A Proximal Gradient Method with Learned Deviations and Inertial Acceleration*

Ernesto G. Birgin† Gabriel R. S. Grillo‡ Elias S. Helou§

July 1, 2026

Abstract

We consider convex composite optimization problems in which the objective function is the sum of a smooth convex term and a nonsmooth convex term admitting an efficiently computable proximal operator. We introduce a learning-enhanced inertial proximal gradient method that combines learned deviation terms with momentum-based acceleration, while also allowing for inexact evaluation of the proximal operator. The proposed scheme is motivated by a limitation commonly observed in learning-based optimization methods: acceleration induced by learned components is typically effective only within the finite iteration regime used during training. To address this issue, our method combines learned deviations with inertial acceleration. It leverages learned deviations to improve performance within the training horizon, while the inertial mechanism ensures robust asymptotic convergence independently of the training regime. We establish convergence results showing that the proposed algorithm preserves the optimal $O(1/k^2)$ worst-case convergence rate of accelerated proximal gradient methods despite the presence of learned perturbations and inexact proximal evaluations. Numerical experiments indicate that the learned deviations significantly improve performance during the training horizon, while the inertial structure ensures stable and competitive behavior in later iterations. The proposed framework therefore combines the practical advantages of learning-based acceleration with the robustness of classical accelerated optimization methods.

Keywords: Convex Optimization, Machine Learning, Linear Inverse Problems

MSC codes: 68Q25, 68R10, 68U05

*This work has been partially supported by the Brazilian agencies FAPESP (grants 2013/07375-0, 2022/05803-3, 2023/08706-1, 2023/15603-4, 2024/22384-0, and 2025/13357-1) and CNPq (grants 302073/2022-1 and 308004/2022-1).

†Department of Computer Science, Institute of Mathematics, Statistics and Computer Science, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil (e-mail: egbirgin@ime.usp.br).

‡Department of Applied Mathematics, Institute of Mathematics, Statistics and Computer Science, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil (e-mail: ggrillo@ime.usp.br). Corresponding author.

§Department of Applied Mathematics and Statistics, Institute of Mathematical Sciences and Computation, University of São Paulo, Av. Trabalhador São Carlense, 400, 13566-590, São Carlos, SP, Brazil (e-mail: elias@icmc.usp.br)

1 Introduction

1.1 Convex Composite Optimization Problems

A convex composite optimization problem consists of minimizing an objective function expressed as the sum of two convex functions, each satisfying a different set of assumptions. This formulation is useful because it enables the development of optimization algorithms based on splitting strategies. These approaches involve different types of steps at each iteration, with each step type exploiting the structure of one of the constituent functions.

Convex composite optimization problems have a wide range of applications. An important use case is variational regularization models, which are used to find stable solutions to ill-posed inverse problems. In this application, the objective function comprises a data fidelity and a regularization term. Often, the data fidelity term is a smooth function (e.g., mean squared error), while the regularization term is a non-smooth function (e.g., ℓ_1-norm regularization).

One of the most popular and well-studied splitting methods for solving convex composite optimization problems with one smooth term is the proximal gradient method, which combines the gradient descent iteration with the proximal iteration. The convergence properties of the method are achieved with the constant step-size policy [7]. Line searches that guarantee analogous convergence properties have also been studied in the literature (see [5, §10.3–10.4]).

The proximal gradient method has been the focus of multiple enhancements aimed at acceleration. One of these techniques, known as the Spectral Proximal Gradient Method, involves considering the Barzilai-Borwein step-size [4], which estimates second-order information, associated with a non-monotone line search (e.g., the Grippo–Lampariello–Lucidi line search [15]). This technique was initially considered for the projected gradient method in [9] and was subsequently extended to the proximal gradient method in [26]. Another notable acceleration technique involves incorporating a momentum term. The seminal papers [21, 18] introduced momentum acceleration for the gradient descent method. An algorithm with convergence of order $O(1/k^2)$ on the optimality gap, where k is the iteration counter, was developed in [18]. This order of convergence was proven optimal in the sense that, for any method based only on gradient evaluations of the iterates, there exists a function for which the optimality gap is bounded from below by a quantity of $O(1/k^2)$, as stated in [19, Thm. 2.1.7]. The momentum acceleration method proposed in [18] was extended to composite convex optimization problems in [7] and [20]. The method developed in [7] was named Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) as it is an accelerated version of the Iterative Shrinkage-Thresholding Algorithm (ISTA). ISTA is essentially the proximal gradient method applied to the LASSO problem [24]. Despite its name, which reflects its origins in shrinkage and thresholding methods, FISTA is a general algorithm applicable to a broad class of convex composite optimization problems. The convergence of function values is guaranteed under the constant step-size policy [7]. With analogous convergence results, line search procedures were derived in [7, 22]. In both cases, the methods achieve the optimal $O(1/k^2)$ convergence rate of first-order methods for smooth problems. (Incidentally, note that FISTA can also handle non-smooth problems.) This fast convergence rate is possible because the algorithms use the proximal operator to handle the non-smooth term of the objective function.

1.2 Machine Learning and Optimization

Machine learning (ML) techniques have long relied on optimization theory and methods to achieve success, as most learning paradigms involve solving either a minimization or a maximization problem. For instance, Support Vector Machine classifiers select a hyperplane that provides the largest possible margin of separation in the training data, resulting in a quadratic optimization problem. K-means clustering minimizes the within-cluster sum of squares (akin to variance), resulting in a hard integer optimization problem that is typically solved heuristically. Deep learning techniques require computing parameters for artificial neural networks that minimize network error in the training data, by solving a continuous nonconvex optimization problem. In most cases, computationally expensive numerical optimization only occurs during training. In contrast, the classification or regression steps of ML methods are inexpensive from a computational viewpoint.

Conversely, an emerging research direction is the development of machine learning (ML)-based optimization methods; see [11]. On the one hand, classical optimization algorithms can be unrolled, that is, their iterative steps are explicitly truncated to a finite number and interpreted as layers of a neural network, possibly with learnable parameters. These unrolled methods can then serve as the basis for deep neural network architectures applied to a variety of tasks, such as sparse coding [13] or medical imaging reconstruction [1, 27]. As discussed in [11], these unrolled architectures may also be used directly to solve optimization problems, effectively acting as learned optimization algorithms. On the other hand, ML techniques may be used to learn optimization algorithm parameters based on data representing a class of problems of interest with the objective of accelerating convergence. For example, learning step-sizes and extrapolation/momentum parameters was considered in [23].

Another example is the deviation-based algorithms proposed in [2]. In that paper, the authors introduce deviation vectors into a known optimization method and estimate bounds on the norms of these vectors to guarantee convergence properties. Then, a normalized neural network may be trained to infer suitable deviations to accelerate empirical convergence.

Introducing deviations in the analysis of an optimization algorithm is a standard approach to derive the convergence properties of inexact methods. In these methods, the deviations are unknown because they model errors in the computations. For instance, a deviation could represent an error when calculating the gradient at an iterate, as in incremental or stochastic gradient descent techniques. However, ML-based acceleration—the topic of this paper—aims to produce deviations that meet a specific goal, directing the training procedure. This goal could be, for example, to obtain the smallest possible value of the objective function within a certain number of iterations or to meet a specific termination criterion within the smallest possible number of iterations.

1.3 Contributions

We present a new method for composite convex optimization. The method can be seen as a variant of the proximal gradient method that combines accelerations based on momentum/extrapolation and on deviations. Moreover, the method we introduce allows for the inexact computation of the proximal operator within iterations. From a practical point of view, following [2], we propose

that the deviations be learned by training a neural network. From a theoretical point of view, unlike the methods presented in [2], the optimality gap of the proposed method preserves the $O(1/k^2)$ asymptotic and non-asymptotic convergence orders exhibited by the FISTA method. Thus, the proposed method is the first to simultaneously present the benefits of ML-based accelerated methods and the theoretical properties of momentum/extrapolation-based accelerated methods. The numerical results show improved convergence in all aspects. At each step, our method achieves lower objective function values than traditional momentum-based accelerated proximal gradient methods and recent ML-based accelerated methods.

1.4 Paper Organization

Section 2 presents the problem and the method we propose to solve it; it is a momentum-accelerated proximal-gradient method with additional deviation terms. Section 3 presents the theoretical analysis and properties of the method. Section 4 presents numerical experiments on the application of the proposed method to the computational tomography problem. The section also describes implementation details, including how to learn the deviations. Section 5 provides conclusions and brings suggestions for future research.

Notation: We denote by $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$ the standard norm and inner product in $\mathbb{R}^n$, respectively.

2 Optimization problem and proposed method

In this paper, we consider the optimization problem given by

$$\min_{x\in\mathbb{R}^n} F(x) \coloneqq f(x) + g(x), \tag{1}$$

where the functions f and g satisfy the following assumptions.

Assumption A1 *$f:\mathbb{R}^n\to\mathbb{R}$ is convex and L_f-smooth, i.e., it is differentiable and its gradient satisfies*

$$\|\nabla f(y)-\nabla f(x)\| \le L_f\|y-x\|, \quad \forall x,y\in\mathbb{R}^n.$$

The constant L_f is called the smoothness parameter of f.

Assumption A2 *$g:\mathbb{R}^n\to(-\infty,\infty]$ is proper, closed, and convex.*

Assumption A3 *The set $X_* \coloneqq \operatorname{argmin}_{x\in\mathbb{R}^n} F(x)$ is non-empty and we define $F_* \coloneqq F(x_*)$ for some $x_*\in X_*$.*

The proximal operator associated with g is defined as

$$\operatorname{prox}_g^\rho(w) \coloneqq \operatorname*{argmin}_{x\in\mathbb{R}^n}\left\{g(x)+\frac{1}{2\rho}\|x-w\|^2\right\}, \tag{2}$$

with $w\in\mathbb{R}^n$ and $\rho>0$. By Assumption A2, it holds that, for every $\rho>0$, $\operatorname{prox}_g^\rho(w)$ is a singleton; see [5, Thm. 6.1, p. 131]. From a practical point of view, we assume that g is simple, in the sense that $\operatorname{prox}_g^\rho(w)$ has a closed form or an efficient iterative approximation scheme.

The iterations of the proximal gradient for (1) are written as

$$x_{k+1} = \operatorname{prox}_g^{\rho_k}\left(x_k - \rho_k \nabla f(x_k)\right), \quad \forall k \geq 0, \tag{3}$$

with $\rho_k > 0$. When Assumptions A1–A3 hold, convergence of the algorithm is ensured with a constant step-size $\rho_k = 1/L_f$. If L_f is unknown, a backtracking line search procedure can be used, which also provides the same convergence properties. In both cases, it is possible to prove an $O(1/k)$ order of convergence on the optimality gap, i.e., on the sequence $\{F(x_k) - F_*\}_{k\geq 0}$; see [7] or [5, §10.4].

FISTA is a momentum-based accelerated proximal gradient method. Given an initial estimate $x_0 \in \mathbb{R}^n$ and a sequence of non-increasing step-sizes $\{\rho_k\}_{k\geq 1}$, considering the initializations $y_1 = x_0$ and $t_1 = 1$, the update of the k-th iteration of FISTA is given by

$$x_k = \operatorname{prox}_g^{\rho_k}\left(y_k - \rho_k \nabla f(y_k)\right), \tag{4a}$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \tag{4b}$$

$$y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}), \tag{4c}$$

for $k \geq 1$. Notice that the x_k update in (4a) is a proximal gradient step from y_k, and that the y_{k+1} update in (4c) is an extrapolation (momentum) step from x_k. The parameter t_k determines the extrapolation parameter in (4c). In the original formulation of [7], t_k is computed as in (4b). Alternatively, one can compute $t_k = (k+1)/2$; see [10]. Convergence on function values is guaranteed under the constant step-size policy $\rho_k = 1/L_f$. A line search procedure was also derived in [7] with analogous convergence results. In both cases, the optimal order of convergence $O(1/k^2)$ on the optimality gap for smooth problems is attained, even though FISTA was developed to treat non-smooth problems in the form of (1); see [7] or [5, §10.7].

A deviation-based proximal gradient method was proposed in [2], in which the update in the k-th iteration is given by

$$w_k = x_k + \Delta x_k^1, \tag{5a}$$

$$x_{k+1} = \operatorname{prox}_g^{\rho_k}\left(x_k - \rho_k \nabla f(w_k) + L_f \rho_k \Delta x_k^1 + \Delta x_k^2\right), \tag{5b}$$

with $\rho_k \in (0, 2/L_f)$ and $\Delta x_k^1, \Delta x_k^2 \in \mathbb{R}^n$ for each $k \geq 0$. Under Assumptions A1–A2 and appropriate bounds on $\|\Delta x_k^1\|$ and $\|\Delta x_k^2\|$, convergence properties can be assured; see [2, Thm. 3.4 and Cor. 3.5]. Moreover, it is pointed out in [2] that FISTA, as defined in (4), can be recovered from (5) by setting $\Delta x_k^1 = ({}^{(t_k-1)}/_{t_{k+1}})\,(x_k - x_{k-1})$, $\Delta x_k^2 = 0$, $\rho_k = 1/L_f$, and $\{t_k\}_{k\geq 2}$ as in (4b) with $t_1 = 1$. Since in this case the bounds on $\|\Delta x_k^1\|$ and $\|\Delta x_k^2\|$ hold [2, Rmk. 3.8], both the convergence properties of (4) and (5) hold. However, in the general setting there is no convergence rate analysis for the optimality gap of the iterative scheme given by (5).

