Analysis and synthesis denoisers for forward-backward plug-and-play algorithms

Matthieu Kowalski, Benoît Malézieux, Thomas Moreau, Audrey Repetti

Analysis and Synthesis Denoisers for Forward-Backward Plug-and-Play Algorithms *

Matthieu Kowalski•, Benoît Malézieux◇, Thomas Moreau◇, Audrey Repetti†⋆

• Laboratoire Interdisciplinaire des Sciences du Numériques, Inria, Université Paris-Saclay, CNRS, Gif-sur-Yvette, France

◇ Inria, Université Paris-Saclay, CEA, Palaiseau, France

† School of Mathematics and Computer Sciences and School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, UK

⋆ Maxwell Institute for Mathematical Sciences, Edinburgh, UK

matthieu.kowalski@universite-paris-saclay.fr, benoit.malezieux@free.fr, thomas.moreau@inria.fr, a.repetti@hw.ac.uk

Abstract

In this work we study the behavior of the forward-backward (FB) algorithm when the proximity operator is replaced by a sub-iterative procedure to approximate a Gaussian denoiser, in a Plug-and-Play (PnP) fashion. In particular, we consider both analysis and synthesis Gaussian denoisers within a dictionary framework, obtained by unrolling dual-FB iterations or FB iterations, respectively. We analyze the associated minimization problems as well as the asymptotic behavior of the resulting FB-PnP iterations. In particular, we show that the synthesis Gaussian denoising problem can be viewed as a proximity operator. For each case, analysis and synthesis, we show that the FB-PnP algorithms solve the same problem whether we use only one or an infinite number of sub-iteration to solve the denoising problem at each iteration. To this aim, we show that each "one sub-iteration" strategy within the FB-PnP can be interpreted as a primal-dual algorithm when a warm-restart strategy is used. We further present similar results when using a Moreau-Yosida smoothing of the global problem, for an arbitrary number of sub-iterations. Finally, we provide numerical simulations to illustrate our theoretical results. In particular we first consider a toy compressive sensing example, as well as an image restoration problem in a deep dictionary framework.

Keywords. Plug-and-Play, forward-backward algorithm, unrolling, deep dictionary learning, inverse imaging problems

MSC. 90C59, 65K10, 68T07, 68U10, 94A08

*This work was funded by the Royal Society of Edinburgh and the EPSRC grant EP/X028860.

1 Introduction

Linear inverse problems play a pivotal role in various scientific disciplines, including imaging [44], neurosciences [24], and astrophysics [51]. In these scenarios, an unknown signal $\overline{x} \in \mathbb{R}^N$ is observed through a degraded linear system given by

$$y = A\overline{x} + \varepsilon w, \tag{1.1}$$

where $y \in \mathbb{R}^M$ represents the degraded observations, $A \colon \mathbb{R}^N \to \mathbb{R}^M$ models a linear measurement operator, $w \in \mathbb{R}^M$ is a realization of an *i.i.d.* standard normal random variable, and $\varepsilon > 0$. The inverse problem (1.1) aims to find an estimate $\widehat{x} \in \mathbb{R}^N$ of $\overline{x}$ from the degraded measurements y. This problem is often challenging due to issues such as under-sampling and noise, rendering it ill-posed and/or ill-conditioned.

To address these challenges, practitioners commonly employ *prior* knowledge to guide the selection of a plausible solution. A widely adopted approach involves a maximum *a posteriori* (MAP) strategy, defining $\widehat{x}$ as a minimizer of a penalized least squares objective:

$$\text{find } \widehat{x} \in \operatorname*{Argmin}_{x \in \mathbb{R}^N} \frac{1}{2}\|Ax - y\|^2 + p(x), \tag{1.2}$$

where $p \colon \mathbb{R}^N \to (-\infty, +\infty]$ is a convex, lower semi-continuous, proper function, representing a penalization term that incorporates prior information on the target solution. The choice of the prior is crucial for both reconstruction performance and computational complexity. Notably, functions summarizing signal structures based on sparsity have been extensively studied in the literature [21, 3, 23]. Proximal algorithms [16, 30] are efficient to solve the resulting minimization problem (1.2). They are scalable and versatile, offering convergence guarantees and maintaining their status as the state-of-the-art for solving inverse problems for more than two decades. A celebrated proximal algorithm, extensively used in the literature for solving (1.2), is the forward-backward (FB) algorithm, also known as proximal-gradient or ISTA [17]. This scheme alternates at each iteration between a gradient step on the differentiable least squares function and a proximal step on the non-smooth function p. This algorithm reads

$$\begin{aligned} &x_0 \in \mathbb{R}^N \\ &\text{for } k = 0, 1, \dots \\ &\quad \left\lfloor \; x_{k+1} = \operatorname{prox}_{\tau p}\big(x_k - \tau A^*(Ax_k - y)\big), \right. \end{aligned} \tag{1.3}$$

where $\operatorname{prox}_{\tau p}$ is the so-called proximity operator of p, and $\tau > 0$ is a step-size chosen to ensure the convergence of the generated sequence $(x_k)_{k \in \mathbb{N}}$ to a solution to (1.2). In (1.3), one has to compute the proximity operator of p (defined in (1.8)) at each iteration. This operator can have an explicit formula for many simple choices of functions p (see e.g. [16]). However this is not true in many cases of interest, and the proximity operator must be approximated numerically with sub-iterations. These sub-iterations can become computationally expensive in practice as they need to accurately approximate prox_p. Recently, to improve the reconstruction quality and avoid

these sub-iterations, proximal algorithms have also been paired with deep learning techniques for solving inverse imaging problems. Examples include deep dictionary learning (DDL) [50], Plug-and-Play (PnP) algorithms [7], and unfolded neural networks [25]. However, providing convergence guarantees for these methods is not straightforward.

A common prior's choice is based on the sparsity of the unknown signal in a certain basis, with prior p of the form of

$$(\forall x \in \mathbb{R}^N) \quad p(x) = \lambda g(\Gamma x), \tag{1.4}$$

where $\lambda > 0$ is a regularization parameter balancing data fidelity and regularization terms, $\Gamma \colon \mathbb{R}^N \to \mathbb{R}^S$ is a linear operator modeling a sparsifying transformed domain, and $g \colon \mathbb{R}^S \to \mathbb{R}^N$ is a convex, lower semi-continuous, proper function promoting sparsity in the transformed domain (e.g., ℓ^1 norm). Notable choices for the sparsifying operator include the Fourier, DCT, or Wavelet transforms [37] as well as the Total Variation (TV) regularization [48, 8], depending on the properties of the target solution. The structure of the sought signal, through the operator Γ, can also be learned from a ground truth dataset in a supervised setting, leading to dictionary learning (DL) approaches [2, 36]. For such sparsity-based priors, there exists no close-form formula and one typically needs to resort to expansive sub-iterations. In this paper, we investigate the properties of such approaches, strategies to reduce the sub-iteration computational burden, and their convergence.

Contributions

We adopt a PnP framework, where the proximity operator of p in (1.3) is replaced with a denoiser. Building our PnP algorithm with FB iterations, we investigate the behavior of the following FB-PnP scheme

$$\begin{aligned} &x_0 \in \mathbb{R}^N \\ &\text{for } k = 0, 1, \dots \\ &\quad \left\lfloor \; x_{k+1} = G\big(x_k - \tau A^*(Ax_k - y)\big), \right. \end{aligned} \tag{1.5}$$

where $\tau > 0$ and $G \colon \mathbb{R}^N \to \mathbb{R}^N$ is a sparsity-based denoising operator. Specifically, we consider G to be either an analysis or a synthesis dictionary-based denoiser. Let $v \in \mathbb{R}^N$ be a noisy image. On the one hand, the Analysis Denoiser (AD) is built as

$$\text{find} \quad G_\Gamma(v) \approx \operatorname{prox}_p(v) = \operatorname*{argmin}_{x \in \mathbb{R}^N} \frac{1}{2}\|x - v\|^2 + g_\lambda(\Gamma x), \tag{1.6}$$

where $g_\lambda \equiv \lambda g$ and $\Gamma \colon \mathbb{R}^N \to \mathbb{R}^S$. On the other hand, the Synthesis Denoiser (SD) is built as

$$\text{find} \quad G_D(v) \approx D \operatorname*{argmin}_{z \in \mathbb{R}^S} \frac{1}{2}\|Dz - v\|^2 + \lambda g(z), \tag{1.7}$$

where $D \colon \mathbb{R}^S \to \mathbb{R}^N$. In general these two classes of denoisers do not have a closed-form formula (unless for specific cases, e.g., when Γ is (semi)-orthogonal [22]), thus requiring sub-iterations to

compute their output. We then aim to investigate the behavior of $(x_k)_{k\in\mathbb{N}}$ when G_Γ and G_D are computed with FB-based iterations following an unrolling framework.

Our first contribution is to analyze the minimization problems associated with the proposed AD and SD described in (1.6)-(1.7), when Γ and D are fixed dictionaries. In particular, we show that (1.7) corresponds to a proximity operator, and that the associated problem is equivalent to a synthesis formulation of problem (1.2)-(1.4). Second, we investigate the behavior of algorithm (1.5), where G_Γ and G_D are obtained with either 1 sub-iteration of a FB-based scheme, or when a large enough number of these iterations are run for solving (1.6)-(1.7), leveraging a warm-restart strategy. We further investigate the behavior of a similar algorithm to (1.5) for solving a Moreau envelope smoothing of problem (1.2)-(1.4). In this context, we show that using a warm-restart strategy yield an algorithm which converges toward a solution of the smooth approximated problem, if the operators G_Γ and G_D satisfy some sufficient decrease conditions. Finally, we show through simulations that the proposed unrolling within PnP frameworks is well suited to adopt a DDL approach for learning dictionaries Γ or D.

Outline

The remainder of the paper is organized as follows. Section 2 first presents the denoiser based on the sparse coding of the analysis coefficient using a given dictionary. It relies on the so-called dual forward-backward algorithm. Then, the unrolled version of this denoiser is embedded in a PnP framework to solve general linear inverse problems for which we provide a convergence analysis. Section 3 gives a similar analysis for the synthesis-based approach. In particular, we show the equivalence between the global synthesis-based approach to solve a linear inverse problem and the PnP approach using a synthesis-based denoiser. Using a smooth, regularized version of the original problem, Section 4 provides an analysis of the PnP algorithm using analysis or synthesis-based denoiser exploiting a bi-level optimization framework. Finally, Section 5 illustrates the convergence properties of the two PnP approaches on a toy compressive sensing examples, as well as to an image restoration problem within a DDL framework.

Notation

An element of $\mathbb{R}^N$ is denoted by $x = (x^{(n)})_{1\leq n\leq N}$. The standard Euclidean norm is denoted $\|\cdot\|$, the ℓ_p norm (for $p > 0$) is denoted $\|\cdot\|_p$, and $\|\cdot\|_S$ denotes the spectral norm. We denote $\Gamma_0(\mathbb{R}^N)$ the set of proper, lower semi-continuous convex functions from $\mathbb{R}^N$ to $(-\infty, +\infty]$. The proximity operator of a function $g \in \Gamma_0(\mathbb{R}^N)$ is defined as [38]

$$(\forall v \in \mathbb{R}^N) \quad \operatorname{prox}_g(v) = \operatorname*{arg\,min}_{x\in\mathbb{R}^N} g(x) + \frac{1}{2}\|x - v\|^2 \,. \tag{1.8}$$

Let $\mathcal{C} \subset \mathbb{R}^N$ be a closed, non-empty, convex set. We denote by $\iota_{\mathcal{C}}$ the indicator function of C, defined as, for every $x \in \mathbb{R}^N$, $\iota_{\mathcal{C}}(x) = 0$ if $x \in \mathcal{C}$, and $\iota_{\mathcal{C}}(x) = +\infty$ otherwise. The Fenchel-Legendre conjugate function of g is denoted by $g^* \in \Gamma_0(\mathbb{R}^N)$ and is defined as, for every $v \in \mathbb{R}^N$, $g^*(v) = \sup_{x \in \mathbb{R}^N} \langle v, x \rangle - g(x)$. The Moreau's identity is given by $\mathrm{Id} = \mathrm{prox}_g + \mathrm{prox}_{g^*}$, where Id denotes the identity operator. The infimal post-composition of g by an operator $U : \mathbb{R}^S \to \mathbb{R}^N$ is defined as

$$U \triangleright g \colon \mathbb{R}^N \to [-\infty, +\infty] \colon x \mapsto \inf_{z \in \mathbb{R}^S, x = Uz} g(z). \tag{1.9}$$

An operator $G \colon \mathbb{R}^N \to \mathbb{R}^N$ is β-Lipschitz continuous with parameter $\beta > 0$ if, for every $(x, y) \in (\mathbb{R}^N)^2$, $\|G(x) - G(y)\|_2 \leq \beta \|x - y\|$.

For further background on convex optimization, we refer the reader to [4, 46] and references therein.

2 Analysis denoiser for FB-PnP algorithm

In this section we focus on the analysis denoiser defined in (1.6), for a fixed dictionary Γ. By definition, the proximity operator of $g \circ \Gamma$ can be interpreted as a MAP estimate for a Gaussian denoising problem, where the least-squares function corresponds to the data-fidelity term, and $g \circ \Gamma$ is the penalization function. During the last decade, this interpretation has been leveraged to develop new hybrid optimization algorithms, dubbed PnP methods, where the proximity operator can be replaced by more powerful denoisers [7, 49] to solve inverse problems of the form of (1.1). For instance, the PnP formulation of the FB algorithm (1.3), called FB-PnP hereafter, is given by (1.5). More generally, PnP algorithms have been proposed in the literature using a wide range of proximal algorithms, including HQS algorithm, Douglas-Rachford algorithm, or primal-dual algorithms. Denoisers can be either hand-crafted (e.g. BM3D [18]), or learned (e.g. neural network denoisers [11, 47, 45]). Although PnP methods have shown outstanding performances in many applications, they can be unstable in practice. The main challenge is to guarantee that PnP iterations produce a converging sequence of estimates $(x_k)_{k \in \mathbb{N}}$, without sacrificing reconstruction performances. In particular, it is well known that PnP algorithms output converging sequences $(x_k)_{k \in \mathbb{N}}$ if the denoiser G is firmly non expansive, as a consequence of fixed point theory [4]. Recently, they have been extensively studied, in particular when denoisers are neural networks [41, 52, 26, 27, 29, 43].

In this section, we focus on FB-PnP defined in (1.5), where the denoiser is built with sub-iterations based on the FB scheme for approximating (1.6), for a fixed dictionary Γ. In the following, we call such a denoiser an analysis denoiser (AD).

2.1 Analysis denoiser structure

As mentioned in Section 1, unless Γ is (semi)-orthogonal, the AD problem (1.6) does not have an explicit solution. However, it can be solved by the FB algorithm when applied to the dual problem of (1.6) (in the sense of Fenchel-Rockafellar, see e.g. [4, Chap. 15]), as proposed in [14, 15]. The resulting dual-FB algorithm reads

$$\begin{array}{l} \text{for } \ell = 0, 1, \dots \\ \quad \left\lfloor \; u_{\ell+1} = \operatorname{prox}_{\sigma g_\lambda^*}\left(u_\ell - \sigma\Gamma(\Gamma^* u_\ell - v)\right) \right. \end{array} \tag{2.1}$$

where $u_0 \in \mathbb{R}^S$ and $\sigma > 0$ is a step-size. Then the following convergence result can be deduced from [14, Thm. 3.7].

