Throughout this paper, we assume

$$A \in \mathbb{R}_+^{m\times n}, \qquad \operatorname{rank}(A) = m < n. \tag{2.1}$$

In the case of discrete tomography considered in Section 5, where A represents the incidence relation of projection rays and cells covering a Euclidean domain, the full-rank condition can be ensured by, e.g., selecting a specific ray geometry, as illustrated in Section 5, or by slightly perturbing the nonzero entries in A [46].

Regarding data errors, we adopt the basic Gaussian noise model

$$(Ax - b)_i \sim \mathcal{N}(0, \sigma^2),\ i \in [m]. \tag{2.2}$$

Regarding the definition of $R_\tau \in \mathcal{F}_c^1(L_{R_\tau})$, we use the discrete gradient matrix (1.6) and index by $i \in [n]$ the vertices of the regular image grid. Then $\binom{(D_1x)_i}{(D_2x)_i} \in \mathbb{R}^2$ represents the gradient at location i in terms of the samples x^i, $i \in [n]$ of a corresponding image function. We set

$$R\colon \mathbb{R}^n \to \mathbb{R}_+, \qquad R(x) = \sum_{i\in[n]} \big(|(D_1x)_i| + |(D_2x)_i|\big). \tag{2.3}$$

To obtain a C^1-approximation, we note that R is a support function,

$$R(x) = \sup_{p\in C} \sum_{i\in[n]} \left\langle p_i, \begin{pmatrix}(D_1x)_i\\(D_2x)_i\end{pmatrix}\right\rangle, \tag{2.4a}$$

$$C = \big\{(p_1,\dots,p_i,\dots,p_n) \in \mathbb{R}^{2n} \mid p_i \in \mathbb{R}^2, \|p_i\|_\infty \le 1, i \in [n]\big\}, \tag{2.4b}$$

which enables the smooth approximation of R in a standard way [1, Sect. 2.8], [24], by

$$R_\tau(x) = \tau \sum_{i\in[n]} \left(\sqrt{1 + (D_1x/\tau)_i^2} + \sqrt{1 + (D_2x/\tau)_i^2}\right) \tag{2.5a}$$

$$= \sum_{i\in[n]} \left(\sqrt{\tau^2 + (D_1x)_i^2} + \sqrt{\tau^2 + (D_2x)_i^2}\right), \qquad 0 < \tau \ll 1. \tag{2.5b}$$

Lemma 2.1. *The gradient ∇R_τ is Lipschitz continuous with constant*

$$L_{R_\tau} \le \frac{1}{\tau}\|D\|_2^2 \le \frac{8}{\tau}. \tag{2.6}$$

Proof. We show that $\sup_{x\in\mathbb{R}^n} \|\nabla^2 R_\tau(x)\|_2^2 \le \frac{8}{\tau}$, which implies (2.6). Denoting by $D_{1;i}, D_{2;i}$ the row vectors of the discrete gradient matrix (1.6), we have

$$\nabla R_\tau(x) = \sum_{i\in[n]} \left(\frac{\langle D_{1;i}, x\rangle}{\sqrt{\tau^2 + \langle D_{1;i}, x\rangle^2}} D_{1;i} + \frac{\langle D_{2;i}, x\rangle}{\sqrt{\tau^2 + \langle D_{2;i}, x\rangle^2}} D_{2;i}\right) \tag{2.7}$$

Consider any summand on the right-hand side of (2.7) that has the form

$$s_{1;i}(x)D_{1;i} = \frac{\langle D_{1;i}, x\rangle}{\sqrt{\tau^2 + \langle D_{1;i}, x\rangle^2}} D_{1;i} \tag{2.8}$$

and $s_{2;i}D_{2;i}$ defined similarly. Using

$$\nabla s_{1;i}(x) = \widetilde{s}_{1;i}(x)D_{1;i} = \frac{1}{\sqrt{\tau^2 + \langle D_{1;i}, x\rangle^2}}\Big(1 - \frac{\langle D_{1;i}, x\rangle^2}{\tau^2 + \langle D_{1;i}, x\rangle^2}\Big) D_{1;i} \tag{2.9}$$

and a similar expression for $\nabla s_{2;i}(x) = \widetilde{s}_{2;i}(x)D_{2;i}$, we obtain

$$0 \le \widetilde{s}_{1;i}(x) \le \frac{1}{\tau}, \qquad 0 \le \widetilde{s}_{2;i}(x) \le \frac{1}{\tau} \tag{2.10}$$

and, consequently, with

$$\nabla^2 R_\tau(x) = \sum_{i\in[n]} \Big(D_{1;i}\nabla s_{1;i}(x)^\top + D_{2;i}\nabla s_{2;i}(x)^\top \Big) \tag{2.11}$$

the expression

$$\|\nabla^2 R_\tau(x)\|_2 = \sup_{\substack{y\in\mathbb{R}^n \\ \|y\|=1}} \sum_{i\in[n]} \Big(\big\langle y, D_{1;i}\nabla s_{1;i}(x)^\top y\big\rangle + \big\langle y, D_{2;i}\nabla s_{2;i}(x)^\top y\big\rangle \Big) \tag{2.12a}$$

$$\overset{(2.9)}{=} \sup_{\substack{y\in\mathbb{R}^n \\ \|y\|=1}} \sum_{i\in[n]} \Big(\widetilde{s}_{1;i}(x)\langle y, D_{1;i}D_{1;i}^\top y\rangle + \widetilde{s}_{2;i}(x)\langle y, D_{2;i}D_{2;i}^\top y\rangle \Big). \tag{2.12b}$$

Since all summands are nonnegative,

$$\|\nabla^2 R_\tau(x)\|_2 \overset{(2.10)}{\le} \frac{1}{\tau} \sup_{\substack{y\in\mathbb{R}^n \\ \|y\|=1}} \Big\langle y, \sum_{i\in[n]} (D_{1;i}D_{1;i}^\top + D_{2;i}D_{2;i}^\top) y\Big\rangle = \frac{1}{\tau} \sup_{\substack{y\in\mathbb{R}^n \\ \|y\|=1}} \big\langle y, (D_1^\top D_1 + D_2^\top D_2) y\big\rangle \tag{2.13a}$$

$$\overset{(1.6)}{=} \frac{1}{\tau} \sup_{\substack{y\in\mathbb{R}^n \\ \|y\|=1}} \langle y, D^\top D y\rangle \le \frac{1}{\tau}\|D\|_2^2 \le \frac{8}{\tau}, \tag{2.13b}$$

where the last equality follows from applying the Gerschgorin circle theorem to the matrix $D^\top D$. □

Lemma 2.2. *The function R_τ is globally Lipschitz continuous.*

Proof. We show that $\sup_{x\in\mathbb{R}^n} \|\nabla R_\tau(x)\| \le M$ for some $M > 0$, which implies the claim. By (2.7) and the proof of Lemma 2.1 we have

$$\nabla R_\tau(x) = \sum_{i\in[n]} \left(s_{1;i}(x)D_{1;i} + s_{2;i}(x)D_{2;i}\right), \tag{2.14}$$

where $|s_{1;i}(x)| \le 1$ and $|s_{2;i}(x)| \le 1$. This gives $\|\nabla R_\tau(x)\| \le \sum_{i\in[n]} \left(|D_{1;i}| + |D_{2;i}|\right) =: M$. □

2.2. **The Superiorization Methodology (SM).** In this subsection, we briefly describe the SM in general terms and relate it to the problems studied in this paper. Algorithm 1 will serve as a template for the concrete basic algorithms and their superiorized versions worked out later in Section 3.

Consider some given mathematically-formulated problem $\mathcal{T}$ and its solution set $\Omega_{\mathcal{T}}$. Typical instances of $\mathcal{T}$ are feasibility or optimization problems. Let $\mathcal{A}$ denote an algorithmic operator such that, for any given $x_0 \in \mathbb{R}^n$, it generates sequences $\mathcal{X}_{\mathcal{A}} := (x_k)$, by the iterative process

$$x_{k+1} = \mathcal{A}(x_k), \qquad \forall k \geq 0. \tag{2.15}$$

The following definition, providing a criterion for iterative algorithms that can be superiorized by bounded perturbations, is an adaptation of [15, Definition 1] to our notation used here.

Definition 2.3 (bounded perturbation resilience)**.** Let $\Omega_{\mathcal{T}}$ be the solution set of some given problem $\mathcal{T}$ and let $\mathcal{A}$ be an algorithmic operator. The algorithm (2.15) is said to be bounded perturbation resilient with respect to $\Omega_{\mathcal{T}}$ if the following holds: If the algorithm (2.15) generates sequences $\mathcal{X}_{\mathcal{A}}$ that converge to points in $\Omega_{\mathcal{T}}$ for all $x_0 \in \mathbb{R}^n$ then any sequence $\mathcal{Y}_{\mathcal{A}} = (y_k)$, generated by $y_{k+1} = \mathcal{A}(y_k + \beta_k v_k)$, also converges to a point in $\Omega_{\mathcal{T}}$ for any $y_0 \in \mathbb{R}^n$ provided all terms $\beta_k v_k$ are "bounded perturbations" meaning that the vector sequence (v_k) is bounded, $\beta_k \geq 0$ for all $k \geq 0$, and $\sum_{k=0}^{\infty} \beta_k < +\infty$.

The objective of the SM is to transform a *basic algorithm* defined by an algorithmic opertor $\mathcal{A}$ into a *superiorized version of that algorithm* with respect to a *target function* ϕ. If the basic algorithm (2.15) generates a sequence (x_k) that converges to some x^* and is bounded perturbation resilient, than the bounded perturbations, that typically employ nonascending directions for the target function ϕ, can be used to find another point $y^* \in \Omega_{\mathcal{T}}$ that satisfies $\phi(y^*) \leq \phi(x^*)$.

Definition 2.4 (nonascending direction)**.** Given a function $\phi\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \operatorname{dom} \phi$, we say that $v \in \mathbb{R}^n$ is a *nonascending vector for* ϕ *at* x, if $\|v\| \leq 1$ and there is a $\overline{t} > 0$ such that

$$\phi(x + tv) \leq \phi(x), \quad \text{for all } t \in (0, \overline{t}]. \tag{2.16}$$

A mathematical guarantee has not been found to date that the overall process of the superiorized version of the basic algorithm will not only retain its feasibility-seeking nature but also preserve globally the target function reductions. This fundamental question of the SM, called "the guarantee problem of the SM", received recently partial answers in [22, 29]. However, numerous works cited in [19] show that this global function reduction of the SM occurs in practice in many real-world applications.

Algorithm 1 precisely defines the superiorized version of a basic algorithm (2.15) generated by $\mathcal{A}$. It interleaves the iterations of the basic algorithmic operator $\mathcal{A}$ (line 8 of Algorithm 1) with perturbations by a target function reduction procedure S, defined by lines 4-7 of Algorithm 1, and can be summarized as the iteration

$$y_0 \in \mathbb{R}^n, \quad y_{k+1} = \mathcal{A}\big(S(y_k)\big), \quad \forall k \geq 0. \tag{2.17}$$

We observe the following.

(1) The procedure S works by applying N_k-times target function reduction steps in an additive manner (line 7).

Algorithm 1: Superiorized Version of a Basic Algorithm $\mathcal{A}$

initialization: Set $N \in \mathbb{N}$, $k = 0$ and pick an arbitrary initial point $y_0 \in \mathbb{R}^n$.
repeat
  Given a current iterate y_k, set $y_{k,0} \leftarrow y_k$ and pick $N_k \in \{0, 1, 2, \ldots, N\}$.
  for $i \leftarrow 0$ **to** $N_k - 1$ **do**
    Pick $\beta_{k,i} \in (0, 1]$ so that $\sum_{k=0}^{\infty} \sum_{i=0}^{N_k-1} \beta_{k,i} < \infty$.
    Pick a nonascending direction $v_{k,i}$ with respect to the target function ϕ at $y_{k,i}$ such that $\phi(y_{k,i} + \beta_{k,i} v_{k,i}) \leq \phi(y_{k,i})$.
    Compute the perturbation $y_{k,i} \leftarrow y_{k,i} + \beta_{k,i} v_{k,i}$.
  Apply the basic algorithm and update $y_{k+1} \leftarrow \mathcal{A}(y_{k,N_k-1})$.
  Increment $k \leftarrow k + 1$.
until *a stopping rule is met.*

(2) The procedure S also requires a summable sequence $(\beta_{k,i})$ that is, in practice, realized by choosing it to be a subsequence of $(\gamma_0 a^{k+i})$ where γ_0 and a are positive and $a < 1$.
(3) The perturbations by the procedure S need not be applied after *each* application of the basic algorithm. This corresponds to the choice $N_k = 0$. The question how to coordinate the number of target function reduction steps and the number of basic algorithmic operations in a superiorization algorithm is of prime importance in practical applications. We call it the *"balancing problem of SM"*.

In this paper we study the SM by applying Algorithm 1 with several basic algorithms for $\mathcal{A}$ and several target function reduction procedures for S. Details are given in Section 3. In view of (1.4), specific basic algorithms $\mathcal{A}$ are used, based on the conjugate gradient (CG) iteration [51, Sect. 8.3] and the (projected) Landweber method (LW) [2], respectively, to reduce each of the three functions

$$g_u(x) := \frac{1}{2}\|Ax - b\|^2, \tag{2.18a}$$

$$g_u^\mu(x) := \frac{1}{2}\|Ax - b\|^2 + \frac{\mu}{2}\|x\|^2, \tag{2.18b}$$

$$g_c(x) := \frac{1}{2}\|Ax - b\|^2 + \delta_{\mathbb{R}^n_+}(x), \tag{2.18c}$$

where the unconstrained and nonnegativity-constrained cases are indicated by subscripts '*u*' and '*c*', respectively. The smooth regularization with parameter μ in (2.18b) is chosen to ensure applicability of the CG iteration as a basic algorithm to the unconstrained least-squares problem, which achieves state of the art performance.

In each case, Algorithm 1 uses as stopping rule the notion of ε-compatibility. For a given problem $\mathcal{T}$, a *proximity function* $\mathcal{P}r_{\mathcal{T}} : \mathbb{R}^n \to \mathbb{R}_+$ measures how incompatible x is with it. Given an $\varepsilon > 0$, we say that x is *ε-compatible* with $\mathcal{T}$ if $\mathcal{P}r_{\mathcal{T}}(x) \leq \varepsilon$. We look, for an $\varepsilon > 0$, a given problem $\mathcal{T}$

and a chosen proximity function $\mathcal{P}r_{\mathcal{T}}$ at the set $\Gamma_\varepsilon \subset \mathbb{R}^n$ of the form

$$\Gamma_\varepsilon := \Big\{ x \in \mathbb{R}^n \mid \mathcal{P}r_{\mathcal{T}}(x) \leq \varepsilon \Big\}. \tag{2.19}$$

We refer to such sets in the sequel as "proximity sets".

For the problems of reducing the three functions in (2.18), via the corresponding basic algorithms for each of them, we run the superiorized version of each basic algorithm until a point x^* in the corresponding proximity set

$$\Gamma_{u,\varepsilon} := \{x \in \mathbb{R}^n \mid g_u(x) \leq \varepsilon\}, \tag{2.20a}$$
$$\Gamma^{\mu}_{u,\varepsilon} := \{x \in \mathbb{R}^n \mid g^{\mu}_u(x) \leq \varepsilon\}, \tag{2.20b}$$
$$\Gamma_{c,\varepsilon} := \{x \in \mathbb{R}^n \mid g_c(x) \leq \varepsilon\} \tag{2.20c}$$

is obtained, respectively, where ε is fixed, depending on the noise level (2.2).

The target functions that we use, correponding to (2.18), are defined as

$$\phi_u(x) := R_\tau(x), \tag{2.21a}$$
$$\phi_c(x) := R_\tau(x) + \delta_{\mathbb{R}^n_+}(x), \tag{2.21b}$$

where R_τ is given by (2.5).

2.3. **Convex Optimization (CO).** Analogous to Section 2.2, we elaborate here on the optimization scheme based on forward-backward splitting (FBS), to fix notation and to provide in Algorithm 2 a template for the concrete algorithms worked out in Section 4.

This algorithm is based on a decomposition (1.1) of a given objective function $h = f + g \in \mathcal{F}_c$, where f is continuously differentiable with L_f-Lipschitz continuous gradient ∇f.

Algorithm 2: Accelerated Forward-Backward Iteration with Inexact Proximal Points

initialization: Set $k = 0$, $x_0, y_0 \in \operatorname{dom} g$, $t_0 > 1$, $a \in (0, 2)$. Pick sequences $(\alpha_k) \subset \big(0, \frac{2-a}{L_f}\big]$, $(a_k) \subset [a, 2 - \alpha_k L_f]$, $(\varepsilon_k) \subset \mathbb{R}_{++}$.
repeat
  Set $z_0 \in \mathbb{R}^n$ and $l = 0$.
  repeat
    Generate the next iterate z_{l+1} of a sequence (z_l) converging to $P_{\alpha_k} g\big(y_k - \alpha_k \nabla f(y_k)\big)$.
    Increment $l \leftarrow l + 1$.
  until *z_l has error $\leq \varepsilon_k$; then set* $\overline{z}_k = z_l$.
  Update $x_{k+1} = \overline{z}_k$.
  Update $t_{k+1} = \frac{1}{2}\Big(1 + \Big(1 + 4\frac{a_k \alpha_k}{a_{k+1}\alpha_{k+1}} t_k^2\Big)^{1/2}\Big)$.
  Update $y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k) + (1 - a_k)\frac{t_k}{t_{k+1}}(y_k - x_{k+1})$.
  Increment $k \leftarrow k + 1$.
until *a stopping rule is met.*

Algorithm 2, adopted from [56], generalizes the basic FBS iteration (1.2a) as follows.

- Besides the primal sequence (x_k) with step-sizes (α_k), a sequence of auxiliary point (y_k) and step sizes (t_k) are used in order to accelerate FBS.
- Acceleration is achieved
  - by computing proximal points of $y_k - \alpha_k \nabla f(y_k)$ instead of $x_k - \alpha_k \nabla f(x_k)$;
  - by inexact evaluation of the proximal mapping $P_{\alpha_k} g$.

Note that Algorithm 2 without acceleration reduces to the basic forward-backward iteration (1.2).