The method proposed in this work can be considered as an extension of FISTA with deviations analogous to those described in [2]. Moreover, we also consider inexact proximal evaluations as in [25, 8]. For the inexact notion of the proximal evaluations, let g satisfying Assumption A2, and consider $J_g \colon \mathbb{R}^n \times \mathbb{R}_{++} \times \mathbb{R}_+ \to \mathbb{P}(\mathbb{R}^n)$ such that

$$J_g(w, \rho, \epsilon) \coloneqq \left\{ z \in \mathbb{R}^n \colon 0 \in \partial_\epsilon g(z) + \frac{1}{\rho}(z - w) \right\}, \tag{6}$$

in which $\partial_\epsilon g(z)$ denotes the ϵ-subdifferential of g in z, defined by the following extension of the subdifferential of g in z:

$$\partial_\epsilon g(z) \coloneqq \{y \in \mathbb{R}^n \colon g(x) \geq g(z) + \langle y, x - z\rangle - \epsilon, \forall x \in \mathbb{R}^n\}\,. \tag{7}$$

By Fermat's optimality condition [5, Thm. 3.63, p. 73] and the definition of the proximal operator in (2), one has that

$$\operatorname{prox}_g^\rho(w) = \left\{z \in \mathbb{R}^n \colon 0 \in \partial g(z) + \frac{1}{\rho}(z - w)\right\}.$$

Thus, it holds that $\operatorname{prox}_g^\rho(w) = J_g(w, \rho, 0)$ and $\operatorname{prox}_g^\rho(w) \subseteq J_g(w, \rho, \epsilon)$ for every $w \in \mathbb{R}^n$, $\rho \in \mathbb{R}_{++}$, and $\epsilon \in \mathbb{R}_+$. Moreover, it is easy to see that if $0 < \epsilon_1 \leq \epsilon_2$ then $J_g(w, \rho, \epsilon_1) \subseteq J_g(w, \rho, \epsilon_2)$. Therefore, the operator $J_g(\cdot, \rho, \epsilon)$ could be interpreted as an extension of $\operatorname{prox}_g^\rho(\cdot)$ for inexact evaluations with a tolerance parameter $\epsilon > 0$.

This notion of inexactness is equivalent to the notion described in [25, §2.1], where the authors define the notation $z \approxeq_\epsilon \operatorname{prox}_{\lambda g}(y)$ when $\frac{y-z}{\lambda} \in \partial_{\frac{\epsilon^2}{2\lambda}} g(z)$. Thus, it holds that $z \approxeq_\epsilon \operatorname{prox}_{\lambda g}(y) \iff z \in J_g(y, \lambda, \epsilon^2/2\lambda)$. One remarkable property of this inexact criterion is its formulation in terms of the duality gap. By [25, Prop. 2.2], when $g(x) = \omega(Bx)$, with $B : \mathbb{R}^n \to \mathbb{R}^\ell$ a linear transformation and $\omega : \mathbb{R}^\ell \to (-\infty, +\infty]$ a proper, closed, convex function, if $v \in \mathbb{R}^\ell$ is such that $G(y - \lambda B^T v, v) \leq \frac{\epsilon^2}{2\lambda}$, in which G is the duality gap function associated with the minimization problem that defines the evaluation of $\operatorname{prox}_g^\lambda(y)$ (see (2)), then $y - \lambda B^T v \approxeq_\epsilon \operatorname{prox}_{\lambda g}(y)$. By setting $\epsilon \leftarrow \sqrt{2\lambda\epsilon}$, $\rho \leftarrow \lambda$, and $y \leftarrow w$, we can conclude that

$$G(w - \rho B^T v, v) \leq \epsilon \implies w - \rho B^T v \in J(w, \rho, \epsilon). \tag{8}$$

The above implication gives us a sufficient condition to evaluate $J(w, \rho, \epsilon)$. Moreover, this condition relies on computable quantities.

The proposed method is fully described by Algorithm 1, in which inexact proximal evaluations have their error modeled by the operator $J_g(\cdot, \rho, \epsilon)$, and the deviations are expressed by Δy_k, Δw_k, and Δz_k.

Concerning the deviations Δy_k, Δw_k, and Δz_k, lines 4 and 6 impose bounds on $\|\Delta y_k\|$ and $\|\Delta w_k\|$, respectively, while line 10 imposes that the deviation Δz_k must produce functional decay. The deviation Δw_k is analogous to an error vector considered in the inexact FISTA proposed in [8]. The deviation Δz_k generalizes the step that guarantees monotone function values in Monotone-FISTA (MFISTA) [6], in which $x_k = \operatorname{argmin}\{F(x) : x \in \{x_{k-1}, z_k\}\}$, by setting $\Delta z_k = 0$ if $F(z_k) \leq F(x_{k-1})$, or setting $\Delta z_k = x_{k-1} - z_k$ if $F(z_k) > F(x_{k-1})$. To the best of our knowledge, the deviation Δy_k is a novelty introduced in the present work.

The sequence $\{R_k\}_{k\geq 0}$ is responsible for the bounds on $\|\Delta y_k\|$, $\|\Delta w_k\|$, and ϵ_k, as can be seen in lines 4, 6, and 8, respectively. The value of R_0 is an input parameter of the algorithm, while subsequent values are computed as described in line 14. The input parameter α is only used in line 14, acting as a relative tolerance parameter for $\|\Delta y_k\|$, $\|\Delta w_k\|$, and the inexactness ϵ_k on the proximal evaluation. The input sequences $\{\beta_{1,k}\}_{k\geq 1}$, $\{\beta_{2,k}\}_{k\geq 1}$, and $\{\beta_{3,k}\}_{k\geq 1}$ have the same property as α for each k, but with individualized actions on $\|\Delta y_k\|$, $\|\Delta w_k\|$, and ϵ_k, respectively. These sequences also allow total freedom in each iteration, given by R_k, to dynamically flow between the bounds on $\|\Delta y_k\|$, $\|\Delta w_k\|$, and ϵ_k throughout the iterations. The input sequence

Algorithm 1: Deviation-based FISTA

Require: $x_0 \in \mathbb{R}^n$, $L_f > 0$, $\{\tau_k\}_{k\geq 1} \subset (0,1]$, $\alpha \in [0,1]$, $R_0 \geq 0$, three sequences $\{\beta_{1,k}\}_{k\geq 1}, \{\beta_{2,k}\}_{k\geq 1}, \{\beta_{3,k}\}_{k\geq 1} \subset \mathbb{R}_+$ such that $\beta_{1,k} + \beta_{2,k} + \beta_{3,k} \leq 1$ for each $k \geq 1$.

1: $y_1 = x_0$, $t_1 = 1$
2: **for** $k = 1, 2, \ldots$ **do**
3: $\quad \rho_k = \frac{\tau_k}{L_f}$
4: $\quad$ Let $\Delta y_k \in \mathbb{R}^n$ be such that $\frac{(1-\tau_k)\tau_k}{2\rho_k}\|\Delta y_k\|^2 \leq \beta_{1,k} R_{k-1}$
5: $\quad \tilde{y}_k = y_k + (1-\tau_k)\Delta y_k$
6: $\quad$ Let $\Delta w_k \in \mathbb{R}^n$ be such that $\frac{1}{2\rho_k}\|\Delta w_k\|^2 \leq \beta_{2,k} R_{k-1}$
7: $\quad w_k = y_k - \rho_k \nabla f(\tilde{y}_k) + \Delta w_k$
8: $\quad \epsilon_k = \beta_{3,k} R_{k-1}$
9: $\quad z_k \in J_g(w_k, \rho_k, \epsilon_k)$
10: $\quad$ Let $\Delta z_k \in \mathbb{R}^n$ be such that $F(z_k + \Delta z_k) \leq F(z_k)$
11: $\quad x_k = z_k + \Delta z_k$
12: $\quad t_{k+1} = \frac{1+\sqrt{1+4\frac{\tau_k}{\tau_{k+1}} t_k^2}}{2}$
13: $\quad y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}) - \frac{t_k}{t_{k+1}}(\Delta z_k + \Delta w_k)$
14: $\quad R_k = (1-\alpha)\frac{\tau_k t_k^2}{\tau_{k+1} t_{k+1}^2}\frac{1-\tau_k}{2\rho_k}\|z_k - y_k + \tau_k \Delta y_k\|_2^2$
15: **end for**

$\{\tau_k\}_{k\geq 1}$ also acts as a relative tolerance parameter for each iteration (see line 14). On the other hand, the value of τ_k also determines the shrinkage of the step size (see line 3).

In terms of computational costs, Algorithm 1 has one mandatory evaluation of ∇f at $\tilde{y}_k$ in line 7 and one mandatory evaluation of J_g at w_k in line 9. The evaluations of $F(z_k + \Delta z_k)$ and $F(z_k)$ in line 10 are not required if $\Delta z_k = 0$ for all k. If the MFISTA step $x_k = \operatorname{argmin}\{F(x) : x \in \{x_{k-1}, z_k\}\}$ is used, then we have that $\Delta z_k = 0$ when $x_k = z_k$ and $\Delta z_k = x_{k-1} - z_k$ when $x_k = x_{k-1}$. Therefore, only the evaluation at z_k is required because $F(z_k + \Delta z_k) = F(x_{k-1})$ has already been computed in the previous iteration.

3 Convergence results

In this section the convergence properties of Algorithm 1 are analyzed under Assumptions A1–A3. The main result is stated in Theorem 3.1, in which the non-asymptotic convergence order $O(1/k^2)$ on the optimality gap is proved.

The lemma below is an extension of the descent result of FISTA-like methods, such as in [7, Lem. 2.3] and [8, Prop. 1], in the presence of deviations and inexact proximal evaluation in Algorithm 1.

Lemma 3.1 *Assume that f and g satisfy Assumptions A1 and A2. Let $\{x_k\}_{k\geq 1}$, $\{y_k\}_{k\geq 1}$, $\{z_k\}_{k\geq 1}$, $\{\Delta y_k\}_{k\geq 1}$, $\{\Delta w_k\}_{k\geq 1}$, $\{\epsilon_k\}_{k\geq 1}$ be given by Algorithm 1. Then for every $x \in \mathbb{R}^n$ and*

$k \geq 1$ *one has*

$$\begin{aligned} F(x) - F(x_k) \quad \geq \quad & \frac{1}{2\rho_k}\Big(2\left\langle y_k - (z_k - \Delta w_k), x - (z_k - \Delta w_k)\right\rangle + \\ & -\|y_k - (z_k - \Delta w_k)\|_2^2\Big) + r_k - c_k, \end{aligned} \tag{9}$$

where

$$c_k \coloneqq \frac{(1-\tau_k)\tau_k}{2\rho_k}\|\Delta y_k\|_2^2 + \frac{1}{2\rho_k}\|\Delta w_k\|_2^2 + \epsilon_k, \tag{10}$$

$$r_k \coloneqq \frac{1-\tau_k}{2\rho_k}\|z_k - y_k + \tau_k \Delta y_k\|_2^2. \tag{11}$$

Proof: Let $x \in \mathbb{R}^n$ be an arbitrary point and let $k \geq 1$. From line 9 of Algorithm 1 and definitions (6) and (7) we have that

$$g(x) \geq g(z_k) + \left\langle \frac{1}{\rho_k}(w_k - z_k), x - z_k \right\rangle - \epsilon_k, \tag{12}$$

while from the convexity of f it follows that

$$f(x) \geq f(\tilde{y}_k) + \left\langle \nabla f(\tilde{y}_k), x - \tilde{y}_k \right\rangle. \tag{13}$$

By summing (12) and (13) one has

$$g(x) + f(x) \geq g(z_k) + f(\tilde{y}_k) + \left\langle \frac{1}{\rho_k}(w_k - z_k), x - z_k \right\rangle + \left\langle \nabla f(\tilde{y}_k), x - \tilde{y}_k \right\rangle - \epsilon_k,$$

and rearranging the terms and noticing that $F = f + g$, it follows that

$$\begin{aligned} F(x) - F(z_k) \quad \geq \quad & f(\tilde{y}_k) + \left\langle \nabla f(\tilde{y}_k), z_k - \tilde{y}_k \right\rangle - f(z_k) + \\ & + \left\langle \tfrac{1}{\rho_k}(w_k - z_k) + \nabla f(\tilde{y}_k), x - z_k \right\rangle - \epsilon_k. \end{aligned}$$

