arXiv:1911.05498v1 [math.OC] 13 Nov 2019

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

YAIR CENSOR, STEFANIA PETRA, CHRISTOPH SCHNÖRR

ABSTRACT. In this paper we conduct a study of both superiorization and optimization approaches for the reconstruction problem of superiorized/regularized solutions to underdetermined systems of linear equations with nonnegativity variable bounds. Specifically, we study a (smoothed) total variation regularized least-squares problem with nonnegativity constraints. We consider two approaches: (a) a superiorization approach that, in contrast to the classic gradient based superiorization methodology, employs proximal mappings and is structurally similar to a standard forward-backward optimization approach, and (b) an (inexact) accelerated optimization approach that mimics superiorization. Namely, a basic algorithm for nonnegative least squares that is enforced by inexact proximal points is perturbed by negative gradients of the the total variation 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. Reversing the roles of the terms treated by accelerated forward-backward optimization, on the other hand, slightly outperforms superiorization, which suggests that optimization can approach superiorization too, using a suitable problem splitting. Extensive numerical results substantiate our discussion of these aspects.

CONTENTS

Date: 4th November.

Key words and phrases. superiorization, perturbation resilience, convex optimization, accelerated inexact proximal iteration, proximal mapping, problem splitting, forward-backward.

1. Introduction

The purpose of this work is to present an advanced comparative study of superiorization and accelerated inexact convex optimization. The motto of the superiorization methodology (SM) is to take a convergent iterative algorithm ("the basic algorithm") which is known to converge to a point in a certain set and perturb its iterates such that the perturbed algorithm ("the superiorized version of the basic algorithm") will still converge to some point of the same set. Such perturbations make the basic algorithm "perturbation resilient". When the perturbations lower the value of a given exogenous function ("the target function") then the output of the superiorized algorithm is expected to be "superior" to the output of the basic algorithm in the sense that it has a lower (not necessarily minimal) value of the target function. For basic algorithms that are feasibility-seeking the SM defines a novel class of approaches that may be classified as being "in between" solving convex feasibility problems and solving optimization problems, respectively. The idea is to improve – superiorize – the basic algorithm that is resilient against bounded perturbations, by perturbing it such that the iterates attain lower values of the target function. As a consequence, the returned vector is superior, but not necessarily optimal, with respect to the target function and any vector produced by the basic algorithm without superiorization. This weaker guarantee – convergence and superiority, but not necessarily optimality – enables simple implementations that only require minimal changes of existing code that implements the 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 can be based on a perturbation resilient basic algorithm.

Introductory and advanced materials about the SM, accompanied by relevant references, can be found in our recent papers [CL19, CGHH19], in particular, [CGHH19, 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 bibliography Internet page that currently contains 104 items [Cen19]. 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 [CHJE17] contains several interesting papers on the theory and practice of SM, such as [CAM17], [HX17], [RZ17], to name but a few. Later papers continue research on perturbation resilience, which lies at the heart of the SM, see, e.g., [BRZ18]. The superiorization method was born when the terms and notions "superiorization" and "perturbation resilience", in the present context, first appeared in the 2009 paper [DHC09] which followed its 2007 forerunner [BDHK07]. The ideas have some of their roots in the 2006 and 2008 papers [BRZ06, BRZ08]. All these culminated in Ran Davidi's 2010 PhD dissertation [Dav10] and the many papers since then cited in [Cen19]. Another recent work attempts at analysing the behavior of the SM via the concept of concentration of measure [CL19].

Superiorization has spurred research during the recent years, focusing on proofs of perturbation resilience of an increasing number of basic algorithms and on numerical experiments that support the claim that in comparison to algorithms for solving constrained optimization problems, superiorized algorithms are simpler, computationally more efficient and return solutions that are often "sufficiently feasible and sufficiently optimal" in practice. A recent example is the work [ZLH18, HLZH18] where the preconditioned conjugate gradient iteration (PCG) is used as a basic algorithm in the algebraic (fully-discretized) model of computed tomography (CT) and is superiorized using the total variation (TV) as a target function. The authors report that superiorized PCG compares favorably to a state-of-the-art optimization algorithm, FISTA [BT09a], applied to the unconstrained convex problem

$$\min_x \left(\frac{1}{2}\|Ax-b\|^2 + \lambda R(x) \right), \quad \lambda > 0, \tag{1.1}$$

where minimizing the first term aims to recover a vector x from linear tomographic model and measurements, whereas minimizing the second, discrete TV, function R enforces a sparse support of the discrete gradient of x. FISTA is well-known to be particularly efficient for convex problems with sparsity-enforcing ℓ_1 or TV regularizers, because proximal maps with respect to these regularizers, when iterated, can be carried out efficiently by shrinkage and TV-based denoising, respectively [GO09, BT09b].

It is interesting to observe the different points of view toward problem (1.1) taken by the SM and by optimization due to treating differently the summands in (1.1). Exemplary implementations [SJP12, XYT+16] of state-of-the-art optimization approaches [CP11a, BT09a] for the TV-based regularization problem (1.1) typically consider a TV minimization algorithm (as proximal term) and modify its iterates by gradients of the least-squares term. Superiorization, on the other hand, considers first a basic algorithm for solving the tomographic recovery problem by a least-squares approach that is known to be resilient against bounded perturbations, and perturbs the iterates with negative subgradients of the TV function to achieve superiorization.

The earlier work [CDH+14] compared superiorization with optimization for a similar application problem. The chosen optimization method there is the projected subgradient method. It has a structure that is similar to superiorization but its convergence rate falls short of state-of-the-art optimization approaches.

Our motivation for the present paper is to conduct a study of comparing superiorization and optimization for the problem

$$\min_{x\geq 0} \left(\frac{1}{2}\|Ax-b\|^2 + \lambda R_\tau(x) \right), \quad \tau, \lambda > 0, \tag{1.2}$$

that is slightly more complex than (1.1) due to the nonnegativity constraints that are often quite relevant in practice. In particular, we wish that the optimization approach that we use for the comparison should retain the similarity with the structure of superiorization, i.e., a basic algorithm related to the inverse problem $Ax = b$ that is perturbed by nonascent directions of the target functional R_τ, which explains why (1.2) comprises a C^1-approximation of R of (1.1) parameterized by τ. In addition, the optimization approach should be based on approaches to convex programming that are more efficient than the projected subgradient method. This mainly concerns two trends:

(i) *Inexact* computations in connection with proximal splitting, and

(ii) *Acceleration* of such inexact proximal iterations in order to achieve $\mathcal{O}(k^{-2})$ convergence rates.

Regarding (i), proximal iterations based on various operator splittings [CP11b] as well as inexactness in terms of summable error sequences [Tos94, Com04] are well-known. More recent work has focused on inexactness criteria that can be checked computationally at each iteration. Regarding (ii), research has focused on extending the acceleration technique introduced by Nesterov to such inexact proximal schemes. We refer to [RDP17] 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 (1.2), however, due to the nonnegativity constraints here.

We point out that convex problems of the form (1.2) have been extensively studied from the viewpoint of convex programming (e.g., [VO98, BT09b, GO09, CZC12]). Selecting the most efficient optimization approach is *not* the main objective of the present paper, however. Rather, we consider also an optimization approach that mimics the structure of superiorization using R_τ as target function, even if this approach is *not* the most efficient one for optimizing (1.2). This restriction makes sense since superiorization also applies to nonconvex problems, in principle, for which the efficiency of optimization algorithms tends to deteriorate anyway.

Organization. After specifying the problem and basic assumptions in Section 3, we work out a superiorization approach in Section 4. Specifically, we consider a state-of-the-art gradient based superiorized conjugate gradient (CG) iteration that we adapt to underdetermined linear systems. Further, we consider superiorization based on proximal mappings in order to handle nonnegativity constraints. In addition to CG we also consider the superiorization of the Landweber and the projected Landweber iteration that is structurally similar to a basic forward-backward iteration for solving (1.2). In Section 5, we work out an optimization approach to (1.2), accelerated inexact forward-backward splitting, that mimics the structure of the superiorization approach: Accelerated inexact proximal iteration, with respect to the nonnegative least-squares recovery problem, is considered as a "basic algorithm" and is perturbed by negative gradients of the target function R_τ. Since the proximal maps cannot be computed in closed-form, they are implemented by an inner iterative loop. We also study the *reverse* forward-backward splitting since it reveals another relation to superiorization that affects the overall performance. These aspects and our findings are discussed in Section 6, based on an a series of quantitative numerical results. We conclude in Section 7.

2. Superiorization Versus Convex Optimization

The superiorization method (SM) interlaces two algorithmic activities under the "fundamental assumption" that one algorithmic activity (called the "basic algorithm") is resilient to the "disturbances" of its iterates caused by the activity of the other algorithm, called "perturbations". The resilience means that an important property of the basic algorithm, such as asymptotic convergence to points in a given set, or ε-compatibility with such a set, are preserved in spite of the repeated perturbations.

The Forward-Backward (FB) splitting optimization method, that plays a prominent role in this paper, also interlaces two algorithmic activities and, therefore, looks from the structural point of view "similar" to the SM. However, the FB applies its interlaced iterative process under different conditions, in particular, it does not obey the "fundamental assumption" of the SM stated above. This difference sets these two methods apart. Such situations of algorithms being similar but not identical are abound.

One example is the sequential Kaczmarz algorithm applied to a consistent system of linear equations $Ax = b$. When it is initialized anywhere in space, i.e., $x^0 \in \mathbb{R}^n$ then it converges asymptotically to any point in the nonempty intersection of the hyperplanes of the linear system. But if it is initialized in the range of $A^\top$, $x^0 \in \mathcal{R}(A^\top)$, then it is guaranteed to converge to the feasible point of $Ax = b$ that is closest to the origin, see, e.g., [CZ97, Algorithm 6.5.1]. Are these then the same algorithm or different algorithms? The first can find only feasible points while the latter does norm-minimization over the linear system. These algorithms share the structure of the iteration process but are different in the problem that they solve and in the assumptions under which they work. Efforts to understand the very good performance of the SM in practice motivated recently Byrne [Byr19] to notice and study similarities between some SM algorithms and optimization methods.

In spite of this we juxtapose experimentally the SM with two versions of FB (plain FB and reversed FB) and investigate their performances. We consider in two different ways how to treat the two algorithmic activities in FB-splitting. We do not claim this to be identical with SM, because - as stated above - the assumptions and the algorithms differ. The point we make is that both versions of the FB splitting can be understood (i) from the SM viewpoint, informally not in a mathematically-strict sense, and (ii) from the

convex optimization viewpoint rigorously, as methods for solving the underdetermined least-squares problem such that the LS-solution (out of many existing ones) achieves an optimal value of the target function R_τ (or $R_\tau + \delta_{\mathbb{R}^n_+}$ in the nonnegatively constrained case).

Pointing out these relations between SM and FB, substantiated with quantitative numerical results, should stimulate further research of both.

3. PRELIMINARIES

3.1. **Basic 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 $\mathbb{R}^n_+$. 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 $C = K$, we simply write $x_+ = \Pi_K(x)$ and, similarly, $x_- = \Pi_{K^*}(x) = x - x_+$. The unit matrix is denoted by I. 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 $A \in \mathbb{R}^{m\times n}$, $\|A\|$ denotes its spectral norm and $A^\top$ is the transformed matrix.

We assume that images are discretized on n grid points in a two dimensional domain in $\mathbb{R}^2$. We define next the discrete gradient matrix obtained by forward differences. We consider first the one-dimensional discrete derivative operator of a $M \times N$ discrete image

$$\partial_d : \mathbb{R}^d \to \mathbb{R}^d, \quad \partial_d = \begin{cases} -1, & i = j < d, \\ +1, & j = i + 1, \\ 0, & \text{otherwise}, \end{cases} \tag{3.1}$$

along each spatial direction, with $d \in \{M, N\}$ and write

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} = \begin{pmatrix} \partial_M \otimes I_N \\ I_M \otimes \partial_N \end{pmatrix} \in \mathbb{R}^{2n\times n}, \tag{3.2}$$

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.

$$\mathcal{F}_c := \{f\colon \mathbb{R}^n \to \overline{\mathbb{R}} \mid f \text{ is convex, proper and lower semicontinuos}\}, \tag{3.3a}$$

$$\mathcal{F}^1_c(L) := \{f \in \mathcal{F}_c \mid f \in C^1 \text{ and } \nabla f \text{ is } L\text{-Lipschitz-continuous}\}, \tag{3.3b}$$

$$\mathcal{F}^1_c(L, \mu) := \{f \in \mathcal{F}^1_c(L) \mid f \text{ is } \mu\text{-strongly monotone convex}\}. \tag{3.3c}$$

We use subscripts L_f, μ_f to specify the corresponding concrete function f. f^* denotes the Legendre-Fenchel conjugate of $f \in \mathcal{F}_c$.

Given a function $f\colon \mathbb{R}^n \to \overline{\mathbb{R}}$ and a point $x \in \operatorname{dom} f$, we say that a vector $v \in \mathbb{R}^n$ is a *nonascending vector for f at x* if $\|v\| \le 1$ and there is a $\bar{t} > 0$ such that

$$f(x + tv) \le f(x), \quad \text{for all } t \in (0, \bar{t}]. \tag{3.4}$$

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

$$P_\alpha f(x) := \arg\min_y \Big\{ f(y) + \frac{1}{2\alpha}\|y - x\|^2 \Big\}, \tag{3.5}$$

whereas the *Moreau envelope* is the function defined by

$$e_\alpha f(x) := \inf_y \Big\{ f(y) + \frac{1}{2\alpha}\|y - x\|^2 \Big\}. \tag{3.6}$$

This function is continuously differentiable with gradient [RW09, Theorm 2.26]

$$\nabla e_\alpha f(x) = \frac{1}{\alpha}\big(x - P_\alpha f(x)\big). \tag{3.7}$$

In the optimization part of this study that concerns the iteration (5.13) below, the sequence of iterates generated by the outer loop will be denoted by (x_k), whereas (z_l) denotes the sequence of iterates, for any fixed k, that is generated by an inner loop in order to evaluate (possibly inexactly) the proximal map $P_{\alpha_k} g(\cdot)$. In the latter context, to simplify notation, we write (5.13) as

$$\overline{y} = P_\alpha g(x) \tag{3.8}$$

with sequence (z_l) converging to $\overline{y}$. Inexact evaluation means to terminate the inner loop after step l and to continue the outer loop with $x_{k+1} = z_{l+1}$. In connection with (3.8), we write $z = z_{l+1} \approx \overline{y}$ for the latter approximation and $\varepsilon = \varepsilon_k$ for a corresponding error parameter.

3.2. **Problem Formulation.** We consider problem (1.2) and assume

$$A \in \mathbb{R}_+^{m\times n}, \qquad \operatorname{rank}(A) = m < n \tag{3.9}$$

and $R_\tau \in \mathcal{F}_c(L_{R_\tau})$. Regarding the definition of R_τ, we use the discrete gradient matrix (3.2) 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

$$R\colon \mathbb{R}^n \to \mathbb{R}_+, \qquad R(x) = \sum_{i\in[n]} \big(|(D_1x)_i| + |(D_2x)_i|\big). \tag{3.10}$$

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

$$R(x) = \sup_{p\in C} \sum_{i\in[n]} \left\langle p_i, \binom{(D_1x)_i}{(D_2x)_i} \right\rangle, \quad C = \{(p_1,\dots,p_i,\dots,p_n) \in \mathbb{R}^{2n} \colon p_i \in \mathbb{R}^2, \|p_i\|_\infty \le 1, i \in [n]\}, \tag{3.11}$$

which enables the smooth approximation of R in a standard way [AT03, Sect. 2.8], [Com18]

$$R_\tau(x) = \tau \sum_{i\in[n]} \big(\sqrt{1 + |(D_1x/\tau)_i|^2} + \sqrt{1 + |(D_2x/\tau)_i|^2}\big) \tag{3.12a}$$

$$= \sum_{i\in[n]} \big(\sqrt{\tau^2 + |(D_1x)_i|^2} + \sqrt{\tau^2 + |(D_2x)_i|^2}\big), \qquad 0 < \tau \ll 1. \tag{3.12b}$$

R_τ is continuously differentiable. The gradient

$$\nabla R_\tau = D^\top \operatorname{Diag}\Big(\dots, \big(\tau^2 + |(D_1x)_i|^2\big)^{-1/2}, \dots, \big(\tau^2 + |(D_2x)_i|^2\big)^{-1/2}, \dots\Big) D \tag{3.13}$$

is Lipschitz continuous with constant

$$L_{R_\tau} \le \frac{1}{\tau}\|D\|^2 < \frac{8}{\tau}. \tag{3.14}$$

Regarding data errors, we adopt the basic Gaussian noise model

$$(Ax - b)_i \sim \mathcal{N}(0, \sigma^2), \ i \in [m]. \tag{3.15}$$

4. SUPERIORIZATION

Superiorization involves a basic algorithm that has some perturbation resilience properties and concrete perturbations of iterates of the basic algorithm related to a target function whose value one wants to reduce. We examine these aspects in turn, with a focus on the problem (1.2). The basic algorithm discussed here is, in general, an iterative process of the form

$$(x_{k+1}, u^1_{k+1}, \dots u^\nu_{k+1}) = \mathcal{A}(x_k, u^1_k, \dots u^\nu_k), \quad \text{for all } k \geq 0, \tag{4.1}$$

where (x_k) are the primal iterates and (u^i_k), $i \in [\nu]$ some auxiliary sequences of vectors that in general depend on (x_k).

