Inversion of Integral Models: a Neural Network Approach

Emilie Chouzenoux, Cecile Della Valle, Jean-Christophe Pesquet

Inversion of Integral Models: a Neural Network Approach

E. Chouzenoux, C. Della Valle, and J.-C. Pesquet *

Abstract

We introduce a neural network architecture to solve inverse problems linked to a one-dimensional integral operator. This architecture is built by unfolding a forward-backward algorithm derived from the minimization of an objective function which consists of the sum of a data-fidelity function and a Tikhonov-type regularization function. The robustness of this inversion method with respect to a perturbation of the input is theoretically analyzed. Ensuring robustness is consistent with inverse problem theory since it guarantees both the continuity of the inversion method and its insensitivity to small noise. The latter is a critical property as deep neural networks have been shown to be vulnerable to adversarial perturbations. One of the main novelties of our work is to show that the proposed network is also robust to perturbations of its bias. In our architecture, the bias accounts for the observed data in the inverse problem. We apply our method to the inversion of Abel integral operators, which define a fractional integration involved in wide range of physical processes. The neural network is numerically implemented and tested to illustrate the efficiency of the method. Lipschitz constants after training are computed to measure the robustness of the neural networks.

1 Introduction

Inverse problem In this article, we are interested in 1D operators of the form

$$T \;:\; \left| \begin{array}{ccc} \mathscr{X} & \to & \mathscr{Y} \\ x & \to & y(t) = \displaystyle\int_0^1 k(t,s)x(s)\,\mathrm{d}s\;. \end{array} \right. \tag{1}$$

Hereabove, $\mathscr{X}$ and $\mathscr{Y}$ are functional Hilbert spaces, typically $\mathscr{X} = \mathscr{Y} = L^2(0,1)$, $k \in L^2([0,1]^2)$ and T is a linear compact operator. A large variety of inverse problems consist of inverting convolution operators such as signal/image restoration [1, 2], tomography [3], Fredholm equation of the first kind [4] or inverse Laplace transform [5]. In this work, we focus on the inversion of the Abel integral, for which the kernel is of the form

$$k(t,s) = \ell(t,s)(t-s)^{a-1}\delta_{s\leq t}\;,$$

*C. Della Valle (corresponding author) is with Université de Paris, Paris Sorbonne Université.
E. Chouzenoux and J.-C. Pesquet are with Université Paris-Saclay, Inria, CentraleSupélec, Center for Visual Computing. The work by J.-C. Pesquet was supported by Institut Universitaire de France and the ANR Chair in Artificial Intelligence BRIDGEABLE.

where a is a real positive number, ℓ is a continuous function, differentiable and decreasing on its second variable, and $\delta_{s \leq t}$ is equal to one if $s \leq t$ and zero otherwise. The inverse problem investigated in this article is the following: given $y \in \mathscr{Y}$, we seek for $x \in \mathscr{X}$ such that

$$y = Tx \ .$$

In addition, we consider the case when the data y are corrupted with measurement errors or noise. We model an additive noise as follows: we call the upper bound for the noise level $\delta > 0$ measured in $\mathscr{Y}^\delta$ with $\mathscr{Y} \subset \mathscr{Y}^\delta$. Here, $\mathscr{Y}^\delta$ is the Hilbert space $H^{-s}(0,1)$ defined in [6]. Typically, $s = 0$ for a deterministic noise and $s = 1/2$ for a deterministic equivalent of a Gaussian white noise [7]. Solving the inverse problem in the presence of noisy data then amounts to finding $x^\delta \in \mathscr{Y}$ such that

$$y^\delta = Tx^\delta \ , \quad \text{with} \quad \|y - y^\delta\|_{\mathscr{Y}^\delta} \leq \delta \ . \tag{2}$$

The above problem is often ill-posed i.e., a solution might not exist, might not be unique, or might not depend continuously on the data.

Variational problem The well-posedness of the inverse problem defined by (2) is retrieved by regularization. Here we consider Tikhonov type regularization. Let $\tau \in]0, +\infty[$ be the regularization parameter. Solving the inverse problem (2) with such regularization, leads to the resolution of the following optimization problem

$$\underset{x \in C}{\text{minimize}} \, J_\tau(x) \ , \tag{3}$$

where

$$(\forall x \in \mathscr{X}) \quad J_\tau(x) = \frac{1}{2}\|Tx - y^\delta\|^2 + \frac{\tau}{2}\|D^r x\|^2, \tag{4}$$