By the descent lemma (see [5, Lem. 5.7, p. 109]), it follows that

$$F(x) - F(z_k) \geq -\frac{L_f}{2}\|z_k - \tilde{y}_k\|_2^2 + \left\langle \frac{1}{\rho_k}(w_k - z_k) + \nabla f(\tilde{y}_k), x - z_k \right\rangle - \epsilon_k.$$

From lines 3 and 7 of Algorithm 1, we have that

$$F(x) - F(z_k) \geq -\frac{\tau_k}{2\rho_k}\|z_k - \tilde{y}_k\|_2^2 + \frac{1}{\rho_k}\left\langle y_k - z_k, x - z_k \right\rangle + \frac{1}{\rho_k}\left\langle \Delta w_k, x - z_k \right\rangle - \epsilon_k.$$

Since

$$\frac{1}{\rho_k}\left\langle y_k - z_k, x - z_k \right\rangle = \frac{1}{2\rho_k}\left(\|y_k - z_k\|_2^2 + \|x - z_k\|_2^2 - \|y_k - x\|_2^2\right),$$

we have that

$$F(x) - F(z_k) \geq -\frac{\tau_k}{2\rho_k}\|z_k - \tilde{y}_k\|_2^2 + \frac{1}{2\rho_k}\left(\|y_k - z_k\|_2^2 + \|x - z_k\|_2^2 - \|y_k - x\|_2^2\right) + \frac{1}{\rho_k}\langle \Delta w_k, x - z_k\rangle - \epsilon_k.$$

By line 5 of Algorithm 1, it follows that

$$\|z_k - \tilde{y}_k\|_2^2 = \|z_k - y_k\|_2^2 - 2\langle z_k - y_k, (1-\tau_k)\Delta y_k\rangle + \|(1-\tau_k)\Delta y_k\|_2^2,$$

which implies that

$$F(x) - F(z_k) \geq \frac{1-\tau_k}{2\rho_k}\|z_k - y_k\|_2^2 + \frac{1}{2\rho_k}\left(\|x - z_k\|_2^2 - \|y_k - x\|_2^2\right) + \frac{\tau_k}{\rho_k}\langle z_k - y_k, (1-\tau_k)\Delta y_k\rangle - \frac{\tau_k}{2\rho_k}\|(1-\tau_k)\Delta y_k\|_2^2 + \frac{1}{\rho_k}\langle \Delta w_k, x - z_k\rangle - \epsilon_k.$$

Since

$$\|x - z_k\|_2^2 + 2\langle \Delta w_k, x - z_k\rangle = \|(z_k - x) - \Delta w_k\|_2^2 - \|\Delta w_k\|_2^2,$$

we have that

$$F(x) - F(z_k) \geq \frac{1-\tau_k}{2\rho_k}\|z_k - y_k\|_2^2 + \frac{1}{2\rho_k}\left(\|(z_k - x) - \Delta w_k\|_2^2 - \|y_k - x\|_2^2\right) + \frac{\tau_k}{\rho_k}\langle z_k - y_k, (1-\tau_k)\Delta y_k\rangle - \frac{\tau_k}{2\rho_k}\|(1-\tau_k)\Delta y_k\|_2^2 - \frac{1}{2\rho_k}\|\Delta w_k\|_2^2 - \epsilon_k.$$

Moreover, since

$$\begin{aligned}\frac{\tau_k}{\rho_k}\langle z_k - y_k, (1-\tau_k)\Delta y_k\rangle &= \frac{1-\tau_k}{\rho_k}\langle z_k - y_k, \tau_k \Delta y_k\rangle \\ &= \frac{1-\tau_k}{2\rho_k}\Big(\|z_k - y_k + \tau_k\Delta y_k\|_2^2 - \|z_k - y_k\|_2^2 - \|\tau_k \Delta y_k\|_2^2\Big),\end{aligned}$$

it follows that

$$\begin{aligned}F(x) - F(z_k) &\geq \frac{1-\tau_k}{2\rho_k}\|z_k - y_k + \tau_k\Delta y_k\|_2^2 + \frac{1}{2\rho_k}\left(\|(z_k - x) - \Delta w_k\|_2^2 - \|y_k - x\|_2^2\right) + \\ &\quad - \frac{1-\tau_k}{2\rho_k}\|\tau_k\Delta y_k\|_2^2 - \frac{\tau_k}{2\rho_k}\|(1-\tau_k)\Delta y_k\|_2^2 - \frac{1}{2\rho_k}\|\Delta w_k\|_2^2 - \epsilon_k. \\ &= \frac{1-\tau_k}{2\rho_k}\|z_k - y_k + \tau_k\Delta y_k\|_2^2 + \frac{1}{2\rho_k}\left(\|(z_k - x) - \Delta w_k\|_2^2 - \|y_k - x\|_2^2\right) + \\ &\quad - \left(\frac{(1-\tau_k)\tau_k}{2\rho_k}\|\Delta y_k\|_2^2 + \frac{1}{2\rho_k}\|\Delta w_k\|_2^2 + \epsilon_k\right).\end{aligned}$$

Thus, by lines 10 and 11 of Algorithm 1 and the definitions of c_k and r_k in (10) and (11), respectively, it holds that

$$F(x) - F(x_k) \geq \frac{1}{2\rho_k}\left(\|z_k - \Delta w_k - x\|_2^2 - \|y_k - x\|_2^2\right) + r_k - c_k,$$

and (9) follows from the fact that

$$\|z_k - \Delta w_k - x\|_2^2 - \|y_k - x\|_2^2 = 2\langle y_k - (z_k - \Delta w_k), x - (z_k - \Delta w_k)\rangle + \\ -\|y_k - (z_k - \Delta w_k)\|_2^2.$$

□

The three lemmas that follow present technical results that will be used to prove the main convergence result in the sequel.

Lemma 3.2 *The sequences $\{t_k\}_{k\geq 1}$ and $\{\rho_k\}_{k\geq 1}$ defined in Algorithm 1 satisfy*

$$\rho_k t_k^2 \geq \frac{k^2}{4L_f}\overline{\tau}_k, \quad \forall k \geq 1, \tag{14}$$

in which

$$\overline{\tau}_k \coloneqq \left(\frac{\sum_{i=1}^{k}\sqrt{\tau_i}}{k}\right)^2, \quad \forall k \geq 1. \tag{15}$$

Proof: Let us prove (14) by induction. For the base case $k = 1$, since $t_1 = 1$ and $\rho_1 = \tau_1/L_f$ (see lines 1 and 3 of Algorithm 1), inequality (14) reduces to $\tau_1 \geq \overline{\tau}_1/4$, which holds by the definition of $\overline{\tau}_1$ in (15). For the inductive step, assume that (14) holds for some $k \geq 1$. By the definition of t_{k+1} in line 12 of Algorithm 1, the relation between ρ_k and τ_k in line 3 of Algorithm 1, and the inductive hypothesis, we have that

$$t_{k+1} = \frac{1 + \sqrt{1 + 4\frac{\tau_k}{\tau_{k+1}}t_k^2}}{2} = \frac{1 + \sqrt{1 + 4\frac{\rho_k}{\rho_{k+1}}t_k^2}}{2} \geq \frac{1 + \sqrt{1 + 4\frac{1}{\rho_{k+1}}\frac{k^2}{4L_f}\overline{\tau}_k}}{2},$$

respectively. Thus, it holds that

$$t_{k+1} \geq \frac{1}{2} + \frac{k}{2}\sqrt{\frac{\overline{\tau}_k}{\rho_{k+1}L_f}}.$$

By multiplying the above expression by $\sqrt{\rho_k}$ one has

$$\sqrt{\rho_{k+1}}t_{k+1} \geq \frac{1}{2\sqrt{L_f}}\left(\sqrt{L_f\rho_{k+1}} + k\sqrt{\overline{\tau}_k}\right).$$

From line 3 of Algorithm 1 and the definition of $\overline{\tau}_{k+1}$ in (15) we have that

$$\sqrt{\rho_{k+1}}t_{k+1} \geq \frac{1}{2\sqrt{L_f}}(k+1)\sqrt{\overline{\tau}_{k+1}}.$$

□

Lemma 3.3 *Suppose that the assumptions in Lemma 3.1, together with Assumption A3, hold. Then for all $k \geq 1$ and $x_* \in X_*$ one has*

$$\begin{aligned} 2\rho_{k-1}t_{k-1}^2(F(x_{k-1}) - F_*) - 2\rho_k t_k^2(F(x_k) - F_*) \geq \\ \|u_k(x_*)\|_2^2 - \|u_{k-1}(x_*)\|_2^2 + 2\rho_k t_k^2(r_k - c_k), \end{aligned} \tag{16}$$

in which

$$\rho_0 \coloneqq 0,\ t_0 \coloneqq 0, \tag{17a}$$

$$u_0(x_*) \coloneqq x_0 - x_*, \tag{17b}$$

$$u_k(x_*) \coloneqq t_k(z_k - \Delta w_k) - (t_k - 1)x_{k-1} - x_*,\ \textit{for all } k \geq 1. \tag{17c}$$

Proof: Let $x_* \in X_*$ and $k \geq 1$. By considering (9) with $x = x_*$ and with $x = x_{k-1}$, we obtain

$$\begin{aligned} F(x_*) - F(x_k) \quad \geq \quad \tfrac{1}{2\rho_k}\Big(2\left\langle y_k - (z_k - \Delta w_k), x_* - (z_k - \Delta w_k)\right\rangle + \\ -\|y_k - (z_k - \Delta w_k)\|_2^2\Big) + r_k - c_k, \end{aligned} \tag{18}$$

and

$$\begin{aligned} F(x_{k-1}) - F(x_k) \quad \geq \quad \tfrac{1}{2\rho_k}\Big(2\left\langle y_k - (z_k - \Delta w_k), x_{k-1} - (z_k - \Delta w_k)\right\rangle + \\ -\|y_k - (z_k - \Delta w_k)\|_2^2\Big) + r_k - c_k, \end{aligned} \tag{19}$$

respectively. By lines 1 and 12 of Algorithm 1 it holds that $t_k \geq 1$. Moreover, we have that the stepsize is positive, i.e., $\rho_k > 0$. Thus, adding (18) and (19) multiplied by $t_k - 1$ and multiplying the result by $2\rho_k t_k$, we obtain that

$$\begin{aligned} 2\rho_k t_k(t_k - 1)(F(x_{k-1}) - F_*) - 2\rho_k t_k^2(F(x_k) - F_*) \geq \\ 2\left\langle t_k(y_k - (z_k - \Delta w_k)), x_* + (t_k - 1)x_{k-1} - t_k(z_k - \Delta w_k)\right\rangle + \\ -\|t_k(y_k - (z_k - \Delta w_k))\|_2^2 + 2\rho_k t_k^2(r_k - c_k). \end{aligned} \tag{20}$$

Note that, by $\rho_0 = t_0 = 0$ in (17a), the definition $t_1 = 1$ in line 1, and the definitions of ρ_k and t_{k+1} for all $k \geq 1$ in lines 3 and 12 of Algorithm 1, respectively, we have that, for all $k \geq 1$,

$$\rho_k t_k(t_k - 1) = \rho_{k-1}t_{k-1}^2. \tag{21}$$

Applying (21) in (20), one has

$$\begin{aligned} 2\rho_{k-1}t_{k-1}^2(F(x_{k-1}) - F_*) - 2\rho_k t_k^2(F(x_k) - F_*) \geq \\ 2\left\langle t_k(y_k - (z_k - \Delta w_k)), x_* + (t_k - 1)x_{k-1} - t_k(z_k - \Delta w_k)\right\rangle + \\ -\|t_k(y_k - (z_k - \Delta w_k))\|_2^2 + 2\rho_k t_k^2(r_k - c_k). \end{aligned}$$

Rearranging the term in the right hand side, it follows that

$$\begin{aligned} 2\rho_{k-1}t_{k-1}^2(F(x_{k-1}) - F_*) - 2\rho_k t_k^2(F(x_k) - F_*) \geq \\ - \|t_k y_k - (x_* + (t_k - 1)x_{k-1})\|_2^2 + \|x_* + (t_k - 1)x_{k-1} - t_k(z_k - \Delta w_k)\|_2^2 \\ + 2\rho_k t_k^2(r_k - c_k). \end{aligned}$$