Theorem 2.1 *Assume that there exists $\underline{\sigma} > 0$ such that $\underline{\sigma} < \sigma < (2 - \underline{\sigma})\|\Gamma\|_S^{-2}$. Then $(u_\ell)_{\ell\in\mathbb{N}}$ converges to a solution $u_\Gamma^\dagger$ to the dual problem of (2.5), and $x_\Gamma^\dagger = v - \sigma\Gamma^* u_\Gamma^\dagger$ is a solution to (2.5).*

algorithm (2.1) can be used as sub-iterations when included within the global FB algorithm (1.5), to approximate the computation of $\operatorname{prox}_{g_\lambda\circ\Gamma}$. However, for (1.5) to still benefit from convergence guarantees, the computation of the proximity operator must be very accurate, which can be difficult in practice as this is only achieved at asymptotic convergence of (2.1). In this section, we investigate the behavior of (1.5) with the AD, when only one iteration of algorithm (2.1) is computed.

Model 2.2 (Analysis denoiser (AD)) Let $(v, u_0) \in \mathbb{R}^N \times \mathbb{R}^S$, $\Gamma\colon \mathbb{R}^N \to \mathbb{R}^S$ be a linear operator, and $\lambda > 0$ be a regularization parameter. Let $L \in \mathbb{N}^*$. Let $\widetilde{G}^{\mathcal{A}}_{L,\lambda,v}\colon \mathbb{R}^S \to \mathbb{R}^S$ be defined as

$$\widetilde{G}^{\mathcal{A}}_{L,\lambda,v}(u_0) = \underbrace{T^{\mathcal{A}}_{\lambda,v} \circ \cdots \circ T^{\mathcal{A}}_{\lambda,v}}_{L \text{ compositions}}(u_0), \tag{2.2}$$

where

$$T^{\mathcal{A}}_{\lambda,v}\colon \mathbb{R}^S \to \mathbb{R}^S\colon u \mapsto \operatorname{prox}_{\sigma g_\lambda^*}\left(u - \sigma\Gamma^*(\Gamma u - v)\right) \tag{2.3}$$

with $\sigma > 0$. The unrolled AD $G^{\mathcal{A}}_{L,\lambda,v}\colon \mathbb{R}^S \to \mathbb{R}^N$ is obtained as follows:

$$G^{\mathcal{A}}_{L,\lambda,v}(u_0) = v - \Gamma\widetilde{G}^{\mathcal{A}}_{L,\lambda,v}(u_0). \tag{2.4}$$

A few comments can be made regarding Model 2.2.

Remark 2.3

(i) In [43, 32], the authors have used the unrolled Model 2.2 to design denoising NNs, where the dictionary Γ is learned and is different at each iteration. In their work, a slightly different structure of operator $T^{\mathcal{A}}_{\lambda,v}$ was used, accounting for a simple convex constraint on the signal domain, and rescaling the dual variable.

(ii) It can be noticed that, since the layers of Model 2.2, defined in (2.3), correspond to FB iterations, it can be formulated as a feed-forward network, acting in the dual domain. Precisely, $T^{\mathcal{A}}_{\lambda,v}$ can be reformulated as

$$(\forall u \in \mathbb{R}^S) \quad T^{\mathcal{A}}_{\lambda,v}(u) = \operatorname{prox}_{\sigma g^*_\lambda}\left((\mathrm{Id} - \sigma\Gamma^*\Gamma)u + \sigma\Gamma^* v\right).$$

In [32], the authors describe the layers of Model 2.2 using a primal-dual formulation, where each layer is composed of two feed-forward layers, one acting in the dual domain, and one acting in the primal domain.

(iii) As a direct consequence of Theorem 2.1, it can be noticed that, for any $u_0 \in \mathbb{R}^S$, if there exists $\underline{\sigma} > 0$ such that $\underline{\sigma} < \sigma < (2 - \underline{\sigma})\|\Gamma\|_S^{-2}$, then $\lim_{L\to+\infty} G^{\mathcal{A}}_{L,\lambda,v}(u_0) = x^\dagger_\Gamma$, where $x^\dagger_\Gamma$ is the solution to the problem to

$$\text{find} \quad x^\dagger_\Gamma = \operatorname{prox}_{g_\lambda \circ \Gamma}(v) = \operatorname*{argmin}_{x\in\mathbb{R}^N} \frac{1}{2}\|x - v\|^2 + g_\lambda(\Gamma x). \tag{2.5}$$

2.2 FB-PnP algorithm with approximated AD

In this section we aim to

$$\text{find } \widehat{x}_{\mathcal{A}} \in \operatorname*{argmin}_{x\in\mathbb{R}^N} \frac{1}{2}\|Ax - y\|^2 + g_\lambda(\Gamma x). \tag{2.6}$$

We study the asymptotic behavior of the FB-PnP algorithm (1.5) with the AD given in Model 2.2. This algorithm boils down to

$$\begin{aligned} &x_0 \in \mathbb{R}^N, z_0 \in \mathbb{R}^S, \\ &\text{for } k = 0, 1, \dots \\ &\left\lfloor \begin{aligned} v_k &= x_k - \tau A^*(Ax_k - y), \\ u_{k+1} &= G^{\mathcal{A}}_{L,\tau\lambda,v_k}(u_k), \\ x_{k+1} &= v_k - \Gamma u_{k+1}, \end{aligned} \right. \end{aligned} \tag{2.7}$$

where $\tau > 0$ is the stepsize of the FB algorithm and $L \in \mathbb{N}^*$ is the number of iterations of the AD in Model 2.2.

The following convergence result directly follows from Theorem 2.1 and [17, Thm. 3.4].

Theorem 2.4 *Let $(x_k)_{k\in\mathbb{N}}$ be generated by algorithm (2.7). Assume that there exist $\underline{\tau} > 0$ and $\underline{\sigma} > 0$ such that $\underline{\tau} \leq \tau \leq (2 - \underline{\tau})\|A\|_S^{-2}$ and $\underline{\sigma} \leq \sigma \leq (2 - \underline{\sigma})\|\Gamma\|_S^{-2}$. When $L \to \infty$,* i.e. *when we use an infinite number of iterations in Model 2.2, then $(x_k)_{k\in\mathbb{N}}$ converges to a solution to problem (2.6).*

Theorem 2.4 only holds when $L \to \infty$, i.e. when $G^{\mathcal{A}}$ has a very large number of sub-iterations. In practice, this is often intractable and only a fixed finite number of iterations are used. For instance, in the context of DDL, typically L will be smaller than 20. We propose to investigate

to what extent this impacts the convergence and the nature of the final solution. Note that in algorithm (2.7), the unrolled denoiser benefits from a warm restart using the output u_k of the inner iterations $\widetilde{G}^{\mathcal{A}}$ obtained at iteration k. We will take advantage of this warm restart to show the convergence of $(x_k)_{k\in\mathbb{N}}$ when $L = 1$. In this particular case, at iteration $k \in G$, the update of u_k in algorithm (2.7) boils down to

$$u_{k+1} = G^{\mathcal{A}}_{1,\tau\lambda,v_k}(u_k) = T^{\mathcal{A}}_{\tau\lambda,v_k}(u_k), \tag{2.8}$$

where $T^{\mathcal{A}}_{\tau\lambda,v_k}$ is defined in (2.3).

In the following result, we show that algorithm (2.7) with $L = 1$ corresponds to the scaled primal-dual algorithm proposed by [35] (see Section 6 in this article, equation (40)). We can then deduce convergence guarantees for $(x_k)_{k\in\mathbb{N}}$, leveraging [35, Thm. 1].

Theorem 2.5 *Let $(x_k)_{k\in\mathbb{N}}$ and $(u_k)_{k\in\mathbb{N}}$ be sequences generated by algorithm (2.7) with $L = 1$. Assume that there exist $\underline{\tau} > 0$ and $\underline{\sigma} > 0$ such that $\underline{\tau} \leq \tau \leq (2 - \underline{\tau})\|A\|_S^{-2}$ and $\underline{\sigma} \leq \sigma \leq (2 - \underline{\sigma})\|\Gamma\|_S^{-2}$. Then the following statements hold:*

(i) *$(x_k)_{k\in\mathbb{N}}$ converges to a solution to (2.6).*

(ii) *$(x_k, \tau^{-1}u_k)_{k\in\mathbb{N}}$ converges to a solution to the saddle-point problem*

$$\underset{x\in\mathbb{R}^N}{\text{minimize}} \ \max_{u\in\mathbb{R}^S} \ \frac{1}{2}\|Ax - y\|^2 + \langle \Gamma x, u\rangle - g^*_\lambda(u). \tag{2.9}$$

Proof. Let $(x_k)_{k\in\mathbb{N}}$ and $(u_k)_{k\in\mathbb{N}}$ be generated by algorithm (2.7) with $L = 1$. We have

$$x_{k+1} = v_k - \Gamma u_{k+1} = x_k - \tau A^*(Ax_k - y) - \Gamma u_{k+1}$$

and

$$\begin{aligned} u_{k+1} &= \operatorname{prox}_{\sigma g^*_{\lambda\tau}}\Big(u_k - \sigma\Gamma^*\Gamma u_k + \sigma\Gamma^* v_k\Big) \\ &= \operatorname{prox}_{\sigma g^*_{\lambda\tau}}\Big(u_k - \sigma\Gamma^*\Gamma u_k + \sigma\Gamma^*\big(x_k - \tau A^*(Ax_k - y)\big)\Big) \\ &= \operatorname{prox}_{\sigma g^*_{\lambda\tau}}\Big(u_k + \sigma\Gamma^*\Big(x_k - \tau A^*\big(Ax_k - y\big) - \Gamma u_k\Big)\Big). \end{aligned}$$

Hence, algorithm (2.7)-(2.8) is equivalent to

$$\left\lfloor \begin{array}{l} \text{for } k = 0, 1, \ldots \\ \quad u_{k+1} = \operatorname{prox}_{\sigma g^*_{\lambda\tau}}\Big(u_k + \sigma\Gamma^*\Big(x_k - \tau A^*\big(Ax_k - y\big) - \Gamma u_k\Big)\Big) \\ \quad x_{k+1} = x_k - \tau A^*(Ax_k - y) - \Gamma u_{k+1}. \end{array} \right. \tag{2.10}$$

Further, noticing that $g_{\lambda\tau} = \tau g_\lambda$, and applying subsequently Prop. 24.8(v) and Prop. 13.23(i) from [4], we have

$$(\forall u \in \mathbb{R}^S) \quad \operatorname{prox}_{\sigma g^*_{\lambda\tau}}(u) = \operatorname{prox}_{\sigma\tau g^*_\lambda}(u) = \operatorname{prox}_{\sigma\tau g^*_\lambda(\tau^{-1}\cdot)}(u) = \tau \operatorname{prox}_{\sigma\tau^{-1} g^*_\lambda}(\tau^{-1}u).$$

Applying this result to (2.10) leads to

$$\begin{aligned}&\text{for } k = 0, 1, \dots\\ &\left\lfloor \begin{aligned} u_{k+1} &= \tau \operatorname{prox}_{\sigma\tau^{-1} g_\lambda^*}\Big(\tau^{-1}\Big(u_k + \sigma\Gamma^*\Big(x_k - \tau A^*\big(Ax_k - y\big) - \Gamma u_k\Big)\Big)\Big)\\ x_{k+1} &= x_k - \tau A^*(Ax_k - y) - \Gamma u_{k+1}. \end{aligned}\right.\end{aligned} \tag{2.11}$$

Hence, by setting, for every $k \in \mathbb{N}$, $\widetilde{u}_k = \tau^{-1} u_k$, we obtain

$$\begin{aligned}&\text{for } k = 0, 1, \dots\\ &\left\lfloor \begin{aligned} \widetilde{u}_{k+1} &= \operatorname{prox}_{\frac{\sigma}{\tau} g_\lambda^*}\Big(\widetilde{u}_k + \tfrac{\sigma}{\tau}\Gamma^*\big(x_k - \tau A^*(Ax_k - y) - \tau\Gamma\widetilde{u}_k\big)\Big)\\ x_{k+1} &= x_k - \tau A^*(Ax_k - y) - \tau\Gamma\widetilde{u}_{k+1}. \end{aligned}\right.\end{aligned} \tag{2.12}$$

By noticing that algorithm (2.12) corresponds to the scaled primal-dual algorithm proposed by [35] (see Section 6 in this article, equation (40)), [35, Thm. 1] can directly be applied to deduce the convergence results, using the scaled version of the algorithm given in Section 6 of the same article. □

Theorem 2.5 shows that using a single sub-iteration $L = 1$ in algorithm (2.7) is sufficient to ensure the convergence toward a solution to the same problem as the one solved when using $L \to \infty$, provided that we use a warm-starting scheme. It also informs us on the structure of the recovered solutions as the solution of a min-max problem. Let us give a practical implication of (ii) in the case where the ℓ_1 norm is used as a sparsity-inducing regularization.

Example 2.6 In the particular case $g = \lambda\|\cdot\|_1$, then Theorem 2.5 shows that

$$\min_{x\in\mathbb{R}^N} \frac{1}{2}\|Ax - y\|^2 + \lambda\|\Gamma x\|_1 = \min_{x\in\mathbb{R}^N} \max_{x_0 \in \mathcal{F}_\lambda^{\mathcal{A}}(\Gamma)} \frac{1}{2}\|Ax - y\|^2 + \langle x, x_0\rangle, \tag{2.13}$$

where $\mathcal{F}_\lambda^{\mathcal{A}}(\Gamma) = \big\{x \mid \inf\{\|z\|_\infty \mid \Gamma z = x\} \leq \lambda\big\}$.

3 Synthesis denoiser for FB-PnP algorithm

We now focus on the synthesis denoiser (SD) defined in (1.7) for a fixed dictionary D. The objective will be to investigate the behavior of the FB-PnP algorithm (1.5), where the operator G is such an SD. Before investigating suitable unrolling techniques to build an SD within the FB-PnP algorithm, we will show that (1.7) is a proximity operator of a convex function. We will further investigate the link between the associated minimization problem, and the problem that would be obtained if a full synthesis formulation of (1.2)-(1.4) was considered. We will then propose an unrolled FB algorithm for solving (1.7), and investigate the behavior of algorithm (1.5) with this unrolled SD.

3.1 Synthesis problem formulations

In this section we first define the synthesis problem of interest, and investigate its properties.

Synthesis denoiser as proximity operator Unlike for the AD studied in Section 2, it is not obvious that the SD corresponds to a proximity operator. The first result of this section will show that (1.7) indeed is a proximity operator of a convex function, making it a suitable candidate for the FB-PnP algorithm (1.5). For a fixed synthesis dictionary D, we introduce the notation $(x_D^\dagger, z_D^\dagger)$, that is a solution to the problem that aims to

$$\text{find} \quad x_D^\dagger = Dz_D^\dagger \quad \text{where} \quad z_D^\dagger = \operatorname*{argmin}_{z \in \mathbb{R}^S} \frac{1}{2}\|Dz - v\|^2 + g_\lambda(z). \tag{3.1}$$

Problems (2.5) and (3.1) are equivalent when the dictionary is invertible, which is generally not the case. Detailed comparisons between analysis and synthesis problem formulations can be found in the literature, e.g., in [22].