Corresponding to the discussion preceding the two problems (1.4), the following four variants of the decomposition (1.1) are applied to (1.4) in Section 4. The first two splittings handle the unconstrained case indicated by the subscript 'u'.

$$h_u(x) = f_u(x) + g_u(x), \tag{2.22a}$$

$$f_u(x) = \lambda R_\tau(x), \qquad g_u(x) = \frac{1}{2}\|Ax - b\|^2, \tag{2.22b}$$

$$f_u(x) = \frac{1}{2}\|Ax - b\|^2, \qquad g_u(x) = \lambda R_\tau(x). \tag{2.22c}$$

The last two splittings handle the constrained case indicated by the subscript 'c'.

$$h_c(x) = f_u(x) + g_c(x), \tag{2.23a}$$

$$f_u(x) = \lambda R_\tau(x), \qquad g_c(x) = \frac{1}{2}\|Ax - b\|^2 + \delta_{\mathbb{R}^n_+}, \tag{2.23b}$$

$$f_u(x) = \frac{1}{2}\|Ax - b\|^2, \qquad g_c(x) = \lambda R_\tau(x) + \delta_{\mathbb{R}^n_+}. \tag{2.23c}$$

The reason for considering the splittings (2.22c) and (2.23c), in addition to (2.22b), (2.23b), is discussed in the following subsection.

2.4. **SM vs. CO: Common Aspects and Conceptual Differences.** The two splittings (2.22b) and (2.23b) resemble structurally the SM in that the least-squares term can be viewed as the task to which the 'basic algorithm' is applied, by evaluating the proximal map $P_{\alpha_k} g$ (this does not change the set of minima of g). Furthermore, this 'basic algorithm' is perturbed by function reduction steps of the target function f, performed by nonascending directions via negative gradients.

However, a key structural difference is that the roles of inner and outer iterations in Algorithms 1 and 2 are interchanged. The SM uses multiple inner iterations (lines 4-7 of Algorithm 1) in order to accumulate bounded perturbations, whereas the basic algorithm $\mathcal{A}$ is updated in a single step of the outer loop. By contrast, CO performs several iterations of its 'basic algorithm' of evaluating the proximal map $P_{\alpha_k} g$ in an inner loop, whereas the perturbation is updated in a single step of the outer loop. Another difference is that the number N_k of inner iterations in Algorithm 1 is a tuning parameter of the SM, whereas the corresponding number in the CO scheme follows from a mathematical termination criterion, that ensures a sufficiently descreasing error level so as to guarantee convergence of the outer loop to a global minimizer of the objective function h.

These differences motivated us to study also the above splitting after interchanging the roles of the functions f and g. This defines the two splittings (2.22c) and (2.23c). Adopting the viewpoint of the SM, this entails many iterative reduction steps with respect to the target function g through evaluating the proximal mapping $P_{\alpha_k} g$.

3. SUPERIORIZATION

In this section, we present our SM algorithms, outlined in Subsection 2.2, by specifying – cf. (2.17) – different basic algorithmic operators $\mathcal{A}(\cdot)$, and various target function reduction procedures $S(\cdot)$. Instances of $\mathcal{A}$ solve problems (2.18), whereas instances of S take into account the target functions (2.21) in order to superiorize the basic algorithm implemented by $\mathcal{A}$. Algorithm 1 (see page 25) details the interplay between $\mathcal{A}$ (line 8) and its superiorization through S (lines 4-7).

Specific instances $\mathcal{A}_{\mathrm{CG}}$, $\mathcal{A}_{\mathrm{LW}}$ and $\mathcal{A}_{\mathrm{LW}+}$ of the basic algorithm $\mathcal{A}$ are worked out in Subsection 3.1, based on the conjugate gradient (CG) iteration [51, Sect. 8.3] and the (projected) Landweber method (LW) [2], respectively. Specific instances S_{∇} and S_{prox} or $S_{\mathrm{prox}+}$ of the target function reduction procedures S are worked out in Subsection 3.3, based on negative gradient steps or on generalized gradient steps, respectively. The latter are defined via the proximal mapping in order to handle nonsmooth convex constraints, using a strategy which is common practice in convex optimization – cf. (1.2).

Subsection 3.2 is devoted to the notion of strong perturbation resilience. Table 1 lists all combinations of $\mathcal{A}$ and $\mathcal{S}$ that we examine as the SM approaches for solving the unconstrained and constrained least-squares problems (2.18) with target functions (2.21). These include the novel approaches that interleave either of the basic algorithmic operators $\mathcal{A}_{\mathrm{CG}}$, $\mathcal{A}_{\mathrm{LW}}$ or $\mathcal{A}_{\mathrm{LW}+}$ with the target function reduction procedures S_{prox} or $S_{\mathrm{prox}+}$ based on proximal mappings.

Method	Basic algorithm	Target function reduction procedure	Perturbation resilience
`GradSupCG`	$\mathcal{A}_{\mathrm{CG}}$	S_{∇} from Algorithm 9	✓, see [59, Thm. A.1]
`GradSupLW`	$\mathcal{A}_{\mathrm{LW}}$	S_{∇} from Algorithm 9	✓, see [34]
`ProxSupCG`	$\mathcal{A}_{\mathrm{CG}}$	$S_{\mathrm{prox}}(x, \beta_k)$ from (3.7)	follows from Prop 3.5 and [59, Thm. A.1]
`ProxSupLW`	$\mathcal{A}_{\mathrm{LW}}$	$S_{\mathrm{prox}}(x, \beta_k)$ from (3.7)	follows from [42, 44]
`ProxCSupCG`	$\mathcal{A}_{\mathrm{CG}}$	$S_{\mathrm{prox}+}(x, \beta_k)$ from (3.8)	open
`ProxCSupLW`	$\mathcal{A}_{\mathrm{LW}}$	$S_{\mathrm{prox}+}(x, \beta_k)$ from (3.8)	open
`GradSupProjLW`	$\mathcal{A}_{\mathrm{LW}+}$	S_{∇} from Algorithm 9	✓, see [42, 34]
`ProxSupProjLW`	$\mathcal{A}_{\mathrm{LW}+}$	$S_{\mathrm{prox}}(x, \beta_k)$ from (3.7)	follows from [44]

TABLE 1. List of superiorized algorithms. The left column shows the name tags used for the superiorized algorithms that are numerically evaluated in Section 5. The second column from the left shows the basic algorithm used while the third column indicates our strategy for the target function reduction. The last column indicates if the perturbation resilience property holds or is still open. Checkmark means that the property has been proven.

3.1. **Basic Algorithms.** We first consider the basic algorithms for the task of minimizing the unconstrained least-squares objective g_u (2.18a). This problem always has a solution and $\min_{x\in\mathbb{R}^n} g_u(x) = 0$ holds due to the *underdetermined* full-rank matrix A. The set of minimizers is an affine

subspace that has the dimension of the nullspace of A. In view of noisy measurements however, we merely wish to find an approximate minimizer of g_u, one that is ε-compatible with the set of minimizers, i.e., an element of $\Gamma_{u,\varepsilon}$ (2.20a), where ε is fixed, depending on the noise level (2.2). To this end we consider a basic algorithm that monotonically decreases the function g_u.

A straightforward choice is the Landweber iteration, presented in Algorithm 3, which is a damped gradient descent method with a constant step-size parameter $\gamma \in (0, 2/\|A\|_2^2)$, see line 3 of Algorithm 4, which describes the basic algorithmic operator $\mathcal{A}_{LW}$. It makes use of the spectral norm of A, which has to be computed or estimated beforehand. We terminate the iteration when g_u has reached an ε-compatiblity dictated by the noise level.

Algorithm 3: The Landweber Algorithm

1 **initialization:** Set $k = 0$, choose an arbitrary initial point $x_0 \in \mathbb{R}^n$, $\varepsilon > 0$ and $\gamma \in (0, 2/\|A\|_2^2)$.
2 **while** $g(x_k) > \varepsilon$ **do**
3 Given a current iterate x_k calculate $x_{k+1} = \mathcal{A}_{\text{LW}}(x_k, \gamma)$ by Algorithm 4.
4 Increment $k \leftarrow k + 1$.

Algorithm 4: $x_{\text{new}} \leftarrow \mathcal{A}_{\text{LW}}(x, \gamma)$

input : Current iterate x.
output : Updated iterate x_{new}.
parameter : A step-size parameter $\gamma \in (0, 2/\|A\|_2^2)$.
1 **begin**
2 $g = A^\top(Ax - b)$,
3 $x_{\text{new}} = x - \gamma g$.

The Landweber iteration can be easily extended to handle nonnegativity constraints and be used as a basic algorithm for the task of minimizing g_c (2.18c) by, simple to execute, projections onto the nonnegative orthant. The resulting basic algorithm for finding an approximate, i.e., ε-compatible, solution of g_c is Algorithm 5. We note that the nonnegative least-squares system might have an empty solution set as opposed to the plain least-squares problem. Therefore, we assume that for an appropriate choice of ε the proximity set $\Gamma_{c,\varepsilon}$ (2.20c) is nonempty.

Algorithm 5: The Projected Landweber Algorithm

1 **initialization:** Set $k = 0$, choose an arbitrary initial point $x_0 \in \mathbb{R}^n$, $\varepsilon > 0$ and $\gamma \in (0, 2/\|A\|_2^2)$.
2 **while** $g_c(x_k) > \varepsilon$ **do**
3 Given a current iterate x_k calculate $x_{k+1} = \mathcal{A}_{\text{LW}_+}(x_k, \gamma)$ by Algorithm 6.
4 Increment $k \leftarrow k + 1$.

Algorithm 6: $x_{\text{new}} \leftarrow \mathcal{A}_{\text{LW}+}(x, \gamma)$

input : Current iterate x.
output : Updated iterate x_{new}.
parameter : A step-size parameter $\gamma \in (0, 2/\|A\|_2^2)$.

1 **begin**
2 $\quad g = A^\top(Ax - b)$,
3 $\quad x_{\text{new}} = \Pi_{\mathbb{R}^n_+}(x - \gamma g) = \max(x - \gamma g, 0)$.

It is well-known that the (projected) Landweber iteration exhibits slow convergence for ill-conditioned problems, becomes unstable in further iterations and needs to be stopped early because of semi-convergence. This motivates us to consider an additional basic algorithm that copes better with ill-conditioned problems. For this purpose we adopt the following CG algorithm, listed as Algorithm 7, which is a slight modification of [59, Algorithm 8]. Our choice will be justified by the good numerical results obtained by a superiorized CG algorithm for an undersampled tomographic problem, reported and discussed in Section 5.

Algorithm 7: The Conjugate Gradient Algorithm

1 **initialization:** Set $k = 0$, pick an arbitrary initial point $x_0 \in \mathbb{R}^n$ and choose $\mu > 0$ small and $\varepsilon > 0$. Set $p_0 = A^\top(b - Ax_0) + \mu x_0$ and $h_0 = A^\top A p_0 + \mu p_0$.
2 **while** $g_\mu(x_k) > \varepsilon$ **do**
3 $\quad$ Given a current iterate x_k and auxilary vectors p_k and h_k,
4 $\quad$ calculate $(x_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\text{CG}}(x_k, p_k, h_k)$ by Algorithm 8.
5 $\quad$ Increment $k \leftarrow k + 1$.

Algorithm 8: $(x_{\text{new}}, p_{\text{new}}, h_{\text{new}}) \leftarrow \mathcal{A}_{\text{CG}}(x, p, h)$

input : Current iterates x, p, h.
output : Updated iterates $x_{\text{new}}, p_{\text{new}}, h_{\text{new}}$.

1 **begin**
2 $\quad g = A^\top(Ax - b) + \mu x$,
3 $\quad \beta = \langle g, h\rangle / \langle p, h\rangle$,
4 $\quad p_{\text{new}} = -g + \beta p$,
5 $\quad h_{\text{new}} = A^\top A p_{\text{new}} + \mu p_{\text{new}}$,
6 $\quad \gamma = -\langle g, p_{\text{new}}\rangle / \langle p_{\text{new}}, h_{\text{new}}\rangle$,
7 $\quad x_{\text{new}} = x + \gamma p_{\text{new}}$.

The updates of the current iterates in lines 2 and 6 of Algorithm 8 differ from the corresponding updates in [59, Algorithm 8], because in our case the CG iteration minimizes g_u^μ (2.18b) and is

applied to the regularized least-squares problem

$$\min \frac{1}{2}\left\| \begin{pmatrix} A \\ \sqrt{\mu} I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|^2, \tag{3.1}$$

with a small parameter $\mu > 0$, in order to obtain a well-defined algorithm for the underdetermined scenario (2.1) considered here. We show however that for an appropriate choice of parameter μ minimizing g_u^μ is equivalent to finding the minimal Euclidean norm solution of g_u (2.18a).

Lemma 3.1. *Let $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m < n$ and define g_u and g_u^μ as in (2.18a) and (2.18b) respectively. If x_{LS} is the minimizer of*

$$\min \|x\| \quad \textit{subject to} \quad x \in \operatorname{argmin}_{y\in\mathbb{R}^n} g_u(y) \tag{3.2}$$

then there exists a parameter $\mu = \mu(x_{\mathrm{LS}}) \geq 0$ such that x_{LS} is also a minimizer of g_u^μ in (2.18b).

Proof. Due to the assumption $\operatorname{rank}(A) = m$ we always have $b \in \mathcal{R}(A)$. Consequently, the set of minimizers of g_u is equal to the solution set of the consistent linear system $Ax = b$. Thus, problem (3.2) is equivalent to

$$\min \|x\| \quad \text{subject to} \quad Ax = b. \tag{3.3}$$

Now problem (3.3) can be equivalently rewritten as

$$\min \frac{1}{2}\|x\|^2 \quad \text{subject to} \quad \frac{1}{2}\|Ax - b\|^2 \leq 0, \tag{3.4}$$

in view of the monotonicity of $t \mapsto \frac{1}{2}t^2$ and the positivitivity of the Euclidean norm. Clearly, x_{LS} also solves (3.3) and (3.4). The full rank assumption implies that the relative interior of the feasible set in (3.4) is nonemepty. Thus, strong duality holds for (3.4) and, therefore, there exists a dual solution y^* of (3.4). We consider the Lagrangian of (3.4) that reads

$$L(x, y) = \frac{1}{2}\|x\|^2 + y\Big(\frac{1}{2}\|Ax - b\|^2\Big).$$

The primal-dual optimal pair (x_{LS}, y^*) is a saddle-point of L and $L(x_{\mathrm{LS}}, y^*) \leq L(x, y^*)$ for all $x \in \mathbb{R}^n$. Thus $x \mapsto \mu L(x, y^*)$ is minimized by x_{LS}, which differs from the objective function g_u^μ in (2.18b) only by the scalar μy^* in front of the least-squares term. In conclusion, we can chose $\mu = 1/y^*$. Since y^* is connected to x_{LS} through the optimality conditions, the parameter μ depends on x_{LS} too. □

Note that the CG Algorithm 7, as well as [59, Algorithm 8], differ from the classic CG algorithm for least-squares in that the gradient of the least-squares term is evaluated at every individual iterate, and not by the, commonly used, computationally more efficient update

$$g_{k+1} = g_k + A^\top A p_k + \mu p_k, \tag{3.5}$$

see [51, Sect. 8.3]. As discussed in [59], by doing so, one obtains an algorithm that performs exactly as CG, but is resilient to bounded perturbations.

The recent work [59, 39] also considers the *preconditioned* CG iteration as a basic algorithm for the SM. While in case of overdetermined systems preconditioning is mandatory, CG without preconditioning works well in our underdetermined scenario. However, the CG method cannot be directly applied to a consistent linear system with nonnegativity constraints. There exist CG

versions for nonnegativity and box-constraints that use an active set strategy but are hampered by frequent restarts of the CG iteration. A more efficient box-constrained CG method for nonnegative matrices can be found in [55], but even its convergence theory it still open, let alone its perturbation resilience.

As a viable alternative, we include constraints in the CG method via superiorization, as detailed below in Subsection 3.3.

3.2. **Perturbation Resilience.** In the sequel, let $\mathcal{A}$ denote either of the operators $\mathcal{A}_{\mathrm{LW}}$, $\mathcal{A}_{\mathrm{LW}+}$, $\mathcal{A}_{\mathrm{CG}}$ that define the basic algorithms in Algorithms 3, 5 or 7, respectively. Likewise, let Γ_ε denote either of the proximity sets $\Gamma_{u,\varepsilon}$ of (2.20a), $\Gamma_{c,\varepsilon}$ of (2.20c) and $\Gamma^{\mu}_{u,\varepsilon}$ of (2.20b), defined as the sub-level sets of the functions g_u, g_c and g^{μ}_u in (2.18), respectively. We assume that $\varepsilon > 0$ has been chosen large enough such that $\Gamma_\varepsilon \neq \emptyset$. As a consequence, for any $x_0 \in \mathbb{R}^n$, the sequence $\mathcal{X}_{\mathcal{A}} := (x_k)$, generated by by the iterative process (2.15) will terminate at a point in $x^* \in \Gamma_\varepsilon$.

Bounded perturbation resilience with respect to a nonempty solution set, recall Definition 2.3, is known to hold for the basic Landweber and projected Landweber iteration, see [42, 34] and Table 1. This implies bounded perturbation resilience of $\mathcal{A}_{LW}$ and $\mathcal{A}_{LW+}$.

A notion of perturbation resilience that is more relevant to concrete applications considers algorithmic *termination* instead of asymptotic convergence of the basic algorithm and its superiorized version. Termination is defined in terms of the ε-output of a sequence.

Definition 3.2 (ε-output of a sequence with respect to Γ_ε)**.** For some $\varepsilon > 0$, a nonempty proximity set Γ_ε and a sequence $\mathcal{X} := (x_k)$ of points, the ε-output $O\left(\Gamma_\varepsilon, \mathcal{X}\right)$ of the sequence $\mathcal{X}$ with respect to Γ_ε is defined to be the element x_k with smallest $k \in \mathbb{N}$ such that $x_k \in \Gamma_\varepsilon$.

Remark 3.3. Due to our assumption that $\Gamma_\varepsilon \neq \emptyset$, the ε-output $O\left(\Gamma_\varepsilon, \mathcal{X}_{\mathcal{A}}\right)$ of the sequences $\mathcal{X}_{\mathcal{A}}$ exists, for either instance $\mathcal{A}_{\mathrm{LW}}$, $\mathcal{A}_{\mathrm{LW}+}$ or $\mathcal{A}_{\mathrm{CG}}$ of $\mathcal{A}$, with respect to either proximity set in (2.20). If $\mathcal{X}_{\mathcal{A}}$ is an *infinite* sequence generated by the iterative process (2.15), then $O\left(\Gamma_\varepsilon, \mathcal{X}_{\mathcal{A}}\right)$ is the *output* produced by that algorithm when we add to it a stopping rule based on Γ_ε, as done in Algorithms 3, 5 and 7.