C is a nonempty closed convex subset of $\mathscr{X}$, and D^r acts as a derivative operator with order $r \geq 0$. Often, we have an a priori of smoothness on the solution, in $H^q(0,1)$ with $q \geq 0$, which justifies the use of such a derivative-based regularization. Problem (3) is an instance of the more general problem stated above, encountered in many signal/image processing tasks:

$$\underset{x \in \mathscr{X}}{\text{minimize}} \, J_\tau(x) + \mu \, g(x) \ , \tag{5}$$

where $\mu \in [0, +\infty[$ is an additional regularization constant and g is a proper lower-semicontinuous convex function from some Hilbert space $\mathscr{X}$ to $]-\infty, +\infty]$. Indeed, Problem (3) corresponds to the case when g is the indicator function ι_C of set C.

Neural network We focus our attention on seeking for a solution to the addressed inverse problem through nonlinear approximation techniques making use of neural networks. Thus, instead of considering the solution to the regularized problem (5), we define the solution to the inverse problem (2) as the output of a neural network, whose structure is similar to a recurrent network [8].

Namely, by setting an initial value x_0, we are interested in the following m-layers neural network where $m \in \mathbb{N} \setminus \{0\}$:

$$\begin{cases} \textbf{Initialization:} \\ \quad b_0 = T^* y^\delta, \\ \textbf{Layer } n \in \{1, \dots, m\}\textbf{:} \\ \quad x_n = R_n(W_n x_{n-1} + V_n b_0) \ , \end{cases} \tag{6}$$

where, for every $n \in \{1, \dots, m\}$,

$$R_n = \text{prox}_{\lambda_n \mu_n g} \tag{7}$$

$$W_n = \mathbb{1} - \lambda_n T^* T - \lambda_n \tau_n D^* D \tag{8}$$

$$V_n = \lambda_n \mathbb{1}. \tag{9}$$

Hereabove, prox_φ states for the proximity operator of a lower-semicontinuous proper convex function φ [9, Chapter 9], $\mathbb{1}$ denotes the identity operator, and for every $n \in \{1, \dots, m\}$, λ_n, μ_n, and τ_n are positive constants, which are learned during training. Throughout this paper, L^* denotes the adjoint of a bounded linear operator L defined on Hilbert spaces.

Model (6) can be viewed as unrolling m iterations of an optimization algorithm, so leading to Algorithm 1. Note that, when $\mu_n \equiv \mu$ and $\tau_n \equiv \tau$, we recognize a forward-backward algorithm [10, 11] applied to the variational problem (5).

Algorithm 1 Proximal forward-backward splitting method

1: Set x_0 ,
2: **for** $n = 1, 2, \dots, m$ **do**
3: Set λ_n, τ_n, μ_n ,
4: $x_n = \text{prox}_{\lambda_n \mu_n g} \ (x_{n-1} - \lambda_n \nabla J_{\tau_n}(x_{n-1}))$,
5: **end for**
6: **return** x_m

From a theoretical standpoint, there is no guarantee that such a model constitutes a regularizing family, and there is no equivalence between the regularized inverse problem (5) and the output of Model (6), since the number of iterations m is fixed in advance. However, we can quantify the robustness of Model (6) to perturbations on its initialization x_0 and on its bias b_0, by an accurate estimation of its Lipschitz constant.

Related works and contributions There has been a plethora of techniques developed to invert integrals of the form (1). Among these methods, Tikhonov-type methods are attractive from a theoretical viewpoint, especially because they provide good convergence rate as the noise level decreases, as shown in [12] or [13]. However, limitations of such methods may be encountered in their implementation. Indeed, certain parameters such as gradient descent steps or the regularization coefficient need to be set, as discussed in [14] or [15] for the Abel integral operator. The latter parameter depends on the noise level, as shown in [16], which is not always easy to

estimate. In practical cases, methods such as the L-curve method, see [17], can be implemented to set the regularization parameter, but they require a large number of resolutions and therefore a significant computational cost. Moreover, incorporating constraints on the solution may be difficult in such approaches, and often reduces to projecting the resulting solution onto the desired set. These points justify the use of a neural network structure to avoid laborious calibration of the parameters and to easily incorporate constraints on the solution.

The use of neural networks for solving inverse problems has become increasingly popular, especially in the image processing community. A rich panel of approaches have been proposed, either adapted to the sparsity of the data [18, 19], or mimicking variational models [20, 21], or iterating learned operators [22, 23, 24, 25, 26], or adpating Tikhonov method [27]. The successful numerical results of the aforementioned works raise two theoretical questions: when these methods are based on the iteration of a neural network, do they converge (in the sense of the algorithm)? Are these inversion methods stable or robust?