We consider two such iterations in Section 4.1 below. In particular, we consider the conjugate gradient (CG) iteration and the Landweber iteration that are both appropriate for the least-squares problem. The Landweber iteration, with a slight modification, is also appropriate the nonnegative least-squares problem. Superiorization of the basic algorithm 4.1 can be written as

$$(x_{k+1}, u_{k+1}) = \mathcal{A}(x_{k+1/2}, u^1_k, \dots u^\nu_k), \quad \text{for all } k \geq 0, \tag{4.2}$$

where the basic algorithmic operator $\mathcal{A}$ is applied to the *perturbed iterations*

$$x_{k+1/2} = x_k + y_k. \tag{4.3}$$

An algorithm like 4.2 is called in the literature on superiorization "the superiorized version of the basic algorithm 4.1". Classic conditions on the *superiorization* sequence (x_k) that guarantee a converging algorithm after perturbation (perturbation resilience) are discussed in Section 4.2. In Section 4.3 we adopt a classic gradient based strategy to superiorize the basic algorithms from Section 4.1 and suggest an alternative strategy based on proximal points which brings us closer to the optimization algorithms considered in Section 5.

Specifically, we first address the unconstrained least-squares problem and consider:

- a superiorized CG iteration, tagged `GradSupCG`, that uses scaled negative gradients, to potentially decrease the target function R_τ,
- a superiorized CG iteration based on proximal points, tagged `ProxSupCG`,
- a gradient based superiorized Landweber iteration, called `GradSupLW` and
- a proximal-point based superiorized Landweber iteration, called `ProxSupLW`.

Further, to address the nonnegative least-squares problem we consider:

- a gradient based superiorized projected Landweber iteration, called `GradSupProjLW`, and
- a superiorized projected Landweber iteration, tagged `ProxSupProjLW`, that uses proximal points to potentially decrease the target function R_τ.

We will see that the superiorized projected Landweber algorithm will approach a nonnegative least-squares solution too. We can steer however the iterates towards the nonnegative orthant through perturbation by proposing

- a superiorized CG iteration, tagged `ProxCSupCG`, based on proximal points that also includes constraints, and
- a constrained-proximal-point based superiorized Landweber iteration, tagged `ProxCSupLW`.

Table 1 summaries these superiorization approaches explored in this paper.

Method	Basic algorithm	Nonascent direction by	Perturbation resilience
`GradSupCG`	$\mathcal{A}_{\mathrm{CG}}$	S_∇ from Alg. 3	✓, see [ZLH18, Thm. 1]
`GradSupLW`	$\mathcal{A}_{\mathrm{LW}}$	S_∇ from Alg. 3	✓, see [GC18]
`ProxSupCG`	$\mathcal{A}_{\mathrm{CG}}$	$S_{\mathrm{prox}}(x,\beta_k)$ from (4.16)	open
`ProxSupLW`	$\mathcal{A}_{\mathrm{LW}}$	$S_{\mathrm{prox}}(x,\beta_k)$ from (4.16)	follows from [JCJ16, LZZ$^+$19]
`ProxCSupCG`	$\mathcal{A}_{\mathrm{CG}}$	$S_{\mathrm{prox}+}(x,\beta_k)$ from (4.18)	open
`ProxCSupLW`	$\mathcal{A}_{\mathrm{LW}}$	$S_{\mathrm{prox}+}(x,\beta_k)$ from (4.18)	open
`GradSupProjLW`	$\mathcal{A}_{\mathrm{LW}+}$	S_∇ from Alg. 3	✓, see [JCJ16, GC18]
`ProxSupProjLW`	$\mathcal{A}_{\mathrm{LW}+}$	$S_{\mathrm{prox}}(x,\beta_k)$ from (4.16)	follows from [LZZ$^+$19]

TABLE 1. List of superiorized algorithms. The left column show the name tags used for the superiorized algorithms that are numerically evaluated in Section 6. The second column from the left shows the basic algorithm used while the third column indicates our strategy for computing the perturbations. The last column indicates if perturbation resilience is solved or still open.

4.1. **Basic Algorithms.** Regarding the problem to recover x from a linear model represented by $Ax = b$, one could adopt any algorithm for solving the linear feasibility problem

$$\text{find } x \in \begin{cases} H, & \text{without nonnegativity constraints,} \\ H \cap R^n_+, & \text{with nonnegativity constraints,} \end{cases} \tag{4.4a}$$

$$H = \bigcap_{j\in[m]} H_j, \qquad H_j = \{x \in \mathbb{R}^n \colon \langle A_j, x\rangle = b_j\}, \tag{4.4b}$$

as a basic algorithm [BB96, Ceg13], where A_j, $j \in [m]$ denote the rows vectors of A. This would be possible in view of an *underdetermined* linear system with a full rank matrix A, as considered here. However, we only consider the more general case of least-squares, i.e., minimizing the first term of (1.2), which is appropriate, in particular, for noisy measurements b_j, $j \in [m]$, and inconsistent linear systems. Specifically, we choose the conjugate gradient (CG) iteration [Saa03, Sect. 8.3] and the Landweber method [BB96] as basic algorithms. First, we adopt the following CG algorithm that is a slight modification of [ZLH18, Algorithm 8].

Initialization: Set $\mu > 0$ small; choose $x_0 \in \mathbb{R}^n$, $p_0 = A^\top(b - Ax_0) + \mu x_0$, and $h_0 = A^\top A p_0 + \mu p_0$.
Iteration for $k \geq 0$: Compute via Algorithm 1 below

$$(x_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\mathrm{CG}}(x_k, p_k, h_k), \quad \text{for all } k \geq 0. \tag{4.5}$$

The updates of the current iterates in lines 2 and 6 of Algorithm 1 below differ from the corresponding updates in [ZLH18, Algorithm 8], because in our case the CG iteration is applied to the regularized least-squares problem

$$\min \frac{1}{2}\left\| \begin{pmatrix} A \\ \sqrt{\mu} I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|^2, \tag{4.6}$$

with a small parameter $\mu > 0$, in order to obtain a well-defined algorithm for the underdetermined scenario (3.9) considered here. Note that the CG iteration (4.5), as well as [ZLH18, Algorithm 8], differs from the classic CG iteration for least-squares in that the gradient of the least-squares term, see line 2, is evaluated at each current iterate, and not by the computationally more efficient update

$$g_{k+1} = g_k + A^\top A p_k + \mu p_k, \tag{4.7}$$

Algorithm 1: $(x_{\text{new}}, p_{\text{new}}, h_{\text{new}}) \leftarrow \mathcal{A}_{\text{CG}}(x, p, h)$

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

begin
- $g = A^\top(Ax - b) + \mu x$
- $\beta = \langle g, h\rangle / \langle p, h\rangle$
- $p_{\text{new}} = -g + \beta p$
- $h_{\text{new}} = A^\top A p_{\text{new}} + \mu p_{\text{new}}$
- $\gamma = -\langle g, p_{\text{new}}\rangle / \langle p_{\text{new}} h_{\text{new}}\rangle$
- $x_{\text{new}} = x + \gamma p_{\text{new}}$

commonly used, see [Saa03, Sect. 8.3]. As discussed in [ZLH18], by doing so, one obtains an algorithm that performs exactly as CG, but is resilient to perturbations. See also Section 4.2.

The recent work in [ZLH18, HLZH18] also considers the preconditioned CG iteration as a basic algorithm. 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 [Vol14], but even its convergence theory it still open, let alone its perturbation resilience. As an alternative, we include constraints in the CG method via superiorization as detailed in Section 4.3 below.

By contrast, the Landweber method can be easily extended to nonnegativity constraints by simple projections onto the nonnegative orthant. We consider the following Landweber iteration. It makes use of the spectral norm of A, denoted here by $\|A\|$, which has to be computed before hand.

Initialization: Choose $x_0 \in \mathbb{R}^n$.
Iteration for $k \geq 0$**:** Compute by Algorithm 2 below

$$x_{k+1} = \mathcal{A}_{\text{LW}+}(x_k), \quad \text{for all } k \geq 0. \tag{4.8}$$

Algorithm 2: $x_{\text{new}} \leftarrow \mathcal{A}_{\text{LW}+}(x)$

Input: Current iterate x; choose $\gamma_x \in (0, 2/\|A\|^2)$.
Output: Updated iterate x_{new}.

begin
- $g = A^\top(Ax - b)$
- $x_{\text{new}} = \Pi_{\mathbb{R}^n_+}(x - \gamma_x g) = \max(x - \gamma_x g, 0)$

4.2. **Perturbation Resilience.** In the sequel let $\mathcal{A}$ denote either the operator $\mathcal{A}_{\text{CG}}$ or $\mathcal{A}_{\text{LW}+}$ of the basic algorithms 4.6 or 4.8, respectively, that, for any given $x_0 \in \mathbb{R}^n$, generate sequences

$$\mathcal{X}_{\mathcal{A}} := (x_k), \text{ by the iteration } x_{k+1} = \mathcal{A}(x_k), \qquad k \geq 0. \tag{4.9}$$

For a given feasibility problem, denoted by T, a *proximity function* $\mathcal{P}r_T : \mathbb{R}^n \to \mathbb{R}_+$ must be decided upon such that $\mathcal{P}r_T(x)$ measures how incompatible x is with the constraints of T. Given an $\varepsilon > 0$, we say that x

is ε-*compatible* with T if $\mathcal{P}r_T(x) \leq \varepsilon$. We can look, for $\varepsilon > 0$, a given problem T and a chosen proximity function $\mathcal{P}r_T$ at the set $C_\varepsilon \subset \mathbb{R}^n$ of the form

$$C_\varepsilon := \Big\{x \in \mathbb{R}^n \mid \mathcal{P}r_T(x) \leq \varepsilon\Big\}. \tag{4.10}$$

We will refer to such sets in the sequel as "proximity sets" and list some instances related to our scenario.

For example, the sub-level set

$$C_{\varepsilon,1} := \Big\{x \in \mathbb{R}^n \mid \frac{1}{2}\|Ax - b\|^2 \leq \varepsilon\Big\}, \qquad \varepsilon \geq 0, \tag{4.11}$$

contains all ε-*compatible* (i.e., approximate) least-squares solutions up to a given noise level ε. Another example is the sub-level set of approximate regularized least-squares solutions, see (4.6), for some fixed $\mu > 0$,

$$C_{\varepsilon,2} := \Big\{x \in \mathbb{R}^n \mid \frac{1}{2}\|Ax - b\|^2 + \frac{\mu}{2}\|x\|^2 \leq \varepsilon\Big\}, \qquad \varepsilon \geq 0, \tag{4.12}$$

or all such approximate regularized least-squares solutions that are also nonnegative, namely,

$$C^+_{\varepsilon,1} := C_{\varepsilon,1} \cap \mathbb{R}^n_+, \qquad C^+_{\varepsilon,2} := C_{\varepsilon,2} \cap \mathbb{R}^n_+ \tag{4.13}$$

with $\varepsilon > 0$ large enough such that the sets $C^+_{\varepsilon,1}, C^+_{\varepsilon,2}$ are nonempty. Each of the above sets describes the "proximity" with respect to the solution set of the (nonnegative/regularized) least-squares problem considered here in terms of an intrinsic proximity function (defined here by the least-squares term) and an accuracy parameter $\varepsilon > 0$.

The underlying idea of the superiorization method is to strive for a point $x \in C_\varepsilon$, in one of the sets of interest, that is superior with respect to a given target function, by merely applying the basic algorithm to perturbed iterates, recall (4.2). For the superiorized version of the basic algorithm to retain its feasibility-seeking nature to a point in the solution set these perturbations have to be bounded.

There are two research directions in the general area of the superiorization method (SM). One is the direction when only "bounded perturbation resilience" is used and the constraints are assumed to be consistent (having nonempty intersection). Then the "superiorized version" of the original unperturbed basic algorithm is treated as a recursion formula that produces an infinite sequence of iterates, and convergence questions are meant in their asymptotic nature.

The second direction does not assume consistency of the constraints but uses instead a proximity function that "measures" the violation of the constraints. Instead of seeking asymptotic feasibility, it looks at ε-compatibility with the constraints and uses the notion of "strong perturbation resilience", see [HGDC12, Subsection II.C] where this direction has been initiated. The same core "superiorized version" of the original unperturbed algorithm might be investigated in each of these directions, but the second is the more useful one for practical applications, whereas the first makes only asymptotic statements. A recent work on the guarantee question of the SM is solely focused on weak superiorization [CL19]. The terms "weak superiorization" and "strong superiorization" were proposed as a nomenclature for the first and second directions, respectively, in [CZ15, Section 6] and [Cen15].

The following definition is an adaptation of [CDH10, Definition 1] to our notation used here.

Definition 4.1 (**Bounded perturbation resilience**). Let $\mathcal{S}$ denote the solution set of some given task and let $\mathcal{A}$ be an algorithmic operator as in (4.9). The algorithm (4.9) is said to be bounded perturbations resilient if the following holds: If the algorithm (4.9) generates sequences $\mathcal{X}_\mathcal{A}$ that converge to points in $\mathcal{S}$ for all $x_0 \in \mathbb{R}^n$ then any sequence $\mathcal{X}'_\mathcal{A} = (x'_k)$, generated by $x'_{k+1} = \mathcal{A}(x'_k + \beta_k v_k)$ where the vector sequence (v_k) is bounded, $\beta_k \geq 0$ for all $k \geq 0$, and $\sum_{k=0}^\infty \beta_k < +\infty$, also converges to a point in $\mathcal{S}$ for any $x'_0 \in \mathbb{R}^n$.

This definition can be used whenever the constraints which define the set are consistent, i.e., $\mathcal{S} \neq \emptyset$. In that case the superiorized version (4.2) of the original unperturbed basic algorithm (4.1) is treated as a

recursion formula that produces an infinite sequence of iterates, and convergence questions are meant in their asymptotic nature.

The next notion, wherein one does not have to assume consistency of the constraints, uses a proximity function that measures the violation of the constraints. Instead of seeking asymptotic feasibility, it looks at ε-compatibility, see [HGDC12, Subsection II.C], specialized in our case by introducing the sets C_ε (4.11)–(4.13) defined above.

Definition 4.2 (**The ε-output of a sequence**). For an $\varepsilon \in \mathbb{R}_+$, a proximity set C_ε, as in (4.10) above, and a sequence $\mathcal{X} := (x_k)$ of points, we use $O\left(C_\varepsilon, \mathcal{X}\right)$ to denote the x that has the following properties: $x \in C_\varepsilon$, and there is a nonnegative integer N such that $x_N = x$ and, for all nonnegative integers $k < N$, $x_k \notin C_\varepsilon$. Such an x is called the "ε-output of the sequence". If there is such an x, then it is unique. If there is no such x, then we say that $O\left(C_\varepsilon, X\right)$ is *undefined,* otherwise it is *defined*.

If $\mathcal{X}$ is an infinite sequence generated by an algorithm such as (4.9), then $O\left(C_\varepsilon, \mathcal{X}\right)$ is the *output* produced by that algorithm when we add to it a stopping rule that makes it terminate when it reaches a point that is an ε-output of the sequence. With this definition of ε-output of a sequence we define the notion of strong perturbation resilience from [HGDC12, Subsection II.C], adapted to our notations.