We define the notion of strong perturbation resilience from [36, Subsection II.C], adapted to our notation.

Definition 3.4 (strong perturbation resilience)**.** Let Γ_ε denote the proximity set and let $\mathcal{A}$ be an algorithmic operator as in (2.15). The algorithm (2.15) is said to be strongly perturbation resilient if the following conditions hold:

(1) there is an $\varepsilon > 0$ such that the ε-output $O\left(\Gamma_\varepsilon, \mathcal{X}_{\mathcal{A}}\right)$ of the sequence $\mathcal{X}_{\mathcal{A}}$ exists, for every $x_0 \in \mathbb{R}^n$;

(2) for all $\varepsilon \geq 0$ such that $O\left(\Gamma_\varepsilon, \mathcal{X}_{\mathcal{A}}\right)$ is defined for every $x_0 \in \mathbb{R}^n$, we also have that $O\left(\Gamma_{\varepsilon'}, \mathcal{Y}_{\mathcal{A}}\right)$ is defined, for every $\varepsilon' \geq \varepsilon$, and for every sequence $\mathcal{Y}_{\mathcal{A}} = (y_k)$ generated by

$$y_{k+1} = \mathcal{A}(y_k + \beta_k v_k), \qquad \forall k \geq 0, \tag{3.6}$$

where the terms $\beta_k v_k$ are bounded perturbations as specified by Definition 2.3.

Note that if a problem $\mathcal{T}$ has a nonempty solution set $\Omega_{\mathcal{T}} \neq \emptyset$ which is contained in a proximity set $\Omega_{\mathcal{T}} \subset \Gamma_\varepsilon$, then bounded perturbation resilience implies strong perturbation resilience. Sufficient conditions for strong perturbation resilience appeared in [36, Theorem 1]. We note that A_{CG} has been shown to be strongly perturbation resilient in [59, Thm. A.1].

3.3. **Superiorization by Bounded Perturbations.** In this subsection we specify in detail the superiorized versions of the basic algorithms 3, 5 and 7 discussed above that fit into the general framework of Algorithm 1, see also [36, page 5537]. Since we already fixed the algorithmic operators to be either $\mathcal{A}_{\mathrm{LW}}$, $\mathcal{A}_{\mathrm{LW}+}$ or $\mathcal{A}_{\mathrm{CG}}$ we need to specify the target function reduction procedures S, that we will use to compute the perturbations of the basic algorithms.

3.3.1. *Target Function Reduction Procedures.* The general target function reduction procedure described in Algorithm 1, lines 4-7, can be easily adapted for reducing the differentiable target function ϕ_u (2.21) using nonascent directions based on normalized negative gradients. This leads to Algorithm 9, that is similar to [39, Algorithm 1]. Note that S_∇ repeats κ-times steps along nor-

Algorithm 9: $(x_{\mathrm{new}}, \ell_{\mathrm{new}}) \leftarrow S_\nabla(x, \ell, a, \gamma_0, \kappa)$

input : Current iterate x, current exponent ℓ.
output : Superior iterate x_{new}, updated exponent ℓ_{new}.
parameter : The parameters ℓ, a, γ_0, κ.

1 **begin**
2 $y \leftarrow x$.
3 **for** $i = 1, \ldots, \kappa$ **do**
4 **if** $\nabla R_\tau(y) \neq 0$ **then**
5 $v \leftarrow -\nabla R_\tau(y)/\|\nabla R_\tau(y)\|$, see (2.7),
6 **else**
7 $v \leftarrow 0$;
8 **repeat**
9 $\gamma \leftarrow \gamma_0 a^\ell$
10 $y_{\mathrm{new}} \leftarrow y + \gamma v$
11 $\ell \leftarrow \ell + 1$
12 **until** $R_\tau(y_{\mathrm{new}}) \leq R_\tau(y)$.
13 $y \leftarrow y_{\mathrm{new}}$.
14 $\ell_{\mathrm{new}} \leftarrow \ell$,
15 $x_{\mathrm{new}} \leftarrow y$.

malized negative gradients (line 10) to reduce the target function ϕ_u. Each normalized negative gradient direction is scaled by parameter γ of the form $\gamma_0 a^\ell$ for some $\ell \in \mathbb{N}$ that as specified by procedure S_∇. As a consequence, perturbations can become very small since $a \in (0, 1)$.

We consider a second variant of the target function reduction procedure S that allows better control of the perturbation parameters $\beta_{k,i}$ from Algorithm 1 line 7. For the target function ϕ_u (2.21), we define

$$S_{\text{prox}}(x,\beta) := P_\beta \phi_u(x) = \arg\min_z \Big\{ R_\tau(z) + \frac{1}{2\beta}\|z-x\|^2 \Big\}. \tag{3.7}$$

For the reduction of the target function ϕ_c (2.21), we define

$$S_{\text{prox}+}(x,\beta) := P_\beta \phi_c(x) = \arg\min_z \Big\{ R_\tau(z) + \delta_{\mathbb{R}^n_+}(z) + \frac{1}{2\beta}\|z-x\|^2 \Big\}, \tag{3.8}$$

which allows us to incorporate the constraints in the target function reduction procedure.

Perturbation by proximal points has been done in [44], as well as in [40], as we became aware while preparing this manuscript. However, the authors in [44] do not take into account constraints when calculating the perturbations.

One might also argue that computing such nonascent directions is expensive and contradicts the spirit of the SM. However, the proximal point in (3.7) or (3.8) can be computed efficiently in our case, e.g., by the box-constrained L-BFGS method from [7] that computes a highly accurate solution within few iterations, as we demonstrate in Section 5.

Clearly, if $y_{k+\frac{1}{2}} := S_{\text{prox}}(y_k,\beta_k)$, then

$$\phi_u(y_{k+\frac{1}{2}}) \le \phi_u(y_{k+\frac{1}{2}}) + \frac{1}{2\beta_k}\|y_{k+\frac{1}{2}} - y_k\|^2 \tag{3.9a}$$

$$\le \phi_u(y_k) + \frac{1}{2\beta_k}\|y_k - y_k\|^2 = \phi_u(y_k), \tag{3.9b}$$

and equality holds if and only if $y_{k+\frac{1}{2}} = y_k$, in view of the definition of the proximal mapping.

Furthermore, if $y_{k+\frac{1}{2}} := S_{\text{prox}+}(y_k,\beta_k)$, then

$$\phi_c(y_{k+\frac{1}{2}}) \le \phi_u(y_{k+\frac{1}{2}}) + \delta_{\mathbb{R}^n_+}(y_{k+\frac{1}{2}}) + \frac{1}{2\beta_k}\|y_{k+\frac{1}{2}} - y_k\|^2 \tag{3.10a}$$

$$\le \phi_u(y_k) + \delta_{\mathbb{R}^n_+}(y_k) + \frac{1}{2\beta_k}\|y_k - y_k\|^2 = \phi_c(y_k). \tag{3.10b}$$

Therefore, the *decrease* of the target function by applying S_{prox} or $S_{\text{prox}+}$ is guaranteed in either case.

Proposition 3.5. *Consider any basic operator $\mathcal{A}$ and a summable nonnegative sequence (β_k), i.e., $\beta_k \ge 0$ and $\sum_{k=0}^{\infty}\beta_k < +\infty$. Let $\phi := \phi_c$ from (2.21) and define the target function reduction procedure by $S(x,\beta_k) := P_{\beta_k}\phi(x)$. Then the perturbations $y_{k+\frac{1}{2}} = S(y_k,\beta_k)$ of the iterates y_k generated by the superiorized version of the basic algorithm (2.17) are bounded perturbations, i.e.,*

$$y_{k+\frac{1}{2}} = y_k + \beta_k v_k, \tag{3.11}$$

where the vector sequence (v_k) is bounded.

Proof. By definition we have $y_{k+\frac{1}{2}} = P_{\beta_k}\phi(y_k)$. Thus, by (1.10) we have

$$-\beta_k v_k := \beta_k \nabla e_{\beta_k}\phi(y_k) = y_k - P_{\beta_k}\phi(y_k) = y_k - y_{k+\frac{1}{2}}. \tag{3.12}$$

Using the variational inequality that characterizes $y_{k+\frac{1}{2}}$, gives

$$\frac{1}{\beta_k}\langle y_k - y_{k+\frac{1}{2}}, y - y_{k+\frac{1}{2}}\rangle + \phi(y_{k+\frac{1}{2}}) - \phi(y) \leq 0, \qquad \forall y \in \operatorname{dom}\phi. \tag{3.13}$$

Setting $y = y_k$, we get

$$\|y_k - y_{k+\frac{1}{2}}\|^2 \leq \beta_k|\phi(y_k) - \phi(y_{k+\frac{1}{2}})| \leq M\beta_k\|y_k - y_{k+\frac{1}{2}}\|, \tag{3.14}$$

for some $M > 0$. The second inequality above follows from the Lipschitz continuity of ϕ in view of Lemma 2.2. Now (3.12) and (3.14) imply $\|v_k\| \leq M$. □

3.3.2. *The Superiorized Versions of the Basic Algorithms.* Taking these considerations into account, we investigate the following superiorized versions of the basic algorithms that interleave either $\mathcal{A}_{\mathrm{LW}}$, $\mathcal{A}_{\mathrm{LW}+}$ or $\mathcal{A}_{\mathrm{CG}}$ with S_{∇}, S_{prox} or $S_{\mathrm{prox}+}$, as introduced above.

We start with the Landweber and projected Landweber algorithms and present in Algorithm 10 the superiorized version of the Landweber algorithm, called `GradSupLW`, that combines $\mathcal{A}_{\mathrm{LW}}$ from Algorithm 4 with the target function procedure S_{∇} introduced in Algorithm 9.

Algorithm 10: `GradSupLW`

initialization: Set $k = 0$ and pick an arbitrary initial point $y_0 \in \mathbb{R}^n$ and parameters $\gamma_0 > 0$, $a \in (0,1)$ and $\varepsilon > 0$. Choose $\gamma \in (0, 2/\|A\|^2)$.
while $g(y_k) > \varepsilon$ **do**
  Given a current iterate y_k, calculate the superiorized sequence
  by $(y_{k+\frac{1}{2}}, \ell_{k+1}) = S_{\nabla}(y_k, \ell_k, a, \gamma_0, \kappa)$ using Algorithm 9.
  Apply the basic algorithm 4 and update $y_{k+1} = \mathcal{A}_{\mathrm{LW}}(y_{k+\frac{1}{2}}, \gamma)$
  and $\beta_{k+1} = \gamma_0 a^{k+1}$.
  Increment $k \leftarrow k + 1$.

Furthermore, we present in Algorithm 11 a second superiorized version of the Landweber algorithm, called `ProxCSupLW`, by combining the basic algorithmic operator $\mathcal{A}_{\mathrm{LW}}$ from Algorithm 4 with the target function reduction procedure $S_{\mathrm{prox}+}$ (3.8). In view of the perturbation resilience

Algorithm 11: `ProxCSupLW`

initialization: Set $k = 0$ and pick an arbitrary initial point $y_0 \in \mathbb{R}^n$ and parameters $\gamma_0 > 0$, $a \in (0,1)$ and $\varepsilon > 0$. Set $\beta_0 = \gamma_0$ and $\gamma \in (0, 2/\|A\|^2)$.
while $g(y_k) > \varepsilon$ **do**
  Given a current iterate y_k, calculate the superiorized sequence
  by $y_{k+\frac{1}{2}} = S_{\mathrm{prox}+}(y_k, \beta_k)$, see (3.8).
  Apply the basic algorithm 4 and update $y_{k+1} = \mathcal{A}_{\mathrm{LW}}(y_{k+\frac{1}{2}}, \gamma)$
  and $\beta_{k+1} = \gamma_0 a^{k+1}$.
  Increment $k \leftarrow k + 1$.

of the Landweber Algorithm 3, the iterates of `GradSupLW` and `ProxCSupLW` are guaranteed to converge to least-squares solutions. On the other hand, the analogue combination of $\mathcal{A}_{\mathrm{LW}+}$ with the target function reduction procedure S_{prox} produces a sequence that converges to a nonnegative least-squares solutions. Despite their different theoretical behavior, these last two superiorized versions of the Landweber algorithm generate sequences that show a nearly identical numerically behavior, as demonstrated in Section 5.

Proposition 3.6 below shows the relation of algorithm `ProxCSupLW` with an FBS iteration, recall (1.2). We point out that the parameter choice assumed below no longer qualifies `ProxCSupLW` as a *superiorized* Landweber algorithm, since the sequence (β_k) then fails to be summable. However, the modified algorithm converges and is guaranteed to return an optimal solution.

Proposition 3.6. *Let $\gamma \in (0, 2/L_{g_u})$ and $\lambda > 0$. The algorithm obtained by setting $\varepsilon = 0$, $\gamma_0 = \lambda\gamma$ and $a = 1$ in the iteration procedure described in Algorithm 11 generates a sequence $(y_{k+\frac{1}{2}})$ that converges to*

$$\operatorname*{arg\,min}_{x\in\mathbb{R}^n}\Big\{\lambda R_\tau(x) + \frac{1}{2}\|Ax-b\|^2 + \delta_{\mathbb{R}^n_+}(x)\Big\}. \tag{3.15}$$

Proof. We show that the iteration described in Algorithm 11 and the FBS iteration, recall (1.2),

$$x_0 \in \mathbb{R}^n \quad x_{k+1} = P_\gamma \phi_c(x_k - \gamma\nabla g_u(x_k)), \tag{3.16}$$

with $\phi_c = \lambda R_\tau + \delta_{\mathbb{R}^n_+} \in \mathcal{F}_c$, $g_u = \frac{1}{2}\|A\cdot - b\|^2 \in \mathcal{F}_c^1(L)$, are equivalent for an appropriate choice of the parameter β_k. The iterates (x_k) generated by (3.16) are converging to the minimizer of $\phi_c + g_c$, the objective in (1.4b), see [26], provided that

$$0 < \gamma < \frac{2}{L_{g_u}}, \tag{3.17}$$

where L_{g_u} denotes the global Lipschitz constant of the gradient of the least-squares term g_u and that it can be estimated by

$$L_{g_u} \le \|A\|_2^2. \tag{3.18}$$

Furthermore, Fermat's optimality condition for problem (3.15) reads

$$0 \in \lambda\nabla R_\tau(\bar{x}) + A^\top(A\bar{x} - b) + \partial\delta_{\mathbb{R}^n_+}(\bar{x}).$$

As done in [44, Thm. 3.8] in a different context, we introduce an auxiliary variable and write

$$0 \in \lambda\nabla R_\tau(\bar{x}) + \frac{1}{\gamma}(\bar{x} - x) + \partial\delta_{\mathbb{R}^n_+}(\bar{x}), \tag{3.19a}$$

$$x = \bar{x} - \gamma A^\top(A\bar{x} - b). \tag{3.19b}$$

Using the fact that (3.19a) is equivalent to

$$\bar{x} = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(x) = \operatorname*{arg\,min}_z \Big\{R_\tau(z) + \delta_{\mathbb{R}^n_+}(z) + \frac{1}{2\lambda\gamma}\|z - x\|^2\Big\},$$

we recast (3.19) as the iterative process

$$\bar{x}_k = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(x_k), \tag{3.20a}$$

$$x_{k+1} = \bar{x}_k - \gamma A^\top(A\bar{x}_k - b). \tag{3.20b}$$

Setting $y_{k+\frac{1}{2}} := \bar{x}_k$, $y_k = x_y$ and $\beta_k := \lambda\gamma$, the iterative process in (3.20) takes the form

$$y_{k+\frac{1}{2}} = S_{\text{prox}+}(y_k, \beta_k), \tag{3.21}$$
$$\beta_{k+1} = \beta_k a, \tag{3.22}$$
$$y_{k+1} = \mathcal{A}_{\text{LW}}(y_{k+\frac{1}{2}}), \tag{3.23}$$

and coincides with the iteration in Algorithm 11 for $a = 1$. On the other hand, the iterative process in (3.20) can be summarized by

$$\bar{x}_k = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(\bar{x}_{k-1} - \gamma A^\top(A\bar{x}_{k-1} - b)), \quad k \geq 1,$$

which is the FBS iteration (3.16). Hence, $(\bar{x}_k)$ and thus $(y_{k+\frac{1}{2}})$ converge to a minimizer of (3.15). □

Combinations between the Landweber and projected Landweber algorithms and the suggested choices of target function reduction procedures lead to different SM algorithms, that are summarized in Table 1.

Algorithm 12 presents the superiorized version of the CG algorithm, called `GradSupCG`, taken from [59, Algorithm 7, Algorithm 8], but applied here to the regularized least-squares problem corresponding to g_u^μ in (2.18b).

Algorithm 12: `GradSupCG`

initialization: Set $k = 0$ and pick an arbitrary initial point $y_0 \in \mathbb{R}^n$ and parameters $\kappa \in \mathbb{N}$, $\gamma_0 > 0$, $a \in (0, 1)$ and $\varepsilon > 0$. Set $\ell_0 = 0$, $p_0 = A^\top(b - Ax_0) + \mu x_0$ and $h_0 = A^\top Ap_0 + \mu p_0$.
while $g_\mu(y_k) > \varepsilon$ **do**
  Given a current iterate y_k, set $N_k := \kappa$ and calculate the superiorized sequence
  by $(y_{k+\frac{1}{2}}, \ell_{k+1}) = S_\nabla(y_k, \ell_k, a, \gamma_0, \kappa)$ using Algorithm 9.
  Apply the basic algorithm and update $(y_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\text{CG}}(y_{k+\frac{1}{2}}, p_k, h_k)$.
  Increment $k \leftarrow k + 1$.

A second superiorized version of the CG algorithm, called `ProxCSupCG`, presented in Algorithm 13, employs the target function reduction procedure $S_{\text{prox}+}$ by proximal points.