By the definition of $u_k(x_*)$ in (17c), we have that

$$\begin{aligned} 2\rho_{k-1}t_{k-1}^2(F(x_{k-1})-F_*) - 2\rho_k t_k^2(F(x_k)-F_*) \geq \\ -\|t_k y_k - (x_* + (t_k-1)x_{k-1})\|_2^2 + \|u_k(x_*)\|_2^2 + 2\rho_k t_k^2(r_k - c_k). \end{aligned} \tag{22}$$

Therefore, it only remains to prove that $t_k y_k - (x_* + (t_k - 1)x_{k-1}) = u_{k-1}(x_*)$ for all $k \geq 1$. In fact, if $k = 1$, then by line 1 of Algorithm 1 we have that

$$t_1 y_1 - (x_* + (t_1 - 1)x_0) = x_0 - x_* = u_0(x_*),$$

in which the last equation follows from the definition of $u_0(x_*)$ in (17b). On the other hand, if $k \geq 2$, then, by line 13 of Algorithm 1, we have that

$$\begin{aligned} t_k y_k - (x_* + (t_k-1)x_{k-1}) = t_k x_{k-1} + (t_{k-1}-1)(x_{k-1} - x_{k-2}) + \\ -t_{k-1}(\Delta z_{k-1} + \Delta w_{k-1}) - (x_* + (t_k - 1)x_{k-1}), \end{aligned}$$

and, by line 11 of Algorithm 1, we conclude that

$$t_k y_k - (x_* + (t_k-1)x_{k-1}) = t_{k-1}(z_{k-1} - \Delta w_{k-1}) - (t_{k-1}-1)x_{k-2} - x_* = u_{k-1}(x_*).$$

Thus, by replacing $t_k y_k - (x_* + (t_k - 1)x_{k-1}) = u_{k-1}(x_*)$ in (22), we obtain (16). □

Lemma 3.4 *Suppose that the assumptions in Lemma 3.3 hold and let ρ_0, t_0, $u_0(x^*)$, and $u_k(x^*)$ be defined as in* (17)*. Then for all $k \geq 1$ and $x_* \in X_*$, it follows that*

$$\begin{aligned} 2\rho_k t_k^2(F(x_k)-F_*) + \|u_k(x_*)\|_2^2 + 2(1-\alpha)\rho_k t_k^2 r_k + 2\alpha\sum_{i=1}^{k}\rho_i t_i^2 r_i \leq \\ \|x_0 - x_*\|_2^2 + 2\rho_1 R_0. \end{aligned} \tag{23}$$

Proof: First, let us prove that, for all $k \geq 1$,

$$c_k \leq R_{k-1} \tag{24}$$

and, for all $k \geq 2$,

$$\rho_k t_k^2 c_k \leq (1-\alpha)\rho_{k-1}t_{k-1}^2 r_{k-1}. \tag{25}$$

In fact, by lines 4, 6, and 8 of Algorithm 1, the definition of c_k in (10), and the fact that $\beta_{1,k} + \beta_{2,k} + \beta_{3,k} \leq 1$ for each $k \geq 1$, we have that (24) holds. Furthermore, by the definition of ρ_k and R_k for each $k \geq 1$ in lines 3 and 14 of Algorithm 1, respectively, the definition of r_k in (11), and (24), it follows that (25) holds.

Now, in order to prove (23), let $k \geq 1$, and consider the sum of (16) in Lemma 3.3 from iteration 1 up to iteration k. By identifying the telescoping sums in the right and left-hand sides of the resulting inequality, we have that

$$2\rho_0 t_0^2(F(x_0)-F_*) - 2\rho_k t_k^2(F(x_k)-F_*) \geq -\|u_0(x_*)\|_2^2 + \|u_k(x_*)\|_2^2 + 2\sum_{i=1}^{k}\rho_i t_i^2(r_i - c_i).$$

By their definitions in (17a) and (17b), $t_0 = \rho_0 = 0$ and $u_0(x_*) = x_0 - x_*$. Thus, it follows that

$$-2\rho_k t_k^2(F(x_k) - F_*) \geq -\|x_0 - x_*\|_2^2 + \|u_k(x_*)\|_2^2 + 2\sum_{i=1}^{k} \rho_i t_i^2(r_i - c_i).$$

From (25), we have that

$$\begin{aligned}
&\sum_{i=1}^{k} \rho_i t_i^2(r_i - c_i) \\
&\quad \geq \quad \rho_1 t_1^2(r_1 - c_1) + \sum_{i=2}^{k} \Big(\rho_i t_i^2 r_i - (1-\alpha)\rho_{i-1} t_{i-1}^2 r_{i-1}\Big) \\
&\quad = \quad \rho_1 t_1^2(r_1 - c_1) + \sum_{i=2}^{k} \big(\rho_i t_i^2 r_i - \rho_{i-1} t_{i-1}^2 r_{i-1}\big) + \alpha \sum_{i=2}^{k} \rho_{i-1} t_{i-1}^2 r_{i-1} \\
&\quad = \quad \rho_1 t_1^2(r_1 - c_1) + \rho_k t_k^2 r_k - \rho_1 t_1^2 r_1 + \alpha \sum_{i=1}^{k-1} \rho_i t_i^2 r_i \\
&\quad = \quad -\rho_1 t_1^2 c_1 + \rho_k t_k^2 r_k + \alpha \sum_{i=1}^{k-1} \rho_i t_i^2 r_i \\
&\quad = \quad -\rho_1 t_1^2 c_1 + (1-\alpha)\rho_k t_k^2 r_k + \alpha \sum_{i=1}^{k} \rho_i t_i^2 r_i,
\end{aligned}$$

which implies that

$$\begin{aligned}
&-2\rho_k t_k^2(F(x_k) - F_*) \geq \\
&\quad -\|x_0 - x_*\|_2^2 + \|u_k(x_*)\|_2^2 - 2\rho_1 t_1^2 c_1 + 2(1-\alpha)\rho_k t_k^2 r_k + 2\alpha \textstyle\sum_{i=1}^{k} \rho_i t_i^2 r_i.
\end{aligned}$$

By line 1 of Algorithm 1, we have that $t_1 = 1$. Moreover, by (24), we have that $c_1 \leq R_0$. Thus, by rearranging the terms in the last inequality, we have that (23) holds. □

Theorem 3.1 below is the main convergence result, which guarantees the $O(1/k^2)$ order of convergence on the optimality gap under suitable assumptions on the sequence $\{\tau_k\}_{k\geq 1}$.

Theorem 3.1 *Suppose that the assumptions in Lemma 3.3 hold and let ρ_0, t_0, $u_0(x^*)$, and $u_k(x^*)$ be defined as in* (17)*. Then for all $k \geq 1$ and $x_* \in X_*$, it follows that*

$$F(x_k) - F_* \leq \frac{2}{k^2} \frac{L_f}{\overline{\tau}_k} \Big(\|x_0 - x_*\|_2^2 + 2\rho_1 R_0\Big). \tag{26}$$

Proof: Let $k \geq 1$ and $x_* \in X_*$. Since $\|u_k(x_*)\|_2^2 \geq 0$, $2(1-\alpha)\rho_k t_k^2 r_k \geq 0$, and $2\alpha \sum_{i=1}^{k} \rho_i t_i^2 r_i \geq 0$, by (23) in Lemma 3.4, we have that

$$2\rho_k t_k^2(F(x_k) - F_*) \leq \|x_0 - x_*\|_2^2 + 2\rho_1 R_0.$$

Therefore, by Lemma 3.2, it follows that (26) holds. □

If $\{\tau_k\}_{k\geq 1}$ is lower bounded by a constant $\tau_{\min} > 0$, by the definition of $\overline{\tau}_k$ in (15), we have that $\overline{\tau}_k \geq \tau_{\min}$ for each $k \geq 1$. Thus, by (26), for each $k \geq 1$ it holds that

$$F(x_k) - F_* \leq \frac{2}{k^2}\frac{L_f}{\tau_{\min}}\Big(\|x_0 - x_*\|_2^2 + 2\rho_1 R_0\Big), \tag{27}$$

which is a non-asymptotic convergence result, in the sense that it holds for each $k \geq 1$. On the other hand, if $\{\tau_k\}_{k\geq 1}$ converges to $\tilde{\tau} > 0$, by the definition of $\overline{\tau}_k$ in (15), we have that $\{\overline{\tau}_k\}_{k\geq 1}$ also converges to $\tilde{\tau}$. From (26) it follows that

$$F(x_k) - F_* \leq \left[\frac{2L_f}{k^2\tilde{\tau}} + \frac{2L_f}{k^2}\left(\frac{1}{\overline{\tau}_k} - \frac{1}{\tilde{\tau}}\right)\right]\Big(\|x_0 - x_*\|_2^2 + 2\rho_1 R_0\Big).$$

Since $\overline{\tau}_k \xrightarrow{k\to\infty} \tilde{\tau}$, we have that

$$\frac{2L_f}{k^2}\left(\frac{1}{\overline{\tau}_k} - \frac{1}{\tilde{\tau}}\right) = o\left(\frac{1}{k^2}\right),$$

which implies that, for sufficiently large k, one has

$$F(x_k) - F_* \lessapprox \frac{2L_f}{k^2\tilde{\tau}}\Big(\|x_0 - x_*\|_2^2 + 2\rho_1 R_0\Big). \tag{28}$$

This is the asymptotic behavior of the optimality gap, in the sense that it holds for $k \gg 1$. By comparing Theorem 3.1 with FISTA's optimality gap convergence result in [7, Thm. 4.4], one observes that they are, in the worst case, analogous up to a constant.

The following theorem examines the convergence of the sequence $\{R_k\}_{k\geq 1}$, responsible for the bounds on $\|\Delta y_k\|$, $\|\Delta w_k\|$, and ϵ_k; see lines 4, 6, and 8 in Algorithm 1. It states that R_k decays as fast as the optimality gap (compare (26) and (29)). Moreover, under suitable assumptions, the asymptotic convergence of R_k to zero is faster than that of the optimality gap (see (30)).

Theorem 3.2 *Suppose that the assumptions in Lemma 3.3 hold and let ρ_0, t_0, $u_0(x^*)$, and $u_k(x^*)$ be defined as in* (17)*. Then for all $k \geq 1$ and $x_* \in X_*$, it follows that*

$$R_k \leq \frac{2}{(k+1)^2}\frac{L_f}{\overline{\tau}_{k+1}}\Big(\|x_0 - x_*\|_2^2 + 2\rho_1 R_0\Big). \tag{29}$$

Moreover, if $\alpha > 0$ and $\inf \tau_k > 0$ then

$$R_k = o\left(\frac{1}{k^3}\right). \tag{30}$$

Proof: Let $k \geq 1$ and $x_* \in X_*$. Since $\|u_k(x_*)\|_2^2 \geq 0$ and $2\rho_k t_k^2(F(x_k) - F_*) \geq 0$, by (23) in Lemma 3.4, we have that

$$2(1-\alpha)\rho_k t_k^2 r_k + 2\alpha\sum_{i=1}^{k}\rho_i t_i^2 r_i \leq \|x_0 - x_*\|_2^2 + 2\rho_1 R_0. \tag{31}$$

Moreover, from lines 3 and 14 of Algorithm 1 and the definition of r_k in (11) it follows that

$$\rho_{k+1} t_{k+1}^2 R_k = (1-\alpha)\rho_k t_k^2 r_k. \tag{32}$$

In order to prove (29), replace (32) in (31) and notice that $2\alpha \sum_{i=1}^{k} \rho_i t_i^2 r_i \geq 0$, which implies that

$$2\rho_{k+1} t_{k+1}^2 R_k \leq \|x_0 - x_*\|_2^2 + 2\rho_1 R_0.$$

Thus, by Lemma 3.2 it follows that (29) holds.

On the other hand, in order to prove (30), notice that by (14) and (32) it holds that

$$\frac{(k+1)^2}{4L_f} \overline{\tau}_{k+1} R_k \leq (1-\alpha)\rho_k t_k^2 r_k.$$

By the definition of $\overline{\tau}_{k+1}$ in (15), we have that $\overline{\tau}_{k+1} \geq \inf \tau_k$, which implies that

$$(\inf \tau_k) \frac{(k+1)^2}{4L_f} R_k \leq (1-\alpha)\rho_k t_k^2 r_k. \tag{33}$$

Furthermore, since $2(1-\alpha)\rho_k t_k^2 r_k \geq 0$ in (31), we have that

$$2\alpha \sum_{i=1}^{k} \rho_i t_i^2 r_i \leq \|x_0 - x_*\|_2^2 + 2\rho_1 R_0.$$

Since $\alpha > 0$, we have that $\sum_{i=1}^{k} \rho_i t_i^2 r_i$ is bounded by a constant for each $k \geq 1$. Therefore, it holds that $\{\rho_k t_k^2 r_k\}_{k\geq 1}$ is summable, which implies that $\rho_k t_k^2 r_k = o(1/k)$. Finally, since $\inf \tau_k > 0$ and $\rho_k t_k^2 r_k = o(1/k)$ in (33), we can conclude that $R_k = o(1/k^3)$. □

4 Numerical experiments

In this section, we describe numerical experiments designed to validate the proposed algorithm in practical tomographic image reconstruction settings. We consider two types of regularization: ℓ_1-based regularization and Total Variation (TV). The former admits a closed-form proximal operator, which is a requirement of the method introduced in [2] for nonsmooth-regularized least-squares inversion, making a direct comparison with that approach possible. In contrast, Total Variation regularization does not admit a closed-form proximal operator, so its evaluation relies on numerical approximation. This setting is used to illustrate the flexibility of our Algorithm 1, which does not require a closed-form proximal operator.