In iterative approaches, a regularization operator is learned, either in the form of a proximity (or denoiser) operator as [23, 22, 26], of a regularization term [27], of a pseudodiffential operator [28], or of its gradient [29, 3]. Strong connections also exist with Plug and Play methods [30, 31, 24], where the regularization operator is a pre-trained neural network. Such objects have in particular enable high-quality imaging restoration or tomography inversion [3]. Here, the non-expensiveness of the neural network is a core property to establish convergence of the algorithm [3, 24]. But our proposed neural network is not based on this idea.

Other recent works solve linear inverse problems by unrolling the optimization iterative process in the form of a network architecture as in [32, 33]. Here the number of iterations is fixed, instead of iterating until convergence, and the network is trained in an end-to-end fashion. Since neural network frameworks offer powerful differential programming capabilities, such architecture are also used for learning hyper-parameters in an unrolled optimization algorithm as in [34, 35].

All of the above strategies have shown very good numerical results. However, few studies have been conducted on their theoretical properties, especially their stability. The study of the robustness of such a structure is often based on a series of numerical tests, as performed in [36]. In [27], they provide very large assumption under which the convergence and the regularization property of their network is ensured. But their result is not subject to verification during the numerical implementation. A fine characterization of the convergence conditions of recurrent neural network and of their stability via the estimation of a Lipschitz constant is done in [37, 38]. In particular, the Lipschitz constant estimated in [38] is more accurate than in basic approaches which often rely in computing the product of the norms of the linear weight operators of each layer as in [39, 40]. Thanks to the aforementioned works, proofs of convergence and stability have been demonstrated on specific neural networks applied to inverse problems as in [24, 25, 34]. The analysis carried out in this article is in the line of these references.

Our contributions in this paper are the following.

i) We propose an algorithm based on a neural network architecture to solve the inverse problem (2), where a constraint is imposed on the sought solution. One of the main advantages is that the structure of the neural network is interpretable and that it contains few parameters which are learnt.

ii) We study theoretically and numerically the stability of the so-built neural network. The sensitivity analysis is performed with respect to the observed data y^δ which corresponds to a bias term in each of the layers of (6). This analysis is more general than the one performed in [34], in which only the impact of the initialization was considered.

iii) We show how to implement the neural network in the case of Abel operators. Such operators arise in various physical applications. The proposed neural network performs numerically well compared to other classical inversion methods. Neural network techniques have been widely applied to imaging inverse problems, but few are tested on experimental 1D signal inverse problems.

Outline The outline of the paper is as follows. In Section 2, we recall the theoretical background of our work. We specify our notation, which is based on [9]. In Section 3, we establish the stability of the neural network defined by (6), based on the results in [37] and [38]. By stability, we mean that the output of the neural network is controlled not only with respect to its initial input x_0, but also also with respect to the bias term T^*y^δ. The objective is to guarantee that a small difference or error on these vectors is not amplified through the network. Our theoretical study concerns a class of dynamical systems including a leakage factor, which is more general than the neural network defined by (6). In Section 4, the numerical resolution of the problem (2) is described. We define the Abel operator, its characteristics as well as its discretization. Then, we detail the construction of the training data set. The architecture of the neural network is explained, as well as the sub-network used to estimate the parameters. Finally, we compute the Lipschitz constants of our trained networks. We also compare the obtained results with those delivered by two other methods classically used to solve inverse problems involving an Abel integral.

2 Notation

We introduce the theory of convex analysis we will be dealing with, namely monotone operator in Hilbert spaces. We also cover the bits of operator theory that will be needed throughout.

Let us consider the Hilbert space $\mathscr{X}$ endowed with the norm $\|\cdot\|$ and the scalar product $\langle\cdot,\cdot\rangle$. In the following, $\mathscr{X}$ shall always refer to spaces of functions defined on the interval $]0,1[$. The notation $\|\cdot\|$ will also refer to the operator norm of bounded operators from $\mathscr{X}$ onto $\mathscr{X}$. The identity operator over $\mathscr{X}$ will be referred to as $\mathbb{1}$.