Proposition 3.1 *Let $D\colon \mathbb{R}^S \to \mathbb{R}^N$, $v \in \mathbb{R}^N$, $\lambda > 0$, and let $g \in \Gamma_0(\mathbb{R}^N)$. Then, for $z_D^\dagger$ defined as in (3.1), we have*

$$x_D^\dagger = Dz_D^\dagger = \operatorname{prox}_{(g_\lambda^* \circ D^*)^*}(v), \tag{3.2}$$

$$= \operatorname{prox}_{D \triangleright g_\lambda}(v), \tag{3.3}$$

where the infimal post-composition operator $\triangleright$ is defined in (1.9).

Proof. According to Fenchel-Rockafellar duality (see, e.g., [4]), the dual problem associated with (3.1) aims to

$$\text{find} \quad u_D^\dagger = \operatorname*{arg\,min}_{u \in \mathbb{R}^N} \frac{1}{2}\|u - v\|^2 + g_\lambda^*(D^*u) = \operatorname{prox}_{g_\lambda^* \circ D^*}(v). \tag{3.4}$$

In addition, according to [4, Thm. 19.1], we have

$$-u_D^\dagger \in \partial_{\frac{1}{2}\|\cdot - v\|_2^2}(Dz_D^\dagger) = \{Dz_D^\dagger - v\} \quad \Leftrightarrow \quad Dz_D^\dagger = v - u_D^\dagger. \tag{3.5}$$

Combining (3.4) and (3.5) leads to $Dz_D^\dagger = (\mathrm{Id} - \operatorname{prox}_{g_\lambda^* \circ D^*})(v)$, which, combined with Moreau's identity gives (3.2). Further, by combining Prop. 12.36(ii) and Prop. 13.24(iv) from [4], we have $(g_\lambda^* \circ D^*)^* = D \triangleright g_\lambda$, hence the final result (3.3). □

Is it important to note that here, we don't consider the traditional output of the synthesis formulation $z_S^\dagger$ but the denoised image $x_S^\dagger$ that is recovered from it. While $z_S^\dagger$ might be non-unique and the application linking some input v to $z_S^\dagger(v) = \operatorname{prox}_{g_\lambda}(v)$ is non-smooth, it is not the case for $x_S^\dagger$, which results from the proximity operator. We can thus deduce from Proposition 3.1 that the PnP-FB algorithm (1.5) with G being the SD aims to

$$\text{find } \widehat{x}_{\mathcal{S}} \in \operatorname*{argmin}_{x \in \mathbb{R}^N} \frac{1}{2}\|Ax - y\|^2 + D \triangleright g_\lambda(x). \tag{3.6}$$

Synthesis problem equivalence Another standard strategy is to consider a full synthesis formulation of (1.2)-(1.4). In this context, the objective is to

$$\text{find } \widehat{z}_{\mathcal{S}} \in \operatorname*{Argmin}_{z \in \mathbb{R}^S} \frac{1}{2}\|ADz - y\|^2 + g_\lambda(z). \tag{3.7}$$

Then, the signal estimate is *synthesized* from the dictionary by computing $D\widehat{z}_{\mathcal{S}}$. This problem can be solved by any sparse coding algorithm. For instance, when g is proximable, algorithms of choice include FB algorithm, and its accelerated versions such as FISTA [5, 9]. In particular, the sparse code estimate $\widehat{z}_{\mathcal{S}}$ can be obtained using FB iterations, as follows

$$\begin{array}{l} \text{for } k = 0, 1, \dots \\ \left\lfloor \begin{array}{l} v_k = z_t - \gamma D^* A^*(ADz_t - y)), \\ z_{k+1} = \operatorname{prox}_{\gamma g_\lambda}(v_k), \end{array} \right. \end{array} \tag{3.8}$$

where $z_0 \in \mathbb{R}^S$, and $\gamma > 0$ is the step-size chosen to enable convergence of $(z_k)_{k\in\mathbb{N}}$ to a solution to (3.7) [17, Thm. 3.4].

The result below highlights the links between the full synthesis problem (3.6) and the analysis problem with SD (3.7).

Theorem 3.2 *Let $f = \frac{1}{2}\|A \cdot -y\|^2$. Then we have*

$$\min_{x\in\mathbb{R}^N} f(x) + D \triangleright g_\lambda(x) = \min_{z\in\mathbb{R}^S} f(Dz) + g_\lambda(z). \tag{3.9}$$

In addition,

(i) *if $\widehat{z}_{\mathcal{S}} \in \mathbb{R}^S$ is a solution to (3.7), then $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}} \in \mathbb{R}^N$ is a solution to (3.6),*

(ii) *if $\widehat{x}_{\mathcal{S}} \in \mathbb{R}^N$ is a solution to (3.6), then there exists a solution $\widehat{z}_{\mathcal{S}} \in \mathbb{R}^S$ to (3.7) such that $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$.*

Proof. Using twice Fenchel-Rockafellar strong duality (see, e.g., Def. 15.19 and Prop. 15.21 in [4]), we have

$$\begin{aligned} \min_{z\in\mathbb{R}^S} f(Dz) + g_\lambda(z) &= -\min_{v\in\mathbb{R}^N} f^*(v) + g_\lambda^*(-D^*v) = -\min_{v\in\mathbb{R}^N} f^*(-v) + g_\lambda^* \circ D^*(v) \\ &= \min_{x\in\mathbb{R}^N} f(x) + (g_\lambda^* \circ D^*)^*(x) \qquad (3.10) \\ &= \min_{x\in\mathbb{R}^N} f(x) + D \triangleright g_\lambda(x), \qquad (3.11) \end{aligned}$$

where in (3.10), the Fenchel-Rockafellar strong duality is applied to f^* and $(g_\lambda^* \circ D^*)$, with $f^{**} = f$ as $f \in \Gamma_0(\mathbb{R}^N)$, and the last equality is obtained using (1.9). This shows (3.9), i.e., that the values of the two problems minima are the same.

(i) Let $\widehat{z}_{\mathcal{S}}$ be a solution to (3.7). We now want to show that $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$ is solution to (3.6). Using the definitions of the infimal post-composition and conjugate functions, we have

$$\begin{aligned} f(D\widehat{z}_{\mathcal{S}}) + D \triangleright g_\lambda(D\widehat{z}_{\mathcal{S}}) &= f(D\widehat{z}_{\mathcal{S}}) + (g_\lambda^* \circ D^*)^*(D\widehat{z}_{\mathcal{S}}) = f(D\widehat{z}_{\mathcal{S}}) + \sup_{x \in \mathbb{R}^N} \langle x, D\widehat{z}_{\mathcal{S}} \rangle - g_\lambda^*(D^*x) \\ &= f(D\widehat{z}_{\mathcal{S}}) + \sup_{x \in \mathbb{R}^N} \langle D^*x, \widehat{z}_{\mathcal{S}} \rangle - g_\lambda^*(D^*x). \end{aligned}$$

Since, for every $x \in \mathbb{R}^N$, $D^*x \in \mathbb{R}^S$, we obtain

$$f(D\widehat{z}_{\mathcal{S}}) + D \triangleright g_\lambda(D\widehat{z}_{\mathcal{S}}) \leq f(D\widehat{z}_{\mathcal{S}}) + \sup_{v \in \mathbb{R}^S} \langle v, \widehat{z}_{\mathcal{S}} \rangle - g_\lambda^*(v) = f(D\widehat{z}_{\mathcal{S}}) + g_\lambda(\widehat{z}_{\mathcal{S}}) \tag{3.12}$$

where the last equality is obtained using the definition of conjugate functions (and $g_\lambda^{**} = g_\lambda$ as $g_\lambda \in \Gamma_0$). Further, according to (3.10), since $\widehat{z}_{\mathcal{S}}$ is a solution to (3.7), we have

$$f(D\widehat{z}_{\mathcal{S}}) + g_\lambda(\widehat{z}_{\mathcal{S}}) = \min_{x \in \mathbb{R}^N} f(x) + D \triangleright g_\lambda(x). \tag{3.13}$$

Let $\widehat{x}_{\mathcal{S}}$ be a solution to (3.6). Combining (3.12) and (3.13), we obtain

$$f(D\widehat{z}_{\mathcal{S}}) + D \triangleright g_\lambda(D\widehat{z}_{\mathcal{S}}) \leq f(\widehat{x}_{\mathcal{S}}) + D \triangleright g_\lambda(\widehat{x}_{\mathcal{S}}).$$

Hence $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$.

(ii) Let $\widehat{x}_{\mathcal{S}}$ be a solution to (3.6). We will show that there exist some $\widehat{z}_{\mathcal{S}}$ such that $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$ and $\widehat{z}_{\mathcal{S}}$ solution to (3.7). Assume that the component of $\widehat{x}_{\mathcal{S}}$ contained in $\mathrm{im}(D)^\perp = \ker D^*$ is not 0, then

$$\sup_{v \in \mathbb{R}^N} \langle v, \widehat{x}_{\mathcal{S}} \rangle - g_\lambda^*(D^*v) \geq \sup_{v \in \ker D^*} \langle v, \widehat{x}_{\mathcal{S}} \rangle - g_\lambda^*(0) \geq +\infty.$$

As the value of (3.10) is finite, this would violate the optimality of $\widehat{x}_{\mathcal{S}}$. Thus $\widehat{x}_{\mathcal{S}}$ is contained in the image of D, and there indeed exists $\widehat{z}_{\mathcal{S}} \in \mathbb{R}^S$ such that $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$. By definition of the infimal post-composition, we have

$$\begin{aligned} f(\widehat{x}_{\mathcal{S}}) + D \triangleright g_\lambda(\widehat{x}_{\mathcal{S}}) &= f(\widehat{x}_{\mathcal{S}}) + \inf_{z \in \mathbb{R}^S, \widehat{x}_{\mathcal{S}} = Dz} g_\lambda(z) \\ &= \min_{z \in \mathbb{R}^S, \widehat{x}_{\mathcal{S}} = Dz} f(Dz) + g_\lambda(z). \end{aligned} \tag{3.14}$$

Then, there exists $\widehat{z}_{\mathcal{S}}$ such that $\widehat{x}_{\mathcal{S}} = D\widehat{z}_{\mathcal{S}}$, solution to (3.14) and we have

$$f(D\widehat{z}_{\mathcal{S}}) + g_\lambda(\widehat{z}_{\mathcal{S}}) = f(\widehat{x}_{\mathcal{S}}) + D \triangleright g_\lambda(\widehat{x}_{\mathcal{S}}).$$

Hence, thanks to (3.11), $\widehat{z}_{\mathcal{S}}$ is a minimizer of (3.7) and this conclude the proof. □

Theorem 3.2 shows that the minimum values of Problems (3.6) and (3.7) are the same. It also highlights that any solution to (3.6) can be projected through D to find a solution to (3.7). Reciprocally, any solution to (3.7) can be generated by some synthesis coefficients of D which are also solutions to (3.6).

Example 3.3 In the particular case when $g = \lambda\|\cdot\|_1$, then Theorem 3.2 ensures that

$$\begin{aligned}\min_{z\in\mathbb{R}^S} \frac{1}{2}\|y - ADz\|_2^2 + \lambda\|z\|_1 &= \min_{x\in\mathbb{R}^N} \frac{1}{2}\|y - Ax\|_2^2 + \iota^*_{\mathcal{F}^S_\lambda(D)}(x) \\ &= \min_{x\in\mathbb{R}^N} \max_{x_0\in\mathcal{F}^S_\lambda(D)} \frac{1}{2}\|Ax - y\|^2 + \langle x, x_0\rangle, \end{aligned} \tag{3.15}$$

where $\mathcal{F}^S_\lambda(D) = \{x \mid \|D^*x\|_\infty \leq \lambda\}$.

One can note the similarity between the analysis and synthesis problems formulations (3.15) and (2.13). Indeed the analysis and the synthesis Lasso formulations for solving the inverse problem (1.1) share the same min-max optimization structure, with different convex constraints on the max variable.

3.2 Synthesis denoiser structure

As explained in Section 1, and similarly to the AD problem, the SD problem (1.7) does not have an explicit solution. However, it can be solved using FB iterations when g is proximable [54, 20, 17, 16]. The resulting iterations read

$$\begin{array}{l}\text{for } \ell = 0, 1, \dots \\ \quad \left\lfloor\ z_{\ell+1} = \operatorname{prox}_{\zeta g_\lambda}\big(z_\ell - \zeta D^*(Dz_\ell - v)\big),\right.\end{array} \tag{3.16}$$

where $z_0 \in \mathbb{R}^S$, and $\zeta > 0$ is a step-size. Then the following convergence result can be deduced from [17, Thm. 3.4].

Theorem 3.4 *Assume that there exists $\underline{\zeta} > 0$ such that $\underline{\zeta} < \zeta < (2 - \underline{\zeta})\|D\|_S^{-2}$. Then the sequence $(z_\ell)_{\ell\in\mathbb{N}}$ converges to $z_D^\dagger$, and $x_D^\dagger = Dz_D^\dagger$ is a solution to (3.1).*

algorithm (3.16) can be used as sub-iterations when included within the global FB algorithm (1.5), to approximate the computation of $\operatorname{prox}_{D\triangleright g_\lambda}$. Similarly to Section 2, in this section we investigate the behavior of (1.5) with the SD, when only one iteration of algorithm (3.16) is computed. To this aim, we define an unfolded iterative scheme approximating the proximal operator of $p = D \triangleright g_\lambda$, i.e., an SD as defined in (1.7).

Model 3.5 (Synthesis denoiser (SD)) Let $(v, z_0) \in \mathbb{R}^N \times \mathbb{R}^S$, $D\colon \mathbb{R}^S \to \mathbb{R}^N$ be a linear operator, and $\lambda > 0$ be a regularization parameter. Let $L \in \mathbb{N}^*$ and $\widetilde{G}^{\mathcal{S}}_{L,\lambda,v}\colon \mathbb{R}^S \to \mathbb{R}^S$ be defined as

$$\widetilde{G}^{\mathcal{S}}_{L,\lambda,v}(z_0) = \underbrace{T^{\mathcal{S}}_{\lambda,v} \circ \cdots \circ T^{\mathcal{S}}_{\lambda,v}}_{L \text{ compositions}}(z_0), \tag{3.17}$$

where

$$T^{\mathcal{S}}_{\lambda,v}\colon \mathbb{R}^S \to \mathbb{R}^S\colon z \mapsto \operatorname{prox}_{\zeta g_\lambda}\big(z - \zeta D^*(Dz - v)\big) \tag{3.18}$$

with $\zeta > 0$. The unrolled SD $G^{\mathcal{S}}_{L,\lambda,v} \colon \mathbb{R}^S \to \mathbb{R}^N$ is obtained as follows:

$$G^{\mathcal{S}}_{L,\lambda,v}(z_0) = D\widetilde{G}^{\mathcal{S}}_{L,\lambda,v}(z_0). \tag{3.19}$$

A few comments can be made on Model 3.5.