The remainder of this section is organized as follows. In Section 4.1, we describe the two problems in detail, explain how they fit within the class of problems addressed by Algorithm 2.1, and discuss the computation of the exact and approximate proximal operators in each case. Section 4.2 is devoted to the procedure used to learn the deviations in Algorithm 1. Implementation details are provided in Section 4.3, and the experimental results are presented in Section 4.4.

4.1 Applications to linear inverse problems

A discretized linear inverse problem is written as

$$Ax = b_{\text{exact}}, \tag{34}$$

in which $A : \mathbb{R}^n \to \mathbb{R}^m$ is the linear discretized forward operator, which models the direct problem, $x \in \mathbb{R}^n$ is the sought solution, and $b_{\text{exact}} \in \mathbb{R}^m$ is the exact data. However, only a noisy version $b \in \mathbb{R}^m$ of the exact data is known in practice. Thus, due to the ill-conditioning of the operator A inherited from the ill-posedness of A's infinite-dimension counterpart, one should consider regularization methods to solve (34) (see e.g. [12, Chap. 3–5]).

The variational regularization approach involves solving the optimization problem

$$\min_{x\in\mathbb{R}^n} d(Ax, b) + \lambda r(x), \tag{35}$$

in which $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is the data fidelity measure, $r : \mathbb{R}^n \to \mathbb{R}$ is the regularization measure, and $\lambda > 0$ is the regularization parameter. The data fidelity term $d(Ax, b)$ induces $Ax \approx b$, whereas the regularization term $r(x)$ induces an a priori knowledge on the solution. The regularization parameter λ balances these two objectives. In many cases of interest, problem (35) fits into the framework of problem (1) under Assumptions A1–A3, which implies that Algorithm 1 can be applied.

In this paper we consider two variational regularization approaches for solving the 2D computed tomography problem. First, we use the ℓ_1–wavelet regularized least squares model:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|DWx\|_1, \tag{36}$$

in which $A : \mathbb{R}^n \to \mathbb{R}^m$ is the discretized Radon forward operator (or ray tracing operator), $b \in \mathbb{R}^m$ is the observed sinogram, $W : \mathbb{R}^n \to \mathbb{R}^n$ is the 2D discrete wavelet transform from the Symlets family with 5 vanishing moments and 5 levels of decomposition, and $D : \mathbb{R}^n \to \mathbb{R}^n$ is such that $Dy = (d_i y_i)_{i=1}^n$ for each $y \in \mathbb{R}^n$. For each $i \in \{1, \dots, n\}$, the coefficients d_i are chosen according to the wavelet decomposition. If $(Wx)_i$ corresponds to a trend component, then $d_i = 1$. Otherwise, if $(Wx)_i$ corresponds to a detail component at level ℓ, then $d_i = 1.8^{\frac{5-\ell+1}{5}}$.

Second, we consider the total variation regularized least squares model:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \lambda\,\mathrm{TV}(x), \tag{37}$$

in which $A : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$ are as in (36), and $\mathrm{TV} : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\begin{aligned}\mathrm{TV}(x) \coloneqq &\sum_{i=1}^{p-1}\sum_{j=1}^{q-1}\Big\{|X_{i+1,j} - X_{i,j}| + |X_{i,j+1} - X_{i,j}|\Big\} \\ &+ \sum_{j=1}^{q-1}\Big\{|X_{p,j}| + |X_{p,j+1} - X_{p,j}|\Big\} + \sum_{i=1}^{p-1}\Big\{|X_{i+1,q} - X_{i,q}| + |X_{i,q}|\Big\} + 2|X_{p,q}|,\end{aligned} \tag{38}$$

where the vector $x \in \mathbb{R}^n$ with $n = pq$ is the stacking of the columns (or rows) of a $p \times q$ image X, e.g., obtained by identifying $x_{(i-1)q+j} = X_{i,j}$. Both optimization problems given by (36) and (37) have the structure of problem (1) under Assumptions A1–A3. Indeed, considering $f(x) \coloneqq \frac{1}{2}\|Ax-b\|_2^2$, one has that f is convex and $\|A^TA\|_{2,2}$-smooth (see [5, Example 5.2]). Both $g_1(x) \coloneqq \lambda\|DWx\|_1$ and $g_2(x) \coloneqq \lambda\,\mathrm{TV}(x)$ are continuous real-valued functions, which implies that they are proper and closed. In addition to that, functions g_1 and g_2 are convex, since they can be written as the composition of a linear transform with the sum of absolute value functions multiplied by a positive scalar. Finally, the existence of a solution for both problems follows from the fact that both g_1 and g_2 are coercive functions.

Therefore, Algorithm 1 can be applied to solve (36) or (37). We now describe how to evaluate ∇f and J_g in practice; this is needed in order to use Algorithm 1. For both problems we have that $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, which implies that $\nabla f(x) = A^T(Ax - b)$. In the experiments we used the Radon Transform implemented in [16], which is a GPU-accelerated implementation with the adjoint's numerical precision in the machine precision level. Computing J_{g_1} for solving (36) amounts to computing the proximal operator of $g_1(x) = \lambda\|DWx\|_1$ exactly. In fact, by rewriting g_1 as $g_1 = h_1 \circ W$, in which $h_1(y) \coloneqq \sum_{i=1}^n \lambda d_i|y_i|$ for each $y \in \mathbb{R}^n$, it is possible to see that g_1 is the composition of an orthogonal linear operator and a separable function. In this manner, considering [5, Thms. 6.6 and 6.15 and Lem. 6.5], one can compute $\mathrm{prox}_{g_1}^{\rho}$ for each $\rho > 0$ as

$$\mathrm{prox}_{g_1}^{\rho}(u) = W^T\Big(\mathcal{T}_{\rho\lambda d_i}\big((Wu)_i\big)\Big)_{i=1}^n, \tag{39}$$

in which $(Wu)_i$ is the i-th component of $Wu \in \mathbb{R}^n$ and

$$\mathcal{T}_\beta(t) \coloneqq \begin{cases} t-\beta, & t > \beta, \\ 0, & |t| \le \beta, \\ t+\beta, & t < -\beta, \end{cases} \tag{40}$$

for each $\beta \ge 0$. Function $\mathcal{T}_\beta$ in (40) is known as the soft-thresholding function.

For problem (37), on the other hand, there is no closed formula for the proximal operator of the function $g_2(x) = \lambda\,\mathrm{TV}(x)$. Nevertheless, since Algorithm 1 supports inexact proximal point evaluations, one may use an efficient iterative scheme to approximate the proximal operator of g_2, such as the methods described in [3, 6]. The implementation must have a stopping criterion that meets the ϵ tolerance of J_g as defined in (6) or the sufficient condition in (8). From the definition of the function TV in (38), it follows that there is a linear transformation $B : \mathbb{R}^n \to \mathbb{R}^{2n}$ such that $\mathrm{TV}(x) = \|Bx\|_1$. Thus, from (8), it suffices to calculate $v \in \mathbb{R}^{2n}$ such that $G(w - \rho B^T v, v) \le \epsilon$ to have $w - \rho B^T v \in J_{g_2}(w, \rho, \epsilon)$. The algorithm proposed in [6] can compute such v by applying FISTA to the dual problem associated with the minimization problem defined in the evaluation of $\mathrm{prox}_{g_2}^{\rho}(w)$. Therefore, in our experiments we integrated the stopping criterion $G(w-\rho B^T v, v) \le \epsilon$ into the algorithm proposed in [6] to compute $J_{g_2}(w, \rho, \epsilon)$.

4.2 Learning the deviations

We now describe the training framework for the deviations of Algorithm 1, which is closely related to the training framework considered in [2]. Learning the deviations in Algorithm 1

consists of choosing appropriate deviation vectors from data to meet a specific objective. The deviations Δy_k and Δw_k, in particular, are well-suited for this purpose, since one only needs to constrain their norms, being completely independent from the oracles f, g, ∇f, or J_g. The deviation Δz_k, on the other hand, is required to be a descent step, which naturally involves the oracles. In preliminary experiments, we identified that this extra computational cost prevents one from learning the deviation Δz_k efficiently. Thus, in our experiments, we set $\Delta z_k = 0$ for each $k \geq 1$.

In the context of learning deviations in an optimization algorithm, the data consists of instances of optimization problems. For a supervised training, one must have a paired data $(\text{problem}, \text{solution})$, whereas, for an unsupervised training, one would only require the optimization problems. For variational regularization of linear inverse problems, by fixing the linear discretized forward operator A, the data fidelity measure d, the regularization parameter λ, and the regularization measure r, the optimization problem in (35) can be parametrized by the observed data b. In this way, the dataset would consist of realizations of the observable data $b \in \mathbb{R}^m$, and each respective optimization problem would have the objective function written as $F_b(x) \coloneqq d(Ax, b) + \lambda r(x)$. For the sake of notation, when identifying F_b in the form of the objective function in (1), we will write $F_b = f_b + g_b$.

Rather than learning the deviations themselves, we aim to learn the parameters of parametrized functions that will generate the deviations. Consider the functions $h^y_{\theta^y} : \mathbb{E}_y \to R^n$ and $h^w_{\theta^w} : \mathbb{E}_w \to R^n$ parameterized by $\theta^y \in \mathbb{R}^{n^y}$ and $\theta^w \in \mathbb{R}^{n^w}$, respectively, in which $\mathbb{E}_y$ and $\mathbb{E}_w$ are finite dimensional spaces. We index the functions and their respective parameters and domains with the letters y and w since they will all be used to compute Δy_k and Δw_k in each iteration k of Algorithm 1, respectively. Moreover, consider a differentiable normalization function $\mathcal{N} : \mathbb{R}^n \times \mathbb{R}_{++} \to \mathbb{R}^n$ such that

$$\|\mathcal{N}(x, \eta)\| \leq \eta, \quad \forall x \in \mathbb{R}^n, \forall \eta \in \mathbb{R}_{++}. \tag{41}$$

A possible choice of normalization function, used in our experiments, is

$$\mathcal{N}(x, \eta) = \frac{\eta}{\sqrt{\|x\|^2 + \eta^2}} x. \tag{42}$$

Consider that $v^y_k \in \mathbb{E}_y$ and $v^w_k \in \mathbb{E}_w$ are the inputs of $h^y_{\theta^y}$ and $h^w_{\theta^w}$ at the k-th iteration of Algorithm 1, respectively. Then, one computes Δy_k and Δw_k as

$$\Delta y_k = \mathcal{N}\left(h^y_{\theta^y}(v^y_k), \sqrt{\frac{2\rho_k \beta_{1,k}}{(1-\tau_k)\tau_k} R_{k-1}}\right) \tag{43}$$

and

$$\Delta w_k = \mathcal{N}\left(h^w_{\theta^w}(v^w_k), \sqrt{2\rho_k \beta_{2,k} R_{k-1}}\right), \tag{44}$$

respectively. In this manner, the bounds on $\|\Delta y_k\|$ and $\|\Delta w_k\|$ respectively specified in lines 4 and 6 of Algorithm 1 are satisfied. Furthermore, the architecture of the functions $h^y_{\theta^y}$ and $h^w_{\theta^w}$ used in our experiments is described in Figure 1. The inputs to $h^y_{\theta^y}$ and $h^w_{\theta^w}$ at the k-th iteration are given by $(y_k, \nabla f(\tilde{y}_{k-1}), \Delta y_{k-1})$ and $(y_k, \nabla f(\tilde{y}_k), \Delta y_k, \Delta w_{k-1})$, respectively. This architecture is equivalent to the one used in [2] for computing Δx^1_k and Δx^2_k in algorithm (5).