Definition 4.3 (**Strong perturbation resilience**). Let C_ε denote the proximity set and let $\mathcal{A}$ be an algorithmic operator as in (4.9). The algorithm (4.9) is said to be strongly perturbation resilient if the following conditions hold:

(1) there is an $\varepsilon > 0$ such that the ε-output of the sequence $\mathcal{X}_\mathcal{A}$ exists for every $x_0 \in \mathbb{R}^n$;
(2) for all $\varepsilon \geq 0$ such that $O\left(C_\varepsilon, \mathcal{X}_\mathcal{A}\right)$ is defined for every $x_0 \in \mathbb{R}^n$ we also have that $O\left(C_{\varepsilon'}, \mathcal{X}'_\mathcal{A}\right)$ is defined for every $\varepsilon' \geq \varepsilon$, and for every sequence

$$\mathcal{X}'_\mathcal{A} = (x'_k), \text{ generated by } x'_{k+1} = \mathcal{A}(x'_k + \beta_k v_k) \text{ for all } k \geq 0,$$

where the sequence (v_k) is bounded, $\beta_k \geq 0$ for all $k \geq 0$, and $\sum_{k=0}^{\infty} \beta_k < +\infty$.

Note that if the constraints are consistent, i.e., $\mathcal{S} \neq \emptyset$, and if $\mathcal{S} \subset C_\varepsilon$, then bounded perturbation resilience implies strong perturbation resilience. Sufficient conditions for strong perturbation resilience appeared in [HGDC12, Theorem 1]. With respect to a target function f, we adopt the convention that a point in the domain of f for which its value is smaller is considered *superior* to a point for which the value of f is larger. The essential idea of the SM is to use the perturbations (4.3) to transform a strongly perturbation resilient basic algorithm that seeks a constraints-compatible solution into a *superiorized version* whose outputs are equally good from the point of view of constraints-compatibility, but are superior (not necessarily optimal) with respect to the target function f. This can be done by using nonascent vectors.

4.3. **Superiorization by Bounded Perturbations.** In this subsection we specify in detail the two superiorized versions of the basic CG iteration (4.5) and the basic Landweber iteration (4.8) Discussed above. We start with CG and use the notation

$$g_\mu(x) := \frac{1}{2}\|Ax - b\|^2 + \frac{\mu}{2}\|x\|^2$$

to denote the objective function of the regularized least-squares problem (4.6). In the sequel g_0 will correspond to the plain least-squares term $g_0 := {}^1\!/_2\|A \cdot -b\|^2$. The next superiorized CG iteration is taken from [ZLH18, Algorithm 7, Algorithm 8] but applied to the regularized least-squres problem (4.6). It uses nonascent directions (3.4) based on negative gradients and fits into the general framework in [HGDC12, page 5537].

`GradSupCG`**:**

Initialization: Set $\varepsilon, \mu > 0$ small; Choose $x_0 \in \mathbb{R}^n$, set $y_0 = x_0$, $p_0 = A^\top(b - Ax_0) + \mu x_0$, $h_0 = A^\top A p_0 + \mu p_0$, $\gamma = \|p_0\|^2/\langle p_0, h_0\rangle$, $x_1 = x_0 + \gamma p_0$, $\ell_1 = 0$

Iteration for $k \geq 1$: Compute

$$\textbf{while} \quad g_\mu(x_{k-1/2}) > \varepsilon \quad \textbf{do}$$

$$(s_k, \ell_{k+1}) = S_\nabla(x_k, \ell_k, a, \gamma_0, \kappa), \text{by Algorithm 3} \tag{4.14a}$$

$$x_{k+1/2} = x_k + s_k \tag{4.14b}$$

$$(x_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\mathrm{CG}}(x_{k+1/2}, p_k, h_k) \tag{4.14c}$$

$$k \leftarrow k + 1 \tag{4.14d}$$

The inputs of the superiorized CG algorithm are the initial vector x_0, and a, κ and γ_0 have the same role as in Algorithm 3, while R_τ and ∇R_τ are as in (3.12b) and (3.13), respectively.

Algorithm 3: $(s, \ell) \leftarrow S_\nabla(x, \ell, a, \gamma_0, \kappa)$

Input: Current iterate x.
Output: Nonascent direction $s := y - x$, exponent ℓ.

begin
  $y \leftarrow x$
  for $i = 1, \ldots, \kappa$ **do**
    if $\nabla R_\tau(y) \neq 0$ **then**
      $d \leftarrow -\nabla R_\tau(y)/\|\nabla R_\tau(y)\|$, see (3.13),
    else
      $d \leftarrow 0$;
    repeat
      $\gamma \leftarrow \gamma_0 a^\ell$
      $y_{\mathrm{new}} \leftarrow y + \gamma d$
      $\ell \leftarrow \ell + 1$;
    until $R_\tau(y_{\mathrm{new}}) \leq R_\tau(y)$.
    $y \leftarrow y_{\mathrm{new}}$.

We consider a second variant of the superiorized CG iteration that employs nonascent directions computed by proximal points. The idea is to replace (4.14a) in Algorithm 3 by a proximal point of x_k with respect to the target function

$$f_0(x) = \lambda R_\tau(x), \tag{4.15}$$

i.e., by

$$S_{\mathrm{prox}}(x_k, \beta_k) := P_{\beta_k} f(x_k) = \arg\min_z \left\{ R_\tau(z) + \frac{1}{2\beta_k}\|z - x_k\|^2 \right\}. \tag{4.16}$$

Alternatively, we can use a proximal point of x_k with respect to the target function f_0 *together* with nonnegative constraints

$$f(x) = \lambda R_\tau(x) + \delta_{\mathbb{R}^n_+}(x) \tag{4.17}$$

by imposing nonnegative constraints in the proximal point computation

$$S_{\mathrm{prox}+}(x_k, \beta_k) := P_{\beta_k} f(x_k) = \arg\min_z \left\{ R_\tau(z) + \delta_{\mathbb{R}^n_+}(z) + \frac{1}{2\beta_k}\|z - x_k\|^2 \right\}. \tag{4.18}$$

Superiorization by proximal points has been done in [LZZ+19], as we became aware while preparing this manuscript. However, the authors in [LZZ+19] do not take into account constraints when computing the perturbations.

Clearly, if $x_{k+1/2} := S_{\text{prox}+}(x_k, \beta_k)$ then

$$f(x_{k+1/2}) \leq f_0(x_{k+1/2}) + \delta_{\mathbb{R}^n_+}(x_{k+1/2}) + 1/2\|x_{k+1/2} - x_k\|^2 \tag{4.19a}$$
$$\leq f_0(x_k) + \delta_{\mathbb{R}^n_+}(x_k) + 1/2\|x_k - x_k\|^2 = f(x_k) \tag{4.19b}$$

and equality holds if and only if $x_{k+1/2} = x_k$, in view of the definition of the proximal mapping. Therefore, the decrease of the target function in the iteration is guaranteed. One might argue that computing such an nonascent direction is expensive and contradicts the spirit of superiorization. The proximal point in (4.16) or (4.18) can be computed efficiently in our case, e.g. by the box-constrained L-BFGS method from [BLNZ95] that computes a highly accurate solution within few iterations as we demonstrate in Section 6. Taking these considerations into account we propose the following superiorized CG iteration.

`ProxCSupCG`**:**
Initialization: Set $\varepsilon, \mu > 0$ small; Choose $x_0 \in \mathbb{R}^n$, $\gamma_0 > 0$ and $a \in (0,1)$; Set $x_{1/2} = x_0$, $p_0 = A^\top(b - Ax_0) + \mu x_0$, $h_0 = A^\top A p_0 + \mu p_0$, $\gamma = \|p_0\|^2/\langle p_0, h_0\rangle$, $x_1 = x_0 + \gamma p_0$ and $\beta_1 = \gamma_0$.
Iteration for $k \geq 1$**:** Compute

while $g_\mu(x_{k-1/2}) > \varepsilon$ **do**
$$x_{k+1/2} = S_{\text{prox}+}(x_k, \beta_k), \quad \text{see (4.18)} \tag{4.20a}$$
$$\beta_{k+1} = \gamma_0 a^k \tag{4.20b}$$
$$(x_{k+1}, p_{k+1}, h_{k+1}) = \mathcal{A}_{\text{CG}}(x_{k+1/2}, p_k, h_k) \tag{4.20c}$$
$$k \leftarrow k + 1 \tag{4.20d}$$

Leaving out the nonnegative constraints by replacing in line (4.20a) above $S_{\text{prox}+}$ by S_{prox} leads to a slighly different iteration that we call `ProxSupCG`. Clearly the sequence (β_k) is summable for $a \in (0,1)$. Despite the proximal mapping being nonexpansive it is not straightforward to show that the iterates $(x_{k+1/2})$ are bounded and that `ProxSupCG` (4.20) converges to a minimzer of g_μ, following the lines of the proof from [LZZ+19], where the basic algorithmic operator $\mathcal{A}$ is iteration independent and nonexpansive. We leave this open problem for future research and turn our attention to the superiorization of the Landweber and projected Landweber iterations.

`ProxCSupLW`**:**
Initialization: Set $\varepsilon > 0$ small; Choose $x_1 \in \mathbb{R}^n$, $\gamma_0 > 0$ and $a \in (0,1)$; Set $x_{1/2} = x_1$ and $\beta_1 = \gamma_0$.
Iteration for $k \geq 1$**:** Compute

while $g_0(x_{k-1/2}) > \varepsilon$ **do**
$$x_{k+1/2} = S_{\text{prox}+}(x_k, \beta_k) \tag{4.21a}$$
$$\beta_{k+1} = \gamma_0 a^k \tag{4.21b}$$
$$x_{k+1} = \mathcal{A}_{\text{LW}}(x_{k+1/2}) \tag{4.21c}$$
$$k \leftarrow k + 1 \tag{4.21d}$$

Combinations between the two basic algorithms (CG and Landweber) and the considered selections of nonascent directions lead to different iterations, that are summarized in Table 1.

Proposition 4.4 below shows the relation of the algorithm `ProxCSupLW` with an optimization algorithm. We point out that the parameter choice assumed below does no longer qualify `ProxCSupLW` as a *superiorized* Landweber iteration, since the sequence $(\beta_k)_{k\geq 0}$ then fails to be summable. However, the modified algorithm converges and is guaranteed to return an optimal solution.

Proposition 4.4. *Let $\gamma \in (0, 2/L_{g_0})$ and $\lambda > 0$. The algorithm obtained by setting $\gamma_0 = \lambda\gamma$ and $a = 1$ in the iteration procedure (4.21) generates a sequence $(x_{k+1/2})$ that converges to a minimizer of*

$$\min \lambda R_\tau(x) + \frac{1}{2}\|Ax - b\|^2 + \delta_{\mathbb{R}^n_+}(x). \tag{4.22}$$

Proof. We will show that the iteration (4.21) and the proximal gradient algorithm, a.k.a. Forward-Backward (FB) iteration, see next section,

$$y_{k+1} = P_\gamma f(y_k - \gamma \nabla g_0(y_k)), \quad y_0 \in \mathbb{R}^n, \tag{4.23}$$

with $f := \lambda R_\tau + \delta_{\mathbb{R}^n_+} \in \mathcal{F}_c$, $g_0 := 1/2\|A \cdot -b\|^2 \in \mathcal{F}^1_c(L)$, are equivalent for an appropriate choice of the parameter β_k. The iterates (y_k) in (4.23) above are converging to the minimizer of $f + g_0$, the objective from (1.2), see [CP11b], provided

$$0 < \gamma < \frac{2}{L_{g_0}} \tag{4.24}$$

holds, where L_{g_0} denotes the global Lipschitz constant of the gradient of least-squares term g_0 and that can be estimated by

$$L_{g_0} \leq \|A\|^2. \tag{4.25}$$

Further, the Fermat condition for problem (4.22) reads

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

As done in [LZZ+19, Thm. 3.8], we can introduce an auxiliary variable and write

$$0 \in \lambda \nabla R_\tau(y) + \frac{1}{\gamma}(y - x) + \partial\delta_{\mathbb{R}^n_+}(y) \tag{4.26a}$$

$$x = y - \gamma A^\top(Ay - b) \tag{4.26b}$$

Using that (4.26a) is equivalent to

$$y = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(x) = \arg\min_z \left\{ R_\tau(z) + \delta_{\mathbb{R}^n_+}(z) + \frac{1}{2\lambda\gamma}\|z - x\|^2 \right\}$$

we can recast (4.26) as the following fix-point iteration

$$y_k = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(x_k) \tag{4.27a}$$

$$x_{k+1} = y_k - \gamma A^\top(Ay_k - b). \tag{4.27b}$$

Now $x_{k+1/2} := y_k$ and $\beta_k := \lambda\gamma$ this coincides with algorithm `ProxCSupLW` (4.21) as it can be rewritten as

$$x_{k+1/2} = S_{\mathrm{LW}+}(x_k, \beta_k) \tag{4.28}$$

$$\beta_{k+1} = \beta_k a \tag{4.29}$$

$$x_{k+1} = \mathcal{A}_{\mathrm{LW}}(x_{k+1/2}) \tag{4.30}$$

for $a = 1$. On the other hand, the iteration (4.27) can be summarized by

$$y_k = P_{\lambda\gamma}(R_\tau + \delta_{\mathbb{R}^n_+})(y_{k-1} - \gamma A^\top(Ay_{k-1} - b)), \quad k \geq 1$$

that is the forward backward iteration (4.23). Now (y_k) converges to a minimizer of (4.22). □

5. Optimization

We examine two particular decompositions of problem (1.2) in Sections 5.1 and 5.4 and corresponding forward-backward (FB) approaches. We do so in order to obtain iterative optimization schemes that are similar to the superiorization approach of Section 4, see Remarks 5.2 and 5.4 below. Plain, accelerated and inexact versions of the FB-iteration are considered next. Corresponding notions of inexactness provide concrete criteria for terminating the inner-loop iterations in Section 5.3. Inexact versions are only considered for the first decomposition from Section 5.1.

5.1. **Problem Splitting, Proximal Map.** Regarding the objective (1.2), we define functions $f_0 \in \mathcal{F}_c^1(L_{f_0})$, $g, h \in \mathcal{F}_c$ and $g_0 \in \mathcal{F}_c^1(L_{g_0})$ by

$$h(x) := f_0(x) + g(x), \tag{5.1a}$$

$$f_0(x) := \lambda R_\tau(x), \qquad L_{f_0} = \lambda L_{R_\tau}, \tag{5.1b}$$

$$g(x) := g_0(x) + \delta_K(x) = \frac{1}{2}\|Ax - b\|^2 + \delta_K(x). \tag{5.1c}$$

Setting

$$B_\alpha := A^\top A + \frac{1}{\alpha} I_n, \qquad c_\alpha := \frac{1}{\alpha} x + A^\top b, \tag{5.2}$$

and

$$\phi(y) := \phi_0(y) + \delta_K(y), \qquad \phi_0(y) := \frac{1}{2}\langle y, B_\alpha y\rangle - \langle c_\alpha, y\rangle, \qquad \phi_0 \in \mathcal{F}_c^1\Big(\|A\|^2 + \frac{1}{\alpha}\Big), \tag{5.3}$$

the proximal map with respect to g reads

$$P_\alpha g(x) = \overline{y} = \arg\min_y \Big\{ \frac{1}{2\alpha}\|y - x\|^2 + \frac{1}{2}\|Ay - b\|^2 + \delta_K(y) \Big\} \tag{5.4a}$$

$$= \arg\min_y \phi(y) + \text{const.} \tag{5.4b}$$

The unique proximal point $\overline{y}$ of x with respect to g is characterized by

$$N_K(\overline{y}) \ni -\nabla\phi_0(\overline{y}) = c_\alpha - B_\alpha \overline{y} \tag{5.5a}$$

$$0 \geq \big\langle c_\alpha - B_\alpha \overline{y}, z - \overline{y}\big\rangle = \big\langle \frac{1}{\alpha}(x - \overline{y}) - \nabla g_0(\overline{y}), z - \overline{y}\big\rangle, \qquad \forall z \in K \tag{5.5b}$$

and satisfies the fixed point equation

$$\overline{y} = \Pi_K\big(\overline{y} + \beta(c_\alpha - B_\alpha\overline{y})\big) = \Pi_K\Big(\overline{y} + \beta\Big(\frac{1}{\alpha}(x - \overline{y}) - \nabla g_0(\overline{y})\Big)\Big), \qquad \beta > 0. \tag{5.6}$$

We next compute the duality gap for a feasible point $z \in K$ with respect to the convex optimization problem (5.4) defining $\overline{y} = P_\alpha g(x)$.