Remark 3.6

(i) The layers of Model 3.5 consisting of FB iterations, they correspond to feed-forward layers, and $T^{\mathcal{S}}_{\lambda,v}$ can be reformulated as

$$(\forall z \in \mathbb{R}^S) \quad T^{\mathcal{S}}_{\lambda,v}(z) = \operatorname{prox}_{\zeta g_\lambda}\Big(\big(\operatorname{Id} - \zeta D^* D\big)z + \zeta D^* v\Big).$$

(ii) The feed-forward layers $T^{\mathcal{A}}_{\lambda,v}$ and $T^{\mathcal{S}}_{\lambda,v}$ share the same structure, except that the activation function in $T^{\mathcal{A}}_{\lambda,v}$ corresponds to the proximity operator of g^*_λ, while for $T^{\mathcal{S}}_{\lambda,v}$ it corresponds to the proximity operator of g_λ.

(iii) A direct consequence of Theorem 3.4 is that, for any $z_0 \in \mathbb{R}^S$, if there exists $\underline{\zeta} > 0$ such that $\underline{\zeta} \leq \zeta \leq (2 - \underline{\zeta})\|D\|_S^{-2}$, then $\lim_{L \to +\infty} G^{\mathcal{S}}_{L,\lambda,v}(z_0) = x^\dagger_D$, where $x^\dagger_D$ is defined in (3.1).

3.3 FB-PnP algorithm with approximated SD

In this section we aim to solve (3.6). We thus study the asymptotic behavior of the FB-PnP algorithm (1.5) with the SD given in Model 3.5. In this context, this algorithm boils down to

$$\begin{aligned}
&x_0 \in \mathbb{R}^N, z_0 \in \mathbb{R}^S, \\
&\text{for } k = 0, 1, \dots \\
&\quad v_k = x_k - \tau A^*(Ax_k - y), \\
&\quad z_{k+1} = G^{\mathcal{S}}_{L,\tau\lambda,v_k}(z_k), \\
&\quad x_{k+1} = Dz_{k+1},
\end{aligned} \tag{3.20}$$

where $\tau > 0$ is the stepsize of the FB algorithm and $L \in \mathbb{N}^*$ is the number of iterations of the AD in Model 2.2. The following convergence result is a direct application of results from [17].

Theorem 3.7 *Let $(x_k)_{k\in\mathbb{N}}$ be generated by algorithm (3.20). Assume that there exist $\underline{\tau} > 0$ and $\underline{\zeta} > 0$ such that $\underline{\tau} \leq \tau \leq (2 - \underline{\tau})\|A\|_S^{-2}$ and $\underline{\zeta} < \zeta < (2 - \underline{\zeta})\|D\|_S^{-2}$. When $L \to \infty$,* i.e. *i.e. when we use an infinite number of iterations in Model 3.5, then $(x_k)_{k\in\mathbb{N}}$ converges to a solution to problem (3.6).*

Theorem 3.7 only holds when $L \to \infty$, i.e. when $G^{\mathcal{S}}$ has a very large number of sub-iterations (i.e., layers). As for the AD case, this is often intractable, and only a fixed (low) number of iterations are considered in practice. In the remainder of this section, we focus on the case when

only $L = 1$ sub-iteration is computed in Model 3.5, and investigate the behavior of $(x_k)_{k\in\mathbb{N}}$ in this context, as well as the nature of the provided solution. In this particular case, at iteration $k \in G$, the update of z_k in algorithm (3.20) boils down to

$$z_{k+1} = G^{\mathcal{S}}_{1,\tau\lambda,v_k}(z_k) = T^{\mathcal{S}}_{\tau\lambda,v_k}(z_k), \tag{3.21}$$

where $T^{\mathcal{S}}_{\tau\lambda,v_k}$ is defined in (3.18).

We will now show that algorithm (3.20)-(3.21) is equivalent to the FB iterations in (3.8) for solving (3.7). This will enable us to deduce asymptotic convergence guarantees of algorithm (3.20) when $L = 1$.

Theorem 3.8 *Let $(z_k)_{k\in\mathbb{N}}$ and $(x_k)_{k\in\mathbb{N}}$ be sequences generated by algorithm (3.20) with $L = 1$. Assume that there exists $\varepsilon > 0$ such that $\varepsilon \leq \tau\zeta \leq (2-\varepsilon)\|AD\|_S^{-2}$, then*

(i) *$(z_k)_{k\in\mathbb{N}}$ converges to a solution $\widehat{z}_S$ to problem (3.7).*

(ii) *$(x_k)_{k\in\mathbb{N}}$ converges to a solution $\widehat{x}_S$ to problem (3.6).*

Proof. Expending algorithm (3.20)-(3.21), we obtain, for every $k \in \mathbb{N}$,

$$v_k = x_k - \tau A^*(Ax_k - y) \tag{3.22}$$
$$\widetilde{z}_k = z_k - \zeta D^*(Dz_k - v_k) \tag{3.23}$$
$$z_{k+1} = \operatorname{prox}_{\tau\zeta g_\lambda}(\widetilde{z}_k) \tag{3.24}$$
$$x_{k+1} = Dz_{k+1}. \tag{3.25}$$

Combining (3.22) and (3.23), and then using (3.25), we obtain

$$\begin{aligned}\widetilde{z}_k &= z_k - \zeta D^*\Big(Dz_k - \big(x_k - \tau A^*(Ax_k - y)\big)\Big)\\ &= z_k - \zeta D^*\Big(Dz_k - \big(Dz_k - \tau A^*(ADz_k - y)\big)\Big)\\ &= z_k - \tau\zeta D^*A^*(ADz_k - y).\end{aligned}$$

Hence, algorithm (3.20)-(3.21) is equivalent to

$$\begin{array}{l}\text{for } k = 0, 1, \dots\\ \quad \left|\begin{array}{l}\widetilde{z}_k = z_k - \tau\zeta D^*A^*(ADz_k - y),\\ z_{k+1} = \operatorname{prox}_{\tau\zeta g_\lambda}(\widetilde{z}_k),\\ x_{k+1} = Dz_{k+1}.\end{array}\right.\end{array} \tag{3.26}$$

algorithm (3.26) corresponds to the FB iterations given in (3.8) with step-size $\gamma = \tau\zeta$. Then, according to [17, Thm. 3.4], $(z_k)_{k\in\mathbb{N}}$ converges to a solution $\widehat{z}_S$ to (3.7) if there exists $\varepsilon > 0$ such that $\varepsilon \leq \tau\zeta \leq (2-\varepsilon)\|AD\|_S^2$. Further, according to Theorem 3.2(i), $\widehat{x}_S = D\widehat{z}_S$ is a solution to (3.6). □

To summarize, in Theorem 3.8, as in the case of AD, we show that we can recover the solution of the full problem (3.6) using only $L = 1$ sub-iterations, provided that we use a restarting scheme.

4 Approximated AD and SD for smooth minimization

Algorithms (2.7)-(2.8) and (3.20)-(3.21) enable using a unique sub-iteration of an unrolled FB-based algorithm within a FB-PnP framework, for solving (2.6) and (3.6), respectively. In particular, we showed in Theorem 2.5 and Theorem 3.8 that such an approximation enables solving the problems of interest (2.6) and (3.6), respectively, as if an infinite number of sub-iterations were computed. In this section, we focus on a particular case when a smoothed version of the problems of interest is considered.

In this context, we do not need to consider the AD and SD problems separately. Instead, we focus on a generic problem of the form

$$\underset{x\in\mathbb{R}^{\widetilde{N}}}{\text{minimize}}\ \widetilde{f}(x)+\widetilde{g}_\lambda(x), \tag{4.1}$$

where $\widetilde{f}\colon \mathbb{R}^{\widetilde{N}}\to]-\infty,+\infty]$ is assumed to be convex and $\widetilde{\beta}$-Lipschitz differentiable, for $\widetilde{\beta}>0$, and $\widetilde{g}_\lambda\colon \mathbb{R}^{\widetilde{N}}\to]-\infty,+\infty]$ is a convex, proper and lower-semicontinuous function. Then, the synthesis problem is a particular case of (4.1) where

$$\begin{cases} \widetilde{f}\colon \mathbb{R}^{S}\to]-\infty,+\infty]\colon z\mapsto \|ADz-y\|^2 \\ \widetilde{g}_\lambda\colon \mathbb{R}^{S}\to]-\infty,+\infty]\colon z\mapsto g_\lambda(z). \end{cases} \tag{4.2}$$

Similarly, the analysis problem is a particular case of (4.1) where

$$\begin{cases} \widetilde{f}\colon \mathbb{R}^{N}\to]-\infty,+\infty]\colon x\mapsto \|Ax-y\|^2 \\ \widetilde{g}_\lambda\colon \mathbb{R}^{N}\to]-\infty,+\infty]\colon x\mapsto g_\lambda(\Gamma x). \end{cases} \tag{4.3}$$

Remark 4.1 We would emphasize that, according to Theorem 3.2, if $\widehat{z}$ is a solution to (4.1)-(4.2), then $D\widehat{z}$ is a solution to (3.6).

In this section, we focus on a smooth regularized version of problem (4.1), aiming to

$$\text{find } x^{\ddagger}=\underset{x\in\mathbb{R}^{\widetilde{N}}}{\operatorname{argmin}}\ \widetilde{f}(x)+{}^{\mu}\widetilde{g}_\lambda(x), \tag{4.4}$$

where ${}^{\mu}\widetilde{g}_\lambda$ denotes de Moreau-Yosida envelope of $\widetilde{g}_\lambda$ with parameter μ, defined as

$$(\forall x\in\mathbb{R}^{\widetilde{N}})\quad {}^{\mu}\widetilde{g}_\lambda(x)=\min_{u\in\mathbb{R}^{\widetilde{N}}}\ \widetilde{g}_\lambda(u)+\frac{\mu}{2}\|x-u\|^2.$$

According to [4, Prop. 12.30], ${}^{\mu}\widetilde{g}_\lambda$ is μ-Lipschitz-differentiable, with gradient given by

$$(\forall x\in\mathbb{R}^{\widetilde{N}})\quad \nabla{}^{\mu}\widetilde{g}_\lambda(x)=\mu\big(x-\operatorname{prox}_{\frac{\widetilde{g}_\lambda}{\mu}}(x)\big).$$

Hence, Problem (4.4) can be solved using a gradient descent algorithm, given by

$$\begin{array}{l} x_0 \in \mathbb{R}^N, u_0 \in \mathbb{R}^N, \\ \text{for } k = 0, 1, \dots \\ \left\lfloor \begin{array}{l} x_{k+1} = x_k - \tau \nabla_x F(x_k, u_k), \\ u_{k+1} = \operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x_{k+1}), \end{array} \right. \end{array} \tag{4.5}$$

where, for every $(x, u) \in \mathbb{R}^{\widetilde{N}} \times \mathbb{R}^{\widetilde{N}}$, $F(x, u) = \widetilde{f}(x) + \widetilde{g}_\lambda(u) + \frac{\mu}{2}\|x - u\|^2$, and

$$\nabla_x F(x, u) = \nabla \widetilde{f}(x) + \mu(x - u) \tag{4.6}$$

is Lipschitz-differentiable, with constant $\widetilde{\beta} + \mu$. The sequence $(x_k)_{k\in\mathbb{N}}$ generated by algorithm (4.5) is then ensured to converge to the solution $x^\ddagger$ to (4.4), if there exists $\underline{\zeta} > 0$ such that $\underline{\zeta} \leq \tau \leq (2 - \underline{\zeta})(\widetilde{\beta} + \mu)^{-1}$.

In practice, as the proximity operator of $\widetilde{g}_{\frac{\lambda}{\mu}}$ often does not have a closed form, the computational complexity of this algorithm is too high. However, the solution of the proximity operator can be computed using iterative schemes such as the FB or the dual-FB algorithms, as described in Sections 2 and 3. Using a bilevel framework similar to the one proposed by [19], we show that one can use inexact proximity operator computation powered by iterative proximal algorithms with warm restarts to build an algorithm that converges to a solution to (4.4). In this context, we then focus on the following algorithm

$$\begin{array}{l} x_0 \in \mathbb{R}^N, u_0 \in \mathbb{R}^N, \\ \text{for } k = 0, 1, \dots \\ \left\lfloor \begin{array}{l} x_{k+1} = x_k - \tau \nabla_x F(x_k, u_k), \\ u_{k+1} = \widetilde{G}_{L,\lambda\mu^{-1}}(x_{k+1}, u_k), \end{array} \right. \end{array} \tag{4.7}$$

where, for every $k \in \mathbb{N}$, $\widetilde{G}_{L,\lambda\mu^{-1}} \colon \mathbb{R}^{\widetilde{N}} \times \mathbb{R}^{\widetilde{N}} \to \mathbb{R}^{\widetilde{N}}$ is an operator that aims to approximate $\operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x_{k+1})$ by computing L sub-iterations of some iterative algorithm. The first input corresponds to the output x_{k+1} of the gradient descent step, *i.e.* the point at which the proximity step should be computed. The second input corresponds to the output of $\widetilde{G}_{L,\lambda\mu^{-1}}$ from the previous iteration $k - 1$, which is aimed to be used for warm-restart, i.e. initializing the sub-iterations to compute the approximation of $\operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x_{k+1})$. Unlike in previous sections, in this section, this operator is not necessarily based on FB or dual-FB iterations. However, we will assume that it satisfies some sufficient decrease property (see Theorem 4.3 Condition (i)).

Remark 4.2 It can be noticed that the regularized problem (4.4) can equivalently be rewritten as the following bi-level optimization problem

$$\text{find } x^\ddagger = \operatorname*{argmin}_{x \in \mathbb{R}^{\widetilde{N}}} \widetilde{f}(x) + \widetilde{g}_\lambda(u_x^\ddagger) + \frac{\mu}{2}\|x - u_x^\ddagger\|^2 \quad \text{such that} \quad u_x^\ddagger = \operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x), \tag{4.8}$$

where $\mu > 0$. Hence, the sequence $(x_k)_{k\in\mathbb{N}}$ generated by (4.5) is ensured to converge to a solution to (4.8) if there exists $\underline{\tau} > 0$ such that $\underline{\tau} \leq \tau \leq (2 - \underline{\tau})(\widetilde{\beta} + \mu)^{-1}$.

In the following result, we analyse the asymptotic behavior of algorithm (4.7). The two conditions on $\widetilde{G}_{L,\lambda\mu^{-1}}$ and on the step size τ are commented in Remark 4.4.