Omitting the nonnegativity constraints by replacing $S_{\text{prox}+}$ with S_{prox} in line 4 of Algorithm 13 leads to a slightly different algorithm that we call `ProxSupCG`. Clearly the sequence (β_k) is summable for $a \in (0, 1)$. It is not straightforward to show that `ProxSupCG` converges to a minimizer of g_u^μ, by following the lines of the proof from [44]. However, it follows from Proposition 3.5 and [59, Theorem A.1] that `ProxSupCG` terminates at a point in $\Gamma_{u,\varepsilon}^\mu$ (2.20b). To show that Algorithm 13 also terminates with an ε-compatible point we need a counterpart of Proposition 3.5 for the *constrained* target function ϕ_c (2.21). We leave this open problem for future work.

Algorithm 13: `ProxCSupCG`

initialization: Set $k = 0$ and pick an arbitrary initial point $y_0 \in \mathbb{R}^n$ and parameters $\kappa \in \mathbb{N}$, $\gamma_0 > 0$, $a \in (0, 1)$ and $\varepsilon > 0$. Set $\beta_0 = \gamma_0$, $p_0 = A^\top(b - Ax_0) + \mu x_0$ and $h_0 = A^\top A p_0 + \mu p_0$.

while $g(y_k) > \varepsilon$ **do**

  Given a current iterate y_k, calculate the superiorized sequence

  by $y_{k+\frac{1}{2}} = S_{\text{prox}+}(y_k, \beta_k)$, see (3.8).

  Apply the basic algorithm and update $(y_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\text{CG}}(y_{k+\frac{1}{2}}, p_k, h_k)$

  and $\beta_{k+1} = \gamma_0 a^{k+1}$.

  Increment $k \leftarrow k + 1$.

4. Optimization

In Subsections 4.1 and 4.2 we work out the details of the convex optimization algorithms for solving problems (2.22a) and (2.23a), based on the splittings (2.22b) and (2.23b), in Sections 4.1 and 4.2. Plain, accelerated and inexact versions of the FB-iteration are considered – cf. (1.2) and Algorithm 2. The reverse splittings (2.22c) and (2.23c) are briefly considered in Subsection 4.3. Here, we confine ourselves to exact evaluations of the corresponding proximal mapping.

Throughout this section, we use the symbols ϕ and ϕ_0 as shorthand for the objective functions that define the proximal mapping (4.1), for the constrained and unconstrained cases, respectively. This should not be confused with the usage of ϕ for denoting a given target function in Algorithm 1.

4.1. Proximal Map, Inexact Evaluation.

In this subsection, we consider the first splitting (2.22b) and (2.23b) with and without the nonnegativity constraint $x \in K = \mathbb{R}^n_+$ and examine the proximal map

$$P_\alpha g_c(x) = \arg\min_y \Big\{ \frac{1}{2\alpha}\|y - x\|^2 + \frac{1}{2}\|Ay - b\|^2 + \delta_K(y) \Big\}, \tag{4.1}$$

whose evaluation is required both in the FBS iteration (1.2a) and in the accelerated FBS defined in Algorithm 2. Note that the unconstrained case $P_\alpha g_u$ is the special case $K = \mathbb{R}^n$.

4.1.1. *Optimality Condition, Duality Gap.*

We rewrite (4.1) as

$$P_\alpha g_c(x) = \overline{y} = \arg\min_y \phi(y), \tag{4.2a}$$

where

$$\phi(y) = \phi_0(y) + \delta_K(y), \qquad \phi_0(y) = \frac{1}{2}\langle y, B_\alpha y\rangle - \big\langle c_\alpha, y\big\rangle, \tag{4.2b}$$

$$B_\alpha = A^\top A + \frac{1}{\alpha} I_n, \qquad c_\alpha = \frac{1}{\alpha} x + A^\top b. \tag{4.2c}$$

Then $\phi_0 \in \mathcal{F}_c^1\Big(\|A\|^2 + \frac{1}{\alpha}\Big)$ and we have

$$\nabla\phi_0(y) = B_\alpha y - c_\alpha = \frac{1}{2}(y - x) + A^\top(Ay - b) \tag{4.3a}$$

$$= \frac{1}{2}(y - x) + \nabla g_u(y), \tag{4.3b}$$

with g_u given by (2.22b). Applying Fermat's optimality condition to the proximal point $\overline{y}$, which is the unique minimizer of ϕ, yields the variational inequality

$$-\nabla\phi_0(\overline{y}) = c_\alpha - B_\alpha\overline{y} \in N_K(\overline{y}) \tag{4.4a}$$

$$\Leftrightarrow \qquad \big\langle c_\alpha - B_\alpha\overline{y}, z - \overline{y}\big\rangle \le 0, \qquad \forall z \in K, \tag{4.4b}$$

with $N_K(\overline{y})$ denoting the normal cone of K at $\overline{y}$.

We consider any feasible point $z \in K$, $z \neq y$, and compute and error measure in terms of the duality gap induced by a dual feasible point associated with z.

Lemma 4.1. *Let $z \in K$ and define the dual feasible point*

$$p = p(z) = (c_\alpha - B_\alpha z)_-. \tag{4.5}$$

Then the duality gap induced by the inexactness $z \approx \overline{y} = P_\alpha g(x)$ is denoted by dgp *and given by*

$$\mathrm{dgp}(z) = \frac{1}{2}\|(c_\alpha - B_\alpha z)_+\|^2_{B_\alpha^{-1}} - \langle p, z\rangle \ge 0. \tag{4.6}$$

Proof. Denote by $p \in K^* = \mathbb{R}^n_-$ the multiplier vector corresponding to the constraint $z \in K$. Then the dual problem corresponding to the primal problem (4.2a) reads

$$\max_p \psi(p), \qquad \psi(p) = -\frac{1}{2}\langle c_\alpha - p, B_\alpha^{-1}(c_\alpha - p)\rangle - \delta_{K^*}(p). \tag{4.7}$$

The unique pair of optimal primal and dual points $(\overline{y}, \overline{p})$ are connected by

$$\overline{p} = p(\overline{y}) = c_\alpha - B_\alpha\overline{y}. \tag{4.8}$$

Choosing any point $z \in K$ and the corresponding dual feasible point

$$p = p(z) = (c_\alpha - B_\alpha z)_- = c_\alpha - B_\alpha z - (c_\alpha - B_\alpha z)_+ \tag{4.9}$$

yields the duality gap

$$\mathrm{dgp}(z) = \phi(z) - \psi\big(p(z)\big) \ge 0 \tag{4.10}$$

which, after rearranging, becomes (4.6). □

Clearly, choosing $z = \overline{y}$ yields a zero duality gap: comparing (4.8) and (4.9) shows that $(c_\alpha - B_\alpha\overline{y})_+ = 0$. In addition, we have $\overline{p} \perp \overline{y}$ by (4.4a). Both relations imply $\mathrm{dgp}(\overline{y}) = 0$ by (4.6).

4.1.2. *Inexactness Criteria.* We describe two different ways for assessing the inexactness of evaluations of the proximal mapping $P_\alpha g$. These measures enable to specify criteria for terminating *early* the corresponding inner iterative loops (cf. Algorithm 2, lines 4-6), without compromising convergence of the FBS iteration.

Summable error sequences: Let

$$\overline{z}_k \approx P_{\alpha_k} g\big(x_k - \alpha_k \nabla f(x_k)\big) \tag{4.11}$$

be points obtained by inexact evaluations of the proximal mapping $P_{\alpha_k} g$ (4.1), at each iteration $k \in \mathbb{N}$. Denote the corresponding error vectors by

$$e_k = x_{k+1} - \overline{z}_k. \tag{4.12}$$

Then the FBS algorithm (1.2) converges [27, Theorem 3.4] if

$$\sum_{k\in\mathbb{N}} \|e_k\| < +\infty. \tag{4.13}$$

Inexact subgradients: A point $z \in \mathbb{R}^n$ is said to *evaluate* $P_\alpha g(x)$ *with* ε*-precision*, denoted by

$$z \mathrel{\overset{\approx}{\in}}_\varepsilon P_\alpha g(x), \quad \varepsilon > 0 \tag{4.14a}$$

if

$$\frac{1}{\alpha}(x - z) \in \partial_{\frac{\varepsilon^2}{2\alpha}} g(z), \tag{4.14b}$$

where the right-hand side is defined by the ε-subdifferential at z [48, Section 23]

$$\partial_\varepsilon g(z) = \{p \in \mathbb{R}^n \mid g(y) \geq g(z) + \langle p, y - z\rangle - \varepsilon\}, \qquad \forall y \in \mathbb{R}^n \tag{4.15a}$$

$$= \{p \in \mathbb{R}^n \mid g(z) + g^*(p) - \langle z, p\rangle \leq \varepsilon\}. \tag{4.15b}$$

Now consider the sequence of inexact evaluations

$$x_{k+1} = \overline{z}_k \mathrel{\overset{\approx}{\in}}_{\varepsilon_k} P_{\alpha_k} g\big(y_k - \alpha_k \nabla f(y_k)\big) \tag{4.16}$$

corresponding to line 8 of Algorithm 2. Theorem 4.4 in [56] assures the $\mathcal{O}(1/k^2)$ convergence rate of Algorithm 2 provided the error parameter sequence $(\varepsilon_k)_{k\in\mathbb{N}}$ corresponding to (4.16) decays at rate

$$\varepsilon_k = \mathcal{O}\Big(\frac{1}{k^{3/2+\delta}}\Big), \quad \delta > 0. \tag{4.17}$$

The second criterion (4.14a) requires to recognize when z satisfies condition (4.14b). Taking into account the specific form $g_c = g_u + \delta_K$ of g as defined by (2.22b) and (2.23b), the following lemma parametrizes any subgradient $p \in \partial_\varepsilon g(z)$ by auxiliary points $z_p, w \in \mathbb{R}^n$ through conjugation, in order to obtain a more explicit form of relation (4.14a).

Lemma 4.2. *Let $g = g_u + \delta_K \in \mathcal{F}_c$ with $g_u \in \mathcal{F}_c^1(L_{g_0})$ and $z \in \mathbb{R}^n$. Then finding a vector $p \in \partial_\varepsilon g(z)$ is equivalent to finding a pair of vectors $z_p \in K$, $w \in N_K(z_p)$ such that p is given by*

$$p = \nabla g_u(z_p) + w \qquad \textit{with} \qquad z_p \in K, \quad w \in N_K(z_p) \tag{4.18}$$

satisfies

$$g_u(z) - g_u(z_p) - \langle \nabla g_u(z_p), z - z_p\rangle - \langle w, z\rangle \leq \varepsilon. \tag{4.19}$$

In the unconstrained case $K = \mathbb{R}^n$, we have $w = 0$ and no constraint imposed on $z_p \in \mathbb{R}^n$.

Proof. Denoting by g^* the conjugate function of g, we can write

$$g^*(p) = \sup_z\{\langle p, z\rangle - g_u(z) - \delta_K(z)\} = -\inf_z\{g_u(z) - \langle p, z\rangle + \delta_K(z)\}, \tag{4.20}$$

and Fermat's optimality condition for the latter minimization problem reads $p \in \nabla g_u(z_p)+N_K(z_p)$, which is equivalent to (4.18). Substituting p and taking into account that $w \perp z_p$ due to $z_p \in K$ and $w \in N_K(z_p)$ gives $g^*(p) = \langle \nabla g_u(z_p), z_p\rangle - g_u(z_p)$ and substitution into (4.15b) yields

$$g(z) + g^*(p) - \langle z, p\rangle = g_u(z) + \langle \nabla g_u(z_p), z_p\rangle - g_u(z_p) - \langle z, \nabla g_u(z_p) + w\rangle \le \varepsilon, \tag{4.21}$$

which is relation (4.19). □

In the following two subsections, we examine criterion (4.14), for both the unconstrained and the constrained case using Lemma 4.2, and compare that to using instead the error vectors (4.12).

4.1.3. *Recognizing $z \approx_\varepsilon P_\alpha g_u(x)$: The Unconstrained Case.* Let $K = \mathbb{R}^n$. Applying Lemma 4.2 shows that condition (4.14b) for recognizing $z \approx_\varepsilon P_\alpha g(x)$ can be expressed as finding another point z_p such that

$$\frac{1}{\alpha}(x - z) = \nabla g_u(z_p) \qquad \text{and} \tag{4.22a}$$

$$\frac{\varepsilon^2}{2\alpha} \ge g_u(z) - g_u(z_p) - \langle \nabla g_u(z_p), z - z_p\rangle. \tag{4.22b}$$

Note that replacing the pair (z, z_p) by a *single* point z that satisfies *both* conditions, implies that $z = \overline{y} = P_\alpha g(x)$ is the *exact* proximal point: equality $z = z_p$ makes the inequality in (4.22b) hold trivially, whereas condition (4.22a) then reads $\nabla\phi_0(z) = 0$ due to (4.3), which implies $z = P_\alpha g(x)$ by (4.4a) since $K = \mathbb{R}^n$.

If $z \ne z_p$, then (4.22a) says that neither point is equal to $P_\alpha g(x)$, whereas the right-hand side of condition (4.22b) is always nonnegative and measures the difference between z and z_p by a Bregman-like distance induced by g_u (it is not a true Bregman distance [3, Definition 3.1] since g_u merely is convex). The decomposition (4.22a) of the optimality condition determining $P_\alpha g(x)$ in terms of (z, z_p) enables to terminate any iterative algorithm that converges to $P_\alpha g(x)$, once the level of inexactness (4.22b) is reached.

We conclude this subsection by comparing (4.22) with (4.12). Let $\overline{y} = P_\alpha g(x)$ denote the exact proximal point as in Subsection 4.1.1 and let $z \approx_\varepsilon P_\alpha g(x)$ satisfy (4.22). Then the error vector (4.12) reads

$$e = \overline{y} - z. \tag{4.23}$$

Since ϕ_0 given by (4.2b) is $\frac{1}{\alpha}$-strongly convex [49, Definition 12.58] and $\nabla\phi_0(\overline{y}) = 0$, we have the inequality

$$\|e\| = \|z - \overline{y}\| \le 2\alpha\|\nabla\phi_0(z)\| \overset{(4.3)}{=} 2\|z - x - \alpha\nabla g_u(z)\|. \tag{4.24}$$

Hence, this error vector evaluates violations of Equation (4.22a) due to $z \ne \overline{y}$, for a *single* point z on both sides.

4.1.4. *Recognizing $z \approx_\varepsilon P_\alpha g(x)$: The Constrained Case.* Let $K = \mathbb{R}^n_+$. In view of (4.14b), we identify $p = \frac{1}{\alpha}(x - z)$ in (4.18). Lemma 4.2 then shows that $z \approx_\varepsilon P_\alpha g$ holds if there is another point z_p such that

$$\frac{1}{\alpha}(x - z) - \nabla g_u(z_p) = w \qquad \text{with} \qquad z_p \perp w, \quad z, z_p \in K, \quad w \in N_K(z_p) \tag{4.25a}$$

$$\frac{\varepsilon^2}{2\alpha} \geq g_u(z) - g_u(z_p) - \langle \nabla g_u(z_p), z - z_p \rangle - \langle w, z \rangle. \tag{4.25b}$$

The discussion above (4.22) applies here analogously. In particular, if $z = z_p$, then (4.25a) reads

$$\frac{1}{\alpha}(x - z) - \nabla g_u(z) \overset{(4.3)}{=} c_\alpha - B_\alpha z \leq 0 = w \in N_K(z) \tag{4.26}$$

which is the optimality condition (4.4) that uniquely determines the proximal point $z = \overline{y} = P_\alpha g(x)$.

We conclude this subsection by comparing (4.25) with (4.12) and consider again the error vector (4.23), taking additionally into account the nonnegativity constraint $z \in K$. Since $\phi(y) - \frac{1}{2\alpha}\|y - x\|^2 = g_c(x)$ is convex, ϕ is $\frac{1}{\alpha}$-strongly convex. Exploiting weak duality (4.10), in particular $\phi(\overline{y}) \geq \psi(p)$, $\forall p \in K^*$, we estimate

$$\|e\|^2 = \|z - \overline{y}\|^2 \leq 2\alpha\big(\phi(z) - \phi(\overline{y})\big) \leq 2\alpha\big(\phi(z) - \psi\big(p(z)\big)\big) = 2\alpha\,\mathrm{dgp}(z), \quad z \in K, \tag{4.27}$$

with $p(z)$ given by (4.9). The explicit expression (4.6) shows how $\mathrm{dgp}(z)$ evaluates the non-optimality of the dual point $p(z)$. Comparing the left-hand sides of (4.25a) (where $z = z_p$) and (4.26) shows that the non-optimality results from $z \neq z_p$.

4.2. **Inner Iterative Loop: Accelerated Primal-Dual Iteration.** We work out in this subsection the inner iterative loop of Algorithm 2, lines 4–7. We apply the primal-dual optimization approach [25] to two different problem decompositions of the corresponding optimization problem (4.1), in order to generate a minimizing sequence (z_l) that is terminated using the inexactness criteria of Subsection 4.1. The first decomposition involves the inversion of the matrix B_α defined by (4.2c). The second alternative decomposition avoids this inversion. There is a tradeoff regarding the overall computational efficiency: more powerful iterative steps converge faster but are more expensive computationally.

Since the *inner* iterative loop will be only considered in this subsection, we drop the fixed arbitrary *outer* loop iteration index k. The argument x of (4.1) either denotes the argument $x = x_k - \alpha_k \nabla f(x_k)$ of the FBS algorithm (1.2a) or the argument $x = y_k - \alpha_k \nabla f(y_k)$ of the accelerated FBS iteration in Algorithm 2. The vector c_α is then defined accordingly by (4.2c). In either case, the exact proximal point is denoted by $\overline{y} = P_\alpha g(x)$ and satisfies the relations of Subsection 4.1.1. As discussed below, the inner iterative loop generates a sequence (z_l) that is terminated once $z_l \approx \overline{y}$ is sufficiently close to the exact proximal point according to the inexactness criteria of Subsection 4.1.

The approach [25] applies to convex optimization problems that can be written as saddle-point problems of the form

$$\min_{x \in X} \max_{y \in Y} \big\{ \langle Mx, y \rangle + G(x) - F^*(y) \big\}, \tag{4.28}$$

where X, Y are finite-dimensional real vector spaces, $M : X \to Y$ is linear and $F, G \in \mathcal{F}_c$. The algorithm involves the proximal maps $P_\alpha F^*$ and $P_\alpha G$ as algorithmic operations. In Sections 4.2.1 and 4.2.2, we apply this algorithm to problem (4.1) in two different ways, based on two different decompositions of problem (4.1) conforming to (4.28). Subsequently, in Section 4.2.3 and 4.2.4, we examine the application of the inexactness criteria of Section 4.1.2 for terminating the primal-dual iteration.