Lemma 5.1. *Let $z \in K$ and define the dual feasible point*

$$p = (c_\alpha - B_\alpha z)_-. \tag{5.7}$$

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

$$\mathrm{dgp}(z) = \frac{1}{2}\|(c_\alpha - B_\alpha z)_+\|^2_{B_\alpha^{-1}} - \langle p, z\rangle \geq 0. \tag{5.8}$$

Proof. $\overline{y} = P_\alpha g(x)$ is the unique solution to problem (5.4b). Denoting by $p \in K^*$ the multiplier vector corresponding to the constraint $z \in K$, the dual problem reads

$$\max_p \psi(p), \qquad \psi(p) = -\frac{1}{2}\langle c_\alpha - p, B_\alpha^{-1}(c_\alpha - p)\rangle - \delta_{K^*}(p). \tag{5.9}$$

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

$$\overline{p} = c_\alpha - B_\alpha \overline{y}. \tag{5.10}$$

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

$$p = p(z) = (c_\alpha - B_\alpha z)_- = c_\alpha - B_\alpha z - (c_\alpha - B_\alpha z)_+ \tag{5.11}$$

yields the duality gap

$$\operatorname{dgp}(z) = \phi(z) - \psi\big(p(z)\big) \geq 0 \tag{5.12}$$

which, after rearranging, becomes (5.8). □

We observe that if $z = \overline{y}$, then by (5.5a) the corresponding dual vector (5.10) is nonpositive and orthogonal to $\overline{y}$. As a consequence, the duality gap (5.8) is zero: $\operatorname{dgp}(\overline{y}) = 0$.

The *Forward-Backward (FB)* iteration generates the sequence (x_k), converging to a minimizer of the objective function h (5.1a), by the process

$$x_{k+1} = P_{\alpha_k} g\big(x_k - \alpha_k \nabla f_0(x_k)\big), \quad k \geq 0, \quad x_0 \in \operatorname{dom} g, \tag{5.13}$$

for a suitable sequence (α_k) of proximal parameters. A common choice is a constant parameter value

$$\alpha_k = \alpha \in \Big(0, \frac{2}{L_{f_0}}\Big). \tag{5.14}$$

A refined scheme that admits inexact evaluations of the proximal map $P_\alpha g$ is considered in Section 5.2.

Remark 5.2 (Relation to superiorization). The FB iteration (5.13) resembles the structure of iteration used in superiorization: Computing iteratively the proximal points $P_{\alpha_k} g(x_k)$ of the iterates x_k can be viewed as basic algorithm that is perturbed by the gradient of the target function $\nabla f_0(x_k)$. Inspecting (5.4) shows that this basic algorithm involves the nonnegative least-squares problem subject to proximal regularization. Choosing a suitable sequence (α_k) of the proximal parameter guarantees to generate a convergent sequence (x_k) minimizing (1.2), which includes the target function.

On the downside, if the proximal map $P_{\alpha_k} g$ cannot be computed in closed-form, then an inner iteration has to be carried out before the next perturbation can affect the primal variable. By contrast, such intermediate iterative computations are not part of the superiorization approach. Superiorization does not provide any quantitative guarantee with respect to the target function, however.

5.2. **Outer Loop: Accelerated Inexact Forward-Backward Iteration.** We work out versions of the FB iteration (5.13) based on [CW05] and [VSBV13] in order to accelerate the iteration (5.13) and to take into account inexact evaluations of the proximal map.

5.2.1. *Summable Error Sequences.* Let error vectors

$$e_k = x_{k+1} - P_{\alpha_k} g\big(x_k - \alpha_k \nabla f_0(x_k)\big) \tag{5.15}$$

represent inexact evaluations of the right-hand side of (5.13). Then the FB iteration (5.13) with parameter choice (5.14) converges if [CW05, Theorem 3.4]

$$\sum_{k \in \mathbb{N}} \|e_k\| < +\infty. \tag{5.16}$$

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

$$z \approxeq_\varepsilon P_\alpha g(x), \quad \varepsilon > 0 \tag{5.17a}$$

if

$$\frac{1}{\alpha}(x - z) \in \partial_{\frac{\varepsilon^2}{2\alpha}} g(z), \tag{5.17b}$$

where the right-hand side is defined by the ε-subdifferential at z [Roc70, Section 23]

$$\partial_\varepsilon g(z) = \{p \in \mathbb{R}^n \mid g(y) \geq g(z) + \langle p, y - z\rangle - \varepsilon\}, \qquad \forall y \in \mathbb{R}^n \tag{5.18a}$$

$$= \{p \in \mathbb{R}^n \mid g(z) + g^*(p) - \langle z, p\rangle \leq \varepsilon\}. \tag{5.18b}$$

Taking into account the specific form $g = g_0 + \delta_K$ of g as defined by (5.1c), the following lemma parametrizes $p \in \partial_\varepsilon g(z)$ by points $z_p \in \mathbb{R}^n$ through conjugation.

Lemma 5.3. *Let $g = g_0 + \delta_K$ with $g_0 \in \mathcal{F}_c^1(L_{g_0})$. Then finding a vector $p \in \partial_\varepsilon g(z)$ is equivalent to finding a pair of vectors $z_p, w \in \mathbb{R}^n$ such that p given by*

$$p = \nabla g_0(z_p) - w, \quad \textit{and} \quad 0 \leq w \perp z_p \geq 0 \tag{5.19}$$

satisfy

$$g_0(z) - g_0(z_p) - \langle \nabla g_0(z_p), z - z_p\rangle + \langle w, z\rangle \leq \varepsilon. \tag{5.20}$$

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

Proof. Writing

$$g^*(p) = \sup_z\{\langle p, z\rangle - g_0(z) - \delta_K(z)\} = -\inf_z\{g_0(z) - \langle p, z\rangle + \delta_K(z)\}, \tag{5.21}$$

Fermat's optimality condition for the latter minimization problem reads $p \in \nabla g_0(z_p) + N_K(z_p)$, which is equivalent to (5.19). Substituting p and taking into account $w \perp z_p \in K$ gives $g^*(p) = \langle \nabla g_0(z_p), z_p\rangle - g_0(z_p)$ and after insertion into (5.18b) Equation (5.20) is obtained. □

We next consider a generalization of the FB iteration (5.13) that includes both acceleration and inexact evaluations of the proximal mapping.

5.2.3. *Algorithm.* We adopt the following algorithm proposed and analyzed by [VSBV13].

Initialization: Choose $x_0 = y_0 \in \operatorname{dom} g$, $t_0 > 1$, $a \in (0, 2)$ and sequences $(\alpha_k) \subset (0, \frac{(2-a)}{L}]$, $(a_k) \subset [a, 2 - \alpha_k L_f]$, $(\varepsilon_k) \subset \mathbb{R}_+$.

Iteration for $k \geq 0$: Compute

$$x_{k+1} \approxeq_{\varepsilon_k} P_{\alpha_k} g\big(y_k - \alpha_k \nabla f_0(y_k)\big), \tag{5.22a}$$

$$t_{k+1} = \frac{1}{2}\left(1 + \left(1 + 4\frac{a_k \alpha_k}{a_{k+1}\alpha_{k+1}} t_k^2\right)^{1/2}\right), \tag{5.22b}$$

$$y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k) + (1 - a_k)\frac{t_k}{t_{k+1}}(y_k - x_{k+1}). \tag{5.22c}$$

Unlike the FB iteration (5.13), inexact evaluations of the proximal map are admissible at step (5.22a). In addition, an auxiliary sequence of vectors (y_k) and corresponding parameters (t_k) are used for acceleration. Theorem 4.4 in [VSBV13] assure an $\mathcal{O}(1/k^2)$ convergence rate of the function values $((g + f)(x_k))$ to its infimum provided the error parameter sequence ε_k corresponding to the steps (5.22a) decays at rate

$$\varepsilon_k = \mathcal{O}\Big(\frac{1}{k^{3/2+\delta}}\Big), \quad \delta > 0. \tag{5.23}$$

We next examine the conditions provided by Lemma 5.3 for the inexact evaluation (5.22a), for both the unconstrained and the nonnegativity constrained case. Recall the simplified notation (3.8) for any fixed outer iteration index k. We also write ε instead of ε_k.

5.2.4. *Recognizing $z \approx_\varepsilon P_\alpha g_0(x)$: The Unconstrained Case.* Let $K = \mathbb{R}^n$. Applying Lemma 5.3 shows that condition (5.17b) for recognizing $z \approx_\varepsilon P_\alpha g(x)$ can be expressed as finding another point z_p such that

$$\frac{1}{\alpha}(x - z) = \nabla g_0(z_p) \qquad \text{and} \tag{5.24a}$$

$$\frac{\varepsilon^2}{2\alpha} \geq g_0(z) - g_0(z_p) - \langle \nabla g_0(z_p), z - z_p \rangle. \tag{5.24b}$$

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 (5.24b) hold trivially, whereas condition (5.24a) then reads $\nabla \phi_0(z) = 0$ which implies $z = P_\alpha g(x)$ by (5.4) if $K = \mathbb{R}^n$.

If $z \neq z_p$, then (5.24a) says that neither point is equal to $P_\alpha g(x)$, whereas the right-hand side of condition (5.24b) is always nonnegative and measures the difference between z and z_p by a Bregman-like distance induced by g_0 (it is not a true Bregman distance since g_0 merely is convex). The decomposition (5.24a) of the optimality condition determining $P_\alpha g(x)$ in terms of (z, z_p) enables to terminate earlier any iterative algorithm that converges to $P_\alpha g(x)$, once the level of inexactness (5.24b) is reached.

We conclude this section by comparing (5.24) with (5.15) which – using the simplified notation – reads

$$e = z - \overline{y}. \tag{5.25}$$

Since ϕ_0 given by (5.3) is $\frac{1}{\alpha}$-strongly convex [RW09, Definition 12.58] and $\nabla \phi_0(\overline{y}) = 0$, we have the inequality

$$\|e\| = \|z - \overline{y}\| \leq 2\alpha \|\nabla \phi_0(z)\| = 2\|z - x - \alpha \nabla g_0(z)\|, \tag{5.26}$$

which evaluates violations of Equation (5.24a), for a single point z on both sides. multiplied by 2, and with the different meaning (5.16) of errors $\varepsilon = \|e\|$.

5.2.5. *Recognizing $z \approx_\varepsilon P_\alpha g(x)$: The Nonnegatively Constrained Case.* Let $K = \mathbb{R}^n_+$. In view of (5.17b), we identify $p = \frac{1}{\alpha}(x - z)$ in (5.19). Lemma 5.3 then shows that $z \approx_\varepsilon P_\alpha g$ holds if there is another point z_p such that

$$\frac{1}{\alpha}(x - z) - \nabla g_0(z_p) = -w \quad \text{and} \quad 0 \leq w \perp z_p \geq 0, \quad z \in K, \tag{5.27a}$$

$$\frac{\varepsilon^2}{2\alpha} \geq g_0(z) - g_0(z_p) - \langle \nabla g_0(z_p), z - z_p \rangle + \langle w, z \rangle. \tag{5.27b}$$

The discussion above of (5.24) applies here analogously. In particular, if $z = z_p$, then (5.27a) reads

$$\frac{1}{\alpha}(x - z) - \nabla g_0(z) = c_\alpha - B_\alpha z \leq 0, \tag{5.28}$$

which is the optimality condition (5.5a) that uniquely determines the proximal point $z = \overline{y} = P_\alpha g(x)$.

We conclude this section by comparing (5.27) with (5.15) and consider again the error vector (5.25). In addition, we take into account the nonnegativity constraint $z \in K$. Since $\phi(y) - \frac{1}{2\alpha}\|y - x\|^2 = g_0(x)$ is convex, ϕ is $\frac{1}{\alpha}$-strongly convex. Exploiting weak duality (5.12), in particular $\phi(\overline{y}) \geq \psi(p)$, $\forall p \in K^*$, we estimate

$$\|e\|^2 = \|z - \overline{y}\|^2 \leq 2\alpha\big(\phi(z) - \phi(\overline{y})\big) \leq 2\alpha\big(\phi(z) - \psi\big(p(z)\big)\big) = 2\alpha \operatorname{dgp}(z), \quad z \in K, \tag{5.29}$$

with $p(z)$ given by (5.11). The explicit expression (5.8) shows how $\operatorname{dgp}(z)$ evaluates the non-optimality of the dual point $p(z)$. Formally, this expression defining $p(z)$ also appears as left-hand side of (5.27a), but its non-optimality results from $z \neq z_p$.

5.3. **Inner Loop: Accelerated Primal-Dual Iteration.** We apply the primal-dual optimization approach [CP11a] to two different decompositions of the optimization problem related to step (5.22a) of the outer loop. The first decomposition involves the inversion of the matrix B_α defined by (5.2). The second alternative decomposition avoids this inversion. The overall efficiency depends on two aspects: more sophisticated iterative steps may converge faster but are also more expensive computationally.

The approach [CP11a] applies to optimization problems written as a saddle-point problem

$$\min_{x\in X}\max_{y\in Y}\big\{\langle Mx,y\rangle + G(x) - F^*(y)\big\}, \tag{5.30}$$

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.

Using definitions the (5.3), we consider problem (5.4b) and two different problem decompositions conforming to (5.30). Each corresponding primal-dual iteration generates a sequence (z_l) minimizing ϕ that is terminated when z_{l+1} satifies the inexactness criterion (5.24) or (5.27),where z plays the role of z_{l+1}. This requires to identify the point z_p during the inner-loop primal-dual iteration.

In this section, we adopt the simplified notation (3.8), namely, $x = y_k - \alpha_k\nabla f_0(y_k)$ denotes the argument of step (5.22a) or $x = x_k - \alpha_k\nabla f(x_k)$ in case of the FB iteration (5.13). This vector x in turn defines the vector c_{α_k} by (5.2), simply denoted by c_α. The vector $z \approx \overline{y}$ denotes an approximation of the proximal point $P_\alpha g(x)$.

5.3.1. *Basic Decomposition.* Rewriting the inner problem in the form (5.30),

$$\min_{y\in\mathbb{R}^n}\max_{p\in\mathbb{R}^n}\Big\{\langle y,p\rangle + \phi_0(y) - \delta_{K^*}(p)\Big\}, \tag{5.31}$$

and noting that $\phi_0 \in \mathcal{F}_c^1(L_{\phi_0},\mu)$ is strongly convex with constant $\mu = \frac{1}{\alpha}$, the application of Algorithm 2 of [CP11a] yields the following iteration.

Initialization: Choose $\tau_0, \sigma_0 > 0$ with $\tau_0\sigma_0 \le 1$, z_0, p_0 arbitrary and $\overline{z}_0 = z_0$.
Iteration for $l \ge 1$: Compute

$$p_{l+1} = (p_l + \sigma_l\overline{z}_l)_- \tag{5.32a}$$
$$z_{l+1} = \big(I_n + \tau_l B_\alpha\big)^{-1}\big(z_l - \tau_l(p_{l+1} - c_\alpha)\big) \tag{5.32b}$$
$$\theta_l = \Big(1 + 2\frac{\tau_l}{\alpha}\Big)^{-1/2}, \quad \tau_{l+1} = \theta_l\tau_l, \quad \sigma_{l+1} = \frac{\sigma_l}{\theta_l} \tag{5.32c}$$
$$\overline{z}_{l+1} = z_{l+1} + \theta_l(z_{l+1} - z_l). \tag{5.32d}$$

Regarding step (5.32b), we note that the inversion of the matrix

$$I_n + \tau_l B_\alpha = \Big(1 + \frac{\tau_l}{\alpha}\Big)I_n + \tau_l A^\top A \tag{5.33}$$

can be simplified using the Sherman-Morrison-Woodbury formula [Hig08]

$$(B + UV^\top)^{-1} = B^{-1} - B^{-1}U(I + V^\top B^{-1}U)^{-1}V^\top B^{-1}, \tag{5.34}$$

to obtain

$$(I_n + \tau_l B_\alpha)^{-1} = \frac{\alpha}{\alpha+\tau_l}\left(I_n - \frac{\alpha}{\alpha+\tau_l}A^\top\Big(\frac{1}{\tau_l}I_m + \frac{\alpha}{\alpha+\tau_l}AA^\top\Big)^{-1}A\right). \tag{5.35}$$

Thus, to do the $n \times n$ matrix inversion on the left-hand side by evaluating the right-hand side, the inversion of an $m \times m$ matrix is only required, which can be considerably less expensive due to assumption (3.9).