Theorem 4.3 *Let $(x_k)_{k\in\mathbb{N}}$ and $(u_k)_{k\in\mathbb{N}}$ be sequences generated by (4.7). Assume that the following conditions hold:*

(i) *there exists $\alpha_L \in]0, 1/\sqrt{2}[$ such that*

$$(\forall k \in \mathbb{N}) \quad \|\widetilde{G}_{L,\lambda\mu^{-1}}(x_{k+1}, u_k) - u^{\ddagger}_{x_{k+1}}\| \leq \alpha_L \|u_k - u^{\ddagger}_{x_{k+1}}\|, \tag{4.9}$$

where $u^{\ddagger}_{x_{k+1}} = \operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x_{k+1})$,

(ii) *the step-size τ in (4.7) satisfies $0 < \tau < \frac{2\phi^{\ddagger}(1-2\alpha_L^2)}{\mu^2}$, where $\phi^{\ddagger} > 0$ is the positive root of the polynomial*

$$\phi \in \mathbb{R}_* \mapsto p(\phi) = (8\alpha_L^2(1-2\alpha_L^2))\phi^2 + 2(\widetilde{\beta}+\mu)(1-2\alpha_L^2)\phi - \mu^2. \tag{4.10}$$

Then, $(x_k)_{k\in\mathbb{N}}$ converges to a solution to (4.8).

Proof. Let $k \in \mathbb{N}$, and let us define, for every $x \in \mathbb{R}^{\widetilde{N}}$, $h(x) = F(x, u^{\ddagger}_x)$, where $u^{\ddagger}_x \in \mathbb{R}^{\widetilde{N}}$ is defined in (4.8). According to (4.6), since $\nabla_x F(\cdot, u^{\ddagger}_x)$ is $(\widetilde{\beta}+\mu)$-Lipschitz, then h is Lipschitz-differentiable, with constant $\widetilde{\beta}+\mu > 0$. Thus we can apply the descent lemma from [6, Prop. A.24] to obtain

$$h(x_{k+1}) = h(x_k - \tau\nabla_x F(x_k, u_k)) \leq h(x_k) - \tau\langle \nabla_x F(x_k, u_k)), \nabla h(x_k)\rangle + \frac{(\widetilde{\beta}+\mu)\tau^2}{2}\|\nabla_x F(x_k, u_k))\|^2. \tag{4.11}$$

Further, we have

$$\frac{1}{2}\|\nabla_x F(x_k, u_k) - \nabla h(x_k)\|^2 = \frac{1}{2}\|\nabla_x F(x_k, u_k)\|^2 + \frac{1}{2}\|\nabla h(x_k)\|^2 - \langle \nabla_x F(x_k, u_k)), \nabla h(x_k)\rangle. \tag{4.12}$$

Then, by combining (4.11) and (4.12), we obtain

$$\begin{aligned} h(x_{k+1}) &\leq h(x_k) - \frac{\tau}{2}\|\nabla_x F(x_k, u_k)\|^2 - \frac{\tau}{2}\|\nabla h(x_k)\|^2 + \frac{\tau}{2}\|\nabla_x F(x_k, u_k) - \nabla h(x_k)\|^2 \\ &\quad + \frac{(\widetilde{\beta}+\mu)\tau^2}{2}\|\nabla_x F(x_k, u_k))\|^2 \\ &= h(x_k) - \frac{\tau}{2}\|\nabla h(x_k)\|^2 - \frac{\tau\big(1-(\widetilde{\beta}+\mu)\tau\big)}{2}\|\nabla_x F(x_k, u_k)\|^2 \\ &\quad + \frac{\tau}{2}\|\nabla_x F(x_k, u_k) - \nabla h(x_k)\|^2. \end{aligned} \tag{4.13}$$

Furthermore, by definition of F and h, we have

$$\begin{aligned} \nabla_x F(x_k, u_k) - \nabla h(x_k) &= \nabla_x\Big(F(x_k, u_k) - F(x_k, u^{\ddagger}_{x_k})\Big) = \mu(x_k - u_k) - \mu(x_k - u^{\ddagger}_{x_k}) \\ &= \mu(u^{\ddagger}_{x_k} - u_k). \end{aligned}$$

Hence, combining this equality with (4.13), we obtain

$$h(x_{k+1}) - h(x_k) \leq -\frac{\tau}{2}\|\nabla h(x_k)\|^2 - \frac{\tau\big(1-(\widetilde{\beta}+\mu)\tau\big)}{2}\|\nabla_x F(x_k,u_k)\|^2 + \frac{\tau\mu^2}{2}\|u^\ddagger_{x_k} - u_k\|^2. \quad (4.14)$$

According to (4.7) and to (4.9), we have

$$\begin{aligned}\|u_{k+1} - u^\ddagger_{x_{k+1}}\| &\leq \alpha_L\|u_k - u^\ddagger_{x_{k+1}}\| = \alpha_L\|(u_k - u^\ddagger_{x_k}) + (u^\ddagger_{x_k} - u^\ddagger_{x_{k+1}})\| \\ &\leq \alpha_L\|u_k - u^\ddagger_{x_k}\| + \alpha_L\|u^\ddagger_{x_k} - u^\ddagger_{x_{k+1}}\|. \end{aligned} \quad (4.15)$$

Since the proximity operator is 1-Lipschitz [4,], we have

$$\|u^\ddagger_{x_k} - u^\ddagger_{x_{k+1}}\| \leq \|x_k - x_{k+1}\| = \tau\|\nabla_x F(x_k,u_k)\|, \quad (4.16)$$

where the last equality is obtained by definition of x_{k+1} in (4.7). By combining (4.15) and (4.16), we have

$$\|u_{k+1} - u^\ddagger_{x_{k+1}}\| \leq \alpha_L\|u_k - u^\ddagger_{x_k}\| + \alpha_L\tau\|\nabla_x F(x_k,u_k)\|,$$

and by applying the Jensen's inequality we obtain

$$\|u_{k+1} - u^\ddagger_{x_{k+1}}\|^2 \leq 2\alpha_L^2\|u_k - u^\ddagger_{x_k}\|^2 + 2\alpha_L^2\tau^2\|\nabla_x F(x_k,u_k)\|^2. \quad (4.17)$$

For $\phi > 0$, we introduce the Lyapunov function

$$(\forall k \in \mathbb{N}) \quad \mathcal{L}_k = h(x_k) + \phi\|u_k - u^\ddagger_{x_k}\|^2 \quad (4.18)$$

and we will show that $(\mathcal{L}_k)_{k\in\mathbb{N}}$ is a decreasing sequence. By definition of $\mathcal{L}_k$, and using (4.14) and (4.17), we obtain

$$\begin{aligned}\mathcal{L}_{k+1} - \mathcal{L}_k &= h(x_{k+1}) - h(x_k) + \phi\big(\|u_{k+1} - u^\ddagger_{x_{k+1}}\|^2 - \|u_k - u^\ddagger_{x_k}\|^2\big) \\ &\leq -\frac{\tau}{2}\|\nabla h(x_k)\|^2 - \frac{\tau\big(1-(\widetilde{\beta}+\mu)\tau\big)}{2}\|\nabla_x F(x_k,u_k)\|^2 + \frac{\tau\mu^2}{2}\|u^\ddagger_{x_k} - u_k\|^2 \\ &\qquad + \phi\Big((2\alpha_L^2-1)\|u_k - u^\ddagger_{x_k}\|^2 + 2\alpha_L^2\tau^2\|\nabla_x F(x_k,u_k)\|^2\Big) \\ &\leq -\frac{\tau}{2}\|\nabla h(x_k)\|^2 - \tau\Big(\frac{1-(\widetilde{\beta}+\mu)\tau}{2} - \phi 2\alpha_L^2\tau\Big)\|\nabla_x F(x_k,u_k)\|^2 \\ &\qquad - \Big(\phi(1-2\alpha_L^2) - \frac{\tau\mu^2}{2}\Big)\|u_k - u^\ddagger_{x_k}\|^2\end{aligned}$$

If $\tau > 0$, $\frac{1-(\widetilde{\beta}+\mu)\tau}{2} - \phi 2\alpha_L^2\tau > 0$, and $\phi(1-2\alpha_L^2) - \frac{\tau\mu^2}{2} > 0$, then $(\mathcal{L}_k)_{k\in\mathbb{N}}$ is a decreasing sequence. Hence $0 < \alpha_L < \frac{1}{\sqrt{2}}$, and τ must satisfy $0 < \tau < \min\Big\{\frac{2\phi(1-2\alpha_L^2)}{\mu^2}, \frac{1}{\widetilde{\beta}+\mu+4\phi\alpha_L^2}\Big\}$. We thus can choose $\phi > 0$ to maximize the upper bound $\min\Big\{\frac{2\phi(1-2\alpha_L^2)}{\mu^2}, \frac{1}{\widetilde{\beta}+\mu+4\phi\alpha_L^2}\Big\}$, *i.e.*, such that $\frac{2\phi(1-2\alpha_L^2)}{\mu^2} = \frac{1}{\widetilde{\beta}+\mu+4\phi\alpha_L^2}$. This is equivalent to find the positive root of polynomial (4.10). The determinant of

this polynomial is positive with a positive solution, as it is negative in $\phi = 0$ and positive in $\phi \to \infty$. By denoting $\phi^\ddagger$ the positive root of this polynomial, we then obtain that if $0 < \tau < \frac{2\phi^\ddagger(1-2\alpha_L^2)}{\mu^2}$, then $(\mathcal{L}_k)_{k\in\mathbb{N}}$ is decreasing, and that

$$\frac{\tau}{2}\|\nabla h(x_k)\|^2 \leq \mathcal{L}_k - \mathcal{L}_{k+1}. \tag{4.19}$$

Hence, by summing (4.19) over $k \in \{0, \ldots, K-1\}$, for $K > 0$, we obtain

$$\sum_{k=0}^{K} \frac{\tau}{2}\|\nabla h(x_k)\|^2 \leq \mathcal{L}_0 - \mathcal{L}_K < +\infty.$$

Thus $(\|\nabla h(x_k)\|^2)_{k\in\mathbb{N}}$ converges to 0. Since h is convex with continuous gradient, the gradient vanishes at a global minimum, and $(x_k)_{k\in\mathbb{N}}$ converges to a solution to (4.8).
□

Remark 4.4

(i) Condition (i) in Theorem 4.3 means that the operator $\widetilde{G}_{L,\lambda\mu^{-1}}$ in (4.7) must be chosen to satisfy some sufficient decrease property in the sense that, for every $k \in \mathbb{N}$, $u_{k+1} = \widetilde{G}_{L,\lambda\mu^{-1}}(x_{k+1}, u_k)$ should be closer to $\operatorname{prox}_{\widetilde{g}_{\frac{\lambda}{\mu}}}(x_{k+1})$ than u_k. In other words, this means that $\widetilde{G}_{L,\lambda\mu^{-1}}$ decreases with respect to its second variable (i.e., the initialisation of that operator). In practice, if we choose a monotone algorithm such as FB or dual-FB, this condition can be satisfied for an L sufficiently large. However, it can be difficult to determine the value of L ensuring $\alpha_L < 1/\sqrt{2}$.

(ii) For $(x, u) \in (\mathbb{R}^N)^2$, we define $\bar{G}_\lambda(x, u) = \frac{1}{2}\|x - u\| + \widetilde{g}_\lambda(u)$. If L iterations of the algorithm that estimates $u_x^\ddagger = \operatorname{prox}_{\widetilde{g}_\lambda}(x) = \min_u \bar{G}_\lambda(x, u)$, warm started with a value u_ℓ produces an estimate $u_{\ell+L}$ such that

$$\bar{G}_\lambda(x, u_{\ell+L}) - \bar{G}_\lambda(x, u_x^\ddagger) \leq \frac{C}{L}\|u_\ell - u_x^\ddagger\|^2$$

where $C > 0$ is a constant that does not depend on x, then, by λ^{-1}-strong convexity of $\bar{G}_\lambda(x, \cdot)$ we have

$$\|u_{\ell+L} - u_x^\ddagger\|_2^2 \leq \frac{2\lambda C}{L}\|u_\ell - u_x^\ddagger\|^2.$$

In this case, L should be chosen larger than $4\lambda C$ to satisfy assumption (4.9). Note that this is typically the case when g is the SD formulation and we use FB as a inner solver [5].

(iii) When α_L is unknown, the root of the polynomial (4.10) cannot be computed. In this case, one can choose $\phi = 1$ in the Lyapunov function (4.18), and $0 < \tau < \min\left\{\frac{2(1-2\alpha_L^2)}{\mu^2}, \frac{1}{\widetilde{\beta}+\mu+4\alpha_L^2}\right\}$. Since $0 < \alpha_L^2 < 1/2$, if τ is chosen to satisfy

$0 < \tau < \min\left\{2/\mu^2, 1/(\widetilde{\beta}+\mu)\right\}$, then $(x_k)_{k\in\mathbb{N}}$ generated by algorithm (4.7) converges to a solution to (4.8).
In practice, since μ is the smoothing parameter, it is often chosen very small. In this context, we recover a similar condition as for the convergence of (4.5), that is $0 < \tau < 1/(\mu+\widetilde{\beta})$.

5 Numerical experiments

We investigate the behavior of algorithm (1.5), in the settings described in Section 2 and Section 3, i.e. with G built as sub-iterations for solving either the analysis or the synthesis Gaussian denoising problem.

In Section 5.1 we study the stability of the FB-PnP iterations described in Section 2 and Section 3. We in particular aim to illustrate results presented in Theorem 2.5 and in Theorem 3.8, for the analysis and synthesis denoisers (AD and SD, respectively). This study is conducted on a toy compressed sensing example.

In Section 5.2 we will investigate the behavior of the proposed approaches in a deep dictionary learning (DDL) framework, where the linear operators Γ and D are learned. We will apply the resulting DDL methods to a deblurring image problem, and for the sake of completeness we will show that they are very competitive with more advanced PnP methods where the denoiser is a full neural network (namely DRUnet).

5.1 Stability analysis of the FB-PnP iterations

In this section we aim to analyze the stability of the FB-PnP iterations when using the proposed AD or SD, depending on the number of layers L. Specifically, we want to illustrate the results presented in Theorem 2.5 and in Theorem 3.8. Although these results only hold when $L = 1$ of $L \to \infty$ (i.e., the proximity operators are computed accurately), we will also empirically investigate the behavior of the AD and SD when $1 < L < +\infty$. To this aim, we consider a toy example consisting of an equation system of the form of $y = A\overline{x}$, where $\overline{x} \in \mathbb{R}^N$ is a vector of dimension $N = 50$ generated randomly following a uniform distribution, $A\colon \mathbb{R}^N \to \mathbb{R}^M$ is a Gaussian measurement operator with $M = 20$, where coefficients are generated randomly following a Gaussian distribution with mean 0 and standard deviation 1. Both the analysis and synthesis dictionaries are generated as Gaussian dictionaries with mean 0 and standard deviation 1, of size $\Gamma \in \mathbb{R}^{100\times 50}$ and $D \in \mathbb{R}^{50\times 100}$, respectively.

5.1.1 Analysis denoiser

In Theorem 2.5 we showed that taking $L = 1$ in algorithm (2.7) is equivalent to using a primal-dual algorithm, namely the Loris-Verhoeven algorithm [35]. In particular, the resulting algorithm, given in (2.8), leads to the same asymptotic solution as when taking $L \to \infty$ in algorithm (2.7). In this section we aim to illustrate this result by comparing the output of algorithm (2.8) (i.e., $L = 1$) with the output of algorithm (2.7) for $L = 10^4$. Furthermore, although the results from Theorem 2.5 only hold for $L = 1$, we show empirically that they remain true for $1 < L < +\infty$ for our problem by considering $L \in \{20, 50, 100\}$.