4.2.1. *Basic Decomposition.* We consider problem (4.1) rewritten as (4.2). Using a dual vector p, we write it in the saddle-point form (4.28)

$$\min_{y\in\mathbb{R}^n} \max_{p\in\mathbb{R}^n} \Big\{ \langle y, p\rangle + \phi_0(y) - \delta_{K^*}(p) \Big\}. \tag{4.29}$$

Note that $\phi_0 \in \mathcal{F}_c^1(L_{\phi_0}, \mu)$ is strongly convex with constant $\mu = \frac{1}{\alpha}$. Applying Algorithm 2 of [25] yields Algorithm 14 listed below.

Algorithm 14: Proximal Map Decomposition, Primal-Dual Iteration

1 **initialization:** Choose $\tau_0, \sigma_0 > 0$ with $\tau_0\sigma_0 \le 1$, $z_0, p_0 \in \mathbb{R}^n$ arbitrary. Set $\overline{z}_0 = z_0$ and $l = 0$.
2 **repeat**
3 $\quad p_{l+1} = (p_l + \sigma_l \overline{z}_l)_-$
4 $\quad z_{l+1} = \big(I_n + \tau_l B_\alpha\big)^{-1}\big(z_l - \tau_l(p_{l+1} - c_\alpha)\big)$
5 $\quad \theta_l = \Big(1 + 2\frac{\tau_l}{\alpha}\Big)^{-1/2}, \quad \tau_{l+1} = \theta_l\tau_l, \quad \sigma_{l+1} = \frac{\sigma_l}{\theta_l}$
6 $\quad \overline{z}_{l+1} = z_{l+1} + \theta_l(z_{l+1} - z_l)$
7 **until** z_{l+1} *satisfies an inexactness criterion.*

Step 4 of Algorithm 14 involves the inversion of the $n \times n$ matrix

$$I_n + \tau_l B_\alpha = \Big(1 + \frac{\tau_l}{\alpha}\Big) I_n + \tau_l A^\top A. \tag{4.30}$$

Due to assumption (2.1), the computational cost can be reduced by using the Sherman-Morrison-Woodbury formula [37], quoted here with arbitrary invertible matrix B and matrices U, V with compatible dimensions,

$$(B + UV^\top)^{-1} = B^{-1} - B^{-1}U(I + V^\top B^{-1}U)^{-1}V^\top B^{-1}. \tag{4.31}$$

Applying this formula yields

$$(I_n + \tau_l B_\alpha)^{-1} = \frac{\alpha}{\alpha + \tau_l}\left(I_n - \frac{\alpha}{\alpha + \tau_l} A^\top \Big(\frac{1}{\tau_l} I_m + \frac{\alpha}{\alpha + \tau_l} AA^\top\Big)^{-1} A\right), \tag{4.32}$$

which only involves the inversion of an $m \times m$ matrix.

For very large problem sizes, however, evaluating this formula may be too costly as well. Therefore, in the next subsection, we consider an alternative problem decomposition that avoids matrix inversion.

4.2.2. *Avoiding Matrix Inversion.* In order to avoid matrix inversion in step (4) of Algorithm 14, we rewrite the objective function ϕ of (4.2) in the form

$$\phi(y) = \frac{1}{2}\|Ay\|^2 + \frac{1}{2\alpha}\|y\|^2 - \langle c_\alpha, y\rangle + \delta_K(y), \tag{4.33}$$

where K either is $\mathbb{R}^n$ or $\mathbb{R}^n_+$. Using a dual vector q, we dualize the term comprising A, to obtain the saddle-point problem

$$\min_y \max_q \left\{ \langle Ay, q\rangle + \frac{1}{2\alpha}\|y\|^2 - \langle c_\alpha, y\rangle + \delta_K(y) - \frac{1}{2}\|q\|^2 \right\}. \tag{4.34}$$

Applying Algorithm 2 of [25] yields Algorithm 15 listed below.

Algorithm 15: Proximal Map Decomposition, Primal-Dual Iteration without Matrix Inversion

initialization: Choose $\tau_0, \sigma_0 > 0$ with $\tau_0\sigma_0 \le \frac{1}{\|A\|_2^2}$, $z_0 \in \mathbb{R}^n, q_0 \in \mathbb{R}^m$ arbitrary. Set $\overline{z}_0 = z_0$, $l = 0$.

repeat

$q_{l+1} = \frac{1}{1+\sigma_l}(q_l + \sigma_l A\overline{z}_l)$

$z_{l+1} = \Pi_K\Big(\frac{\alpha}{\alpha+\tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big)\Big)$

$\theta_l = \Big(1 + 2\frac{\tau_l}{\alpha}\Big)^{-1/2}, \quad \tau_{l+1} = \theta_l\tau_l, \quad \sigma_{l+1} = \frac{\sigma_l}{\theta_l}$

$\overline{z}_{l+1} = z_{l+1} + \theta_l(z_{l+1} - z_l)$

until z_{l+1} *satisfies an inexactness criterion.*

Note that this algorithm only requires a single matrix-vector multiplication with respect to A and $A^\top$ in each iteration. While this is much cheaper than the matrix inversion of Algorithm 14, considerably more iterations are required to satisfy the inexactness criterion.

4.2.3. *Termination: The Unconstrained Case.* Let $K = \mathbb{R}^n$. Then $\overline{y} = P_\alpha g(x)$ uniquely minimizes $\phi = \phi_0$, and is determined by the optimality condition

$$\nabla\phi_0(\overline{y}) = B_\alpha\overline{y} - c_\alpha = 0 \quad \Leftrightarrow \quad \overline{y} = B_\alpha^{-1}c_\alpha = P_\alpha g_u(x). \tag{4.35}$$

Now suppose that applying a direct method for computing $B_\alpha^{-1}c_\alpha = (I_n + \alpha A^\top A)^{-1}c_\alpha$ is numerically intractable, due to the problem size, even when a formula analogous to (4.32) is used to reduce the problem size for matrix inversion from n to $m < n$. Then one chooses an iterative numerical method that generates a sequence (z_l) converging to $\overline{y}$ until the inexactness condition $z_l \approxeq_\varepsilon P_\alpha g(x)$ holds. As discussed in Subsection 4.1.3, in order to make this criterion explicit, a pair of points (z, z_p) has to be chosen such that the two relations (4.22) are satisfied.

Let z_{l+1} be a point generated by the iterative algorithm that approximately satisfies the optimality condition

$$\nabla\phi_0(z_{l+1}) = \nabla g_u(z_{l+1}) - \frac{1}{\alpha}(x - z_{l+1}) \approx 0. \tag{4.36}$$

Setting

$$z_p = z_{l+1}, \qquad z = x - \alpha\nabla g_u(z_p) \tag{4.37}$$

makes condition (4.22a) hold and we can evaluate the right-hand side of (4.22b) in order to check if $z \approxeq_\varepsilon P_\alpha g(x)$. It remains to be checked if the choice (4.37) is compatible with the *specific* Algorithm 15. Step (4) reads

$$0 = \frac{\alpha}{\alpha+\tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big) - z_{l+1} \tag{4.38a}$$

$$= \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l) - \alpha A^\top(q_{l+1} - Az_{l+1})\Big) - A^\top(Az_{l+1} - b), \tag{4.38b}$$

and the choice

$$z_p = z_{l+1}, \qquad z := z_p - \frac{\alpha}{\tau_l}(z_p - z_l) - \alpha A^\top(q_{l+1} - Az_p), \tag{4.39}$$

makes Equation (4.38b) equal to the second equation of (4.37).

A minor disadvantage of (4.37) is the additional evaluation of the gradient $\nabla g_u(z_p)$. In the present unconstrained case, this can be avoided as follows. Since the sequence of dual vectors (q_l) converges to $q_l \to A\overline{y}$ as line 3 of Algorithm 15 reveals, the expression $A^\top(q_{l+1} - b) \approx \nabla g_u(\overline{y})$ approximates the gradient. Hence, rearranging Equation (4.38) into the form

$$0 = \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l)\Big) - A^\top(q_{l+1} - b), \tag{4.40}$$

suggests the choice

$$Az_p = q_{l+1}, \qquad z = z_{l+1} + \frac{\alpha}{\tau_l}(z_{l+1} - z_l). \tag{4.41}$$

Then $A^\top(q_{l+1} - b) = \nabla g_u(z_p)$ and condition (4.22a) is satisfied. Condition (4.22b) can be efficiently checked as well, even though (4.41) does not determine z_p explicitly:

$$\frac{\varepsilon^2}{2\alpha} \geq \frac{1}{2}\|Az - b\|^2 - \frac{1}{2}\|Az_p - b\|^2 - \langle Az_p - b, Az - Az_p\rangle \tag{4.42a}$$

$$= \frac{1}{2}\big\|(Az - b) - (Az_p - b)\big\|^2 = \frac{1}{2}\|Az - q_{l+1}\|^2. \tag{4.42b}$$

Thus, checking these conditions during the primal-dual iteration only requires negligible additional computation. Definition (4.41) of z shows that the single additional vector-matrix multiplication Az is implicitly done by evaluating the primal-dual steps (3) and (6) of Algorithm 15.

We conclude by comparing the alternative error measure (4.24), which for the choice (4.37) translates to

$$\|e\| = \|z_p - z\|, \tag{4.43}$$

with condition (4.13) for summing up these inner-loop errors at the outer iteration level. Conversely, inserting the alternative choice (4.41) into (4.42b) yields the error

$$\varepsilon \geq \alpha^{1/2}\|A(z - z_p)\| \tag{4.44}$$

in terms of vectors of smaller dimension m (cf. assumption (2.1)) and with a factor $\alpha^{1/2} \ll 1$ that typically is much smaller than 1 due to the upper bound $\alpha < \frac{2}{L_f}$ of feasible proximal parameters. The sum of the errors (4.44) for all outer steps $k \in \mathbb{N}$ of the iteration not only have to be summable but should decay with rate (4.17) to achieve fast convergence.

4.2.4. *Termination: The nonnegativity Constrained Case.* Let $K = \mathbb{R}^n_+$. We examine criteria based on the conditions (4.25) for terminating the inner-loop iteration that generates a sequence (z_l) converging to the proximal point $\overline{y} = P_\alpha g(x)$. Algorithm 14 does not comprise a projection onto K and hence does not conform to condition (4.25a). Therefore, we only consider early termination of Algorithm 15.

Based on the projection theorem, [4, Thm. 3.14] we rewrite step (4) as

$$N_K(z_{l+1}) \ni \frac{\alpha}{\alpha+\tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big) - z_{l+1} \tag{4.45a}$$

$$\overset{(4.2c)}{=} \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l) - \alpha A^\top(q_{l+1} - Az_{l+1})\Big) - A^\top(Az_{l+1} - b). \tag{4.45b}$$

Setting

$$z_p = z_{l+1}, \qquad z = z_{l+1} + \frac{\alpha}{\tau_l}(z_{l+1} - z_l) + \alpha A^\top(q_{l+1} - Az_{l+1}), \tag{4.46a}$$

$$w = A^\top(Az_{l+1} - b) - \frac{1}{\alpha}(x - z) \tag{4.46b}$$

satisfies condition (4.25) *except* for the condition $z \in K$. Clearly, z given by (4.46) becomes nonnegative as $l \to \infty$, and once this happens for sufficiently large l we can check $z \approxeq_\varepsilon P_\alpha g(x)$ by evaluating condition (4.25b). Similarly to (4.42), we get

$$\frac{\varepsilon^2}{2\alpha} \geq \frac{1}{2}\|Az - b\|^2 - \frac{1}{2}\|Az_p - b\|^2 - \langle Az_p - b, Az - Az_p\rangle + \langle w, z\rangle \tag{4.47a}$$

$$= \frac{1}{2}\|A(z - z_p)\|^2 + \langle w, z\rangle, \tag{4.47b}$$

which, in view of (4.46), evaluates inexactness of z through a distance and a complementarity term with respect to z_p.

A weakness of criterion (4.47) is that it cannot be applied if $z \notin K$, with z given by (4.46a). As an alternative, we evaluate the error measure (4.27) in order to terminate the inner iterative loop. The duality gap $\mathrm{dgp}(z_l)$ (4.6) requires to compute

$$c_\alpha - B_\alpha z_l = c_\alpha - A^\top A z_l - \frac{1}{\alpha} z_l \tag{4.48}$$

where $A^\top A z_l$ has to be computed anyway in order to execute Algorithm 15. To avoid the expensive evaluation of the first term on the right-hand side of (4.6), we substitute (4.5) and estimate

$$\mathrm{dgp}(z_l) \leq \frac{1}{2}\|B_\alpha^{-1}\|_2 \|(c_\alpha - B_\alpha z_l)_+\|^2 - \langle (c_\alpha - B_\alpha z_l)_-, z_l\rangle \tag{4.49}$$

and, hence, obtain with $\|B_\alpha^{-1}\|_2 \leq \alpha$ and (4.27)

$$\|e_l\| \leq \alpha\Big(\|(c_\alpha - B_\alpha z_l)_+\|^2 - \frac{2}{\alpha}\langle (c_\alpha - B_\alpha z_l)_-, z_l\rangle\Big)^{1/2}. \tag{4.50}$$

After fixing errors $\|e_k\|$ at each outer iteration k so as to satisfy (4.13), each correspoding inner loop can be terminated once the right-hand side of (4.50) drops below $\|e_k\|$.

4.3. **Reverse Problem Splitting, Proximal Map.** As motivated and discussed in Subsection 2.4, we also evaluate the reverse splitting (2.23c), including the corresponding unconstrained splitting (2.22c) as a special case.

The FBS iteration (1.2) now reads

$$x_{k+1} = P_{\alpha_k} g\big(x_k - \alpha_k \nabla f_u(x_k)\big) = P_{\alpha_k} g\big(x_k - \alpha_k A^\top (Ax_k - b)\big), \tag{4.51}$$

where either $g = g_c$ or $g = g_u$. We confine ourselves to using *exact* evaluations of the proximal map $P_{\alpha_k} g$ using the highly accurate box-constrained L-BFGS method from [7]. These iterations replace lines 4–7 of Algorithm 2, that otherwise is applied without change.

A major consequence of using this reversed splitting scheme is that now the target function R_τ defines the proximal map, whereas the least-squares problem defines the task whose algorithm will be perturbed by the target function reductions. As a consequence, more iterations are spent to lower the target function as inner loop iterations of the proximal mapping evaluation, compared to the number of gradient iterations devoted to solving the unconstrained least-squares problem. Our empirical findings are reported and discussed, based on our experimental numerical work, in Section 5.

5. Numerical Results

Subsection 5.1 details the experimental set-up based that we use to compare superiorization with accelerated inexact optimization in Subsections 5.2 and 5.3.

5.1. **Data, Implementation.** The linear system

$$Ax = b, \qquad A \in \mathbb{R}^{m \times n} \tag{5.1}$$

considered for our experiments uses linear tomographic measurements b of the Shepp-Logan MATLAB 128×128 phantom shown in Figure 5.1 in a highly underdetermined scenario, i.e., $m/n = 0.15625$. The vector representing this test phantom is denoted by x_* in the sequel. The subsampling ratio m/n is chosen according to [43] so that recovery error $\|x - x_*\|$ with x obtained as minimizer of (1.4) is small as illustrated in Figure 5.2. We use the MATLAB routine `paralleltomo.m` from the AIR Tools II package [38] that implements such a tomographic matrix for a given vector of projection angles. In particular, we used 20 angles (projections) uniformly spaced between 1 and 180 degrees, and 120 parallel rays per angle of projection, resulting in a full rank 2560×16384 matrix A. We consider both exact and noisy measurements. In the latter case, i.i.d. normally distributed noise was added with variance σ^2 determined by

$$b_j \quad \leftarrow \quad b_j + \xi_j, \qquad \xi_j \sim \mathcal{N}(0, \sigma^2), \qquad j \in [m], \qquad \sigma = \frac{0.02}{m} \sum_{j \in [m]} b_j. \tag{5.2}$$

Figure 5.1 depicts in addition to the original image x_* and its histogram the least-squares solution of minimal norm x_{LS} for noisy measurements along with its histogram.

Figure 5.2 depicts minimizers x of both the unconstrained and the nonnegativity constrained problem (1.4) for exact and for noisy data, respectively, and illustrates that the models in (1.4) are appropriate and lead to a relatively small recovery error $\|x - x_*\|$. The histograms show, in particular, that adding nonnegativity constraints increases accuracy.

Smoothing Parameter. The smoothing parameter of the target function R_τ (2.5b) and of the regularization term in (1.4) was set to $\tau = 0.01$ in all experiments.

Regularization Parameter. The regularization parameter λ in (1.4) was set to

$$\lambda := \begin{cases} 0.01, & \text{for exact data} \\ 1.6529, & \text{for noisy data} \end{cases} \quad \Longrightarrow \quad \begin{cases} R_\tau(x) \approx R_\tau(x_*) \\ \|Ax - b\|^2 \approx \|Ax_* - b\|^2 \end{cases} \tag{5.3}$$

which implies,

- in the case of exact data b, that minimizers x have almost the same total variation as the "unknown" true solution x_*,
- in the case of inexact data b, that minimizers x reach the noise level of x_* induced by (5.2).

Least-Squares Regularization Parameter. The parameter μ in the regularized least-squares objective (2.18b) was set to $\mu = 1/y_*$ according to Lemma 3.1. The dual variable y_*, see the proof of Lemma 3.1, is computed via CVX. This parameter choice guarantees that the CG iteration converges to a solution of (2.18a).

Starting vectors. The zero vector $0 \in \mathbb{R}^n$ was chosen as starting vectors $x_0, y_0 \in \mathbb{R}^n$ in all variants of Algorithm 1 and Algorithm 2.

Termination Criteria. For illustration purposes all algorithms where run for a maximum number of (outer) iterations equal to 2000 *beyond* the used stopping rules of Algorithm 1 and Algorithm 2. Regarding termination of the superiorized versions of the basic algorithms, recall the proximity sets in (2.20), we used

$$g_u(x) \leq \varepsilon \tag{5.4}$$

to terminate the iterative process that concerns the unconstrained least-squares problem, while using

$$g_u(x) \leq \varepsilon \quad \text{and} \quad \min_{i \in [n]} x_i > -10^{-8} \tag{5.5}$$

in the case of the constrained least-squares problem. The parameter ε was set to the noise level $\varepsilon \approx 0.047m$ in the noisy case and $\varepsilon = 0.001$ in the noise free case.