5.3.2. *Avoiding Matrix Inversion.* If matrix inversion in step (5.32b) is too expensive (e.g., by a direct method), then it can be avoided by rewriting the objective function (5.4b) in the form

$$\phi(y) = \frac{1}{2}\|Ay\|^2 + \frac{1}{2\alpha}\|y\|^2 - \langle c_\alpha, y\rangle + \delta_K(y), \tag{5.36}$$

with K either $\mathbb{R}^n$ or $\mathbb{R}^n_+$ and by dualizing the term comprising A, using an additional dual variable q. Due to assumption (3.9), this does not affect strong convexity with respect to the primal variable y. The corresponding saddle-point problem then reads

$$\min_y \max_q \Big\{ \langle Ay, q\rangle + \frac{1}{2\alpha}\|y\|^2 - \langle c_\alpha, y\rangle + \delta_K(y) - \frac{1}{2}\|q\|^2 \Big\} \tag{5.37}$$

and yields the following iteration.

Initialization: Choose $\tau_0, \sigma_0 > 0$ with $\tau_0\sigma_0 \le 1/\|A\|^2$, z_0, q_0 arbitrary and $\overline{z}_0 = z_0$.

Iteration for $l \ge 1$: Compute

$$q_{l+1} = \frac{1}{1+\sigma_l}(q_l + \sigma_l A\overline{z}_l) \tag{5.38a}$$

$$z_{l+1} = \Pi_K\Big(\frac{\alpha}{\alpha+\tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big)\Big) \tag{5.38b}$$

$$\theta_l = \Big(1 + 2\frac{\tau_l}{\alpha}\Big)^{-1/2}, \quad \tau_{l+1} = \theta_l\tau_l, \quad \sigma_{l+1} = \frac{\sigma_l}{\theta_l} \tag{5.38c}$$

$$\overline{z}_{l+1} = z_{l+1} + \theta_l(z_{l+1} - z_l). \tag{5.38d}$$

Note that this algorithm only requires matrix-vector multiplications with respect to A and $A^\top$.

5.3.3. *Termination: The Unconstrained Case.* If $K = \mathbb{R}^n$, then $P_\alpha g(x)$ uniquely minimizes ϕ_0 and any algorithm solving

$$\nabla\phi_0(\overline{y}) = B_\alpha\overline{y} - c_\alpha = 0 \quad \Leftrightarrow \quad \overline{y} = B_\alpha^{-1}c_\alpha \tag{5.39}$$

is exact: $\overline{y} = P_\alpha g(x)$. 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, even when a formula analogous to (5.35) is used to reduce the problem size for matrix inversion from n to $m < n$. Then we would like to know when *any iterative method* generating a sequence (z_l) converging to $\overline{y}$ can be terminated once $z_l \approxeq_\varepsilon P_\alpha g(x)$ holds.

Let z_{l+1} be a point such that

$$\nabla\phi_0(z_{l+1}) = \nabla g_0(z_{l+1}) - \frac{1}{\alpha}(x - z_{l+1}) \approx 0. \tag{5.40}$$

In order to check if z_{l+1} is sufficiently exact according to the conditions (5.24), we set

$$z_p = z_{l+1}, \qquad z = x - \alpha\nabla g_0(z_p). \tag{5.41}$$

Then condition (5.24a) holds and we can evaluate the right-hand side of (5.24b) in order to check if $z \approxeq_\varepsilon P_\alpha g(x)$.

As an illustration, we now consider the specific iterative algorithm (5.38) applied *without* the projection Π_K since here we assume $K = \mathbb{R}^n$. The second step (5.38b) reads

$$0 = \frac{\alpha}{\alpha+\tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big) - z_{l+1} = -\frac{\tau_l+\alpha}{\alpha\tau_l}z_{l+1} + \frac{1}{\tau_l}z_l - A^\top q_{l+1} + c_\alpha \tag{5.42a}$$

$$= -\frac{1}{\alpha}\Big(z_{l+1} + \frac{\alpha}{\tau_l}(z_{l+1} - z_l)\Big) - A^\top q_{l+1} + \frac{1}{\alpha}x + A^\top b \tag{5.42b}$$

$$= \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l) - \alpha A^\top(q_{l+1} - Az_{l+1})\Big) - A^\top(Az_{l+1} - b). \tag{5.42c}$$

The choice

$$z_p = z_{l+1}, \qquad z := z_p - \frac{\alpha}{\tau_l}(z_p - z_l) - \alpha A^\top (q_{l+1} - A z_p), \tag{5.43}$$

makes equation (5.42c) equal to the second equation of (5.41).

A minor disadvantage of (5.41) is the additional evaluation of the gradient $\nabla g_0(z_p)$. In the present unconstrained case, this can be avoided as follows. Since by duality $q_l \to A\overline{y}$, the expression $A^\top(q_{l+1} - b) \approx \nabla g_0(\overline{y})$ approximates the gradient. Hence, rearranging equation (5.42b),

$$0 = \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l)\Big) - A^\top(q_{l+1} - b), \tag{5.44}$$

suggests the choice

$$A z_p = q_{l+1}, \qquad z := z_{l+1} + \frac{\alpha}{\tau_l}(z_{l+1} - z_l). \tag{5.45}$$

Then $A^\top(q_{l+1} - b) = \nabla g_0(z_p)$ and condition (5.24a) is satisfied. Even though (5.45) does not yield z_p explicitly, condition (5.24b) can be efficiently checked as well,

$$\begin{aligned}
\frac{\varepsilon^2}{2\alpha} &\geq \frac{1}{2}\|Az - b\|^2 - \frac{1}{2}\|Az_p - b\|^2 - \langle Az_p - b, Az - Az_p\rangle && \text{(5.46a)}\\
&= \frac{1}{2}\|Az - b\|^2 + \frac{1}{2}\|Az_p - b\|^2 - \langle Az_p - b, Az - Az_p + Az_p - b\rangle && \text{(5.46b)}\\
&= \frac{1}{2}\big\|(Az - b) - (Az_p - b)\big\|^2 && \text{(5.46c)}\\
&= \frac{1}{2}\|Az - q_{l+1}\|^2. && \text{(5.46d)}
\end{aligned}$$

Checking these conditions during the primal-dual iteration only requires negligible additional computation. Definition (5.45) of z shows that the single additional vector-matrix multiplication Az is implicitly done by evaluating the primal-dual steps (5.38a) and (5.38d).

We conclude by comparing to the alternative error measure (5.26) which for the choice (5.41) translates to

$$\varepsilon = \|e\| = \|z_p - z\|, \tag{5.47}$$

and condition (5.16) for summing up these inner-loop errors at the outer iteration level. Conversely, the alternative choice (5.45) inserted into (5.46d) gives the error

$$\varepsilon \geq \alpha^{1/2}\|A(z - z_p)\| \tag{5.48}$$

in terms of vectors of smaller dimension (cf. (3.9)) and with a factor $\alpha^{1/2} \ll 1$ that typically is much smaller than 1. These errors not only have to be summable but should decay with rate (5.23) to achieve fast convergence.

5.3.4. *Termination: The Nonnegatively Constrained Case.* Let $K = \mathbb{R}^n_+$. We derive criteria for terminating the inner-loop iteration (5.38) based on the conditions (5.27). Algorithm (5.32) does not comprise a projection onto K and hence does not conform to condition (5.27a). We, therefore, only consider early termination of algorithm (5.38).

We rewrite step (5.38b) as

$$\begin{aligned}
& N_K(z_{l+1}) \ni \frac{\alpha}{\alpha + \tau_l}\big(z_l - \tau_l(A^\top q_{l+1} - c_\alpha)\big) - z_{l+1} && \text{(5.49a)}\\
\Leftrightarrow\quad & N_K(z_{l+1}) \ni -\frac{\alpha + \tau_l}{\alpha\tau_l} z_{l+1} + \frac{1}{\tau_l} z_l - A^\top q_{l+1} + \frac{1}{\alpha}x + A^\top b && \text{(5.49b)}\\
& \qquad = \frac{1}{\alpha}\Big(x - z_{l+1} - \frac{\alpha}{\tau_l}(z_{l+1} - z_l) - \alpha A^\top(q_{l+1} - A z_{l+1})\Big) - A^\top(A z_{l+1} - b). && \text{(5.49c)}
\end{aligned}$$

Setting

$$z_p = z_{l+1}, \qquad z = z_{l+1} + \frac{\alpha}{\tau_l}(z_{l+1} - z_l) + \alpha A^\top (q_{l+1} - A z_{l+1}), \tag{5.50a}$$

$$w = A^\top (A z_{l+1} - b) - \frac{1}{\alpha}(x - z) \tag{5.50b}$$

satisfies condition (5.27) *except* for the condition $z \in K$. Clearly, z given by (5.50) becomes nonnegative as $l \to \infty$, and once this happens for some l we can check $z \approxeq_\varepsilon P_\alpha g(x)$ by evaluating condition (5.27b). Similarly to (5.46), we get

$$\frac{\varepsilon^2}{2\alpha} \geq \frac{1}{2}\|Az - b\|^2 - \frac{1}{2}\|Az_p - b\|^2 - \langle Az_p - b, Az - Az_p\rangle + \langle w, z\rangle \tag{5.51a}$$

$$= \frac{1}{2}\|A(z - z_p)\|^2 + \langle w, z\rangle, \tag{5.51b}$$

which, in view of (5.50), evaluates inexactness of z through a distance and a complementarity term with respect to z_p.

$$A^\top q_{l+1} - c_\alpha = \frac{1}{1+\sigma_l}(A^\top q_l + \sigma_l A^\top A(z_l + \theta_l(z_l - z_{l-1}))) - c_\alpha \tag{5.52a}$$

$$= \frac{\sigma_l(1+\theta_l)}{1+\sigma_l} A^\top A z_l - c_\alpha + \frac{1}{\sigma_l} A^\top q_l - \frac{\sigma_l + \theta_l}{1+\sigma_l} A^\top A z_{l-1} \tag{5.52b}$$

A weakness of criterion (5.51) is that it cannot be applied if $z \notin K$, with z given by (5.50a). As an alternative, we evaluate the error measure (5.29). The duality gap $\mathrm{dgp}(z)$ (5.8) requires to compute the vector

$$c_\alpha - B_\alpha z = c_\alpha - A^\top A z - \frac{1}{\alpha} z. \tag{5.53}$$

Due to the primal-dual iteration (5.38), we have

$$A^\top q_{l+1} - c_\alpha = \frac{\sigma_l(1+\theta_l)}{1+\sigma_l} A^\top A z_l - c_\alpha + \frac{1}{\sigma_l} A^\top q_l - \frac{\sigma_l + \theta_l}{1+\sigma_l} A^\top A z_{l-1}. \tag{5.54}$$

Thus, recording the vectors $A^\top(Az_l)$ and $A^\top q_l$ during the primal-dual iteration, the vector (5.53) with z_l in place of z can be efficiently computed at each iteration l. In order to avoid the expensive evaluation of the first term on the right-hand side of (5.8), we estimate

$$\mathrm{dgp}(z_l) \leq \frac{1}{2}\|B_\alpha^{-1}\|\|(c_\alpha - B_\alpha z_l)_+\|^2 - \langle (c_\alpha - B_\alpha z_l)_-, z_l\rangle \tag{5.55}$$

and hence obtain with $\|B_\alpha^{-1}\| \leq \alpha$ and (5.29)

$$\|e\| \leq \alpha\Big(\|(c_\alpha - B_\alpha z_l)_+\|^2 - \frac{2}{\alpha}\langle (c_\alpha - B_\alpha z_l)_-, z_l\rangle\Big)^{1/2}. \tag{5.56}$$

Fixing an error $\varepsilon_k = \|e_k\| = \|e\|$ so as to satisfy (5.16), the inner loop can be terminated once the right-hand side of (5.56) drops below ε_k.

5.4. **Reverse Problem Splitting, Proximal Map.** Having in mind the similarity between the superiorized Landweber iteration (4.21) and the FB iteration, see Proposition 4.4, we consider here the reverse splitting for the objective (1.2). We recall the functions $f, h \in \mathcal{F}_c$ and $g_0 \in \mathcal{F}_c^1(L_{g_0})$ defined by

$$h(x) := f(x) + g_0(x), \tag{5.57a}$$

$$f(x) := \lambda R_\tau(x) + \delta_K(x), \tag{5.57b}$$

$$g_0(x) := \frac{1}{2}\|Ax - b\|^2, \qquad L_{g_0} \leq \|A\|^2. \tag{5.57c}$$

For the reverse splitting the FB iteration becomes

$$x_{k+1} = P_{\alpha_k} f\big(x_k - \alpha_k A^\top (Ax_k - b)\big) \tag{5.58a}$$
$$= P_{\alpha_k}(\lambda R_\tau + \delta_{\mathbb{R}^n_+})\big(x_k - \alpha_k \nabla g_0(x_k)\big), \quad k \geq 0, \quad x_0 \in \operatorname{dom} f, \tag{5.58b}$$

where (α_k) is a sequence of proximal parameters. As already discussed, a constant parameter value $\alpha_k = \alpha \in \left(0, 2/L_{g_0}\right)$ is a permissible and common choice. However, it turns out that for the considered problem these step-size parameters become very small due to the bad conditioning of the matrix A, see Section 6. The FB generates sequence (x_k) above is also converging to a minimizer of the objective function h defined in (1.2). As in Section 4 we assume that the proximal points $P_\alpha f$ can be computed efficiently. Hence we do not pursue a refined scheme that allows for inexact evaluations of the proximal map $P_\alpha f$ further and use the highly accurate box-constrained L-BFGS method from [BLNZ95] to compute the proximal points.

Remark 5.4 (**Relation to superiorization**)**.** Focusing solely on the structure of the iterative processes involved in the algorithms, we can notice similarities between the FB and the reverse splitting FB iteration of (5.13) and (5.58), respectively, and the structure of the iterations in superiorization. In (5.13) one would naturally identify the inner iteration with "perturbations" and the outer iteration of $P_{\alpha_k} g$ with a "basic algorithm" in order to see structural similarity with superiorization. With the reverse splitting FB iteration of (5.58) we may change our viewpoint and consider the gradient steps with respect to the least-squares objective, i.e., the mapping $\mathcal{A}(x) := x - \alpha_k A^\top (Ax - b)$, as a "basic algorithm" and view the proximal mapping $P_{\alpha_k} f$, with respect to the target function f as perturbations of the basic algorithm. Optimization theory tells us how to choose the parameter sequence (α_k) so as to make the overall iteration converge to a minimizer. We discussed a special case already in connection with Proposition 4.4.

If the proximal map $P_{\alpha_k} f$ cannot be computed in closed-form, then the inner iterations are applied to successively approximate the proximal mapping. This entails *several* "perturbations" before the next step of the "basic algorithm" is carried out. This resembles structurally the method of superiorization by nonascent vectors, see Algorithm 3, that uses several negative gradient steps to achieve heuristically a decrease of the target function f. Note that here f is nondifferentiable and alternatives to negative gradients have been considered [CGHH19]. Using proximal maps as suggested here, however, turns f into the differentiable function (3.6) with gradient (3.7) and hence enables to apply efficient accelerated optimization schemes.

The downside of this splitting is that the gradient steps with respect to g_0, seen as "basic algorithm", make in our scenario only very slow progress since the matrix A is underdetermined and bad conditioned. We therefore investigate whether it is possible to remedy this by acceleration.

We specialize below the accelerated FB iteration from (5.22) and assume exact computations of the proximal map $P_\alpha f$. We make in (5.22) the following parameter choices: $a = 1$, $a_k = 1$, $t_0 = 1$ and $\alpha_k = \gamma \leq 1/L_{g_0}$. This implies $t_1 = 1$ and $y_1 = x_1$. We can now start with iteration $k = 1$ and rename it below to $k = 0$.

Initialization: Choose $x_0 = y_0 \in \operatorname{dom} g$, $t_0 = 1$ and sequences $\gamma \in (0, 1/L_{g_0})$
Iteration for $k \geq 0$**:** Compute

$$x_{k+1} = P_\gamma f\big(y_k - \gamma \nabla f(y_k)\big), \tag{5.59a}$$
$$t_{k+1} = \frac{1}{2}\left(1 + \left(1 + 4t_k^2\right)^{1/2}\right), \tag{5.59b}$$
$$y_{k+1} = x_{k+1} + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k). \tag{5.59c}$$

As discussed in [VSBV13], this is the well-known FISTA iteration [BT09a].

6. Numerical Results

Section 6.1 details the experimental set-up based on which we compare superiorization with accelerated inexact optimization in Sections 6.2 and 6.3.