In fig. 1 we show the behavior of $(\|x_k - x^*\|)_{k\in\mathbb{N}}$ and $(\|u_k - u^*\|)_{k\in\mathbb{N}}$ with $(x_k, u_k)_{k\in\mathbb{N}}$ generated by algorithm (2.7) with $L \in \{1, 20, 50, 100\}$, and (x^*, u^*) generated by algorithm (2.7) with $L \to \infty$ (obtained after 10^4 iterations). Note that x^* is solution to problem (2.6). We see that the trajectory per iteration is the same independently of the value of L. Hence, according to Theorem 2.5 we can deduce that $(x^*, \tau^{-1}u^*)$ is solution to the saddle-point problem (2.9). We can further observe that the convergence is faster in time for smaller values of L.

5.1.2 Synthesis denoiser

We shown in Theorem 3.8 that taking $L = 1$ in algorithm (3.20) enables solving (3.6) as well as the direct sparse coding problem (3.7). Since algorithm (3.20) for $L \to \infty$ solves (3.6) (theorem 3.7), we deduce that both algorithm (3.20) for $L \to \infty$ and algorithm (3.21) (i.e., $L = 1$) solve the same problem (3.6). In this section we aim to illustrate this result by comparing the output of algorithm (3.21) with the output of algorithm (3.20) for $L = 10^4$. Furthermore, although the results from Proposition 3.1 only hold for $L = 1$, we show empirically that they remain true for $1 < L < +\infty$ for our problem by considering $L \in \{20, 50, 100\}$.

In fig. 2 we show the behavior of $(\|x_k - x^*\|)_{k\in\mathbb{N}}$ and $(\|z_k - z^*\|)_{k\in\mathbb{N}}$ with $(x_k, z_k)_{k\in\mathbb{N}}$ generated by algorithm (3.20) with $L \in \{1, 20, 50, 100\}$, and (x^*, z^*) generated by algorithm (3.20) with $L \to \infty$ (obtained after 10^4 iterations). Note that x^* is solution to problem (3.6). We see that

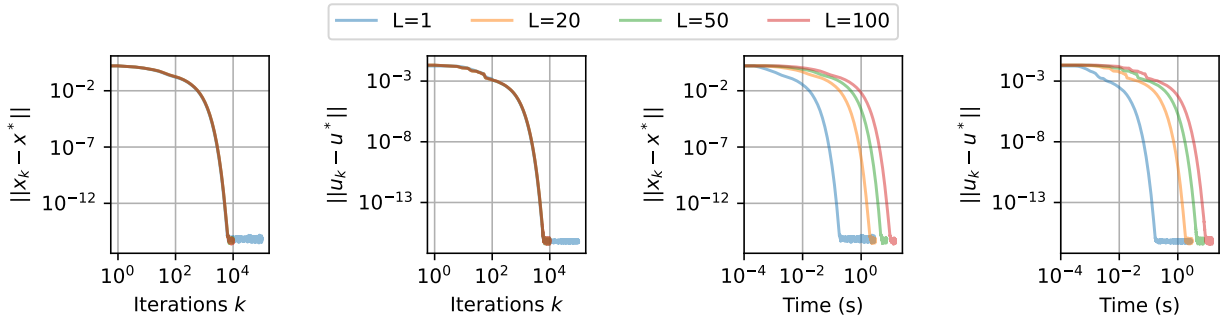

Figure 1: Convergence profile of algorithm (2.7) with $L \in \{1, 20, 50, 100\}$. Solutions (x^*, u^*) are pre-computed considering algorithm (2.7) with $L \to \infty$. The dictionary Γ is learned with $L = 1$.

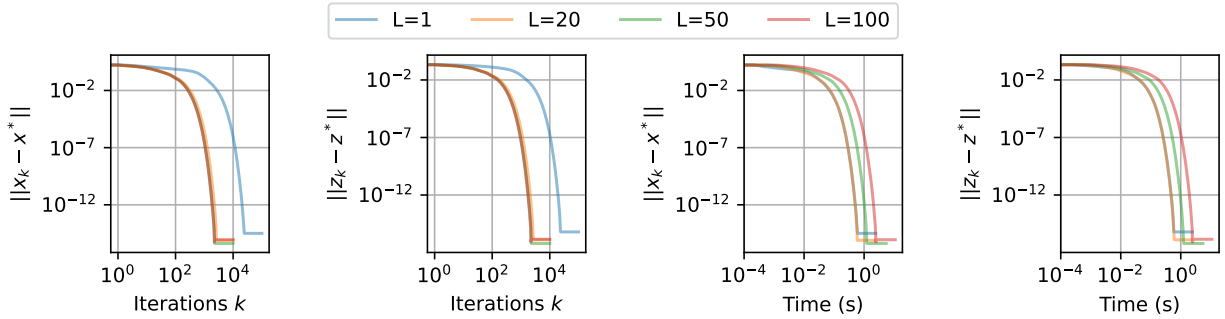

Figure 2: Convergence profile of algorithm (3.20) with $L \in \{1, 20, 50, 100\}$. Solutions (x^*, u^*) are pre-computed considering algorithm (3.20) with $L \to \infty$. The dictionary D is learned with $L = 1$.

the trajectory per iteration is fairly the same when $L \in \{20, 50, 1000\}$, while $L = 1$ is taking a bit more iterations to reach convergence. However, for the convergence speed, all strategies seems to behave similarly, with $L \in \{1, 20\}$ being very slightly faster.

5.2 Application to image deblurring within a DDL framework

Although the results presented in this work are generic and can be applied for any linear operators Γ and D, they are of particular interest in the DDL framework, where these operators are learned. We will explore this case in this section, and apply the resulting methods to an image deblurring problem. For the sake of completeness we will compare the resulting algorithms to a FB-PnP algorithm where the denoiser is a DRUnet.

5.2.1 DDL context and training

As mentioned above, in this section, we focus on a deep learning approach, where the dictionaries Γ and D (see Section 2 and Section 3 for details) are learned as Gaussian denoisers, within a supervised context. Such an approach using unrolled algorithms for dictionary learning in computational imaging has been investigated in multiple works of the literature, either for the analysis formulation [10, 33] or the synthesis formulation [53, 50], leading to the so-called DDL framework.

In this section, we describe the training process for Γ and D. The objective is to find an estimate $x^\dagger \in \mathbb{R}^N$ of an original unknown signal $\overline{x} \in \mathbb{R}^N$ that has been corrupted by some additive white Gaussian noise. Hence the noisy observed signal is given by

$$v = \overline{x} + \upsilon w, \tag{5.1}$$

where $w \in \mathbb{R}^N$ is a realization of an i.i.d. standard normal random variable, and $\upsilon > 0$ is the scale of the noise. Both analysis and synthesis formulations to find the estimate of $\overline{x}$ from v in (5.1) (see

(2.5) and (3.1), respectively) have been extensively studied in the literature (see e.g. [42, 55] for the analysis, and [40, 2, 36] for the synthesis formulation). In the context of DDL, these problems can be reformulated as a joint estimation of the image and the dictionary, or more generally as bi-level optimization problems [36]. Then, the dictionaries are defined as solution to

$$\Gamma^\dagger = \underset{\Gamma \in \mathbb{R}^{S\times N}}{\operatorname{argmin}} \ \ell^{\mathcal{A}}\big(x_\Gamma^\dagger\big) \quad \text{where} \quad x_\Gamma^\dagger \text{ solution to (2.5)} \tag{5.2}$$

and

$$D^\dagger = \underset{D \in \mathbb{R}^{N\times S}}{\operatorname{argmin}} \ \ell^{\mathcal{S}}\big(z_D^\dagger\big) \quad \text{where} \quad z_D^\dagger \text{ solution to (3.1)} \tag{5.3}$$

respectively, where $\ell^{\mathcal{A}}$ and $\ell^{\mathcal{S}}$ are some loss functions. In this context, the computation of the signal $x_\Gamma^\dagger$ or the sparse codes $z_D^\dagger$ is often referred to the *inner problem*, while the global minimization for estimating the dictionary Γ or D is called the *outer problem*.

If in (5.2) (resp. (5.3)), the loss function $\ell^{\mathcal{A}}(\Gamma)$ is the same as the one minimized in (2.5) (resp. if $\ell^{\mathcal{S}}(D)$ is the same as the one minimized in (3.1)), then the bi-level problem is equivalent to the DDL joint estimation problem. This choice is often used in an unsupervised setting. In a supervised setting, where a ground truth dataset can be used, a standard choice for $\ell^{\mathcal{A}}$ and $\ell^{\mathcal{S}}$ is the mean square error (MSE) loss, minimizing the ℓ^2 norm of the difference between the ground truth and the estimated signal $x_\Gamma^\dagger$ or $x_D^\dagger$. In the remainder, we will use this approach. Specifically, we aim to learn Γ and D such that

$$\begin{cases} x_{\mathcal{A}}^\dagger = G^{\mathcal{A}}_{L,\upsilon,\upsilon}(\Gamma^* \upsilon) \approx \overline{x}, \\ x_{\mathcal{S}}^\dagger = G^{\mathcal{S}}_{L,\upsilon,\upsilon}(D^* \upsilon) \approx \overline{x} \end{cases} \tag{5.4}$$

where $G^{\mathcal{A}}$ and $G^{\mathcal{S}}$ are defined in Model 2.2 and Model 3.5, respectively, for $L \geq 1$. Furthermore, we assume that the dictionaries Γ and D are parametrized by learnable parameters θ_Γ and θ_D, respectively (i.e., convolution kernel). Then, given a dataset of pairs of clean/noisy images $(\overline{x}_i, v_i)_{i\in\mathbb{S}_{\mathcal{T}}}$ obtained following (5.1), the unrolled denoisers in (5.4) are trained by solving, respectively

$$\underset{\theta_\Gamma}{\text{minimize}} \ \frac{1}{\#(\mathbb{S}_{\mathcal{T}})} \sum_{i\in\mathbb{S}_{\mathcal{T}}} \|G^{\mathcal{A}}_{L_{\text{train}},\upsilon_i,\upsilon_i}(\Gamma^* \upsilon_i) - \overline{x}_i\|, \tag{5.5}$$

and

$$\underset{\theta_D}{\text{minimize}} \ \frac{1}{\#(\mathbb{S}_{\mathcal{T}})} \sum_{i\in\mathbb{S}_{\mathcal{T}}} \|G^{\mathcal{S}}_{L_{\text{train}},\upsilon_i,\upsilon_i}(0_S) - \overline{x}_i\|, \tag{5.6}$$

where $0_S \in \mathbb{R}^S$ is a constant vector with 0 value on all coefficients. We trained our models on patches of size 150×150 of images from the training dataset from the BSDS500 image bank and we chose a noise level for the training of $\upsilon = 0.05$. The learnable parameters θ_Γ and θ_D correspond to 50 convolutional filters of dimension 5×5, and we fix $\sigma = \frac{1.8}{||\Gamma||^2}$ in Model 2.2 and $\zeta = \frac{1.8}{||D||^2}$ in

Table 1: Average time and standard deviation when applying different denoisers, computed over 150 images from BSDS500 cropped to 150×150, with noise level $\upsilon = 0.05$.

Denoiser	AD		SD		DRUNet
	1 iteration	20 iterations	1 iteration	20 iterations	
Averaged time ($\pm$std) ($\times 10^{-2}$ sec.)	$0.19(\pm 0.08)$	$3.94(\pm 1.46)$	$0.27(\pm 0.16)$	$3.12(\pm 0.96)$	$5.32(\pm 0.76)$

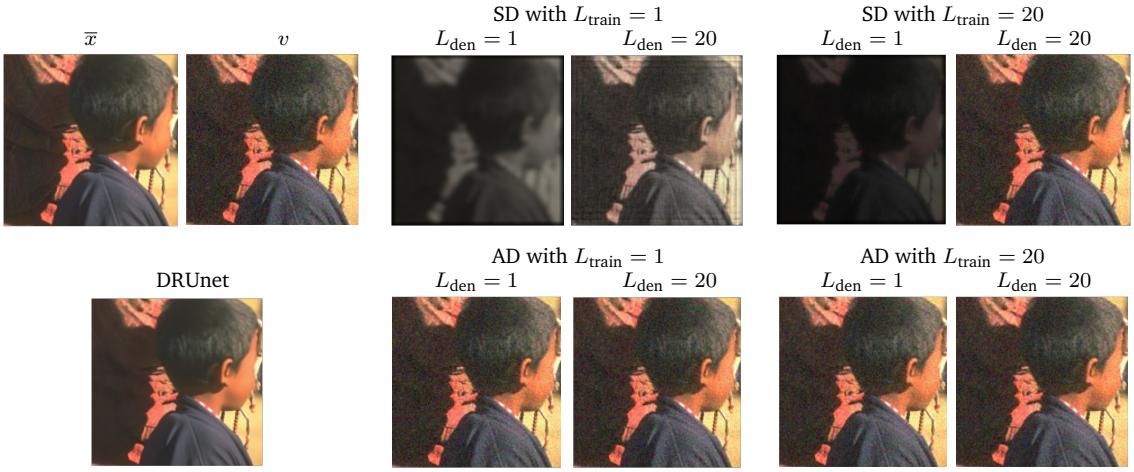

Figure 3: Denoising result example for different denoisers and configurations. Top left two images show original image $\overline{x}$ and noisy image υ. Bottom left image shows denoised image with DRUnet. Central four images show SD (top) and AD (bottom) trained with only $L_{\text{train}} = 1$ iteration, and evaluated with either $L_{\text{den}} = 1$ (left) or $L_{\text{den}} = 20$ (rigth) iterations for denoising. Right four images show SD (top) and AD (bottom) trained with $L_{\text{train}} = 20$ iteration, and evaluated with either $L_{\text{den}} = 1$ (left) or $L_{\text{den}} = 20$ (rigth) iterations for denoising.

Model 3.5. We tested two strategies: either training the denoisers with only $L_{\text{train}} = 1$ iteration or with $L_{\text{train}} = 20$ iterations that share the same dictionary.

In Table 1 we give the inference time for denoising an image, when applying the AD and SD with either $L_{\text{den}} = 1$ or $L_{\text{den}} = 20$ iterations. For comparison, we also provide the inference time when using a DRUnet, where the weights have been taken from the DeepInv PyTorch library. We further provide examples of output images for the different denoisers in Figure 3, evaluated in different configurations. It can be observed that the cases where SD is evaluated with $L_{\text{den}} = 1$ iteration (SD train $L_{\text{train}} = 1, 20$ and $L_{\text{den}} = 1$) lead to very poor results. When SD is trained with $L_{\text{train}} = 1$ and evaluated with $L_{\text{den}} = 20$ iterations, results start to improve, although remaining quite poor. Additional simulations have shown that increasing L_{den} in the evaluation continues improving the output. All other cases look very similar, with DRUnet giving best denoising results.