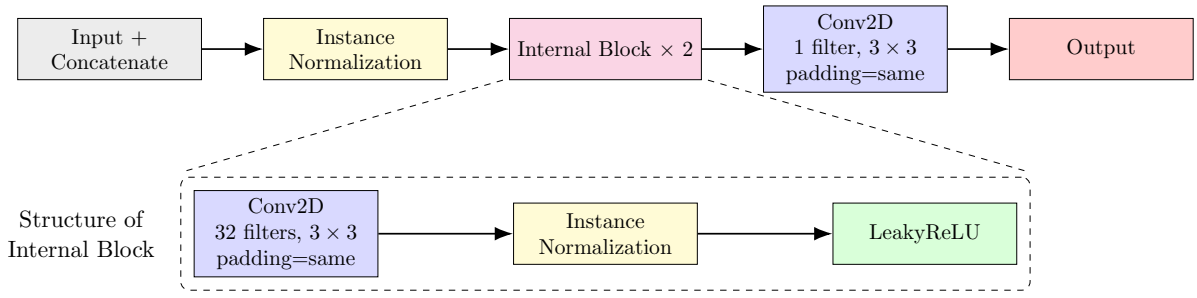


Figure 1: Architecture of the network used in training.

Let $S_{B,N}(\theta) \in \mathbb{R}^n$, with $\theta = (\theta^y, \theta^w)$, denote the N-th iterate produced by Algorithm 1 when minimizing F_B, with the deviations Δy_k and Δw_k computed as in (43) and (44), respectively. Consider the random variable N, uniformly distributed over $[N_{\min}, N_{\max}] \cap \mathbb{Z}$, and let B be a random variable taking values in the set of observable data. We consider the unsupervised training problem defined by

$$\min_{\theta} \mathbb{E}_{B,N} \left[\frac{F_B(S_{B,N}(\theta)) - F_B^\dagger}{F_B(x_{B,N}^{\mathrm{FISTA}}) - F_B^\dagger} \right], \tag{45}$$

where $x_{B,N}^{\mathrm{FISTA}}$ denotes the N-th iterate generated by FISTA applied to minimize F_B, and $F_B^\dagger$ is an approximation of the optimal functional value given by $F_B(x_{B,K}^{\mathrm{FISTA}})$, with $K \gg N_{\max}$. In our experiments, we set $K = 1000$ for problem (36) and $K = 200$ for problem (37). In practice, only a dataset $\{b_i\}_{i=1}^{\ell}$ of observed data is available. The training procedure therefore consists of epochs in which each observed datum is used once to update the parameters θ, possibly within a mini-batch. At the beginning of each epoch, the dataset is shuffled, and at each update step, a number of iterations n is sampled from $[N_{\min}, N_{\max}] \cap \mathbb{Z}$. Algorithm 1 is then executed for n iterations to solve the current problem associated with b, yielding $S_{b,n}(\theta)$, or to solve in parallel the problems corresponding to a mini-batch $Y \subset \{b_i\}_{i=1}^{\ell}$, yielding $S_{b,n}(\theta)$ for each $b \in Y$. The gradient of $\theta \mapsto \frac{F_b(S_{b,n}(\theta)) - F_b^\dagger}{F_b(x_{b,n}^{\mathrm{FISTA}}) - F_b^\dagger}$ is computed by automatic differentiation, and the parameters θ are updated using the chosen optimizer. When mini-batch training is employed, the gradient of $\theta \mapsto \mathrm{mean}_{b \in Y} \left[\frac{F_b(S_{b,n}(\theta)) - F_b^\dagger}{F_b(x_{b,n}^{\mathrm{FISTA}}) - F_b^\dagger} \right]$ is computed instead. In our experiments, we chose Adam as the optimizer, with the standard parameters $\alpha = 10^{-3}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

4.3 Implementation details

For (30) to hold, we need $\alpha > 0$. Experimentally we have observed that the greater the choice of α, the sooner the $o(1/k^3)$ behavior of R_k starts; see Theorem 3.2. In order to tune this parameter and avoid this behavior near the trained iterations horizon, which is associated with poor performance of the algorithm in the transition to the untrained iterations, we tried the values $\alpha \in \{10^{-3.5}, 10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}\}$.

When the proximal operator is computed exactly, such as for solving (36), the input sequence $\{\tau_k\}_{k\geq 1}$ is chosen such that

$$\tau_k = \frac{0.99}{1 + e^{-\gamma(k-1)}}. \tag{46}$$

Notice that $\tau_k \xrightarrow{k\to\infty} 0.99$, which implies, from (28), an asymptotic behavior of the optimality gap similar to the FISTA's behavior. Also, this guarantees that the step size $\rho_k = \frac{\tau_k}{L_f}$ is similar to the FISTA's step size given by $\frac{1}{L_f}$ for sufficiently large k. Moreover, we have that $\inf \tau_k > 0$, which implies that (30) holds. The parameter $\gamma > 0$ in (46) is chosen for the algorithm to have a smooth transition between the training iterations and the non-training iterations. When the proximal operator is computed inexactly, such as for solving (37), we considered the constant sequence $\tau_k = \tau$ for each $k \geq 1$. In our experiments, we studied $\gamma \in \{0.01, 0.025, 0.05, 0.075, 0.1\}$ and $\tau \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$.

The three sequences $\{\beta_{1,k}\}_{k\geq 1}, \{\beta_{2,k}\}_{k\geq 1}, \{\beta_{3,k}\}_{k\geq 1} \subset \mathbb{R}_+$ were chosen such that $\beta_{1,k} + \beta_{2,k} + \beta_{3,k} = 1$ and the upper bounds[1] on $\|(1-\tau)\Delta y_k\|$ and $\|\Delta w_k\|$ coincide for each iteration $k \geq 1$. Therefore, we chose

$$\beta_{1,k} = \tau_k(1 - \beta_3), \tag{47a}$$

$$\beta_{2,k} = (1 - \tau_k)(1 - \beta_3), \tag{47b}$$

$$\beta_{3,k} = \beta_3, \tag{47c}$$

for each $k \geq 1$, in which $\beta_3 \in [0, 1)$ is an input scalar. When the proximal operator is computed exactly, we considered $\beta_3 = 0$. On the other hand, when the proximal operator is computed inexactly, we considered $\beta_3 = 0.5$.

We chose the input parameter R_0 of Algorithm 1 based on the available training data $\{b_i\}_{i=1}^{\ell}$ when the proximal operator is computed exactly. The goal is to set the upper bounds on $\|\Delta w_1\|$ and $\|(1-\tau)\Delta y_1\|$, which coincide by (47), to be equal to the root mean square of the step norm given by the proximal gradient method from x_0:

$$\sqrt{2\rho_1\beta_{2,k}R_0} = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell} L_i \left\| \operatorname{prox}_{g_{b_i}}^{\frac{1}{L_i}}\left(x_0 - \frac{1}{L_i}\nabla f_{b_i}(x_0)\right) - x_0 \right\|^2},$$

in which L_i is the smoothness parameter of f_{b_i} for $i \in \{1, \ldots, \ell\}$. Substituting (47b) with $\beta_3 = 0$ in the above expression and solving for R_0 we have that

$$R_0 = \frac{1}{2\rho_1(1-\tau_1)}\frac{1}{\ell}\sum_{i=1}^{\ell} L_i \left\| \operatorname{prox}_{g_{b_i}}^{\frac{1}{L_i}}\left(x_0 - \frac{1}{L_i}\nabla f_{b_i}(x_0)\right) - x_0 \right\|^2.$$

On the other hand, when the proximal operator is computed inexactly, we chose R_0 large enough so that the first approximation produced by the inner algorithm is accepted. The chosen value was $R_0 = 10^7$.

[1]We consider the upper bound on $\|(1-\tau)\Delta y_k\|$ rather than the upper bound on $\|\Delta y_k\|$ since the actual deviation in line 5 of Algorithm 1 is given by $(1-\tau)\Delta y_k$.

4.4 Results

In this section, we present the results of the numerical experiments. Section 4.4.1 is devoted to the ℓ_1-wavelet regularized least-squares model (problem (36)), for which the proximal operator can be evaluated exactly. Section 4.4.2 considers the total variation regularized least-squares model (problem (37)), where the proximal operator must instead be computed approximately using an inner iterative algorithm. For both experiments, the sinograms were simulated from images of the AAPM Low Dose CT Grand Challenge Dataset [17]. The dataset, containing a total of 2 378 examples, was split into three subsets: training (70%), validation (10%), and testing (20%). All implementations were carried out in TensorFlow 2, and both training and inference were performed on an NVIDIA GeForce RTX 5070 Ti GPU. The entire experimental process, including dataset creation, took approximately six days. The source code is freely available at `https://github.com/gabriel-grillo/FISTA-LD`.

4.4.1 ℓ_1-wavelet regularized least-squares model

In this experiment we intend to solve problem (36). First, we performed a parameter search to select $\alpha \in \{10^{-3.5}, 10^{-3.0}, \ldots, 10^{-1.5}\}$ and $\gamma \in \{0.01, 0.025, 0.05, 0.075, 0.1\}$. We trained FISTA-LD with each pair (α, γ) for $N \in [1, 20]$ and with 5% of the available training data for 20 epochs. Then we ran the trained algorithms over the entire validation set for 1000 iterations. The mean optimality gap results are depicted in Figure 2. We also evaluated the results with two metrics. First, the mean validation loss, given by

$$\mathrm{MVL}(\theta) \coloneqq \mathrm{mean}_{b,k} \left[\frac{F_b(S_{b,k}(\theta)) - F_b^\dagger}{F_b(x_{b,k}^{\mathrm{FISTA}}) - F_b^\dagger} \right], \tag{48}$$

where b is in the validation set and $k \in [1, 20]$. Second, the mean validation penalty, given by

$$\mathrm{MVP}(\theta) \coloneqq \mathrm{mean}_{b,k} \left[\max \left(\frac{F_b(S_{b,k}(\theta)) - F_b(x_{b,k}^{\mathrm{FISTA}})}{F_b(x_{b,k}^{\mathrm{FISTA}})}, 0 \right) \right], \tag{49}$$

where b is in the validation set and $k \in [21, 1000]$. The notation in (48) and (49) follows the definitions introduced in Section 4.2; see (45). The MVL serves as an empirical measure of generalization, since it computes the mean value of the training loss function defined in (45) over unseen validation examples and across the trained iterations $k \in [1, 20]$. In contrast, the MVP compares the given algorithm with FISTA outside the training range, namely for $k \in [21, 1000]$. Whenever the given algorithm yields a functional value greater than that produced by FISTA, the corresponding relative difference is added to the penalization term. Both of this metrics are shown in Tables 1 and 2, respectively. The smallest value in each table is highlighted in bold.

The results indicate that FISTA-LD is robust to variations of α and γ within the considered ranges. Figure 2 shows that the qualitative behavior of the optimality gap is similar across all 25 parameter combinations. In the initial iterations, corresponding to the training horizon, FISTA-LD exhibits a substantially faster decay of the optimality gap compared with FISTA. In later iterations, however, the difference between the two methods gradually vanishes. Moreover, larger values of γ lead the trained algorithms to behave more similarly to FISTA in the asymptotic regime. Table 1 shows that the mean validation loss is relatively insensitive to the choice

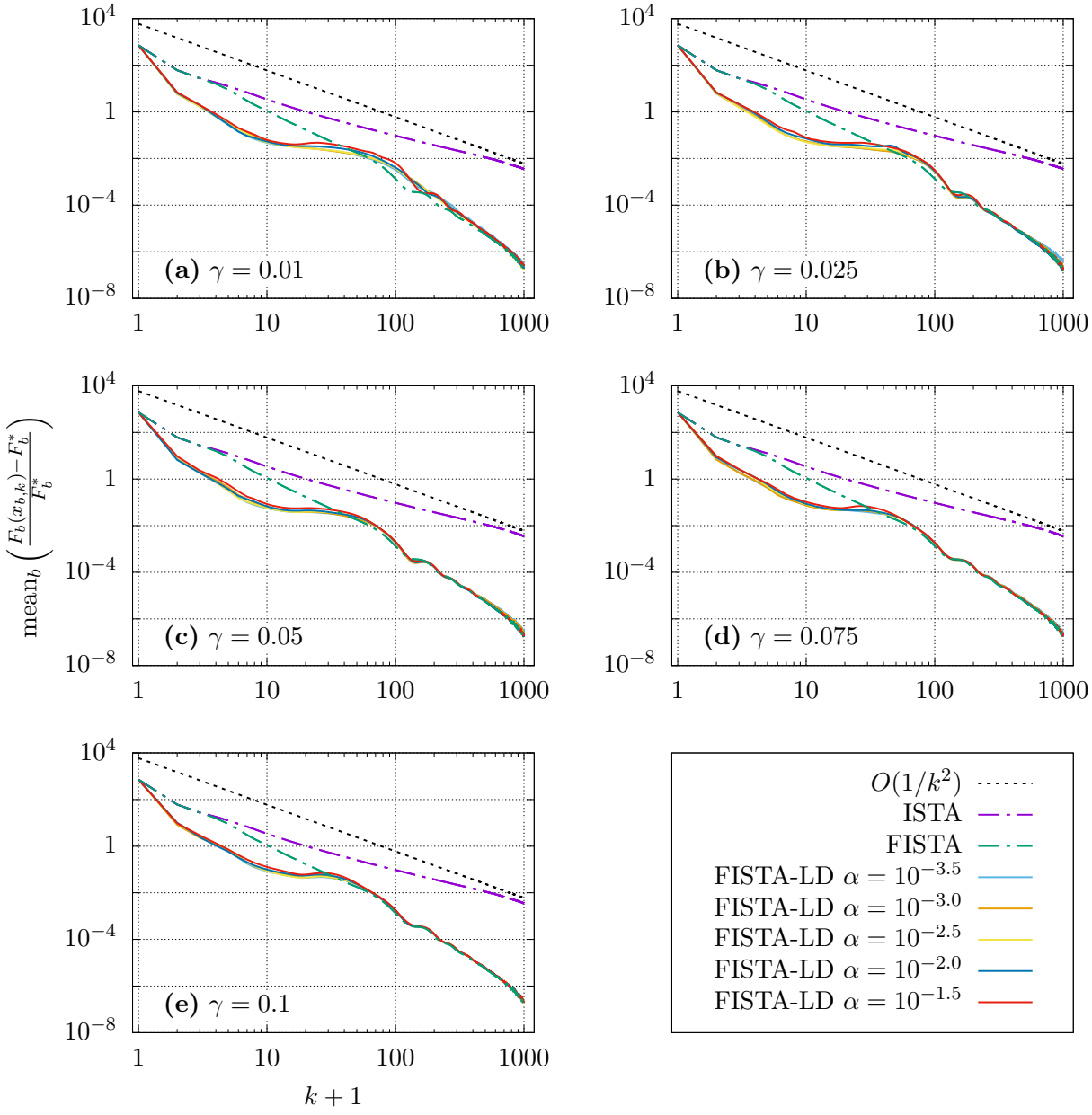


Figure 2: Mean optimality gap results over the validation dataset. Training performed with 5% of the training dataset, $N \in [1, 20]$, 20 epochs, and varying pairs (α, γ).