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

$$Ax = b, \qquad A \in \mathbb{R}^{m \times n} \tag{6.1}$$

considered for experiments collects linear tomographic measurements of the Shepp-Logan MATLAB 128×128 phantom shown in Figure 6.1 in a highly underdetermined scenario, i.e., $\frac{m}{n} = 0.15625$. The vector representing this test phantom is denoted by x_* in the sequel. We use the MATLAB routine routine `paralleltomo.m` from the AIR Tools package [HJ18] 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

$$b_j \quad \leftarrow \quad b_j + \xi_j, \qquad \xi_j \sim \mathcal{N}(0, \sigma^2), \qquad j \in [m], \qquad \sigma = \frac{0.02}{m} \sum_{j \in [m]} b_j. \tag{6.2}$$

Figure 6.2 illustrates the ill-posedness of the problem to recover x from b in terms of the regularized least-squares solutions

$$(A^\top A + \mu I_n)^{-1} b, \qquad \mu = 0.01. \tag{6.3}$$

Termination Criteria. Regarding termination of the superiorized CG iteration we used

$$\big\| A^\top (Ax - b) + \mu x \big\|_\infty \leq 0.001 \tag{6.4}$$

to terminate the iterative process that concerns the regularized least-squares problem (4.6), while using

$$\big\| A^\top (Ax - b) \big\|_\infty \leq 0.001 \tag{6.5}$$

in the case of the superiorized Landweber iteration, and

$$\max_{i \in [n]} \big| x_i \big(A^\top (Ax - b) \big)_i \big| \leq 0.001 \tag{6.6}$$

in the case of the superiorized projected Landweber iteration.

Regarding the optimization problem (1.2), the value $\tau = 0.01$ of the smoothing parameter in R_τ (3.12b) was used in all experiments. In the unconstrained case $x \in \mathbb{R}^n$, the criterion

$$\big\| A^\top (Ax - b) + \lambda \nabla R_\tau(x) \big\|_\infty \leq 0.001 \tag{6.7}$$

was used to terminate iterative optimization and to accept x as approximate minimizer of (1.2). In the nonnegatively constrained case $x \in \mathbb{R}^n_+$, the complementarity condition

$$\max_{i \in [n]} \big| x_i \big(A^\top (Ax - b) + \lambda \nabla R_\tau(x) \big)_i \big| \leq 0.001 \tag{6.8}$$

was used.

Regularization Parameter. The regularization parameter λ was set to

$$\lambda := \begin{cases} 0.01, & \text{for exact data} \\ 1.6529, & \text{for noisy data} \end{cases} \qquad \Longrightarrow \qquad \begin{cases} R_\tau(x) \approx R_\tau(x_*) \\ \|Ax - b\|^2 \approx \|Ax_* - b\|^2 \end{cases} \tag{6.9}$$

which implies ,

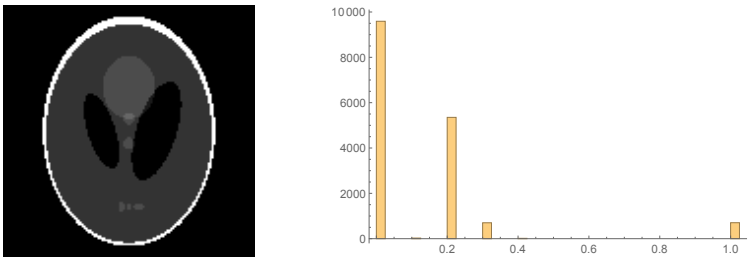

FIGURE 6.1. LEFT: The original image x_* of size $n = 128^2$ to be recovered from $m = 0.15625n$ linear measurements. Both exact and noisy measurements are considered. RIGHT: The histogram of the original image x_* that plots the number of pixels for each intensity value.

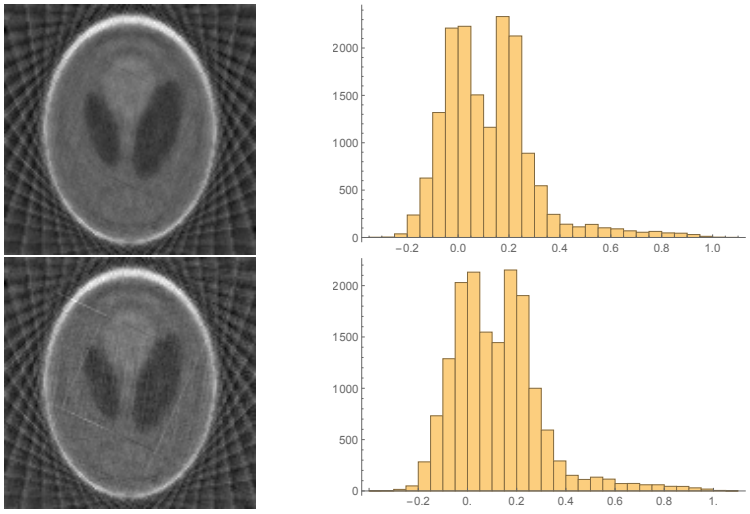

FIGURE 6.2. Regularized least-squares solutions determined by (6.3) for exact b (top) and for noisy b (bottom) illustrate the ill-posedness of the problem to recover the data shown by Figure 6.1 from $m = 0.15625n$ linear measurements comprising the data vector b.

- 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 (6.2).

Figure 6.3 depicts minimizers x of both the unconstrained and the nonnegatively constrained problem (1.2) for exact and for noisy data, respectively. The histograms show, in particular, that adding nonnegativity constraints increases accuracy.

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

(1) The scaled squared residual $\|Ax_k - b\|^2/(2m)$ 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) $R_\tau(x_k)/n$ that should approach the smoothed TV value of the original image x_*;
(3) The scaled error term $\|x_k - x_*\|^2/n$ that should be reduced as much as possible.

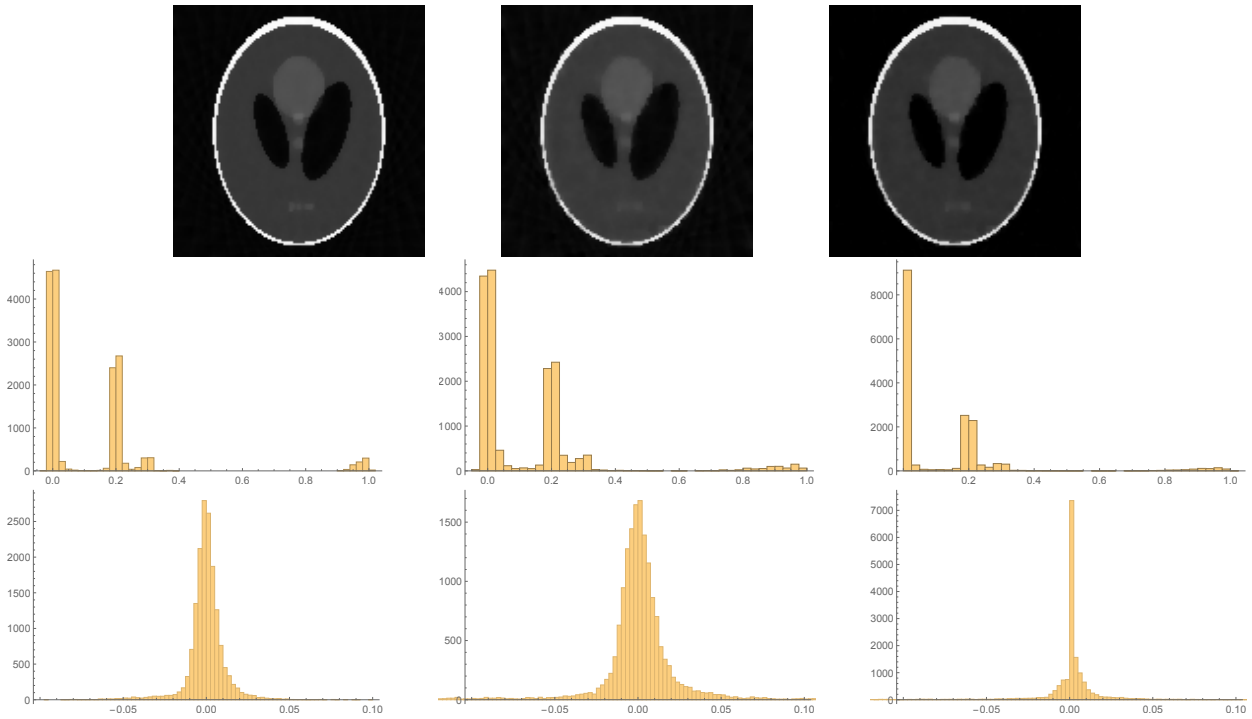

FIGURE 6.3. TOP ROW, FROM LEFT TO RIGHT: Minimizers x of the unconstrained objective function (1.2) for exact data, for noisy data, and minimizer for the nonegatively constrained objective function and noisy data. CENTER ROW: The histograms corresponding to the top row. In the unconstrained cases, a 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), and significantly so on the right (nonnegatively constrained case, noisy data).

Each optimization algorithm is guaranteed to reduce both (1) and (2), and also (3), provided that our model from (1.2) is appropriate, while the superiorized iterations are only guaranteed to reduce the error in (1).

Superiorization. We first address the unconstrained least-squares problem or the regularized least-squares problem (4.6) in order to handle an underdetermined system by CG and consider the superiorized CG and superiorized Landweber iteration by both gradient- and proximal point based target function reduction methods. The nonnegative least-squares problem is addressed by superiorizing the projected Landweber iteration. To additionally steer the iterates of a basic algorithm for the least-squares problem towards the nonnegative orthant we employ perturbation by constrained proximal points.

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 x_k using the relative error $\|x_k - x_*\|^2/n$ 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 3, (4.14) , (4.20) and (4.20). 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 (6.9). The combination of these parameters, that provide the best results are listed in Table 2, while the results are shown in Figure 6.4 and Figure 6.5. For comparison we also plot the results of the FB iteration (5.58) for the reversed splitting, in view of its intimate connection with the proximal-point based superiorized Landweber

iteration (4.21), discussed in Proposition 4.4. Note that the best found values for parameter a are very close to 1 as Proposition 4.4 suggests. For illustration purposes, we show error values even beyond the stopping criterion. Note that `ProxSupCG` has the best error values among all superiorized algorithms and that it is also the superiorized algorithm that satisfied the stopping criterion (6.4) in the least amount of time when it is adapted to incorporate nonnegativity constraints, by `ProxCSupCG`.

Method	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`	$(\gamma_0, a) = {}^{1.9\lambda}/_{\|A\|^2}, 1 - 10^{-6})$
`GradSubLW`, `GradSupProjLW`	$(a, \gamma_0, \kappa) = (1 - 10^{-4}, 0.0025, 20)$

TABLE 2. Optimal parameters for the superiorized basic algorithms.

Optimization. Figure 6.6 shows the results of comparing the FB iteration (5.13) with the accelerated FB iteration (5.22) for solving (1.2) without nonnegativity constraints. To obtain a fair comparison, the proximal maps were evaluated *exactly* using formula (5.35). Depending on the value of λ that affects the Lipschitz constant (5.1b), i.e., a small for exact data and a larger value for noisy data, more iterations are required in the latter case. And acceleration pays: significantly less iterations are then required.

Since the values of λ were chosen such that minimizers match the properties (6.9) of x_*, it seems fair to claim that – from the viewpoint of optimization – our implementation mimics the problem decomposition of superiorization, i.e., least-squares minimization adjusted by gradients of the target function R_τ, most efficiently. Inspecting the plots of Figure 6.6, we noticed that the final values are almost reached after merely ≈ 75 outer iterations. The much larger number of iterations required to meet the termination criterion (6.7), therefore, suggests that from the superiorization point of view, this termination criterion might be considered as too conservative.

Similarly, the accelerated FB iteration (5.59) for the reversed splitting 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 6.7 and Figure 6.8 for the unconstrained and nonnegatively constrained case, respectively.

6.3. **Superiorization vs. Optimization: Computational Complexity. Superiorization.** To evaluate the cost of the superiorized algorithms listed in Table 1 we call an outer iteration an iteration of the basic algorithm and an inner iteration an execution of Algorithm 3 or of a proximal map via (4.16) or (4.18). Within an outer iteration we have for the CG iteration in Algorithm 1 four costly matrix-vector operations in line 2 and line 5, while for the Landweber Algorithm 2 only two costly matrix-vector operations in line 2. The cost of an inner iteration is the amount of gradient and function evaluations of R_τ. In the case of computing nonascent directions by scaled nonnegative gradients, see line (4.14a) in Algorithm 4.14, the number of gradient and function evaluation is controlled by parameter κ and is upper bounded by $(\ell_k - \ell_{k-1})\kappa$. Similarly, the cost of a proximal mapping evaluation via (4.16) or (4.18) depends on the tolerance parameter of the employed optimization algorithm. In our case, the box-constrained L-BFGS method from [BLNZ95] is very efficient in reaching a tolerance level of 10^{-6} within few iterations due to the small values of 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 cost of Algorithm 3 for the considered choice of parameters. We underline that the low cost of the proximal-point based Landweber iteration is almost identical to the FB iteration (5.58) and the accelerated FB iteration (5.59) for the reversed splitting.

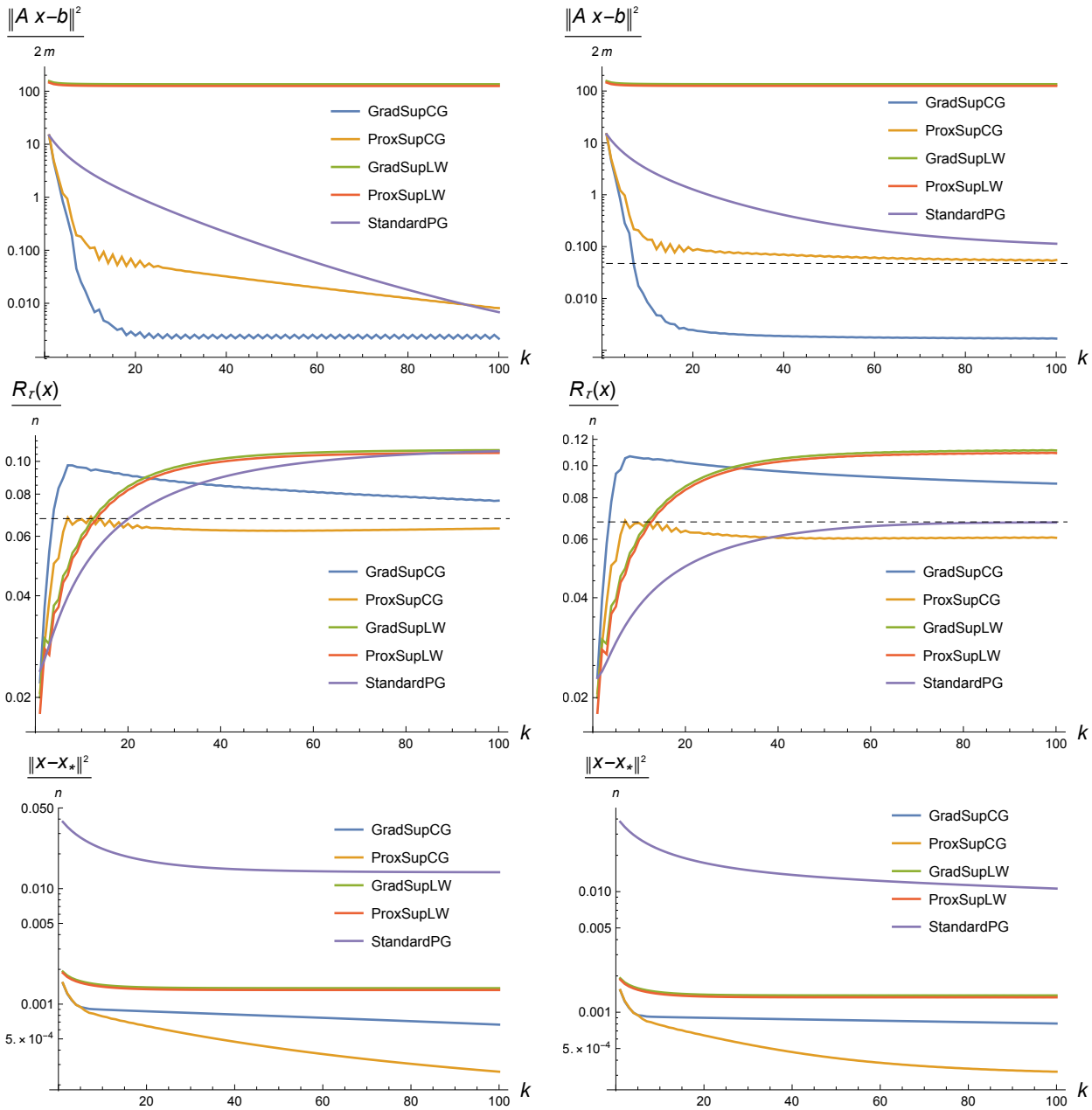

FIGURE 6.4. *Superiorization: Unconstrained Least-Squares.* Scaled values of the data term (top row) and the regularizer (middle row) of the objective (1.2) and the squared error norm (bottom row), as a function of the iterative steps k of the superiorized CG iteration using both perturbations based on gradients, see Algorithm 3 and proximal points, see (4.16), the gradient and proximal-point based superiorized Landweber iteration and the FB iteration (5.58) for the reversed splitting. The dashed lines in the middle of the plots indicate the corresponding values for x_*. The left and right columns correspond to exact and noisy data. Both the basic algorithm and the superiorization strategy does not incorporate nonegativity constraints, respectively. After appropriate parameter tuning the Superiorized CG iteration approaches the solution of (1.2) significantly faster than the FB iteration (5.58). The superiorized Landweber iteration makes slow progress due to very small step-sizes in view of the large value of $\|A\|^2$.