An operator $S\colon \mathscr{X} \to \mathscr{X}$ is nonexpansive if it is $1-$Lipschitz, that is

$$(\forall (x,y) \in \mathscr{X}\times\mathscr{X}) \qquad \|Sx-Sy\| \leq \|x-y\| \ .$$

Moreover, S is said to be

i) firmly nonexpansive if

$$(\forall (x,y) \in \mathscr{X}\times\mathscr{X}) \qquad \|Sx-Sy\|^2 + \|(\mathbb{1}-S)x-(\mathbb{1}-S)y\|^2 \leq \|x-y\|^2 \ ;$$

ii) a Banach contraction if there exists $\kappa \in]0,1[$ such that

$$(\forall (x,y) \in \mathscr{X} \times \mathscr{X}) \qquad \|Sx - Sy\| \leq \kappa \|x - y\| \ . \tag{10}$$

If S is a Banach contraction, then the iterates $(S^n x)_{n\in\mathbb{N}}$ converge linearly to a fixed point of S according to Picard's theorem. On the other hand, when S is nonexpansive, the convergence is no longer guaranteed. A way of recovering the convergence of the iterates is to assume that S is averaged, i.e., there exists $\alpha \in]0,1[$ and a nonexpansive operator $R : \mathscr{X} \to \mathscr{X}$ such that $S = (1-\alpha)\mathbb{1} + \alpha R$. In particular, S is $\alpha-$averaged if and only if

$$(\forall (x,y) \in \mathscr{X} \times \mathscr{X}) \qquad \|Sx - Sy\|^2 + \frac{1-\alpha}{\alpha}\|(\mathbb{1} - S)x - (\mathbb{1} - S)y\|^2 \leq \|x - y\|^2 \ .$$

If S has a fixed point and it is averaged, then the iterates $(S^n x)_{n\in\mathbb{N}}$ converge weakly to a fixed point. Note that S is firmly nonexpansive if and only if it is $1/2-$averaged and that, if S satisfies (10) with $\kappa \in]0,1[$, then it is $(\kappa+1)/2$-averaged.

Let $\Gamma_0(\mathscr{X})$ be the set of proper lower semicontinuous convex function from $\mathscr{X}$ to $]-\infty, +\infty]$. Then we define the proximal operator as

Definition 2.1

Let $f \in \Gamma_0(\mathscr{X})$, $x \in \mathscr{X}$, and $\gamma > 0$. Then $\mathrm{prox}_{\gamma f}(x)$ is the unique point that satisfies

$$\mathrm{prox}_{\gamma f}(x) \ = \ \underset{y\in\mathscr{X}}{\mathrm{argmin}} \left(f(y) + \frac{1}{2\gamma}\|x - y\|^2 \right) \ .$$

The function $\mathrm{prox}_{\gamma f} : \mathscr{X} \to \mathscr{X}$ is the proximity operator of γf.

Finally, the proximity operator has the following property.

Proposition 2.2 (*Proposition 12.28 of [9]*)

The operators $\mathrm{prox}_{\gamma f}$ and $\mathbb{1} - \mathrm{prox}_{\gamma f}$ are firmly nonexpansive.

In the proposed neural network (6), the activation operator is a proximity operator. In practice, this is the case for most activation operators, as shown in [37]. The neural network (6) is thus a cascade of firmly nonexpansive operators and linear operators. If the linear part is also nonexpansive, bounds on the effect of a pertubation of the neural network or its iterates can be established.

3 Stability and α-averagedness

In this section, we study the stability of the proposed neural network (6). This analysis is performed by estimating the Lipschitz constant of the network, and by determining under which conditions this network is α-averaged. To do so, we introduce a virtual network, which takes as inputs the classical ones on top of a new one, which is the bias parameter.

3.1 Virtual neural network with leakage factor

To facilitate our theoretical analysis, we will introduce a virtual network making use of new variables $(z_n)_{n\in\mathbb{N}}$. For every $n \in \mathbb{N} \setminus \{0\}$, we define the n-th layer of our virtual network as follows

$$z_n = \begin{pmatrix} x_n \\ b_n \end{pmatrix}, \quad z_n = Q_n(U_n z_{n-1}) \ , \quad \text{with} \quad \begin{cases} Q_n = \begin{pmatrix} R_n \\ \mathbb{1} \end{pmatrix} \ , \\ \\ U_n = \begin{pmatrix} W_n & \lambda_n \mathbb{1} \\ 0 & \eta_n \mathbb{1} \end{pmatrix} \ . \end{cases} \tag{11}$$

Note that, in order to gain more flexibility, we have included positive multiplicative factors $(\eta_n)_{n\geq 1}$ on the bias. Cascading m such layers yields

$$\begin{cases} \textbf{Initialization:} \\ \quad b_0 = T^* y^\delta, \\ \textbf{Layer } n \in \{1, \dots, m\}\textbf{:} \\ \quad x_n = R_n(W_n x_{n-1} + V_n b_0) \ , \end{cases} \tag{12}$$

where

$$R_n = \text{prox}_{\lambda_n \mu_n g} \tag{13}$$

$$W_n = \mathbb{1} - \lambda_n T^* T - \lambda_n \tau_n D^* D \tag{14}$$

$$V_n = \lambda_n \eta_{n-1} \cdots \eta_1 \mathbb{1} \tag{15}$$

and $\eta_0 = 1$. We thus see that the network defined by Model (6) is equivalent to the virtual one when all the factors η_n are equal to one. When $n \geq 1$ and $\eta_n < 1$. The parameters $(\eta_n)_{n\geq 1}$ can be interpreted as a leakage factor.