Remark 5.1

(i) Since unrolled algorithms have a finite number of iterations $L \in \mathbb{N}^*$, it can be interpreted

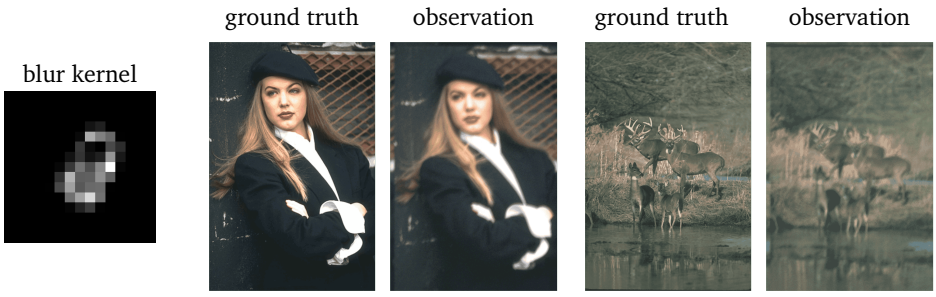

Figure 4: Blur kernel used in the experiments to model operator A in (1.1), and example of ground truth and associated observed images.

as a neural network with L layers, where dictionaries are layer dependent (i.e., L different dictionaries to be learned). This approach has been proposed by [25] for the synthesis formulation (5.3) to speed up the computation of $z_D^\dagger$, where the authors unrolled iterations of FB [20], with g being the ℓ^1 norm. The resulting method is known as LISTA. The work of [25] has been further improved in multiple contributions, considering either a supervised setting [12, 34] or an unsupervised setting [1, 39].

(ii) Unrolled-based networks adapted to the analysis formulation have also been studied [10, 13, 28, 31]. Recently a few works have also proposed to use unfolded proximal analysis denoising networks in PnP algorithms [32, 43]. These works have empirically shown that, despite having 10^2 to 10^3 less parameters than state-of-the-art denoising networks such as DnCNN [57] and DRUnet [56], unfolded denoising networks are more robust, especially when plugged in a FB algorithm for solving deblurring imaging problems. However, it is to be noted that in our approach, we are focusing on learning a dictionary, fixed for all the layers of the unrolled algorithm, leading to an even lighter architecture.

5.2.2 Simulation results

We consider an image deblurring problem of the form of (1.1), where $A \colon \mathbb{R}^N \to \mathbb{R}^M$ models a convolution with kernel given in fig. 4 and $\varepsilon = 0.05$. Both the AD and SD are trained following the DDL procedure described in section 5.2.1. The resulting AD and SD are then plugged in the FB-PnP iterations and applied to the image deblurring problem. We will further compare the proposed approaches with an FB-PnP algorithm with a DRUnet denoiser [56], considered as state-of-the-art for PnP.

In Figure 5, we compare convergence behaviors of the FB-PnP iterations $(\|x_{k+1} - x_k\|)_{k\in\mathbb{N}}$ for different denoisers, for different regularization parameters $\lambda \in \{10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Each plot shows convergence curves when running the algorithm on 5 different images. The first two rows correspond to SD trained with $L_{\text{train}} = 1$, second two rows show SD trained with $L_{\text{train}} = 20$, third two rows show AD trained with $L_{\text{train}} = 1$ and fourth two rows show AD trained with $L_{\text{train}} = 20$. In each case we run the FB-PnP with either $L_{\text{PnP}} = 1$ or $L_{\text{PnP}} = 20$ sub-iterations. The very last

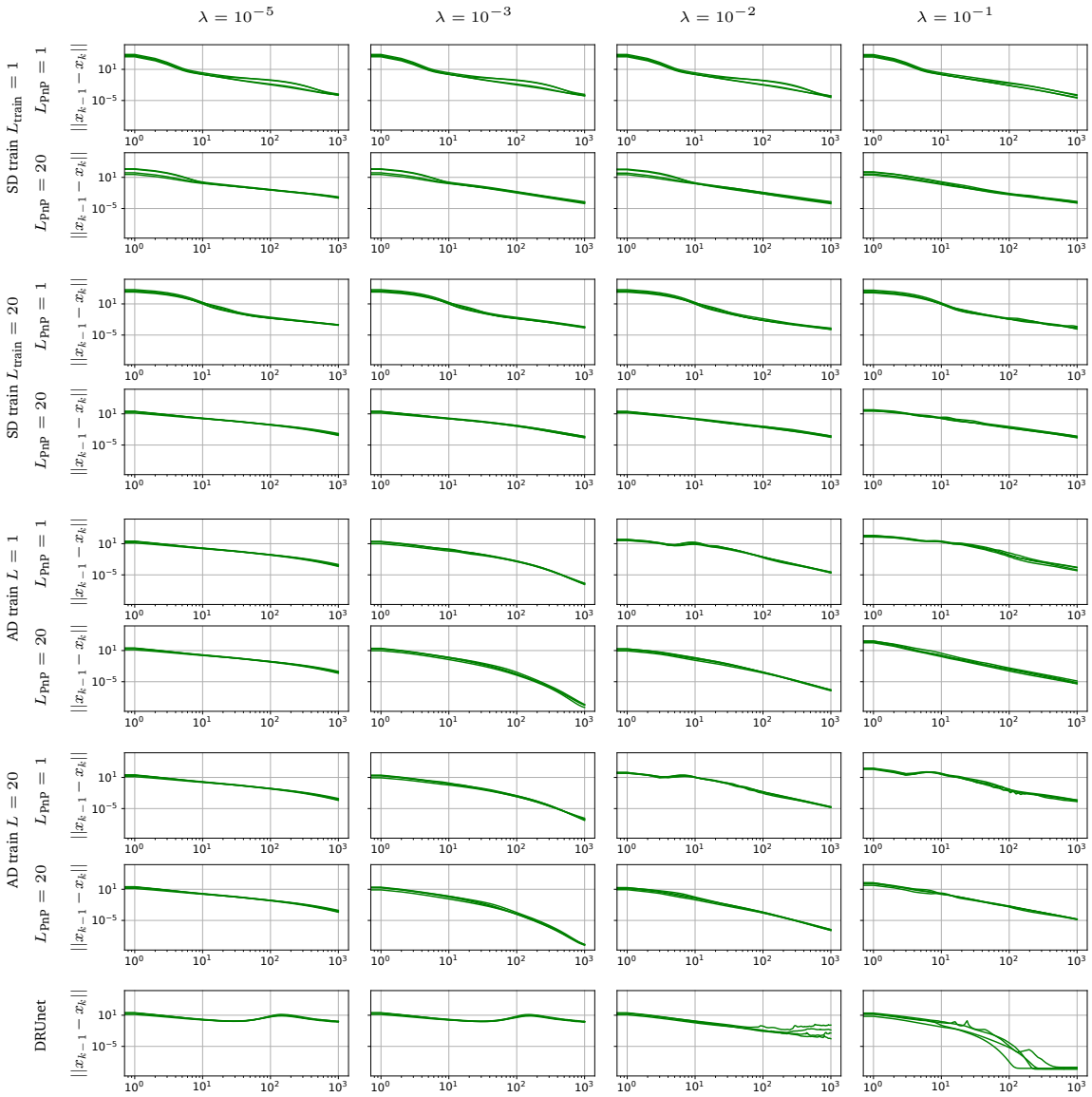

Figure 5: Convergence profiles with respect to iterations for the different FB-PnP algorithms. AD and SD are trained with either $L_{\text{train}} = 1$ or $L_{\text{train}} = 20$, and implemented in the FB-PnP with either $L_{\text{PnP}} = 1$ or $L_{\text{PnP}} = 20$.

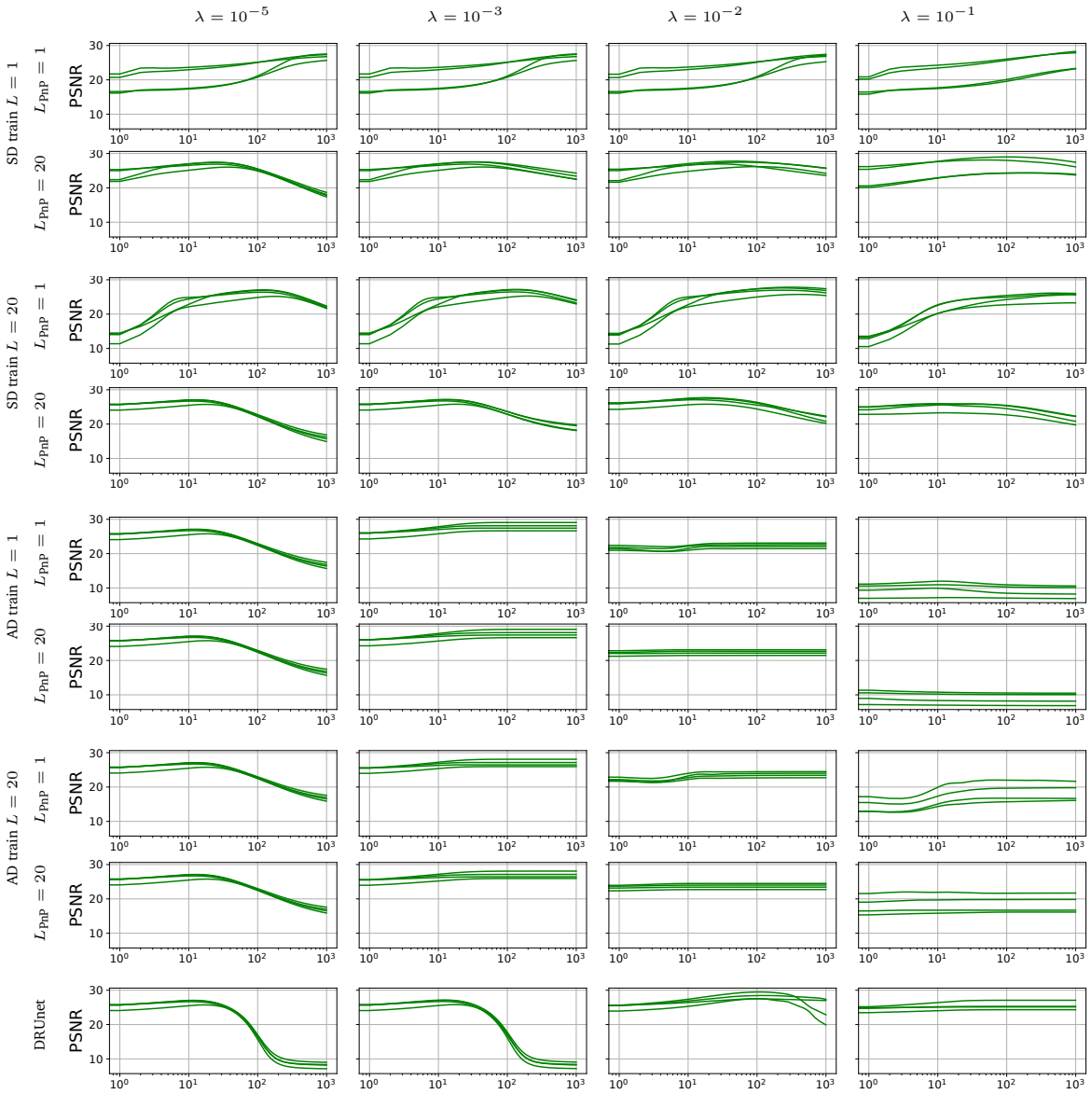

Figure 6: PSNR profiles with respect to iterations for the different FB-PnP algorithms. AD and SD are trained with either $L_{\text{train}} = 1$ or $L_{\text{train}} = 20$, and implemented in the FB-PnP with either $L_{\text{PnP}} = 1$ or $L_{\text{PnP}} = 20$

row in Figure 5 shows FB-PnP behaviour with DRUnet. FB-PnP algorithms with SD and AD exhibit monotonic convergence trends independently of the choice of the regularization parameter λ. This is not the case for DRUnet that seems to converge only for $\lambda = 10^{-1}$, while other choices are not stable[1]. We further provide curves of PSNR values with respect to iterations in Figure 6, for the above mentioned FB-PnP iterations. It can be observed that for AD the PSNR profiles are very similar when fixing λ, and looking at the different strategies $(L_{\text{train}}, L_{\text{PnP}}) \in \{1, 20\}^2$. This suggests that the AD framework is very stable, independently on the number of sub-iterations used for training or for the reconstruction process. This is not the case for SD. We can observe that the strategies $(L_{\text{train}}, L_{\text{PnP}}) = (1, 20)$ and $(L_{\text{train}}, L_{\text{PnP}}) = (20, 20)$ have similar trends, while strategies $(L_{\text{train}}, L_{\text{PnP}}) = (1, 1)$ and $(L_{\text{train}}, L_{\text{PnP}}) = (20, 1)$ are much slower. This was to be expected given the denoising performances of the last two strategies, shown in Figure 3 for $(L_{\text{train}}, L_{\text{den}}) = (1, 1)$ and $(L_{\text{train}}, L_{\text{den}}) = (20, 1)$. DRUnet exhibit very unstable behaviour for $\lambda \in \{10 - 5, 10^{-3}\}$, as expected given Figure 5. More stable results are for $\lambda = 10^{-1}$.

Additionally, we give in Figure 7 reconstruction results for two images. The regularization parameters are chosen according to Figure 5 andFigure 6. For AD we chose $\lambda = 10^{-3}$ that achieves best results for all testing configurations. For SD we chose $\lambda = 10^{-2}$ that looks a compromise between stability and results (for all configurations). It is to be noted however that for $(L_{\text{train}}, L_{\text{PnP}}) = (1, 1)$ and $(L_{\text{train}}, L_{\text{PnP}}) = (20, 1)$, SD does not achieve convergence within the 10^3 iterations, and results displayed in Figure 7 hence should be seen as early stopping results in this case. Finally, for DRUnet we take $\lambda = 10^{-1}$ that is the only value for which DRUnet is stable[2]. Overall it can be observed that AD leads to the best reconstructions, with any configuration $(L_{\text{train}}, L_{\text{den}}) \in \{1, 20\}^2$. Further, for fixed dictionary, doing $L_{\text{PnP}} = 1$ or $L_{\text{PnP}} = 20$ sub-iterations leads to the same reconstructions in our examples. This observation is not true for SD due to early stopping of the algorithms. In particular, SD produces artifacts at the edges of the images for both examples, even when high PSNR values are achieved.

Finally, for the sake of completeness, we provide in Figure 8 reconstructed images obtained with the FB-PnP when varying $\lambda \in \{10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\}$. We show images obtained with SD and AD for $(L_{\text{train}}, L_{\text{PnP}}) = (1, 20)$ (top and middle rows), and with DRUnet (bottom row). Interestingly it seems that SD is less sensitive to the choice of λ than AD and DRUnet.

6 Conclusion

In this work we study convergence behavior of the FB-PnP algorithm where the proximity operator is replaced by a specific unfolded denoiser. In particular, we study two strategies: when the denoising problem is solved with an analysis formulation or when it is solved with a synthesis for-

[1]For DRUnet λ corresponds to the noise level given as an input to the network. Additional simulations have been performed for DRUnet increasing λ, however these choices were observed to lead to lower PSNR values.

[2]Note that $\lambda = 10^{-2}$ gives better PSNR values for DRUnet, however to the price of important artifact in the reconstruction.