Regarding the optimization problem (1.4), in the the unconstrained case $x \in \mathbb{R}^n$, the criterion

$$\|\nabla h_u(x)\|_\infty \leq 0.001 \tag{5.6}$$

was used to terminate the iterative optimization and to accept x as approximate minimizer of h_u (1.4). In the nonnegativity constrained case $x \in \mathbb{R}^n_+$, the complementarity condition

$$\left| \min(x, \nabla h_u(x)) \right| \leq 0.001 \tag{5.7}$$

was used to accept x as approximate minimizer of h_c (1.4).

CVX: Matlab Software for Disciplined Convex Programming, http://cvxr.com/cvx.

The optimality condition for the nonnegativity constrained objective h_c (1.4) can be compactly written as $\min(x, \nabla h_u(x)) = 0 \Leftrightarrow x \geq 0, \nabla h_u(x) \geq 0, x^\top \nabla h_u(x) = 0$, where h_u denotes the function without constraints.

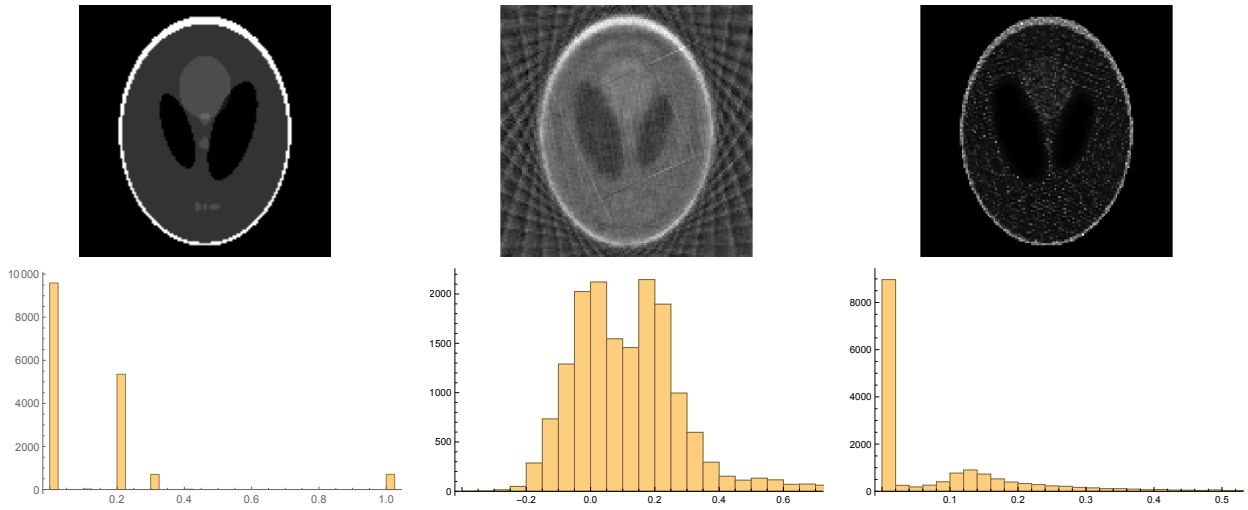

FIGURE 5.1. TOP LEFT: The original image x_* of size $n = 128^2$ to be recovered from $m = 0.15625n$ linear measurements. BOTTOM LEFT: The histogram of x_* depicts the number of pixels for each intensity value. All components of $x_{*,i}$, $i \in [n]$ are confined to the interval $[0, 1]$. TOP MIDDLE AND RIGHT: The least-squares solution of minimal Euclidean norm x_{LS} (middle) and a nonnegative least-squares solution (right) both include severe artefacts. BOTTOM MIDDLE AND RIGHT: The histogram of x_{LS} and the histogram of the nonnegative least-squares solution deviate significantly from that of x_*. This holds for both exact and noisy measurements. The reconstruction problem requires further regularization.

5.2. **Superiorization vs. Optimization: Reconstruction Error Values.** In this subsection we compare the algorithmic performance in terms of reconstruction quality based on three reconstruction errors:

(1) The *scaled squared residual* $\|Ay_k - b\|^2/(2m)$ generated by variants of Algorithm 1 and $\|Ax_k - b\|^2/(2m)$ generated by variants of Algorithm 2 that should approach 0 in the noiseless case and ideally approach the noise level ≈ 0.0473 in the noisy case;
(2) The *scaled target function* (regularization term) $R_\tau(y_k)/n$ generated by the superiorized versions of the basic algorithms and $R_\tau(x_k)/n$ generated by each optimization algorithm that should approach the smoothed TV value of the original image x_*;
(3) The *scaled error term* $\|y_k - x_*\|^2/n$ computed by Algorithms of type 1 and $\|x_k - x_*\|^2/n$ computed by Algorithms of type 2, that should be reduced as much as possible.

Each optimization algorithm is guaranteed to reduce the combination of (1) and (2) as in (1.4), and also (3), as our model (1.4) is appropriate, see Figure 5.2, while the superiorized versions of the basic algorithms are only guaranteed to reduce the error in (1).

Superiorization. We first address the unconstrained least-squares problem and consider the superiorized CG algorithm and the superiorized Landweber algorithm, by gradient- or proximal point

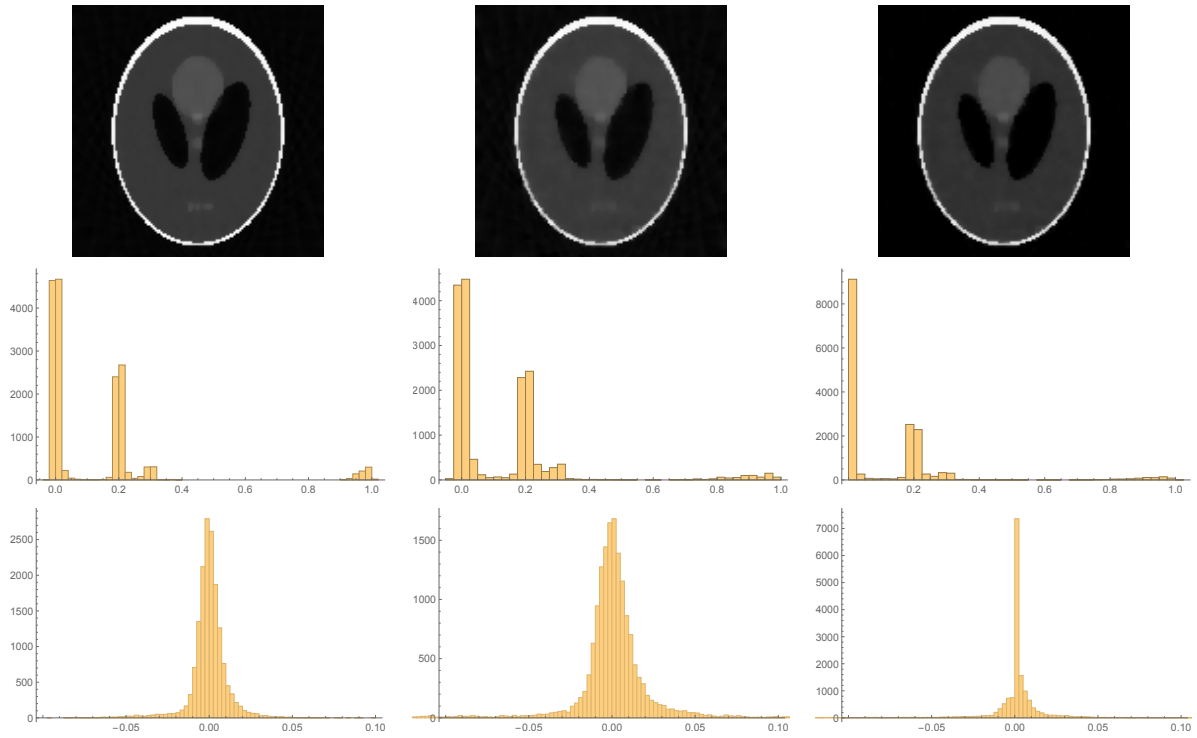

FIGURE 5.2. TOP ROW, FROM LEFT TO RIGHT: Minimizers x of the unconstrained objective function h_u (1.4) for exact data, for noisy data, and minimizer for the nonegatively constrained objective function h_c and noisy data. CENTER ROW: The histograms corresponding to the top row. In the unconstrained cases, a number of components x_i are outside the range $[0, 1]$ of $x_{*,i}$, $i \in [n]$. BOTTOM ROW: The difference histograms of $x - x_*$ corresponding to the top row. The left histogram (unconstrained case, exact data) peaks more sharply around 0 than the histogram in the center (unconstrained case, noisy data), while the right histogram has the sharpest peak (nonnegativity constrained case, noisy data).

based target function reduction procedures S_∇ and S_{prox} respectively. To handle the underdetermined system by CG we consider the regularized least-squares problem (3.1) and an appropriate choice of μ as previously described.

The nonnegative least-squares problem is addressed by superiorizing the projected Landweber algorithm. To additionally steer the iterates of a basic algorithm for the least-squares problem towards the nonnegative orthant we employ perturbations by constrained proximal points with $S_{\text{prox}+}$.

The algorithms, listed in Table 1, have parameters that need to be tuned. To select these parameters, we evaluated the quality of the reconstructed image y_k using the relative error $\|y_k - x_*\|^2/n$ and then choose, for each algorithm separately, the parameters that provided the best reconstructed image within the first 100 iterations. The superiorization strategies share three parameters (κ, a and γ_0), see, e.g., Algorithm 12 and Algorithm 13. These were chosen from the following sets: $\kappa \in \{5, 10, 20\}$, $a \in \{0.5, 1 - 10^{-2}, 1 - 10^{-4}, 1 - 10^{-6}\}$, $\gamma_0 \in \{0.01, 0.001, 0.0025, \beta_1\}$, with

$\beta_1 = 1.9\lambda/\|A\|^2$ and λ from (5.3). The combination of these parameters, that provide the best results are listed in Table 2, while the results are shown in Figure 5.3 and Figure 5.4. For comparison we also plot the results of the FBS algorithm (1.2) for the reversed splitting (2.23c), in view of its intimate connection with the proximal-point based superiorized version of the Landweber algorithm, Algorithm 11, discussed in Proposition 3.6. Note that the best found values for parameter a are very close to 1 as Proposition 3.6 suggests. Note that `ProxSupCG` has the best error values among all superiorized versions of the basic algorithms and that it approaches optimal values in the least amount of time when it is adapted to incorporate nonnegativity constraints, by `ProxCSupCG`.

Algorithm	Optimal parameters
`GradSupCG`	$(a, \gamma_0, \kappa) = (1 - 10^{-4}, 0.001, 20)$
`ProxSupCG`, `ProxSubLW`	$(\gamma_0, a) = (0.001, 1 - 10^{-6})$
`ProxCSupCG`, `ProxCSubLW`, `ProxSupProjLW`	$(\gamma_0, a) = \frac{1.9\lambda}{\|A\|^2}, 1 - 10^{-6})$
`GradSubLW`, `GradSupProjLW`	$(a, \gamma_0, \kappa) = (1 - 10^{-4}, 0.0025, 20)$

TABLE 2. Optimal parameters for the superiorized versions of the basic algorithms.

Optimization. Figure 5.5 shows the results of comparing the FBS algorithm (1.2) with the accelerated FBS iteration in Algorithm 2 for solving (1.4) without nonnegativity constraints. To obtain a fair comparison, the proximal maps were evaluated *exactly* using formula (4.32). Depending on the value of λ that affects the Lipschitz constant, i.e., a small value of λ for exact data and a larger value of λ for noisy data, more iterations are required in the latter case. And acceleration pays: significantly fewer iterations are then required.

Since the values of λ were chosen such that minimizers match the properties (5.3) of x_*, it seems fair to claim that – from the viewpoint of optimization – our implementation structurally resembles the problem decomposition of superiorization, i.e., least-squares minimization adjusted by gradients of the target function R_τ. Inspecting the plots of Figure 5.5, we notice that the final values are almost reached after merely ≈ 75 outer iterations.

Similarly, the accelerated FBS algorithm (1.2) for the reversed splitting (2.23c) reaches error term values close to final values within the first 100 iterations, at significantly lower cost as discussed next. Results for the reversed splitting are shown in Figure 5.6 and Figure 5.7 for the unconstrained and nonnegativity constrained case, respectively.

5.3. **Superiorization vs. Optimization: Computational Complexity.** We look at the computational costs of the superiorization and optimization approaches. **Superiorization.** To evaluate the cost of the superiorized versions of the basic algorithms, listed in Table 1, we call an outer iteration an "iteration of the basic algorithm" $\mathcal{A}(\cdot)$ and an inner iteration "an execution of the target function reduction procedure" $S(\cdot)$ by Algorithm 9 or by a proximal map via (3.7) or (3.8). Within an outer iteration we have for the CG iteration in Algorithm 8 four costly matrix-vector operations in line 2 and line 5, while for the Landweber Algorithm 4 only two costly matrix-vector operations appear in line 2. The cost of an inner iteration depends on the number of gradient and function evaluations of R_τ. In the case of computing nonascent directions by scaled nonnegative gradients, see, e.g., line

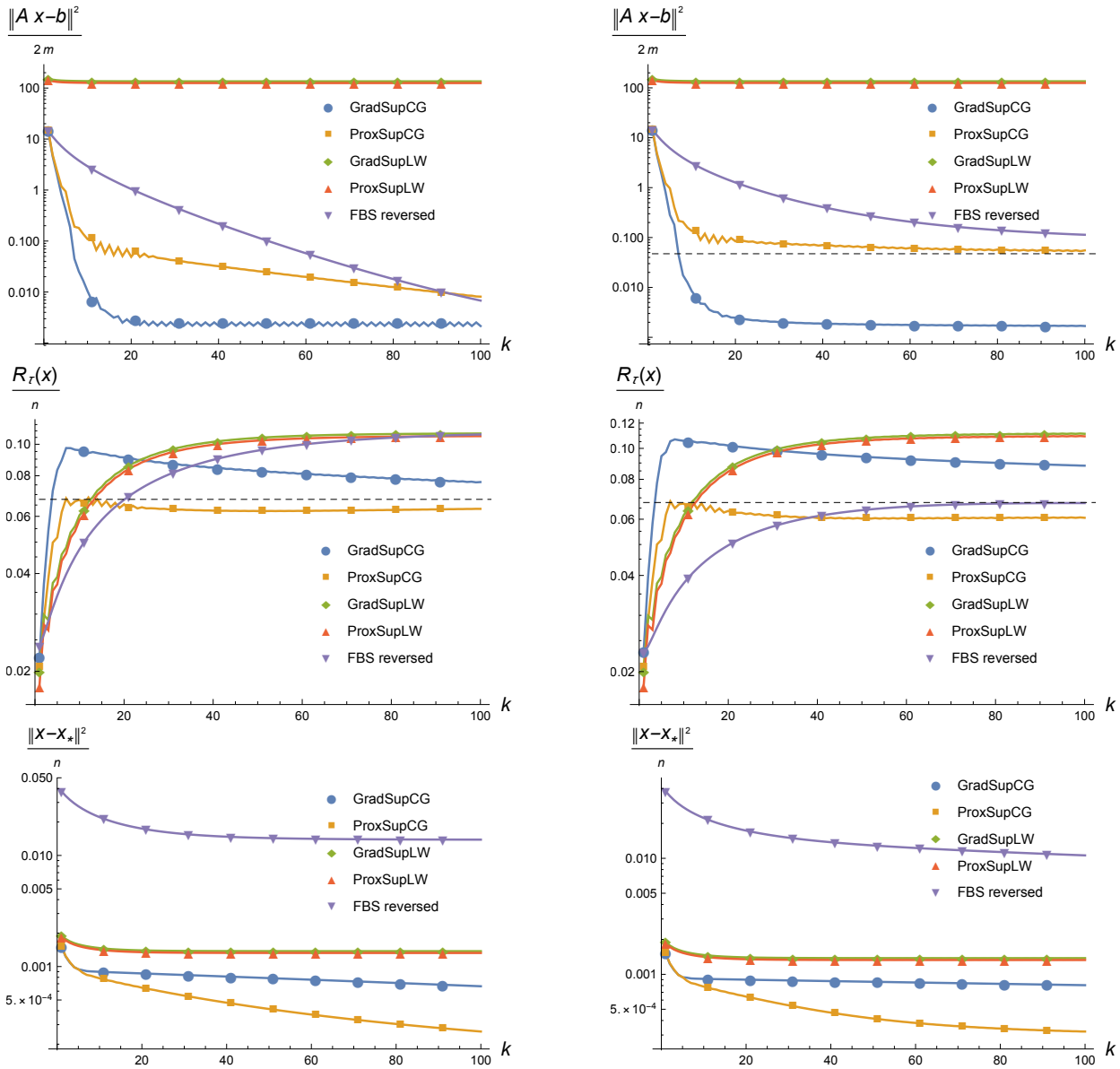

FIGURE 5.3. *Superiorization: Unconstrained Least-Squares.* Scaled values of the data term (top row) and the regularizer (middle row) of the objective (1.4) and the squared error norm (bottom row), as a function of the iteration index k of the superiorized versions of the CG algorithm using perturbations based on gradients, see Algorithm 9 and on proximal points, see (3.7), the negative gradients and proximal-point based superiorized versions of the Landweber algorithm and the FBS algorithm (1.2) for the reversed splitting (2.23c). The horizontal dashed lines indicate the corresponding values for x_*. The left and right columns correspond to exact and noisy data, respectively. Neither the basic algorithms nor their superiorized versions incorporate nonnegativity constraints. After appropriate parameter tuning the superiorized CG iteration approaches the solution of (1.4) significantly faster than the FBS algorithm. The superiorized version of the Landweber algorithm makes slow progress due to very small step-sizes in view of the large value of $\|A\|^2$.