Remark 3.1. In the original forward-backward algorithm, the introduction of $(\eta_n)_{n\geq 1}$ amounts to introducing an error e_n in the gradient step, at iteration n, which is equal to

$$e_n = \lambda_n(\eta_{n-1} \cdots \eta_1 - 1) b_0. \tag{16}$$

From known properties concerning the forward-backward algorithm [11], the convergence of the algorithm is still guaranteed provided that

$$\sum_{n=2}^{+\infty} \lambda_n |\eta_{n-1} \cdots \eta_1 - 1| < +\infty \ . \tag{17}$$

In our analysis, it will be useful to define the triangular linear operator

$$U = \ U_m \circ \cdots \circ U_1 = \begin{pmatrix} W_{1,m} & \widetilde{W}_{1,m} \\ 0 & \eta_{1,m} \mathbb{1} \end{pmatrix} \ , \tag{18}$$

where, for every $n \in \{1, \dots, m\}$ and $i \in \{1, \dots, n\}$

$$\widetilde{W}_{i,n} = \sum_{j=i}^{n} \lambda_j \eta_{i,j-1} W_{j+1,n} \tag{19}$$

and, for every $i \in \{1, \dots, m+1\}$ and $j \in \{0, \dots, m\}$,

$$W_{i,j} = \begin{cases} W_j \circ \cdots \circ W_i & \text{if } j \geq i \\ \mathbb{1} & \text{otherwise,} \end{cases} \tag{20}$$

$$\eta_{i,j} = \begin{cases} \eta_j \cdots \eta_i & \text{if } j \geq i \\ 1 & \text{otherwise.} \end{cases} \tag{21}$$

Since T defined by (1) is a compact operator, we can define its singular value expansion as in [16]. Furthermore, we place ourselves in the case where D^*D and T^*T commutes, for operators T defined by (1) and regularization operators D. Therefore those operators admit the same eigensystem. In particular, they can be diagonalized in the same orthonormal set of eigenvectors $(v_p)_p$. We define their respective eigenvalues $(\beta_{T,p})_p$ and $(\beta_{D,p})_p$, as well as the following quantities, for every eigenspaces $p \in \mathbb{N}$, $n \in \{1, \dots, m\}$, and $i \in \{1, \dots, n\}$,

$$\beta_p^{(n)} = 1 - \lambda_n(\beta_{T,p} + \tau_n \beta_{D,p}) \,, \tag{22}$$

$$\beta_{i,n,p} = \prod_{j=i}^{n} \beta_p^{(j)} \,, \tag{23}$$

$$\widetilde{\beta}_{i,n,p} = \sum_{j=i}^{n-1} \beta_p^{(n)} \cdots \beta_p^{(j+1)} \lambda_j \eta_{i,j-1} + \lambda_n \eta_{n-1} \cdots \eta_i \tag{24}$$

with the convention $\sum_{i=n}^{n-1} \cdot = 0$. Note that $(\beta_{i,n,p}, v_p)_{p \in \mathbb{N}}$ and $(\widetilde{\beta}_{i,n,p}, v_p)_{p \in \mathbb{N}}$ are the eigensystems of $W_{i,n}$ and $\widetilde{W}_{i,n}$, respectively.

3.2 Stability results for the virtual network

We first recall some recent results on the stability of neural networks [37, Proposition 3.6(iii)] [38, Theorem 4.2].