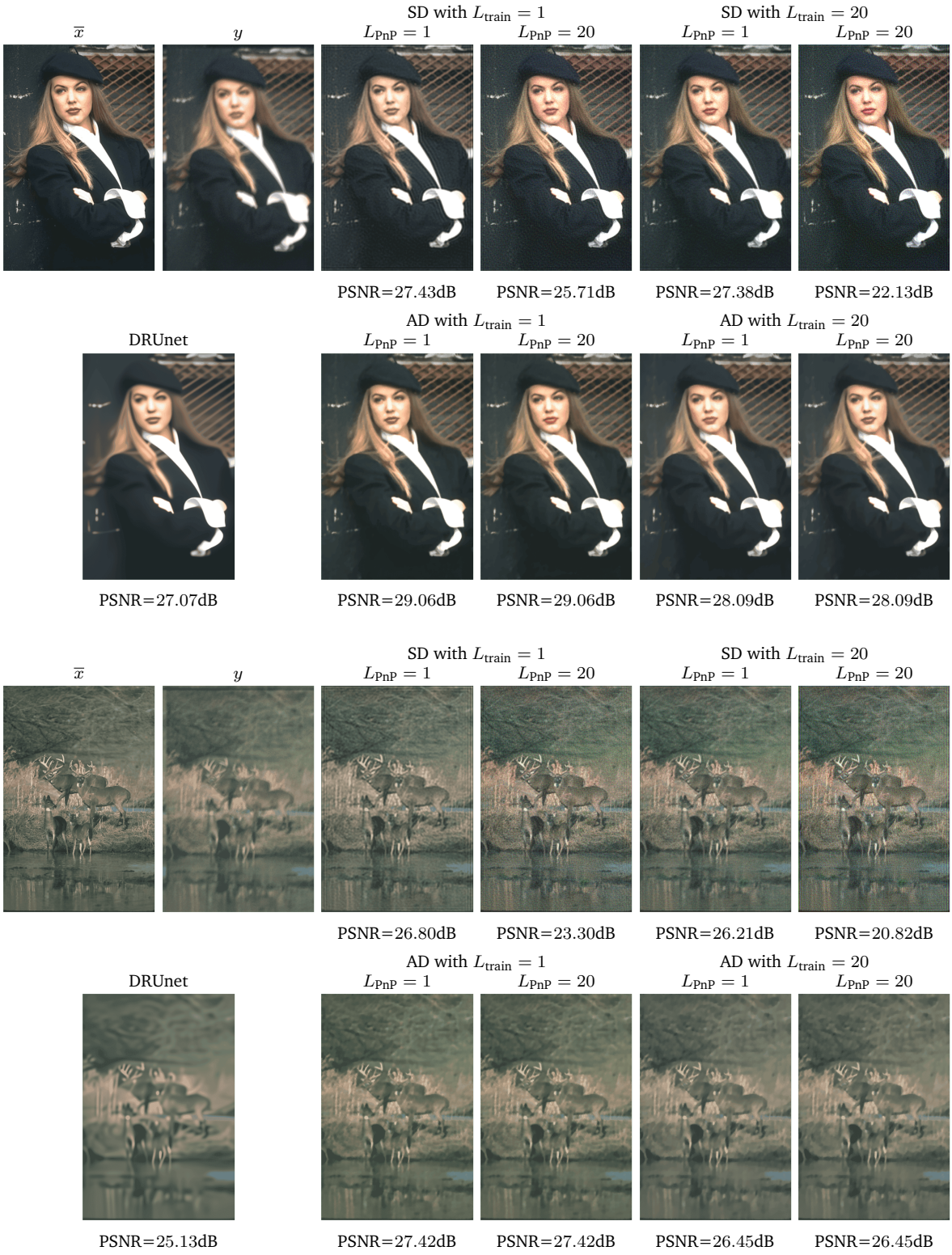

Figure 7: Examples of FB-PnP outputs obtained with the different denoisers tested, for two image examples. Regularization parameters are chosen according to convergence profiles given in Figure 5 and Figure 6: $\lambda_{\mathsf{SD}} = 10^{-2}$, $\lambda_{\mathsf{AD}} = 10^{-3}$ and $\lambda_{\mathsf{DRUnet}} = 10^{-1}$.

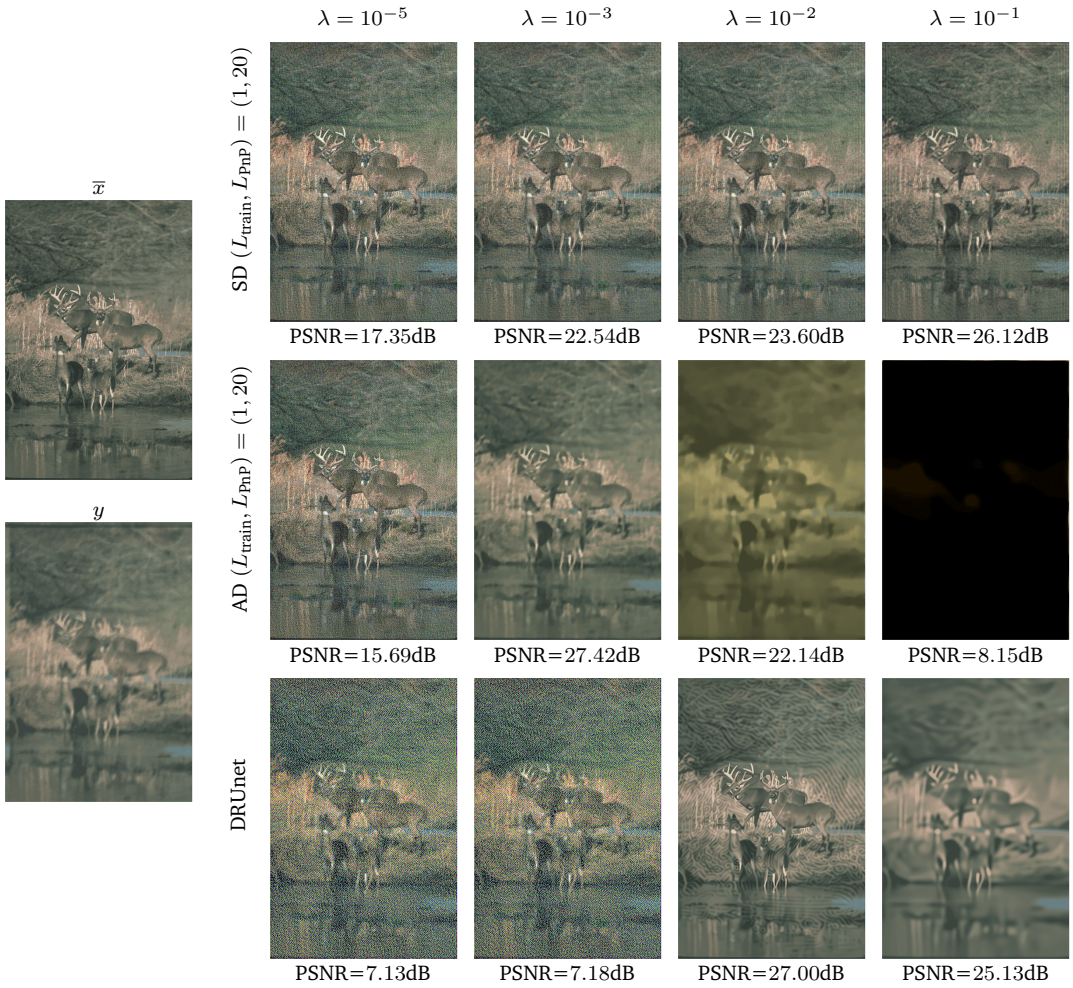

Figure 8: Example of FB-PnP outputs obtained with SD for $(L_{\text{train}}, L_{\text{PnP}}) = (1, 20)$ (top row), AD for $(L_{\text{train}}, L_{\text{PnP}}) = (1, 20)$ (middle row) and DRUnet (bottom row) when varying the regularization parameter $\lambda \in \{10^{-5}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

mulation. The two resulting analysis and the synthesis denoising minimization problems can then be solved with sub-iterations based on the dual FB algorithm and the FB algorithm, respectively. For both strategies, we show that computing only one sub-iteration within the FB-PnP combined with a warm-restart procedure on the sparse coefficients is equivalent to the FB-PnP algorithm when the full analysis or the synthesis denoising minimization problems are solved at each iteration. We further show that when smoothing the denoising problem using a Moreau envelope, this equivalence extends to an arbitrary number of sub-iterations. Finally, we investigated how these different strategies compare on a toy compressive sensing problem as well as on a deblurring imaging problem.

References

[1] Pierre Ablin, Thomas Moreau, Mathurin Massias, and Alexandre Gramfort. Learning step sizes for unfolded sparse coding. In *Advances in Neural Information Processing Systems*, pages 13100–13110, 2019.

[2] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54:4311 – 4322, 2006.

[3] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

[4] Heinz H Bauschke and Patrick L Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2017.

[5] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[6] Dimitri P Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, MA, 2nd edition, 1999.

[7] Alon Brifman, Yaniv Romano, and Michael Elad. Turning a denoiser into a super-resolver using plug and play priors. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1404–1408. IEEE, 2016.

[8] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[9] Antonin Chambolle and Ch Dossal. On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm". *Journal of Optimization theory and Applications*, 166:968–982, 2015.

[10] Antonin Chambolle and Thomas Pock. Learning consistent discretizations of the total variation. *SIAM Journal on Imaging Sciences*, 14(2):778–813, 2021.

[11] Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play ADMM for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

[12] Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. *Advances in Neural Information Processing Systems*, 2018.

[13] Hamza Cherkaoui, Jeremias Sulam, and Thomas Moreau. Learning to solve TV regularised problems with unrolled algorithms. *Advances in Neural Information Processing Systems*, 33, 2020.

[14] Patrick L Combettes, Dinh Dũng, and Bang Công Vũ. Dualization of signal recovery problems. *Set-Valued and Variational Analysis*, 18(3-4):373–404, 2010.

[15] Patrick L Combettes, Dinh Dũng, and Bang Công Vũ. Proximity for sums of composite functions. *Journal of Mathematical Analysis and applications*, 380(2):680–688, 2011.

[16] Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212, 2011.

[17] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200, 2005.

[18] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. BM3D image denoising with shape-adaptive principal component analysis. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[19] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *Advances in Neural Information Processing Systems*, 35:26698–26710, 2022.

[20] Ingrid Daubechies, Michel Defrise, and Christine Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constrains. *Communications on Pure and Applied Mathematics*, 57, 2004.

[21] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

[22] Michael Elad, Peyman Milanfar, and Ron Rubinstein. Analysis versus synthesis in signal priors. *Inverse Problems*, 23:947, 2007.

[23] Simon Foucart and Holger Rauhut. An invitation to compressive sensing. In *A mathematical introduction to compressive sensing*, pages 1–39. Springer, 2013.

[24] Alexandre Gramfort, Matthieu Kowalski, and Matti Hämäläinen. Mixed-norm estimates for the M/EEG inverse problem using accelerated gradient methods. *Physics in medicine and biology*, 57:1937–61, 2012.

[25] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. *International conference on machine learning*, pages 399–406, 2010.

[26] Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. Gradient step denoiser for convergent plug-and-play. In *International Conference on Learning Representations (ICLR'22)*, 2022.

[27] Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization. In *International Conference on Machine Learning*, pages 9483–9505. PMLR, 2022.

[28] Mingyuan Jiu and Nelly Pustelnik. A deep primal-dual proximal network for image restoration. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):190–203, 2021.

[29] Ulugbek S Kamilov, Charles A Bouman, Gregery T Buzzard, and Brendt Wohlberg. Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications. *IEEE Signal Processing Magazine*, 40(1):85–97, 2023.

[30] Nikos Komodakis and Jean-Christophe Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, 2015.

[31] H. T. V. Le, N. Pustelnik, and M. Foare. The faster proximal algorithm, the better unfolded deep learning architecture ? the study case of image denoising. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 947–951, 2022.

[32] Hoang Trieu Vy Le, Audrey Repetti, and Nelly Pustelnik. Unfolded proximal neural networks for robust image gaussian denoising. *IEEE Transactions on Image Processing*, 2024.

[33] Bruno Lecouat, Jean Ponce, and Julien Mairal. A flexible framework for designing trainable priors with adaptive smoothing and game encoding. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[34] Jialin Liu and Xiaohan Chen. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations (ICLR)*, 2019.

[35] Ignace Loris and Caroline Verhoeven. On a generalization of the iterative soft-thresholding algorithm for the case of non-separable penalty. *Inverse Problems*, 27(12):125007, 2011.

[36] Julien Mairal, Francis Bach, J. Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11, 2009.

[37] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic press, 2008.

[38] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[39] Thomas Moreau and Joan Bruna. Understanding neural sparse coding with matrix factorization. In *International Conference on Learning Representation (ICLR)*, 2017.

[40] Bruno A. Olshausen and David J Field. Sparse coding with an incomplete basis set: A strategy employed by \protect{V1}. *Vision Research*, 37(23):3311–3325, 1997.

[41] Jean-Christophe Pesquet, Audrey Repetti, Matthieu Terris, and Yves Wiaux. Learning maximally monotone operators for image recovery. *SIAM Journal on Imaging Sciences*, 14(3):1206–1237, 2021.

[42] Gabriel Peyré and Jalal M Fadili. Learning analysis sparsity priors. In *Sampta'11*, pages 4–pp, 2011.

[43] Audrey Repetti, Matthieu Terris, Yves Wiaux, and Jean-Christophe Pesquet. Dual forward-backward unfolded network for flexible plug-and-play. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 957–961. IEEE, 2022.

[44] Alejandro Ribes and Francis Schmitt. Linear inverse problems in imaging. *IEEE Signal Processing Magazine*, 25(4):84–99, 2008.

[45] JH Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5888–5897, 2017.

[46] R Tyrrell Rockafellar. *Convex analysis*, volume 11. Princeton university press, 1997.

[47] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[48] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[49] Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pages 5546–5557. PMLR, 2019.

[50] Meyer Scetbon, Michael Elad, and Peyman Milanfar. Deep K-SVD denoising. *IEEE Transactions on Image Processing*, 30:5944–5955, 2021.

[51] Jean-Luc Starck. Sparsity and inverse problems in astrophysics. *Journal of Physics: Conference Series*, 699, 2016.

[52] Matthieu Terris, Audrey Repetti, Jean-Christophe Pesquet, and Yves Wiaux. Enhanced convergent PnP algorithms for image restoration. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1684–1688. IEEE, 2021.

[53] Bahareh Tolooshams, Sourav Dey, and Demba Ba. Deep residual autoencoders for expectation maximization-inspired dictionary learning. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–15, 2020.

[54] Paul Tseng. A modified forward-backward splitting method for maximal monotone mappings. *SIAM Journal on Control and Optimization*, 38(2):431–446, 2000.

[55] Mehrdad Yaghoobi, Sangnam Nam, Rémi Gribonval, and Mike E Davies. Constrained overcomplete analysis operator learning for cosparse signal modelling. *IEEE Transactions on Signal Processing*, 61(9):2341–2355, 2013.

[56] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):6360–6376, 2021.

[57] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.