of parameters. In particular, the trained algorithms are able to achieve a mean optimality gap approximately one order of magnitude smaller than that of FISTA over the trained iterations ($k \in [1, 20]$). The mean validation penalty reported in Table 2 further corroborates the observed qualitative behavior, indicating that the iterates produced by the trained algorithms progressively approach those of FISTA beyond the training horizon. Based on these results, we selected

$\alpha \setminus \gamma$	0.01	0.025	0.05	0.075	0.1
$10^{-3.5}$	**8.80e-02**	9.70e-02	1.07e-01	1.33e-01	1.37e-01
$10^{-3.0}$	9.29e-02	9.49e-02	1.15e-01	1.27e-01	1.56e-01
$10^{-2.5}$	8.93e-02	9.51e-02	1.08e-01	1.36e-01	1.38e-01
$10^{-2.0}$	9.93e-02	1.18e-01	1.22e-01	1.39e-01	1.60e-01
$10^{-1.5}$	1.19e-01	1.35e-01	1.53e-01	1.63e-01	1.79e-01

Table 1: Mean validation loss. Training performed with 5% of the training dataset, $N \in [1, 20]$, 20 epochs, and varying pairs (α, γ).

$\alpha \setminus \gamma$	0.01	0.025	0.05	0.075	0.1
$10^{-3.5}$	1.31e-04	1.34e-04	1.43e-04	1.85e-04	1.37e-04
$10^{-3.0}$	1.84e-04	1.32e-04	**1.07e-04**	1.93e-04	1.63e-04
$10^{-2.5}$	1.72e-04	2.10e-04	1.79e-04	1.73e-04	1.59e-04
$10^{-2.0}$	2.85e-04	3.72e-04	1.92e-04	2.22e-04	2.19e-04
$10^{-1.5}$	5.45e-04	3.89e-04	3.18e-04	3.95e-04	3.97e-04

Table 2: Mean validation penalty. Training performed with 5% of the training dataset, $N \in [1, 20]$, 20 epochs, and varying pairs (α, γ).

the parameter pair $\alpha = 10^{-3.0}$ and $\gamma = 0.025$. This choice is motivated by the observation that, although the configurations with $\gamma = 0.01$ achieve lower mean validation losses, they converge more slowly to the FISTA curve in Figure 2. In contrast, the configurations with $\gamma = 0.025$ exhibit faster convergence, and the setting with $\alpha = 10^{-3.0}$ achieves competitive performance across all evaluated metrics.

For the selected parameter pair (α, γ), we investigated the influence of the training length, measured in epochs, on the performance of the method. At this stage, we switched from using only 5% of the dataset to using the entire dataset. Figure 3 shows the mean optimality gap over the validation set when training with the entire training dataset for five different numbers of epochs: 20, 40, 60, 80, and 100. The results indicate that training for only 20 epochs is already sufficient to reproduce the same qualitative behavior observed after 100 epochs of training. From a quantitative perspective, Table 3 reports the corresponding mean validation loss (MVL) and mean validation penalty (MVP) for each training length. The MVL decreases between epochs 20 and 60 but increases thereafter, between epochs 60 and 100. In contrast, the MVP exhibits an overall increasing trend throughout the training process. Considering these results, together with the computational cost of prolonged training, we selected 20 epochs as the standard training duration. In our implementation, each epoch requires approximately 13 min 20 s, resulting in a total training time of about 4 h 30 min for a 20-epoch run.

Finally, we evaluated FISTA-LD on the test set using the selected parameter pair and a model trained with the full training dataset for 20 epochs. The algorithm was executed for 1000 iterations on each test instance. For comparison, we also trained the learned forward–backward algorithm proposed in [2] using the recommended parameter choice $\alpha = 0.5$, under the same

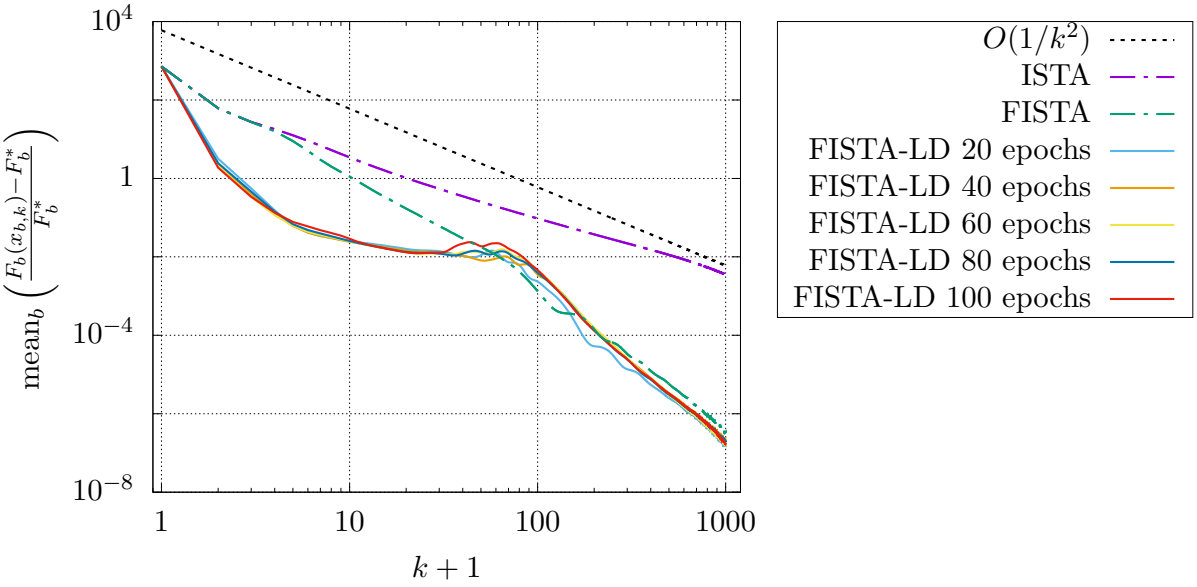


Figure 3: Mean optimality gap results over the validation dataset. Training performed with 100% of the training dataset, $N \in [1, 20]$, $\alpha = 10^{-2.5}$, $\gamma = 0.05$, and varying number of epochs.

Epochs	20	40	60	80	100
MVL	4.70-02	4.23e-02	3.99e-02	4.09e-02	4.14e-02
MVP	1.03e-4	1.73e-4	2.99e-4	2.39e-4	4.53e-4

Table 3: Mean Validation Loss (MVL) and Mean Validation Penalty (MVP) for training with 100% of the training dataset, $N \in [1, 20]$, $\alpha = 10^{-2.5}$, $\gamma = 0.05$, and varying number of epochs.

training protocol, namely, 100% of the training data and 20 training epochs. This protocol produces more than $3 \cdot 10^4$ training iterations, which is sufficient according to the authors. The resulting trained method was likewise evaluated on the test set for 1000 iterations. In fact, analogously to the calibration procedure adopted for FISTA-LD, we also tuned the method from [2] over the parameter set $\alpha \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. The calibration procedure selected $\alpha = 0.5$, which coincides with the parameter value originally recommended in [2]. The mean optimality gaps obtained by both methods are presented in Figure 4. The results show that the two approaches are competitive within the range of iterations used during training. Beyond this regime, however, FISTA-LD preserves the theoretical $O(1/k^2)$ convergence behavior, whereas the competing learned method does not exhibit the same asymptotic performance. Quantitatively, when MVL and MVP are redefined so that b is taken from the testing set rather than the validation set, FISTA-LD achieved an MVL of 4.54e−2 and an MVP of 1.09e−4. In contrast, the trained algorithm from [2] obtained an MVL of 1.06e−1 and an MVP of 3.83e−3, indicating inferior performance in both metrics.

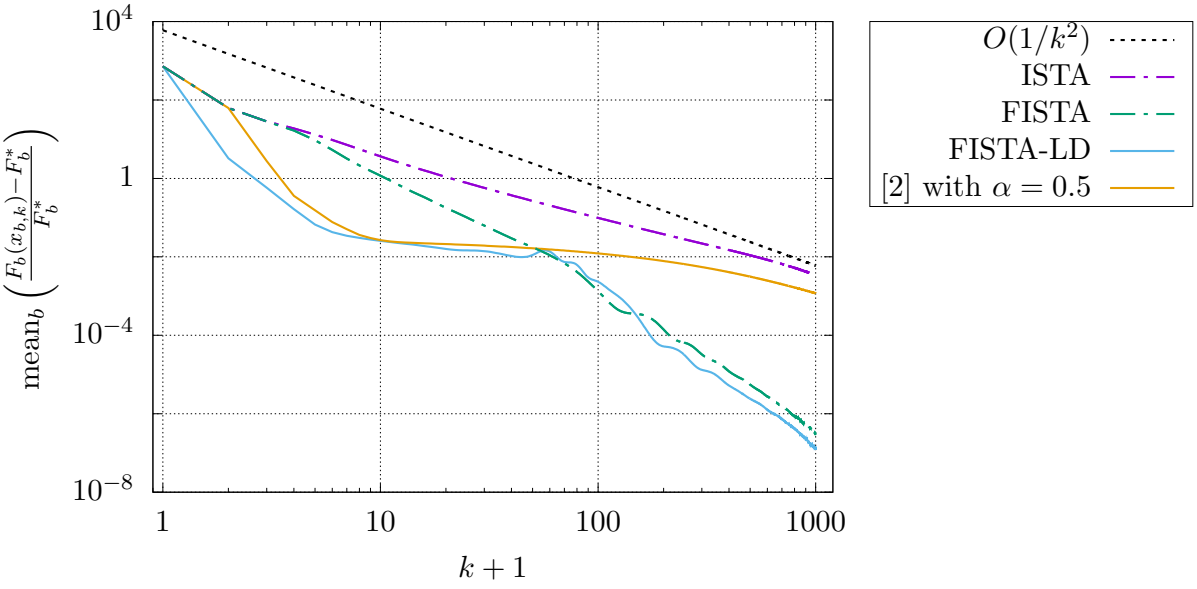


Figure 4: Mean optimality gap results over the test dataset. Both algorithms were trained with 100% of the training dataset, with $N \in [1, 20]$, and for 20 epochs. For FISTA-LD it was used $\alpha = 10^{-2.5}$ and $\gamma = 0.05$.

4.4.2 Total variation regularized least squares model

In this experiment, we aim to solve problem (37) directly using Algorithm 1, that is, by computing the proximal operator associated with the TV term through an iterative procedure. To this end, as discussed in Section 4.1, we employed the algorithm from [6] with a stopping criterion consistent with the definition of J_g in (6). In our implementation, we allow at most 10 iterations of this inner algorithm. We tested Algorithm 1 both with and without training. In the untrained setting, we set $\beta_{1,k} = \beta_{2,k} = 0$ and $\beta_{3,k} = 1$ for all $k \geq 1$ in Algorithm 1, which yields an inexact version of FISTA. In the following, we refer to this method simply as FISTA. In both settings, we considered the constant sequence $\tau_k = \tau$ for all $k \geq 1$, with $\tau \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. Regarding the parameter α, the experiments in Section 4.4.1 showed that both the considered metrics and the qualitative behavior were largely insensitive to variations in this parameter. Accordingly, in the present experiment we fixed α at the same value used in the previous section, namely, $\alpha = 10^{-3.0}$. For this preliminary experiment, we trained FISTA-LD using 5% of the training data for 20 epochs with $N \in [1, 20]$. Figure 5 depicts the resulting optimality gaps.