Proposition 3.2

Let $m > 1$ be an integer, let $(\mathcal{H}_i)$ be nonzero real Hilbert spaces. For every $n \in \{1, \dots, m\}$, let $U_n \in \mathcal{B}(\mathcal{H}_{n-1}, \mathcal{H}_n)$ and let $Q_n \colon \mathcal{H}_n \to \mathcal{H}_n$ be a firmly nonexpansive operator. Set

$U = U_m \circ \cdots \circ U_1$ and

$$\theta_m = \|U\| + \sum_{k=1}^{m-1} \sum_{1 \leq j_1 < \ldots < j_k \leq m-1} \|U_m \circ \cdots \circ U_{j_k+1}\| \, \|U_{j_k} \circ \cdots \circ U_{j_{k-1}+1}\| \cdots \|U_{j_1} \circ \cdots \circ U_1\|. \tag{25}$$

Let $S = Q_m \circ U_m \circ \cdots \circ Q_1 \circ U_1$. Then the following hold

i) *$\theta_m / 2^{m-1}$ is a Lipschitz constant of S.*

ii) *Let $\alpha \in [1/2, 1]$. If $\mathcal{H}_m = \mathcal{H}_0$ and*

$$\|U - 2^m(1-\alpha)\mathbb{1}\| - \|U\| + 2\theta_m \leq 2^m \alpha \ , \tag{26}$$

then S is α-averaged.

In light of these results, we will now analyze the properties of the virtual network (11) based on the singular values of the operators T and D, and the parameters $(\lambda_n)_{1 \leq n \leq m}$ and $(\tau_n)_{1 \leq n \leq m}$. One of the main difficulties with respect to the case already studied by [34] is that here the involved operators $(U_n)_{1 \leq n \leq m}$ are no longer self-adjoint.

A preliminary result will be needed:

Lemma 3.3

Let $m \in \mathbb{N} \setminus \{0\}$ the total number of layers. For every layer $n \in \{1, \ldots, m\}$ and layer $i \in \{1, \ldots, n\}$, the norm of $U_n \circ \cdots \circ U_i$ is equal to $\sqrt{a_{i,n}}$ with

$$a_{i,n} = \frac{1}{2} \sup_{p \in \mathbb{N}} \left(\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2 + \sqrt{(\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2)^2 - 4\beta_{i,n,p}^2 \eta_{i,n}^2} \right) \ , \tag{27}$$

where p covers the eigenspaces of T^*T defined by (1).

Proof. Thanks to expressions (11), (19), (20), and (21), we can calculate the norm of $\|U_n \circ \cdots \circ U_i\|$. For every $z = (x, b)$, $U_n \circ \cdots \circ U_i z = (W_{i,n}x + \widetilde{W}_{i,n}b, \eta_{i,n}b)$ and

$$\begin{aligned} \|U_n \circ \cdots \circ U_i z\|^2 &= \|W_{i,n}x + \widetilde{W}_{i,n}b\|^2 + \eta_{i,n}^2 \|b\|^2 \\ &= \|W_{i,n}x\|^2 + 2\langle W_{i,n}x, \widetilde{W}_{i,n}b \rangle + \|\widetilde{W}_{i,n}b\|^2 + \eta_{i,n}^2 \|b\|^2 \ . \end{aligned}$$

Let $(\beta_{i,n,p}, v_p)_{p \in \mathbb{N}}$ defined by (23) and $(\widetilde{\beta}_{i,n,p}, v_p)_{p \in \mathbb{N}}$ defined by (24) be the respective eigensystems of $W_{i,n}$ and $\widetilde{W}_{i,n}$. Let us decompose (x, b) in the basis of eigenvectors $(v_p)_p$ of T^*T, as

$$\begin{cases} x = \sum_p \xi_p \, v_p \ , \\ b = \sum_p \zeta_p \, v_p \ . \end{cases}$$

We have then

$$\|U_n \circ \cdots \circ U_i z\|^2 = \sum_p \beta_{i,n,p}^2 \xi_p^2 + 2 \sum_p \beta_{i,n,p} \widetilde{\beta}_{i,n,p} \xi_p \zeta_p + \sum_p (\widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2) \zeta_p^2 \ .$$

By definition of the operator norm,

$$\|U_n \circ \cdots \circ U_i\|^2 = \sup_{\|x\|^2 + \|b\|^2 = 1} \left(\sum_p \beta_{i,n,p}^2 \, \xi_p^2 + (\eta_{i,n}^2 + \widetilde{\beta}_{i,n,p}^2) \, \zeta_p^2 + 2 \beta_{i,n,p} \widetilde{\beta}_{i,n,p} \, \xi_p \, \zeta_p \right) \ .$$

Note that, for every integer $p \in \mathbb{N}$ and $\omega_p = (\xi_p, \zeta_p) \in \mathbb{R}^2$,

$$\beta_{i,n,p}^2 \, \xi_p^2 + (\eta_{i,n}^2 + \widetilde{\beta}_{i,n,p}^2) \, \zeta_p^2 + 2 \beta_{i,n,p} \widetilde{\beta}_{i,n,p} \, \xi_p \, \zeta_p = \langle A_{i,n,p} \omega_p, \omega_p \rangle \tag{28}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product and

$$A_{i,n,p} = \begin{pmatrix} \beta_{i,n,p}^2 & \beta_{i,n,p} \widetilde{\beta}_{m,p} \\ \widetilde{\beta}_{i,n,p} \beta_{i,n,p} & \eta_{i,n}^2 + \widetilde{\beta}_{i,n,p}^2 \end{pmatrix} \ .$$