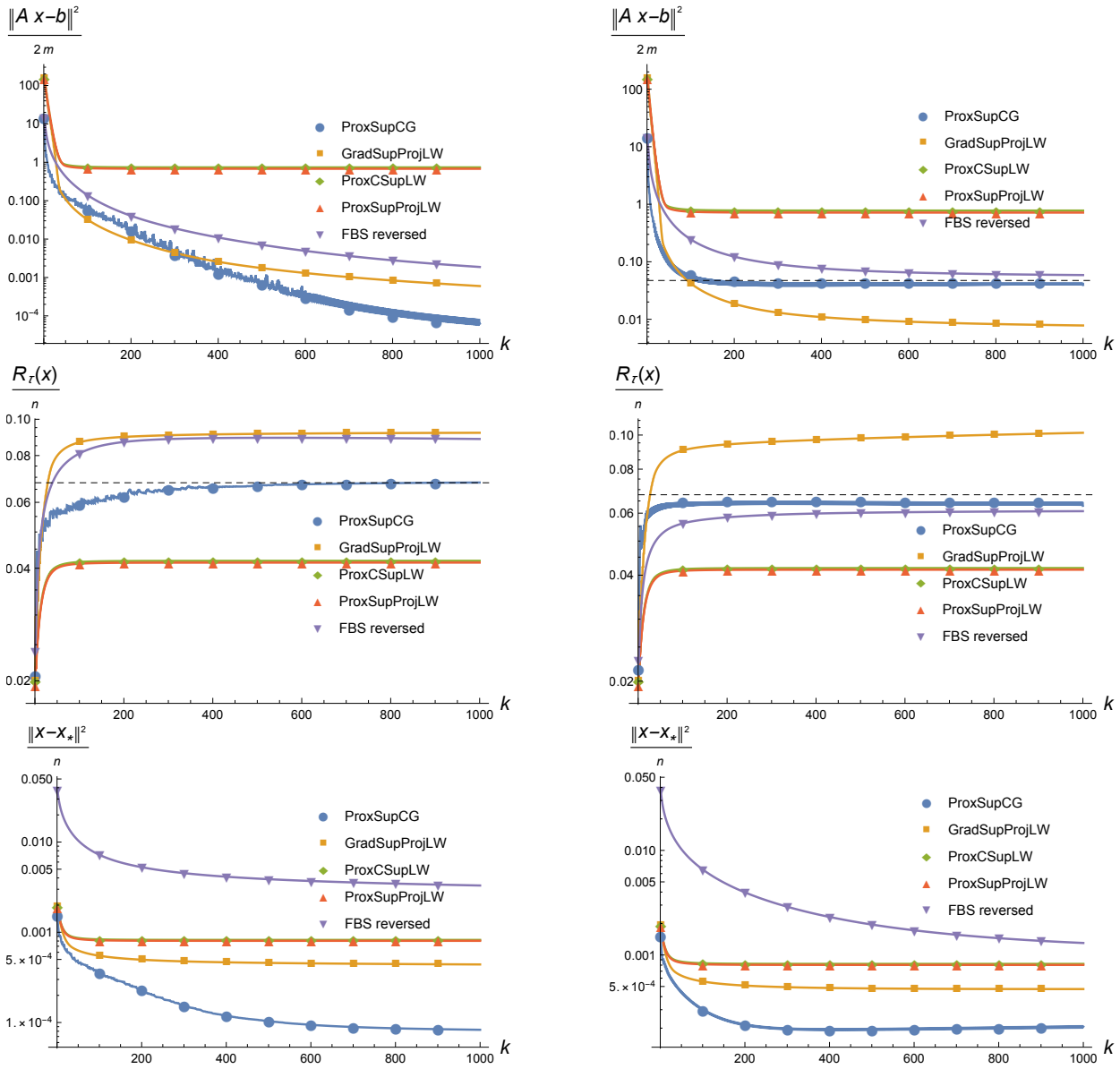

FIGURE 5.4. *Superiorization: Adding Nonnegative Constraints.* Scaled values of the data term (top row) and the regularizer (middle row) of the objective (1.4) and the squared error norm (bottom row), as a function of the iteration index k of the superiorized versions of the CG algorithm 13 and the Landweber algorithm 11 using proximal points, see (3.8), the negative gradient and proximal point based superiorized version of the projected Landweber algorithm and the FBS algorithm (1.2) for the reversed splitting (2.23c). The horizontal dashed lines indicate the corresponding values for x_*. The left and right columns correspond to exact and noisy data, respectively. Constraints are incorporated either in the basic algorithm (`GradSupProjLW`, `ProxSupProjLW`) or via the superiorization strategy (`ProxCSupCG`, `ProxCSubLW`) by constraining proximal points. Superiorized versions of the Landweber algorithm perform similarly making slow progress towards x_* due to the ill-conditioned matrix A, with a surprising better performance of the gradient based superiorized versions of the projected Landweber iteration `GradSupProjLW`. Again, the superiorized version of the CG algorithm approaches the solution of (1.4) significantly faster than the FBS algorithm for the reversed splitting (2.23c) after appropriate parameter tuning.

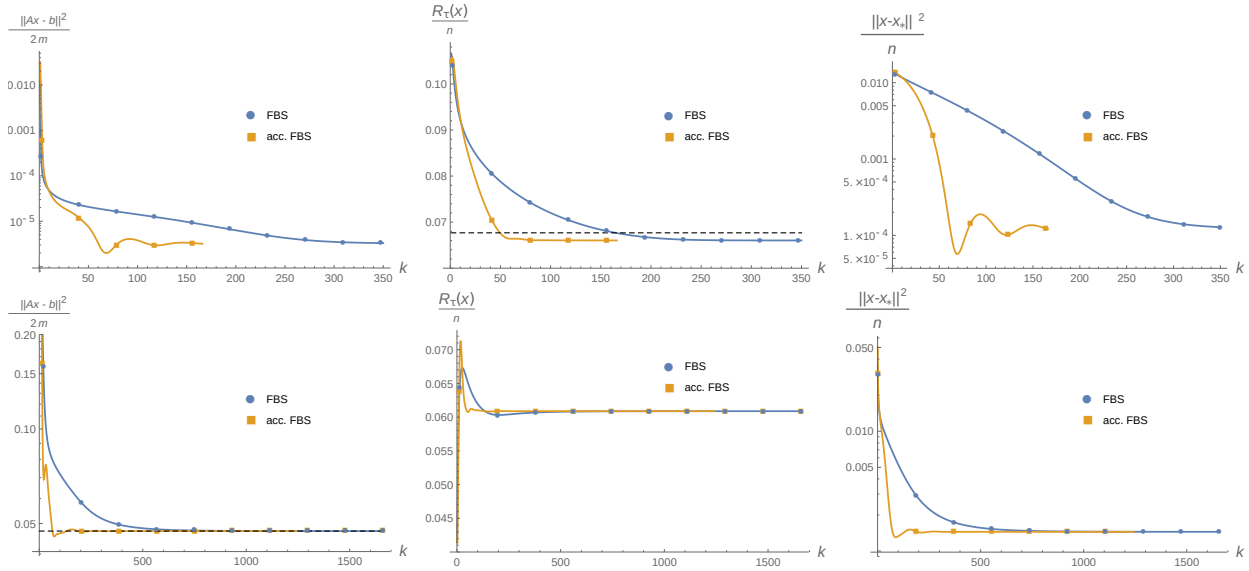

FIGURE 5.5. *Optimization: FBS and accelerated FBS algorithms – No Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the objective (1.4) and the squared error norm (right column), as a function of the iteration index k of the FBS algorithm (1.2) (blue curves) and the accelerated FBS iteration in Algorithm 2. The horizontal dashed lines indicate the corresponding values for x_*. The top and bottom rows correspond to exact and noisy data, respectively: The accelerated FBS algorithm needs about 50% and 25% fewer outer iterations, respectively.

4 in Algorithm 12, the number of gradient and function evaluation is controlled by the parameter κ and is upper bounded by $(\ell_k - \ell_{k-1})\kappa$. Similarly, the cost of a proximal mapping evaluation via (3.7) or (3.8) depends on the tolerance parameter of the employed optimization algorithm. In our case, the box-constrained L-BFGS method from [7] is very efficient in reaching a tolerance level of 10^{-6} within few iterations due to the small values of the parameters β_k used. In particular, we need 3–18 inner iterations while using at most 136 function evaluations of R_τ. This is *comparable* to the cost of Algorithm 9 for the considered choice of parameters. We underline that the low cost of the proximal-point based Landweber algorithm is almost identical to the FBS algorithm (1.2) and the accelerated FBS iteration in Algorithm 2 for the reversed splitting (2.23c).

Optimization. We evaluated accelerated the FBS algorithm with *inexact* evaluations of the proximal maps, for both the unconstrained and nonnegativity constrained problem (1.4). In the former case, the inexactness criterion was evaluated during the primal-dual inner iteration using (4.42). In the latter nonengatively constrained case, we noticed that the corresponding criterion (4.47) could not be used since $z \notin K = \mathbb{R}^n_+$, with z given by (4.46a), happened frequently. We, therefore, resorted to the error estimate (4.50) and chose $\varepsilon_k = \mathcal{O}(k^{-q})$ so as to satisfy (4.13).

Figures 5.8 and 5.9 depict the corresponding results. Inspecting the panels on the left shows that both in the unconstrained and in the constrained case, choosing the proper error decay rate (4.17) significantly affects the accelerated FBS algorithm. Having chosen a proper exponent, an increasing

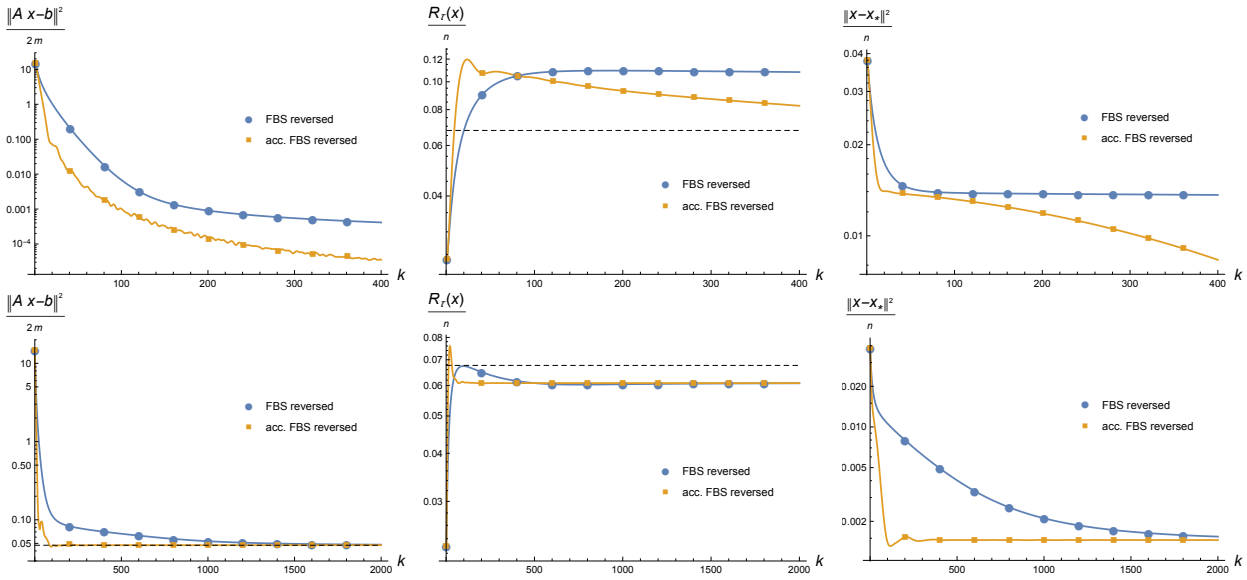

FIGURE 5.6. *Optimization: FBS and accelerated FBS algorithm for the reversed splitting – No Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the *unconstrained* objective h_u (1.4) and the squared error norm (right column), as a function of the iteration indes k of the FBS algorithm (1.2) (blue curves) and the accelerated FBS iteration in Algorithm 2 for the reversed splitting (2.23c). The horizontal dashed lines indicate the corresponding values for x_*. The top and bottom rows correspond to exact and noisy data, respectively.

number of inner iterations, ranging from few dozens to a couple of hundreds, are required for each outer iteration. Even though these inner iterations can be computed efficiently, their total number adds up to few hundreds of thousands for the entire optimization procedure.

5.4. **Discussion.** We discuss our observations according to the following five aspects:

(1) Observations about superiorization;
(2) Observations about optimization;
(3) Superiorization vs. convex optimization: empirical findings;
(4) How superiorization could stimulate convex optimization;
(5) How optimization could stimulate superiorization.

(1) (a) *Superiorization, Figure 5.3, Figure 5.4: The choice of the basic algorithm is essential.* The superiorized versions of the CG algorithm outperforms in our experimental numerical work the superiorized versions of the Landweber algorithm. The superiorized versions of the Landweber algorithm makes slow progress towards x_* due to the bad conditioning of the matrix A and the corresponding small step-size parameters.

(b) *Superiorization, Figure 5.3, Figure 5.4: Incorporating nonnegativity constraints in the superiorization method slows down the iteration process.* Compare the three reconstruction errors in Figure 5.3 to those values in Figure 5.4. Note the disparity in the scale of the x-axis between Figures 5.3 and 5.4. Since the proximal-point based superiorized version

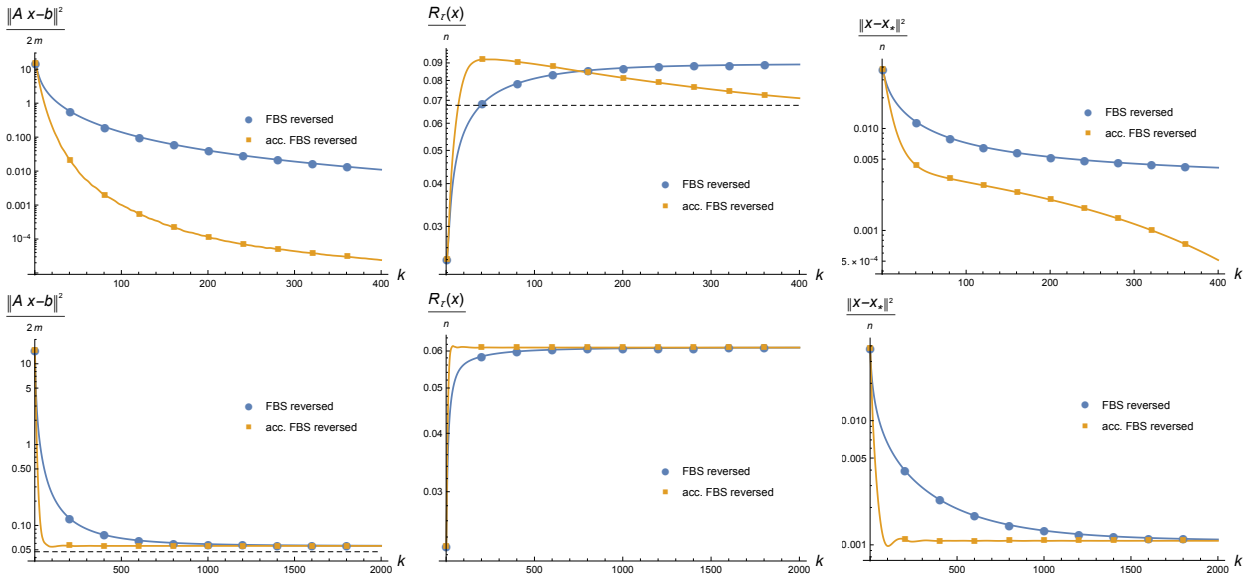

FIGURE 5.7. *Optimization: FBS and accelerated FBS iteration for the reversed splitting – Nonnegativity Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the *nonnegativity constrained* objective h_c (1.4) and the squared error norm (right column), as a function of the iteration index k of the FBS algorithm (blue curves) and the accelerated FBS algorithm for the reversed splitting (2.23c). The horizontal dashed lines indicate the corresponding values for x_*. The top and bottom rows correspond to exact and noisy data, respectively. While the FBS algorithm is damped by the inclusion of constraints, the nonnegativity constraints appear to be beneficial for the accelerated FBS algorithm on the other hand when comparing to the results in Figure 5.6.

of the projected Landweber algorithm (`ProxSupProjLW`) behaves identically to the superiorized version of the Landweber algorithm, which includes nonnegativity constraints via the proximal map (`ProxCSubLW`), we conclude that this is not an artefact of the proposed method of computing nonascent directions but rather a characteristic of projection-based algorithms. We conjecture that a geometric basic algorithm that smoothly evolves on a manifold defined by the constraints, in connection with a proximal point that employs an appropriate Bregman distance, can remedy this situation.

(2) (a) *Optimization, Figure 5.6, Figure 5.7: Incorporating nonnegativity constraints in the FBS algorithm for the reversed splitting* (2.23c) *also slows down the algorithm. On the other hand, the accelerated FBS iteration in Algorithm 2 does not suffer from this shortcoming.*

(b) *Optimization, Figure 5.5: Acceleration is very effective at little additional cost provided proximal maps are computed exactly.* About 50 and 25 outer iterative steps are merely needed to terminate in the noise-free and noisy case, respectively (the latter termination criterion - just reaching the noise level - is more loose). These small numbers of outer iterations result from computing exact proximal maps.

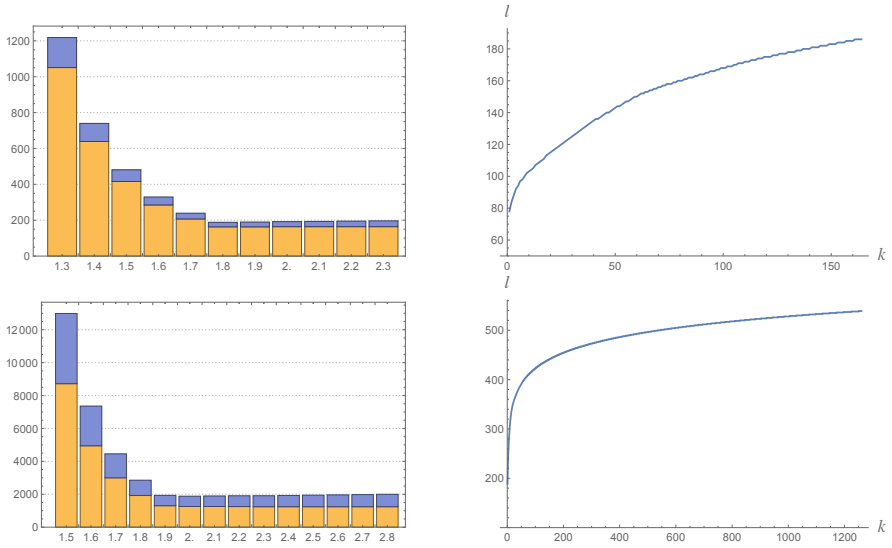

FIGURE 5.8. The inexact accelerated FBS algorithm for the *unconstrained* problem (1.4). LEFT: Relation of the number of outer iterations (not scaled down) and the *total* number of inner iterations (scaled down by $1/1000$) as a function of the exponent q of the error $\varepsilon_k = \mathcal{O}(k^{-q})$ (4.17). RIGHT: The number l of inner iterations at each outer iteration k, for exponents $q = 1.8$ (top) and $q = 2.0$ (bottom) of (4.17).