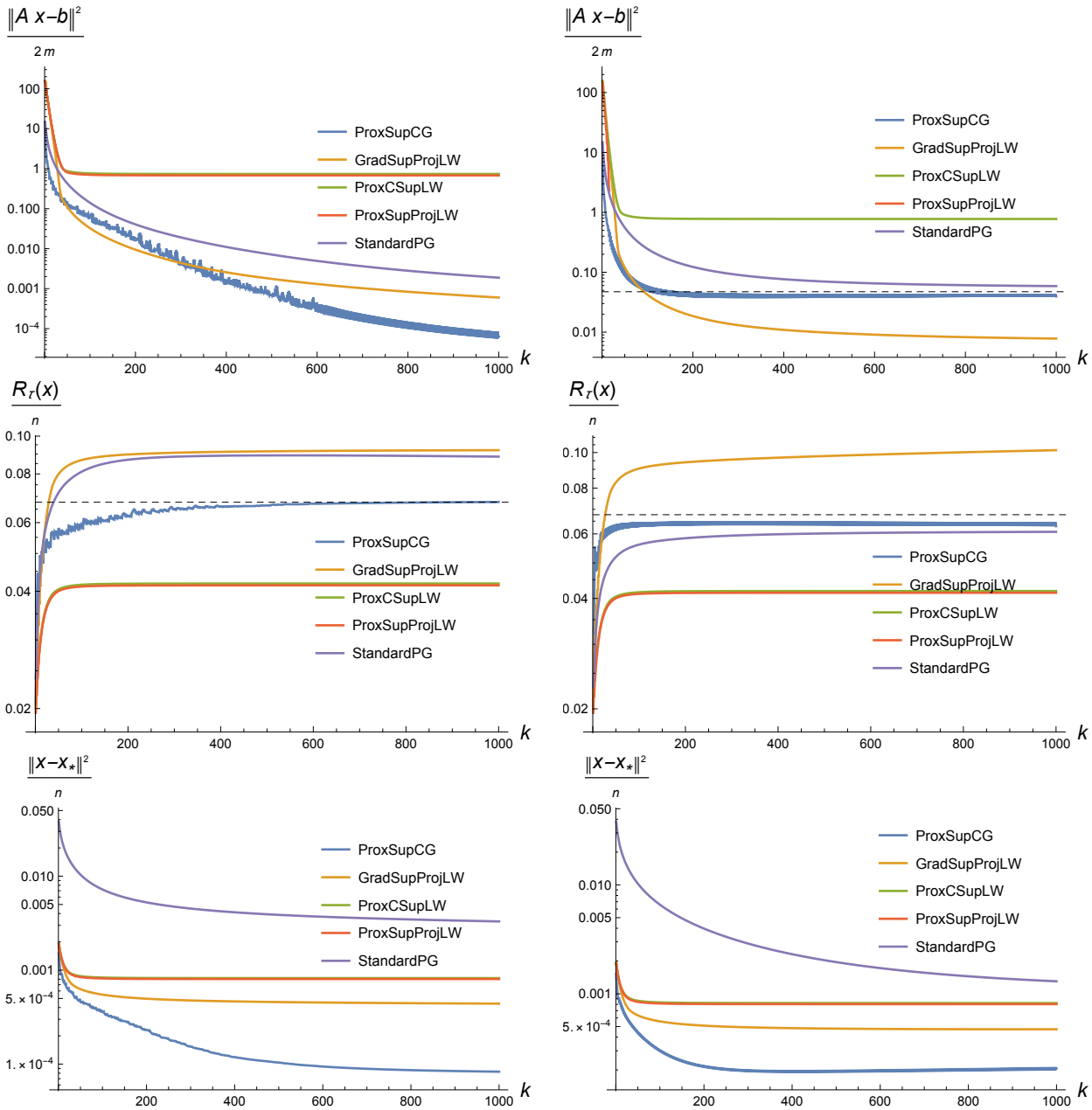

FIGURE 6.5. *Superiorization: Adding Nonnegative Constraints.* Scaled values of the data term (top row) and the regularizer (middle row) of the objective (1.2) and the squared error norm (bottom row), as a function of the iterative steps k of the superiorized CG iteration (4.20) and the Landweber iteration (4.21) using proximal points, see (4.18), the gradient and proximal-point based superiorized projected Landweber iteration and the FB iteration (5.58) for the reversed splitting. The dashed lines in the middle row plots 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 iteration perform similarly making slow progress towards x_* due to the bad conditioning of A, with a surprising better performance of the gradient based superiorized projected Landweber iteration `GradSupProjLW`. Again, the superiorized CG iteration approaches the solution of (1.2) significantly faster than the FB iteration (5.58) after appropriate parameter tuning.

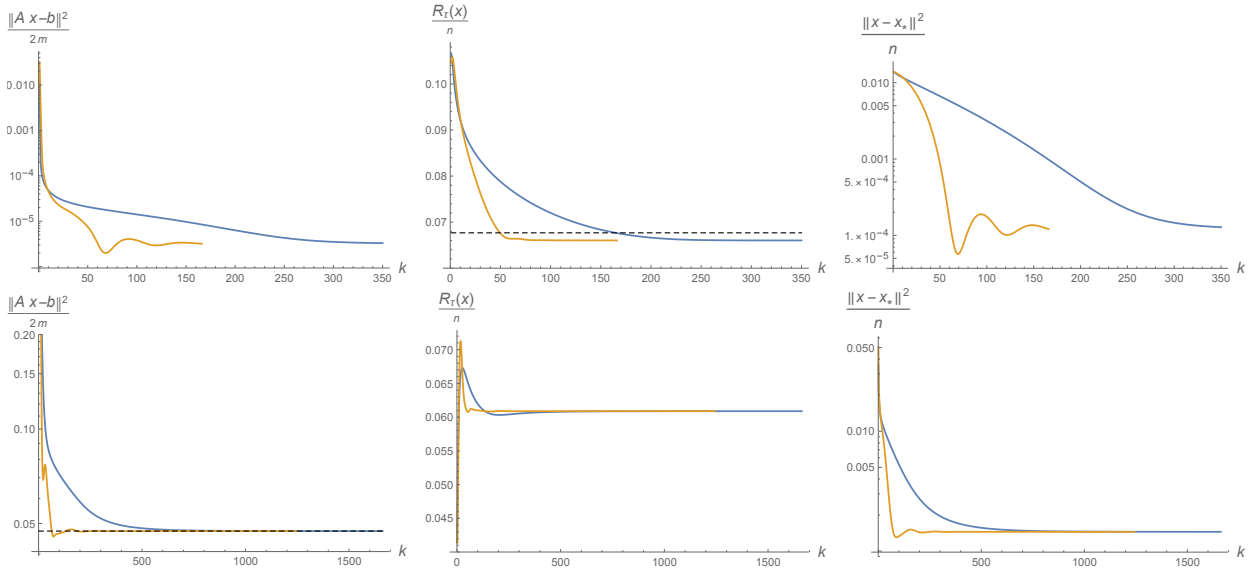

FIGURE 6.6. *Optimization: FB and accelerated FB iteration – No Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the objective (1.2) and the squared error norm (right column), as a function of the iterative steps k of the FB iteration (5.13) (dark curves) and the accelerated FB iteration (5.22). The dashed lines indicate the corresponding values for x_*. The top and bottom row correspond to exact and noisy data: accelerated FB needs about 50% and 25% less outer iterations, respectively.

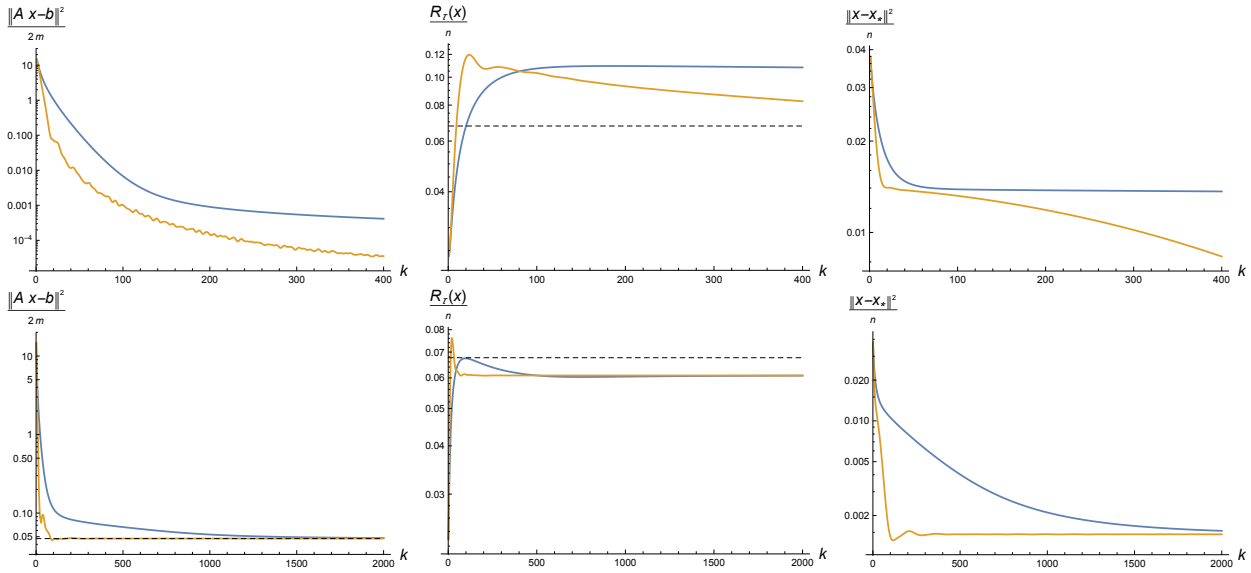

FIGURE 6.7. *Optimization: FB and accelerated FB iteration for the reversed splitting – No Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the *unconstrained* objective (1.2) and the squared error norm (right column), as a function of the iterative steps k of the FB iteration (5.58) (dark curves) and the accelerated FB iteration (5.59) for the reversed splitting. The dashed lines indicate the corresponding values for x_*. The top and bottom row correspond to exact and noisy data.

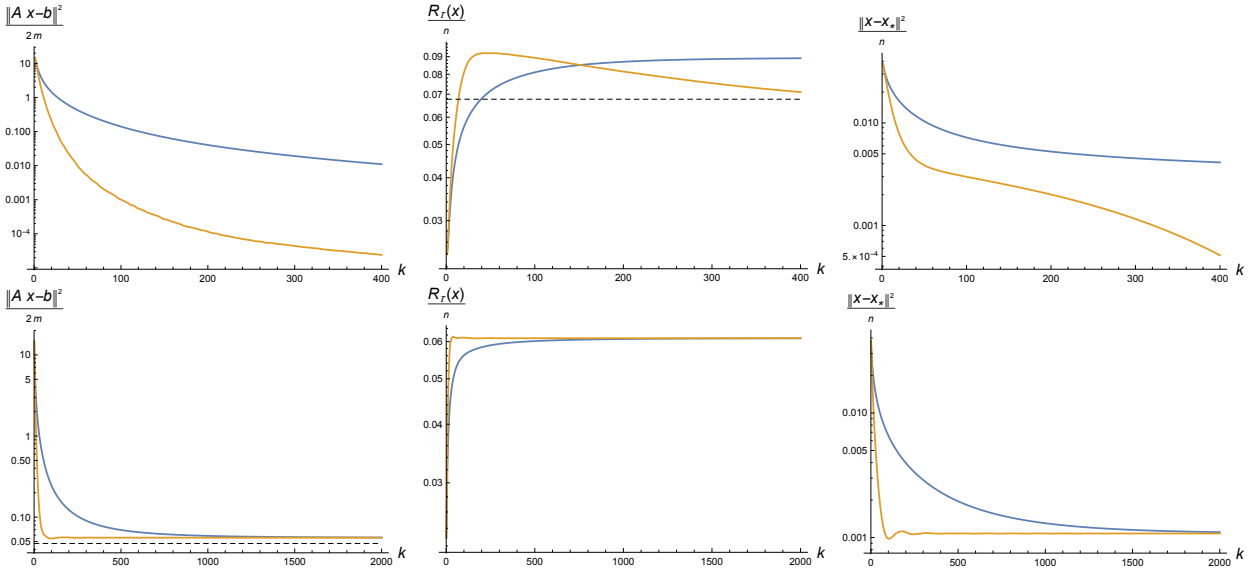

FIGURE 6.8. *Optimization: FB and accelerated FB iteration for the reversed splitting – Nonnegativity Constraints.* Scaled values of the data term (left column) and the regularizer (center column) of the *nonnegatively constrained* objective (1.2) and the squared error norm (right column), as a function of the iterative steps k of the FB iteration (5.58) (dark curves) and the accelerated FB iteration (5.59) for the reversed splitting. The dashed lines indicate the corresponding values for x_*. The top and bottom row correspond to exact and noisy data. While the FB iteration is damped by the inclusion of constraints, the nonnegativity constraints appear to be beneficial for the accelerated FB iteration on the other hand when comparing to the results from Figure 6.7.

Optimization. We evaluated accelerated FB iteration with *inexact* evaluations of the proximal maps, for both the unconstrained and nonnegatively constrained problem (1.2). In the former case, the inexactness criterion was evaluated during the primal-dual inner iteration using (5.46). In the latter nonengatively constrained case, we noticed that the corresponding criterion (5.51) could not be used since $z \notin K = \mathbb{R}^n_+$, with z given by (5.50a), happened frequently. We, therefore, resorted to the error estimate (5.56) and chose and exponent $\varepsilon_k = \mathcal{O}(k^{-q})$ so as to satisfy (5.16).

Figures 6.9 and 6.10 depict the corresponding results. Inspecting the panels on the left shows both in the unconstrained and in the constrained case, that choosing the proper error decay rate (5.23) significantly affects the accelerated FB iteration. Having chosen a proper exponent, an increasing 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 thousends for the entire optimization procedure.

6.4. **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.

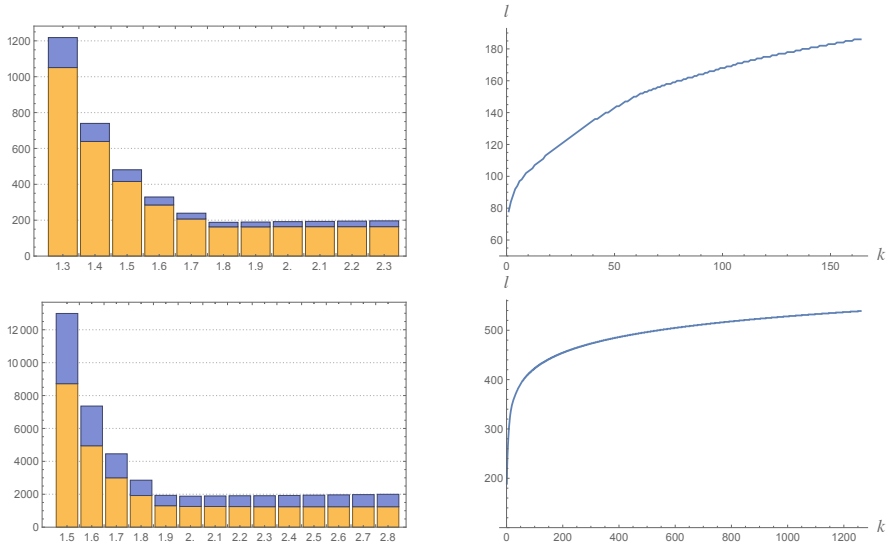

FIGURE 6.9. Inexact accelerated FB iteration for the *unconstrained* problem (1.2). LEFT: Relation of the number of outer iterations (not scaled down) and the *total* number of inner iterations (scaled down by $1/1000$) as a function of the exponent q of the error $\varepsilon_k = \mathcal{O}(k^{-q})$ (5.23). RIGHT: Number ℓ of inner iterations at each outer iteration k, for exponents $q = 1.8$ (top) and $q = 2.0$ (bottom) of (5.23).