Hence,

$$\|U_n \circ \cdots \circ U_i\|^2 = \sup_{z = (\omega_p)_p, \|z\| = 1} \sum_p \langle A_{i,n,p} \, \omega_p, \, \omega_p \rangle \ .$$

Since $A_{i,n,p}$ is a symmetric positive semidefinite matrix,

$$\|U_n \circ \cdots \circ U_i\|^2 = \sup_{p, \|\omega_p\| = 1} \langle A_{i,n,p} \omega_p, \omega_p \rangle = \sup_p \, \nu_{i,n,p} \ , \tag{29}$$

where, for every $p \in \mathbb{N}$, $\nu_{i,n,p}$ is the maximum eigenvalue of $A_{i,n,p}$. The two eigenvalues of this matrix are the roots of the characteristic polynomial

$$\begin{aligned} (\forall \nu \in \mathbb{R}) \quad \det(A_{i,n,p} - \nu \mathbb{1}_2) &= (\beta_{i,n,p}^2 - \nu)(\widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2 - \nu) - \beta_{i,n,p}^2 \widetilde{\beta}_{i,n,p}^2 \\ &= \nu^2 - (\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2) \nu + \beta_{i,n,p}^2 \eta_{i,n}^2 \ . \end{aligned}$$

The discriminant of this second-order polynomial reads

$$\begin{aligned} \Delta_{i,n,p} &= (\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2)^2 - 4 \beta_{i,n,p}^2 \eta_{i,n}^2 \\ &= (\beta_{i,n,p}^2 - \widetilde{\beta}_{i,n,p}^2 - \eta_{i,n}^2)^2 + 4 \beta_{i,n,p}^2 \widetilde{\beta}_{i,n,p}^2 \ \geq 0 \ . \end{aligned}$$

Therefore, for every $p \in \mathbb{N}$,

$$\nu_{i,n,p} = \frac{1}{2} \left(\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2 + \sqrt{(\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 + \eta_{i,n}^2)^2 - 4 \beta_{i,n,p}^2 \eta_{i,n}^2} \right) \ . \tag{30}$$

By going back to (29), we obtain

$$\|U_n \circ \cdots \circ U_i\|^2 = a_{i,n} \ .$$

□

We will now quantify the Lipschitz regularity of the network.

Proposition 3.4

Let $m \in \mathbb{N} \setminus \{0\}$. For every $n \in \{1, \dots, m\}$ and $i \in \{1, \dots, n\}$, let $a_{i,n}$ be given by (27). *Set $\theta_0 = 1$ and define $(\theta_n)_{1 \leq n \leq m}$ recursively by*

$$(\forall n \in \{1, \dots, m\}) \quad \theta_n = \sum_{i=1}^{n} \theta_{i-1} \sqrt{a_{i,n}} \; .$$

Then $\theta_m / 2^{m-1}$ is a Lipschitz constant of the virtual network (11).

Proof. According to Proposition 3.2i), if θ_m is given by (25), then $\theta_m / 2^{m-1}$ is a Lipschitz constant of the virtual network (11). On the other hand, it follows from [37, Lemma 3.3] that θ_m can be calculated recursively as

$$(\forall n \in \{1, \dots, m\}) \quad \theta_n = \sum_{i=1}^{n} \theta_{i-1} \|U_n \circ \cdots \circ U_i\| \; ,$$

with $\theta_0 = 1$. Finally, Lemma (3.3) allows us to substitute $(\sqrt{a_{i,n}})_{1 \leq i \leq n}$ for $(\|U_n \circ \cdots \circ U_i\|)_{1 \leq i \leq n}$ in the above expression. □

We will next provide conditions ensuring that the virtual network is an averaged operator.