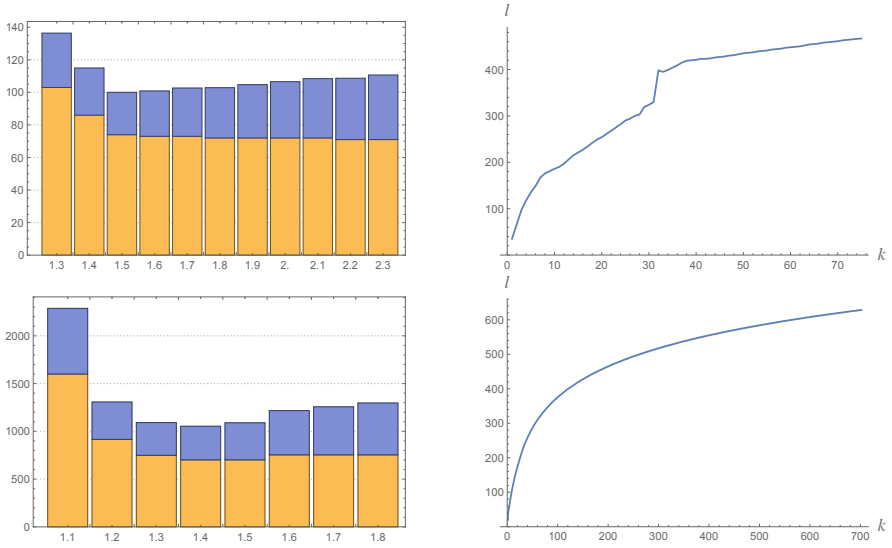

FIGURE 5.9. The inexact accelerated FBS algorithm for the *nonnegativity constrained* problem (1.4). LEFT: Relation of the number of outer iterations (not scaled down) and the *total* number of inner iterations (scaled down by $1/1000$) as a function of the exponent q of the error $\varepsilon_k = \mathcal{O}(k^{-q})$ (4.17). RIGHT: The number l of inner iterations at each outer iteration k, for exponents $q = 1.8$ (top) and $q = 2.0$ (bottom) of (4.17).

(c) *Unconstrained optimization, Figure 5.8: Inexact inner loops considerably increase the number of iterations.* In comparison to the results shown by Figure 5.5, where termination of the accelerated FBS algorithm using *exact* proximal maps was reached after ≤ 75 iterations, Figure 5.8 shows that the number of outer iterations increases to about 150 and 1200 in the case of noise-free and noisy data, respectively, when *inexact* evaluations of the proximal map are used. This contrast with the results shown by Figure 5.5 (exact proximal maps), where the difference between noise-free and noisy data is insignificant.
Each inexact evaluation of the proximal map requires on average 130 and 450 primal-dual iterations when using noise-free and noisy data, respectively, which are computationally inexpensive. The two panels on left-hand side of Figure 5.8 reveal that the efficiency of *acceleration* of the outer FBS iteration requires to choose the exponent q of the error $\varepsilon_k = \mathcal{O}(k^{-q})$ not smaller than a problem-dependent critical value (cf. the caption of Figure 5.8).

(d) *nonnegativity constrained optimization, Figure 5.9: The number of outer iterations decreases relative to Figure 5.8 (so constraints help), but the number of inner iterations increases.* Otherwise, the observations regarding Figure 5.8 hold.

(3) (a) *Superiorization vs. optimization: Convergence.* Each optimization algorithm is guaranteed to reduce the combined recovery errors (1) and (2), listed in the beginning of Subsection 5.2. The sub-sampling ratio m/n is chosen according to [43] so that recovery via (1.4) is stable. As a consequence, also the recovery error (3) in Subsection 5.2 is reduced by the optimization algorithm. While the superiorized versions of the basic algorithms are guaranteed to reduce only the, above mentioned, recovery error in (1), we observe in Figure 5.3 and Figure 5.4 that also recovery errors (2) and (3) are decreased to levels comparable to those achieved by optimization. This favorable performance of superiorization requires some parameter tuning as described in Subsection 5.2.

(b) *Superiorization vs. optimization: Best performance is achieved by the superiorized version of the CG algorithm and by the accelerated FBS algorithm for the reversed splitting.* The evaluation of the implemented algorithms shows that, in the unconstrained case, both the proximal-point based superiorized *version of the CG algorithm* and the accelerated FBS algorithm for the reversed splitting (2.23c) reach almost optimal error term values within the fewest number of iterations at low computational cost (4 vs. 2 matrix-vector evaluations and 1 proximal-point evaluation of R_τ). In the constrained case, the accelerated FBS iteration, Algorithm 2, applied for the reversed splitting (2.23c) performs best while keeping low the computational cost.

(c) *Superiorization vs. optimization: We addressed the balancing question by the two alternative splittings.* By analyzing and evaluating the FBS algorithm for the two alternative splittings, we also addressed the so-called *balancing question* in the SM: How to distribute the efforts that a superiorized versions of a basic algorithm invests in target function reduction steps versus the efforts invested in the basic algorithm iterative steps? In the case of the splitting (2.23b) that results in the inexact (accelerated) FBS algorithm, we performed *many basic algorithm iteration steps*, e.g., via Algorithm 15, and just *one target function reduction step* with respect to R_τ. By contrast, in the case of the reversed splitting (2.23c), we performed *one basic algorithm iteration* and *many target function reduction steps*, to

compute the proximal point in (1.2). Our results show that the *accelerated* reversed FBS is computationally more efficient. They suggest that one basic iteration should be followed by many target function reduction steps for our considered scenario.

(4) *There is a deeper relation between the balancing problem of the SM and the splitting problem of optimization.* Due to the above comment (3)(b), the number of steps of the basic algorithm relative to the number of steps for target function reduction is crucial for the performance of superiorization. For our considered problem, a relatively larger number of target function reduction steps seems to pay. This observation agrees with the better performance of the *reversed* splitting for optimization, since then inexact evaluations of the proximal map entail a larger relative number of target function reduction steps.

(5) *Superiorization vs. optimization: Future work.* Observation (4) above motivates us to consider in future research an inexact Douglas-Rachford (DR) splitting that includes two proximal maps: one corresponding to the least-squares term of (1.4) and one corresponding to the regularizer R_τ that might include nonnegativity constraints as well. Such an inexact DR splitting would not only be structurally close to the superiorization of a basic algorithm for least-squares, but also provide a foundation for a theoretical investigation of the SM.

6. Conclusion

We considered the underdetermined nonnegative least-squares problem, regularized by total variation, and compared, for its solution, superiorization with a state-of-the-art convex optimization approach, namely, accelerated forward-backward (FB) splitting with inexact evaluations of the corresponding proximal mapping. The distinction of the basic algorithm and target function reduction by the SM approach motivated us to contrast superiorization with an FB splitting, such that the more expensive backward part corresponds structurally to the basic algorithm, whereas the forward part takes into account the target function. However, in view of the balancing problem of the SM, we also considered the *reverse* FB splitting, since exchanging the forward and backward parts in connection with inexact evaluations of the proximal mapping is structurally closer to superiorization.

Our experiments showed that superiorization outperforms convex optimization *without* acceleration, after proper parameter tuning. Convex optimization *with* acceleration, however, is on a par with superiorization or even more efficient when using the *reverse* splitting. This raises the natural question: How can *accelerated* superiorization be defined for basic algorithms and target function reduction procedures?

Superiorization performs best when the number of iterative steps used for the basic algorithm and for target function reduction, respectively, are balanced properly. Our results indicate a deeper relationship of the balancing problem of superiorization and the splitting problem of convex optimization. We suggested a splitting approach (point (5) in Subsection 5.4 above) that deems most promising to us for further closing the theoretical gap between superiorization and optimization of composite convex problems.

Finally, we point out that we restricted our study to an overall *convex* problem, which enabled us to apply an advanced optimization scheme that combines acceleration with inexact evaluations.

While superiorization has proven to be useful for *nonconvex* problems as well, more research is required to relate both methodologies to each other in the nonconvex case.

Acknowledgements
We are grateful to the three anonymous reviewers for their constructive comments that helped us improve the paper. The work of the YC was supported by the ISF-NSFC joint research program grant No. 2874/19.

REFERENCES

[1] A. Auslender, M. Teboulle, Asymptotic cones and functions in optimization and variational inequalities, Springer, New York, 2003.
[2] H. H. Bauschke, J. M. Borwein, On projection algorithms for solving convex feasibility problems, SIAM Rev. 38 (1996), 367-426.
[3] H. H. Bauschke, J. M. Borwein, Legendre functions and the method of random Bregman projections, J. Convex Anal. 4 (1997), 27-67.
[4] H. H. Bauschke, P. L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2nd ed., Springer, 2017.
[5] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, 1996.
[6] D. Butnariu, R. Davidi, G. T. Herman, I. G. Kazantsev, Stable convergence behavior under summable perturbations of a class of projection methods for convex feasibility and optimization problems, IEEE J. Sel. Topics Signal Process. 1 (2007), 540-547.
[7] R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. 16 (1995), 1190-1208.
[8] D. Butnariu, S. Reich, A. J. Zaslavski, Convergence to fixed points of inexact orbits of Bregman-monotone and of nonexpansive operators in Banach spaces, Fixed Point Theory and its Applications, In: H. F. Nathansky, B.G. de Buen, K. Goebel, W. A. Kirk, B. Sims, (ed.), pp. 11-32, Yokohama Publishers, Yokohama, 2006,
[9] D. Butnariu, S. Reich, A. J. Zaslavski, Stable convergence theorems for infinite products and powers of nonexpansive mappings, Numer. Funct. Anal. Opt. 29 (2008), 304–323.
[10] C. Bargetz, S. Reich, R. Zalas, Convergence properties of dynamic string-averaging projection kethods in the presence of perturbations, Numer. Algo. 77 (2018), 185-209.
[11] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imag. Sci. 2 (2009), 183-202.
[12] A. Beck, M. Teboulle, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. Image Proc. 18 (2009), 2419-2434.
[13] C.L. Byrne, What do simulations tell us about superiorization? Preprint. Available on ResearchGate at: https://www.researchgate.net/publication/336361338_What_ Do_Simulations_Tell_Us_About_Superiorization_revised_October_8_2019, Revised: October 8, 2019.
[14] A. Cegielski, F. Al-Musallam, Superiorization with level control, Inverse Probl. 33 (2017), 044009.
[15] Y. Censor, R. Davidi, G. T. Herman, Perturbation resilience and superiorization of iterative algorithms, Inverse Probl. 26 (2010), 65008.
[16] Y. Censor, R. Davidi, G. T. Herman, R. W. Schulte, L. Tetruashvili, Projected subgradient minimization versus superiorization, J. Optim. Theory Appl. 160 (2014), 730-747.
[17] A. Cegielski, Iterative Methods for Fixed Point Problems in Hilbert Spaces, Lect. Notes Math., vol. 2057, Springer, 2013.
[18] Y. Censor, Weak and strong superiorization: between feasibility-seeking and minimization, An. St. Univ. Ovidius Constanta 23 (2015), 41-54.

[19] Y. Censor, Superiorization and perturbation resilience of algorithms: A bibliography compiled and continuously updated, http://math.haifa.ac.il/yair/bib-superiorization-censor.html, 2019, Last updated: January 6, 2020.
[20] Y. Censor, E. Garduño, E. S. Helou, G. T. Herman, Derivative-free superiorization: principle and algorithm, arXiv e-prints (2019), arXiv:1908.10100.
[21] Y. Censor, G. T. Herman, M. Jiang (Editors), Special Issue on Superiorization: Theory and Applications, vol. 33 (4), IOP Science, Inverse Probl. 2017.
[22] Y. Censor, E. Levy, An analysis of the superiorization method via the principle of concentration of measure, Appl. Math. Comput. (2019). DOI:10.1007/s00245-019-09628-4.
[23] P. L. Combettes, Solving monotone inclusions via compositions of nonexpansive averaged operators, Optimization 53 (2004), 475-504.
[24] P. L. Combettes, Perspective functions: properties, constructions, and examples, Set-Valued Var. Anal. 26 (2018), 247-264.
[25] A. Chambolle, T. Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, J. Math. Imag. Vision 40 (2011), 120-145.
[26] P. L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, fixed-point algorithms for inverse problems in science and engineering, In: H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, H. Wolkowicz, (ed.) pp. 185-212, Springer, New York, 2011.
[27] P. L. Combettes, V. R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Model. Simul. 4 (2005), 1168-1200.
[28] Y. Censor, S. A. Zenios, Parallel Optimization: Theory, Algorithms, and Applications, Oxford University Press, New York, NY, 1997.
[29] Y. Censor, A. J. Zaslavski, Strict Fejér monotonicity by superiorization of feasibility-seeking projection methods, J. Optim. Theory Appl. 165 (2015), 172-187.
[30] D.-Q. Chen, H. Zhang, L.-Z. Cheng, A fast fixed point algorithm for total variation deblurring and segmentation, J. Math. Imag. Vision 43 (2012), 167-179.
[31] R. Davidi, Algorithms for superiorization and their applications to image reconstruction, Ph.D. thesis, City University of New York, New York, NY, 2010.
[32] R. Davidi, G. T. Herman, Y. Censor, Perturbation-resilient block-iterative projection methods with application to image reconstruction from projections, Int. Trans. Oper. Res. 16 (2009), 505-524.
[33] A. Denitiu, S. Petra, Cl. Schnörr, Ch. Schnörr, Phase transitions and cosparse tomographic recovery of compound solid bodies from few projections, Fundam. Inform. 135 (2014), 73-102.
[34] Y. Guo, W. Cui, Strong convergence and bounded perturbation resilience of a modified proximal gradient algorithm, J. Inequal. Appl. 2018 (2018), 103.
[35] T. Goldstein, S. Osher, The split bregman method for L_1-regularized problems, SIAM J. Imag. Sci. 2 (2009), 323-343.
[36] G. T. Herman, E. Garduño, R. Davidi, Y. Censor, Superiorization: An optimization heuristic for medical physics, Med. Phy. 39 (2012), 5532-5546.
[37] N.J. Higham, Functions of Matrices: Theory and Computation, SIAM, 2008.
[38] P. C. Hansen, J. S. Jørgensen, AIR tools II: Algebraic iterative reconstruction methods, improved implementation, Numer. Algo. 79 (2018), 107-137.
[39] E. S. Helou, C. Lin, M. V. W. Zibetti, G. T. Herman, Superiorization of preconditioned conjugate gradient algorithms for tomographic image reconstruction, Appl. Anal. Optim. 2 (2018), 271-284.
[40] E. S. Helou, M. V. W. Zibetti, E. X. Miqueles, Superiorization of incremental optimization algorithms for statistical tomographic image reconstruction, Inverse Probl. 33 (2017), 044010.
[41] H. He, H.-K. Xu, Perturbation resilience and superiorization methodology of averaged mappings, Inverse Probl. 33 (2017), 044007.
[42] W. Jin, Y. Censor, M. Jiang, Bounded perturbation resilience of projected scaled gradient methods, Comput. Optim. Appl. 63 (2016), 365-392.

[43] J. Kuske, S. Petra, Performance bounds for co-/sparse box constrained signal recovery, An. St. Univ. Ovidius Constanta 1 (2019), 79-109.
[44] S. Luo, Y. Zhang, T. Zhou, J. Song, Y. Wang, XCT image reconstruction by a modified superiorized iteration and theoretical analysis, Optim. Methods Softw. (2019), DOI: 10.1080/10556788.2018.1560442
[45] Y. E. Nesterov, A method for solving the convex programming problem with convergence rate $O(1/k^2)$, Soviet. Math. Dokl. 27/2 (1983), 372-376.
[46] S. Petra, C. Schnörr, Average case recovery analysis of tomographic compressive sensing, Linear Algebra Appl. 441 (2014), 168-198.
[47] D. Reem, A. De Pierro, A new convergence analysis and perturbation resilience of some accelerated proximal forward-backward algorithms with errors, Inverse Probl. 33 (2017), 044001.
[48] R. Rockafellar, Convex Analysis, Princeton Univ. Press, Princeton, 1970.
[49] R. T. Rockafellar, R. J.-B. Wets, Variational Analysis, 3rd ed., Springer, 2009.
[50] S. Reich, A. J. Zaslavski, Convergence to approximate solutions and perturbation resilience of iterative algorithms, Inverse Probl. 33 (2017), 044005.
[51] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, 2003.
[52] E. Y. Sidky, J. H. Jørgensen, X. Pan, Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle-Pock algorithm, Phys. Med. Biol. 57 (2012), 3065-3091.
[53] P. Tossings, The perturbed proximal point algorithm and some of its applications, Appl. Math. Optim. 29 (1994), 125-159.
[54] C. R. Vogel, M. E. Oman, Fast, robust total variation-based reconstruction of noisy, blurred images, IEEE Trans. Image Proc. 7 (1998), 813-824.
[55] E. A. H. Vollebregt, The bound-constrained conjugate gradient method for non-negative matrices, J. Optim. Theory Appl. 162 (2014), 931-953.
[56] S. Villa, S. Salzo, L. Baldassarre, A. Verri, Accelerated and inexact forward-backward algorithms, SIAM J. Optim. 23 (2013), 1607-1633.
[57] Q. Xu, D. Yang, J. Tan, A. Sawatzky, M. A. Anastasio, Accelerated fast iterative shrinkage thresholding algorithms for sparsity-regularized cone-beam CT image reconstruction, Med Phys. 43 (2016), 1849.
[58] M. V. W. Zibetti, E. S. Helou, R. R. Regatte, G. T. Herman, Monotone FISTA with variable acceleration for compressed sensing magnetic resonance imaging, IEEE Trans. Comput. Imaging 5 (2019), 109-119.
[59] M. V. W. Zibetti, C. Lin, G. T. Herman, Total variation superiorized conjugate gradient method for image reconstruction, Inverse Probl. 34 (2018), 034001.