It is possible to observe that, for every pair of FISTA and FISTA-LD sharing the same value of τ, the qualitative behavior of the optimality gap is consistent with that observed in the previous section. Within the range of trained iterations, the optimality gap of FISTA-LD decreases faster than that of FISTA, whereas for later iterations the difference between the two methods becomes negligible. Except for FISTA with $\tau = 0.1$, all algorithms eventually reach an outer iteration k at which the solver for the proximal subproblem fails to produce a solution

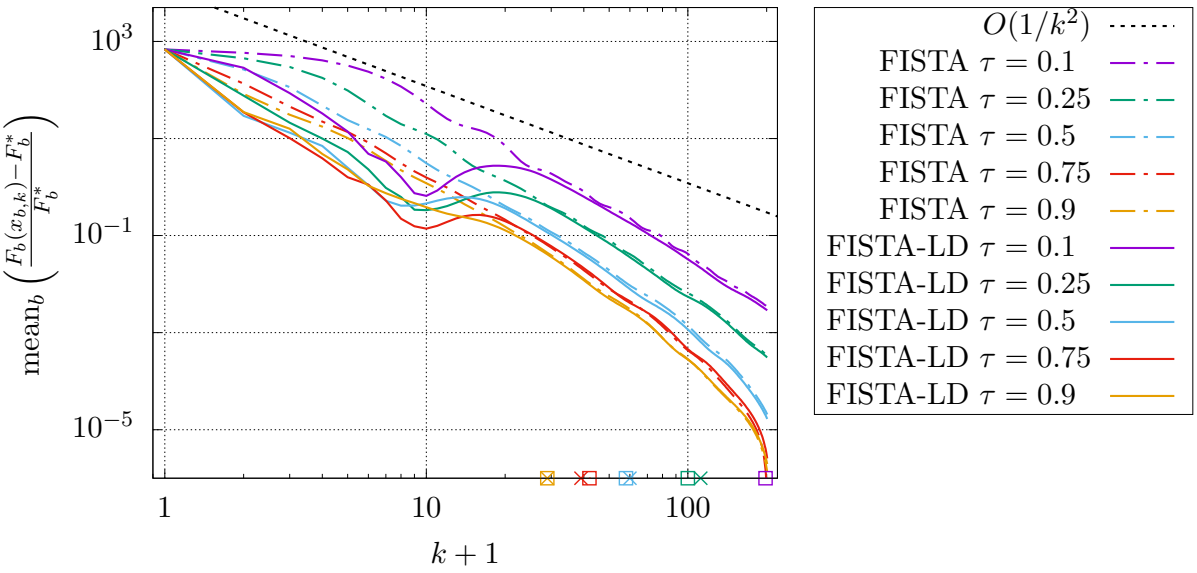


Figure 5: Mean optimality gap results over the validation dataset. Training performed with 5% of the training dataset, $N \in [1, 20]$, 20 epochs, and varying values of τ. For both FISTA and FISTA-LD it was used $\alpha = 10^{-2.5}$. Over the x-axis it is marked ($\times$ for FISTA and $\square$ for FISTA-LD) the mean first iteration in which the proximal approximation does not match the definition of J_g. For FISTA with $\tau = 0.1$ the proximal approximation always matched the definition of J_g

satisfying the definition of J_g in (6). Moreover, larger values of τ lead to earlier failures of the proximal subproblem solver. Nevertheless, both FISTA and FISTA-LD, viewed as instances of Algorithm 1, preserve the $O(1/k^2)$ decay rate of the optimality gap even after the failure of the approximate proximal evaluation.

Among the trained algorithms, the variant with $\tau = 0.75$ achieved the best performance within the range of trained iterations. We therefore selected this variant and

To show the behavior of the algorithms in the test dataset, we arbitrarily chose $\tau = 0.5$ and trained FISTA-LD on the full training dataset for 20 epochs with $N \in [1, 20]$. In our implementations, each epoch takes approximately 9 min 50 secs, which implies in a complete training with 20 epochs in around 3 h 17 min. Figure 6 presents the results for FISTA and FISTA-LD with $\tau = 0.5$ on the test dataset. Compared with Figure 5, the additional training further improved the decay of the optimality gap achieved by FISTA-LD. The optimality gap of FISTA-LD exhibits a nonmonotone behavior, which also appeared, although less prominently, in Figure 4. Nevertheless, in both cases FISTA-LD eventually approaches the behavior of FISTA.

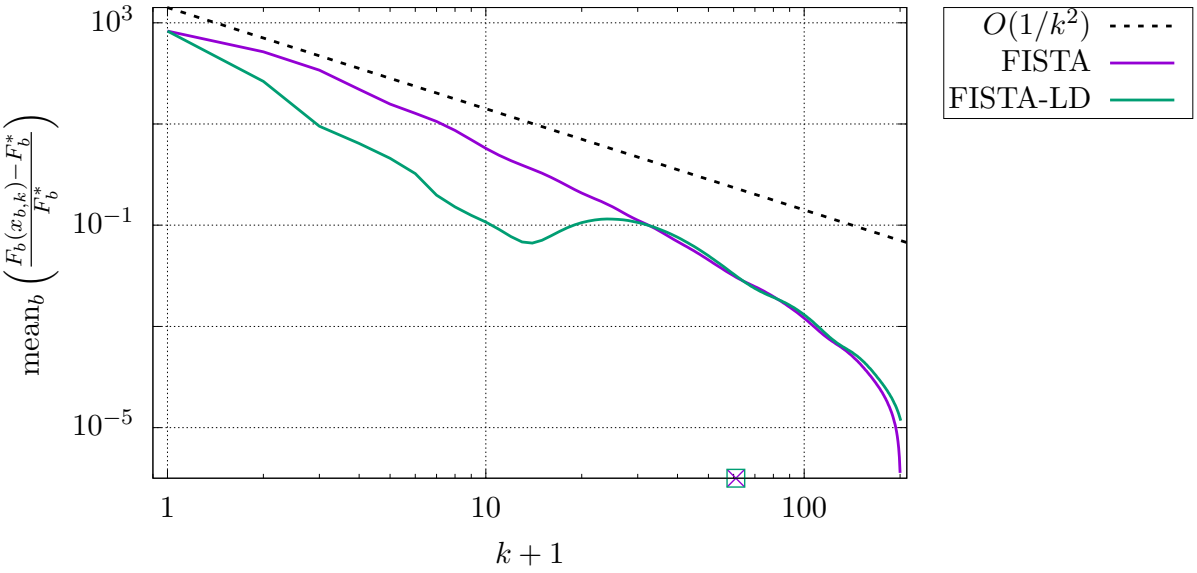


Figure 6: Mean optimality gap over the test dataset. The algorithms share the parameters $\alpha = 10^{-3.0}$ and $\tau = 0.5$. FISTA-LD was trained with 100% of the training dataset, $N \in [1, 20]$, and for 20 epochs. The marks over the x-axis follows the description of Fig. 5.

5 Conclusion and future work

In this work, we introduced a new splitting algorithm for solving convex composite optimization problems in which the objective function is the sum of a smooth convex term with a computable gradient and a convex term whose proximal operator can be evaluated, at least approximately.

Our algorithm combines momentum-based acceleration with learned deviations. The momentum mechanism is active throughout the iterations and ensures an $O(1/k^2)$ convergence rate, as in [6, 7]. The learned deviations, inspired by [2], provide transient acceleration within the finite iteration horizon used during training, while the momentum mechanism governs the asymptotic convergence behavior of the method. Moreover, the method allows for inexact evaluation of the proximal operator, as in [14], a feature not present in [2, 6, 7].

Our numerical experiments indicate that the combination of learned corrections and momentum yields an algorithm that is competitive with state-of-the-art methods across all regimes. In particular, the method of [2] is significantly faster than FISTA during the iterations for which the deviations are trained, but does not maintain the same advantage beyond this regime. Our algorithm achieves comparable or better performance in the early iterations, while matching FISTA in later iterations, consistent with its theoretical convergence analysis.

One of the acceleration techniques of the proposed method is based on deep learning. Consequently, it requires an offline training phase. In deep learning, training consists of optimizing the neural network parameters with respect to a prescribed loss function. In this work, we designed the loss function so that its minimization promotes faster convergence in terms of the objective

function value. However, this is only one possible direction within the proposed framework. As immediate future work, the same theoretical framework could be trained with alternative objectives more directly related to image quality. Furthermore, by decoupling optimization-oriented goals from inverse-problem-oriented goals, promising research directions may include the joint training of both the algorithmic components and the structural elements of the model.

Acknowledgments

We sincerely thank Marcelo Marazzi for his careful editing and helpful suggestions that enhanced the quality of the written text in English.

References

[1] J. Adler and O. Öktem. Learned Primal-Dual Reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1322–1332, June 2018.

[2] S. Banert, J. Rudzusika, O. Öktem, and J. Adler. Accelerated forward-backward optimization using deep learning. *SIAM Journal on Optimization*, 34(2):1236–1263, 2024.

[3] A. Barbero and S. Sra. Modular Proximal Optimization for Multidimensional Total-Variation Regularization. *Journal of Machine Learning Research*, 19(56):1–82, 2018.

[4] J. Barzilai and J. M. Borwein. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

[5] A. Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[6] A. Beck and M. Teboulle. Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, 2009.

[7] A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009. Publisher: Society for Industrial and Applied Mathematics.

[8] Y. Bello-Cruz, M. L. N. Gonçalves, and N. Krislock. On FISTA with a relative error rule. *Computational Optimization and Applications*, 84(2):295–318, Mar. 2023.

[9] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[10] A. Chambolle and C. Dossal. On the Convergence of the Iterates of the "Fast Iterative Shrinkage/Thresholding Algorithm". *Journal of Optimization Theory and Applications*, 166(3):968–982, Sept. 2015.

[11] T. Chen, X. Chen, W. Chen, Z. Wang, H. Heaton, J. Liu, and W. Yin. Learning to optimize: a primer and a benchmark. *J. Mach. Learn. Res.*, 23(1):189:8562–189:8620, 2022.

[12] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems*. Number v. 375 in Mathematics and its applications. Kluwer Academic Publishers, Dordrecht, Boston, 1996.

[13] K. Gregor and Y. LeCun. Learning Fast Approximations of Sparse Coding. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[14] G. R. S. Grillo, E. S. Helou, and S. A. Santos. Inexact fista-like methods with adaptive backtracking. *Computational Optimization and Applications*, 93(3):1259–1304, 2026.

[15] L. Grippo, F. Lampariello, and S. Lucidi. A nonmonotone line search technique for newton's method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.

[16] E. S. Helou. `https://github.com/elias-helou/ray_tracing`. Accessed: 2025-11-06.

[17] C. H. McCollough, A. C. Bartley, R. E. Carter, B. Chen, T. A. Drees, P. Edwards, D. R. Holmes III, A. E. Huang, F. Khan, S. Leng, K. L. McMillan, G. J. Michalak, K. M. Nunez, L. Yu, and J. G. Fletcher. Low-dose ct for the detection and classification of metastatic liver lesions: Results of the 2016 low dose ct grand challenge. *Medical Physics*, 44(10):e339–e352, 2017.

[18] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O\left(1/k^2\right)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983. Issue: 3.

[19] Y. Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Applied Optimization*. Springer US, New York, NY, 2004.

[20] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[21] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[22] K. Scheinberg, D. Goldfarb, and X. Bai. Fast First-Order Methods for Composite Convex Optimization with Backtracking. *Foundations of Computational Mathematics*, 14(3):389–417, June 2014.

[23] M. Sucker and P. Ochs. PAC-Bayesian Learning of Optimization Algorithms. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 8145–8164. PMLR, 2023. ISSN: 2640-3498.

[24] R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[25] S. Villa, S. Salzo, L. Baldassarre, and A. Verri. Accelerated and Inexact Forward-Backward Algorithms. *SIAM Journal on Optimization*, 23(3):1607–1633, Jan. 2013.

[26] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. Sparse Reconstruction by Separable Approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[27] J. Xiang, Y. Dong, and Y. Yang. FISTA-Net: Learning a Fast Iterative Shrinkage Thresholding Network for Inverse Problems in Imaging. *IEEE Transactions on Medical Imaging*, 40(5):1329–1339, 2021.