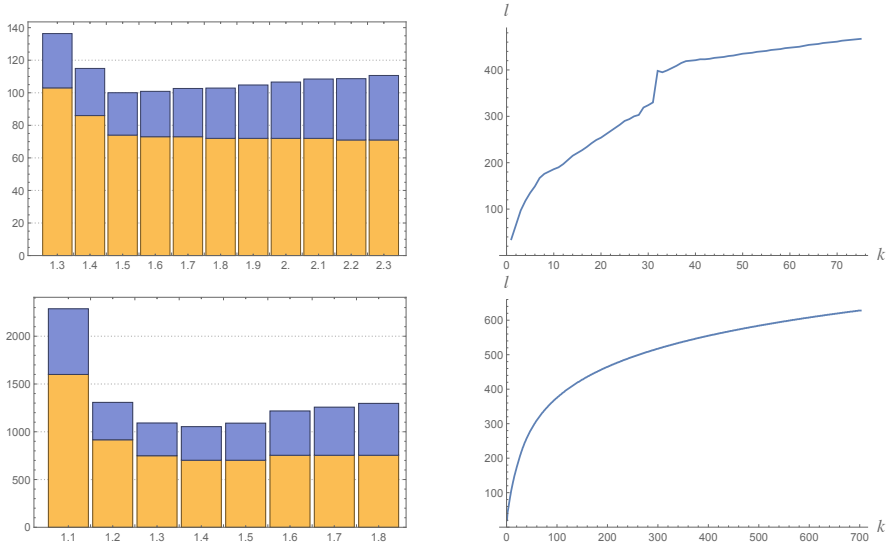

FIGURE 6.10. Inexact accelerated FB iteration for the *nonnegatively constrained* problem (1.2). LEFT: Relation of the number of outer iterations (not scaled down) and the *total* number of inner iterations (scaled down by $1/1000$) as a function of the exponent q of the error $\varepsilon_k = \mathcal{O}(k^{-q})$ (5.23). RIGHT: Number ℓ of inner iterations at each outer iteration k, for exponents $q = 1.8$ (top) and $q = 2.0$ (bottom) of (5.23).

(1) (a) *Superiorization, Figure 6.4, Figure 6.5: the choice of the basic algorithm is essential.* The superiorized versions of the CG iteration outperforms in our experimental numerical work the superiorized versions of the Landweber iteration. The superiorized Landweber iteration makes slow progress towards x_* due to the bad conditioning of matrix A and corresponding small step-size parameters.

(b) *Superiorization, Figure 6.4, Figure 6.5: incorporating nonnegativity constraints in the superiorization method slows down the iteration process.* Compare the three reconstruction errors in Figure 6.4 to values in Figure 6.5. Since the proximal-point based superiorized projected Landweber iteration (tagged `ProxSupProjLW`) behaves identically to the superiorized Landweber iteration, which includes nonnegativity constraints via the proximal map (tagged `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 believe 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 6.7, Figure 6.8: incorporating nonnegative constraints in the FB iteration for the reversed splitting* (5.58) *also slows down the iteration process. On the other hand, the accelerated FB iteration* (5.59) *does not suffer from this shortcoming.*

(b) *Optimization, Figure 6.6: 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, however.

(c) *Unconstrained optimization, Figure 6.9: inexact inner loops dramatically increase the number of iterations.* In comparison to Figure 6.6, the number of outer iterations increases to about 150 and 1200, respectively. Thus, the noisy case is becoming much more expensive than the noise-free case (in contrast to Figure 6.6). Each of these iterations requires on average 130 and 450 primal-dual iterations that inexactly evaluate the proximal map and are computationally inexpensive. The efficiency of inexact accelerated optimization requires to choose the exponent q not smaller than a problem-dependent critical value.

(d) *Nonnegatively constrained optimization, Figure 6.10: The number of outer iterations decreases relative to Figure 6.9 (so constraints help), but the number of inner iterations increases.* Otherwise, the observations regarding Figure 6.9 hold.

(3) (a) *Superiorization vs. optimization: convergence.* Each optimization algorithm is guaranteed to reduce both recovery errors (1) and (2), as listed in the beginning of Section 6.2. The subsampling ratio m/n is chosen according to [KP19] so that recovery via (1.2) is stable. As a consequence, also the recovery error (3) in Section 6.2 is reduced by the optimization algorithm. While the superiorized iterations are guaranteed to reduce only the recovery error in (1), we observe in Figure 6.4 and Figure 6.5 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 Section 6.2.

(b) *Superiorization vs. optimization: best performance is achieved by superiorized CG and accelerated FB iteration for the reversed splitting.* The evaluation of the implemented algorithms shows that, in the unconstrained case, both the proximal-point based superiorized CG iteration and the accelerated FB iteration (5.59) for the reversed splitting 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 FB iteration (5.59) for the reversed splitting 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 FB iteration for the two alternative splittings, we also addressed the so-called *balancing question* in superiorization: How to distribute the efforts that a superiorization algorithm invests in target function reduction steps versus the efforts invested in basic algorithm iterative steps? In the case of the splitting (5.1) that results in the inexact (accelerated) FB iteration (5.22), we performed *many basic algorithm iteration steps*, e.g., via (5.38), and just *one target function reduction step* with respect to R_τ. By contrast, in the case of the reversed splitting (5.58), we performed *one basic algorithm iteration* and *many target function reduction steps*, to compute the proximal point in (5.58). Our results show that the *accelerated* reversed FB splitting 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 superiorization 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) 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 (1.2) and one corresponding to the regularizer R_τ that might include nonnegativity constraints as well. Such an inexact DR splitting would not only be algorithmically very close to the superiorization of a basic algorithm for least-squares, but also provide a basis for a theoretical investigation of superiorization.

7. Conclusion

We considered the underdermined nonnegative least-squares problem, regularized by total variation, and compared, for its solution, superiorization with a state-of-the-art approach to convex optimization, viz. 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 superiorization approach motivated us to contrast superiorization with a first FB splitting, such that the more expensive backward part corresponds to the basic algorithm, whereas the forward part takes into account the target function. However, in view of the balancing problem of superiorization, 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, as regards the balancing issue.

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 question: How can *accelerated* superiorization be defined for convex basic algorithms and target function reductions?

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 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, it will be much harder to relate both methodologies to each other in the nonconvex case.

ACKNOWLEDGEMENTS

The work of Y. C. is supported by the ISF-NSFC joint research program grant No. 2874/19.

REFERENCES

[AT03] A. Auslender and M. Teboulle, *Asymptotic Cones and Functions in Optimization and Variational Inequalities*, Springer, 2003.

[BB96] H. H. Bauschke and J. M. Borwein, *On Projection Algorithms for Solving Convex Feasibility Problems*, SIAM Review **38** (1996), 367–426.

[BDHK07] D. Butnariu, R. Davidi, G. T. Herman, and 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.

[BLNZ95] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A Limited Memory Algorithm for Bound Constrained Optimization*, SIAM J. Sci. Comput. **16** (1995), 1190–1208.

[BRZ06] D. Butnariu, S. Reich, and 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 (H. F. Nathansky, B.G. de Buen, K. Goebel, W. A. Kirk, and B. Sims, eds.), Yokohama Publ., 2006, pp. 11 – 32.

[BRZ08] ———, *Stable Convergence Theorems for Infinite Products and Powers of Nonexpansive Mappings*, Numer. Func. Anal. Opt. (2008), 304–323.

[BRZ18] C. Bargetz, S. Reich, and R. Zalas, *Convergence Properties of Dynamic String-Averaging Projection Methods in the Presence of Perturbations*, Numer. Algorithms **77** (2018), 185–209.

[BT09a] A. Beck and M. Teboulle, *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Probl.*, SIAM J. Imag. Sci. **2** (2009), 183–202.

[BT09b] ———, *Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems*, IEEE Trans. Image Proc. **18** (2009), 2419–2434.

[Byr19] 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.

[CAM17] A. Cegielski and F. Al-Musallam, *Superiorization with Level Control*, Inverse Probl. **33** (2017), 044009.

[CDH10] Y. Censor, R. Davidi, and G. T. Herman, *Perturbation Resilience and Superiorization of Iterative Algorithms*, Inverse Probl. **26** (2010), 65008.

[CDH+14] Y. Censor, R. Davidi, G. T. Herman, R. W. Schulte, and L. Tetruashvili, *Projected Subgradient Minimization versus Superiorization*, J. Optim. Theory Appl. **160** (2014), 730–747.

[Ceg13] A. Cegielski, *Iterative Methods for Fixed Point Problems in Hilbert Spaces*, Lect. Notes Math., vol. 2057, Springer, 2013.

[Cen15] Y. Censor, *Weak and Strong Superiorization: Between Feasibility-Seeking and Minimization*, An. St. Univ. Ovidius Constanta **23** (2015), 41–54.

[Cen19] 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: September 26, 2019.

[CGHH19] Y. Censor, E. Garduño, E. S. Helou, and G. T. Herman, *Derivative-Free Superiorization: Principle and Algorithm*, arXiv e-prints (2019), arXiv:1908.10100.

[CHJE17] Y. Censor, G. T. Herman, and M. Jiang (Editors), *Special Issue on Superiorization: Theory and Applications*, vol. 33 (4), IOP Science, Inverse Probl., 2017.

[CL19] Y. Censor and E. Levy, *An Analysis of the Superiorization Method via the Principle of Concentration of Measure*, Applied Mathematics and Computation, accepted for publication (2019).

[Com04] P. L. Combettes, *Solving Monotone Inclusions via Compositions of Nonexpansive Averaged Operators*, Optimization **53** (2004), 475–504.

[Com18] ———, *Perspective Functions: Properties, Constructions, and Examples*, Set-Valued Var. Anal. **26** (2018), 247–264.

[CP11a] A. Chambolle and T. Pock, *A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging*, J. Math. Imag. Vision **40** (2011), 120–145.

[CP11b] P. L. Combettes and J.-C. Pesquet, *Proximal Splitting Methods in Signal Processing*, Fixed-Point Algorithms for Inverse Probl. in Science and Engineering (H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, eds.), Springer, New York, 2011, pp. 185–212.

[CW05] P. L. Combettes and V. R. Wajs, *Signal Recovery by Proximal Forward-Backward Splitting*, Multiscale Model. Simul. **4** (2005), 1168–1200.

[CZ97] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, New York, NY, USA, 1997.

[CZ15] Y. Censor and A. J. Zaslavski, *Strict Fejér Monotonicity by Superiorization of Feasibility-Seeking Projection Methods*, J. Optim. Theory Appl. **165** (2015), 172–187.

[CZC12] D.-Q. Chen, H. Zhang, and L.-Z. Cheng, *A Fast Fixed Point Algorithm for Total Variation Deblurring and Segmentation*, J. Math. Imag. Vision **43** (2012), 167–179.

[Dav10] R. Davidi, *Algorithms for Superiorization and Their Applications to Image Reconstruction*, Ph.D. thesis, City University of New York, New York, NY, USA, 2010, AAI3426727.

[DHC09] R. Davidi, G. T. Herman, and Y. Censor, *Perturbation-Resilient Block-Iterative Projection Methods with Application to Image Reconstruction From Projections*, Int. T Oper. Res. **16** (2009), 505–524.

[GC18] Y. Guo and W. Cui, *Strong Convergence and Bounded Perturbation Resilience of a Modified Proximal Gradient Algorithm*, J. Inequal. Appl. **2018** (2018), 103.

[GO09] T. Goldstein and S. Osher, *The Split Bregman Method for L1 Regularized Problems*, SIAM J. Imag. Sci. **2** (2009), 323–343.

[HGDC12] G. T. Herman, E. Garduño, R. Davidi, and Y. Censor, *Superiorization: An Optimization Heuristic for Medical Physics*, Med. Phy. **39** (2012), 5532–5546.

[Hig08] N.J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, 2008.

[HJ18] P. C. Hansen and J. S. Jørgensen, *AIR Tools II: Algebraic Iterative Reconstruction Methods, Improved Implementation*, Numer. Algorithms **79** (2018), 107–137.

[HLZH18] E. S. Helou, C. Lin, M. V. W. Zibetti, and G. T. Herman, *Superiorization of Preconditioned Conjugate Gradient Algorithms for Tomographic Image Reconstruction*, Appl. Anal. Optim. **2** (2018), 271–284.

[HX17] H. He and H.-K. Xu, *Perturbation Resilience and Superiorization Methodology of Averaged Mappings*, Inverse Probl. **33** (2017), 044007.

[JCJ16] W. Jin, Y. Censor, and M. Jiang, *Bounded Perturbation Resilience of Projected Scaled Gradient Methods*, Comput. Optim. Appl. **63** (2016), 365–392.

[KP19] J. Kuske and S. Petra, *Performance Bounds For Co-/Sparse Box Constrained Signal Recovery*, An. St. Univ. Ovidius Constanta **1** (2019), 79–109.

[LZZ+19] S. Luo, Y. Zhang, T. Zhou, J. Song, and Y. Wang, *XCT Image Reconstruction by a Modified Superiorized Iteration and Theoretical Analysis*, Optim. Methods and Softw. **0** (2019), 1–18.

[RDP17] D. Reem and A. De Pierro, *A New Convergence Analysis and Perturbation Resilience of Some Accelerated Proximal Forward-Backward Algorithms with Errors*, Inverse Probl. **33** (2017), 044001.

[Roc70] R. Rockafellar, *Convex Analysis*, Princeton Univ. Press, 1970.

[RW09] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, 3rd ed., Springer, 2009.

[RZ17] S. Reich and A. J. Zaslavski, *Convergence to Approximate Solutions and Perturbation Resilience of Iterative Algorithms*, Inverse Probl. **33** (2017), 044005.

[Saa03] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, 2003.

[SJP12] E. Y. Sidky, J. H. Jørgensen, and X. Pan, *Convex Optimization Problem Prototyping for Image Reconstruction in Computed Tomography with the Chambolle–Pock algorithm*, Phys. Med. Biol. **57** (2012), 3065–3091.

[Tos94] P. Tossings, *The Perturbed Proximal Point Algorithm and Some of Its Applications*, Appl. Math. Optim. **29** (1994), 125–159.

[VO98] C. R. Vogel and M. E. Oman, *Fast, Robust Total Variation-Based Reconstruction of Noisy, Blurred Images*, IEEE Trans. Image Proc. **7** (1998), 813–824.

[Vol14] E. A. H. Vollebregt, *The Bound-Constrained Conjugate Gradient Method for Non-Negative Matrices*, J. Optim. Theory Appl. **162** (2014), 931–953.

[VSBV13] S. Villa, S. Salzo, L. Baldassarre, and A. Verri, *Accelerated and Inexact Forward-Backward Algorithms*, SIAM J. Optim. **23** (2013), 1607–1633.

[XYT+16] Q. Xu, D. Yang, J. Tan, A. Sawatzky, and M. A. Anastasio, *Accelerated Fast Iterative Shrinkage Thresholding Algorithms for Sparsity-Regularized Cone-Beam CT Image Reconstruction*, Med Phys. **43** (2016), 1849.

[ZLH18] M. V. W. Zibetti, C. Lin, and G. T. Herman, *Total Variation Superiorized Conjugate Gradient Method for Image Reconstruction* , Inverse Probl. **34** (2018), 034001.

(Y. Censor) DEPT. MATHEMATICS, UNIVERSITY OF HAIFA, ISRAEL
E-mail address: yair@math.haifa.ac.il
URL: http://math.haifa.ac.il/yair

(S. Petra) MATHEMATICAL IMAGING GROUP, HEIDELBERG UNIVERSITY, GERMANY
E-mail address: petra.uni-heidelberg.de
URL: https://www.stpetra.com

(C. Schnörr) IMAGE AND PATTERN ANALYSIS GROUP, HEIDELBERG UNIVERSITY, GERMANY
E-mail address: schnoerr@math.uni-heidelberg.de
URL: https://ipa.math.uni-heidelberg.de