Proposition 3.5

Let $m \in \mathbb{N} \setminus \{0\}$. Let $a_{1,m}$ be defined in Lemma 3.3 and θ_m be defined in Proposition 3.4. Let $\alpha \in [1/2, 1]$. Define

$$b_\alpha = \frac{1}{2} \sup_p \Big((\beta_{1,m,p} - \gamma_\alpha)^2 + (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}_{1,m,p}^2 + \sqrt{\begin{array}{c} \left((\beta_{1,m,p} - \gamma_\alpha)^2 + (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}_{1,m,p}^2\right)^2 \\ -4(\beta_{1,m,p} - \gamma_\alpha)^2 (\eta_{1,m} - \gamma_\alpha)^2 \end{array}} \Big) \; , \tag{31}$$

with $\gamma_\alpha = 2^m (1 - \alpha)$. Then virtual network (11) *is α-averaged if*

$$\sqrt{b_\alpha} - \sqrt{a_{1,m}} \leq 2^m \alpha - 2\theta_m \; . \tag{32}$$

Proof. Let us calculate the operator norms of U and $U - \gamma_\alpha \mathbb{1}$, where U is given by (18).

Norm of U. Applying Lemma 3.3 when $i = 1$ and $n = m$ yields

$$\|U\|^2 = a_{1,m} \; .$$

Norm of $U - \gamma_\alpha \mathbb{1}$. We follow the same reasoning as in the proof of Lemma 3.3. We have

$$\|U - \gamma_\alpha \mathbb{1}\|^2 = \sup_{z = (\omega_p)_p, \|z\| = 1} \sum_p \langle B_p \omega_p, \omega_p \rangle \; ,$$

where B_p is the symmetric positive semidefinite matrix given by

$$B_p = \begin{pmatrix} (\beta_{1,m,p} - \gamma_\alpha)^2 & (\beta_{1,m,p} - \gamma_\alpha)\widetilde{\beta}_{1,m,p} \\ (\beta_{1,m,p} - \gamma_\alpha)\widetilde{\beta}_{1,m,p} & (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}^2_{1,m,p} \end{pmatrix} .$$

By definition of the spectral norm,

$$\|U - 2^m(1-\alpha)\mathbb{1}\|^2 = \sup_p \nu_p \,, \tag{33}$$

where, for every $p \in \mathbb{N}$, ν_p is the maximum eigenvalue of B_p. The two eigenvalues of this matrix are the roots of the polynomial

$$\begin{aligned}(\forall \nu \in \mathbb{R}) \quad \det(B_p - \nu \mathbb{1}_2) = \nu^2 - ((\beta_{1,m,p} - \gamma_\alpha)^2 + (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}^2_{1,m,p})\,\nu \\ + (\beta_{1,m,p} - \gamma_\alpha)^2(\eta_{1,m} - \gamma_\alpha)^2 \,. \end{aligned} \tag{34}$$

Solving the corresponding second-order equation leads to

$$\begin{aligned}\sup_p \nu_p = \frac{1}{2}\Bigg(&(\beta_{1,m,p} - \gamma_\alpha)^2 + (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}^2_{1,m,p} \\ &+ \sqrt{\left((\beta_{1,m,p} - \gamma_\alpha)^2 + (\eta_{1,m} - \gamma_\alpha)^2 + \widetilde{\beta}^2_{1,m,p}\right)^2 - 4(\beta_{1,m,p} - \gamma_\alpha)^2(\eta_{1,m} - \gamma_\alpha)^2}\Bigg) \,. \end{aligned} \tag{35}$$

Then, it follows from (33) that $\|U - \gamma_\alpha \mathbb{1}\|^2 = b_\alpha$.

Conclusion of the proof. Based on the previous calculations, Condition (32) is equivalent to (26). In addition, let us note that for every $n \in \{1, \dots, m\}$, Q_n in (11) is firmly nonexpansive since R_n and $\mathbb{1}$ are. By applying now Proposition 3.2ii), we deduce that, when Condition (32) holds, virtual network (11) is α-averaged. □

Remark 3.6. Condition (32) just provides a sufficient condition for the averagedness of virtual network (11).

3.3 Link with the original neural network – direct approach

In this subsection we go back to our initial model defined by (12). We consider two different inputs $z_1 = (x_1, b_1)$ and $z_2 = (x_2, b_2)$ in $\mathscr{X} \times \mathscr{X}$. The distance between these points is

$$\|z_2 - z_1\| = \sqrt{\|x_2 - x_1\|^2 + \|b_2 - b_1\|^2} \,.$$

Let $z_{i,n} = (x_{i,m}, b_{i,m})$ be the output of the m-th layer of virtual network (11). Then,

$$\begin{aligned}\|z_{2,m} - z_{1,m}\|^2 &= \|x_{2,m} - x_{1,m}\|^2 + \|b_{2,m} - b_{1,m}\|^2 \\ &= \|x_{2,m} - x_{1,m}\|^2 + \eta^2_{1,m}\|b_2 - b_1\|^2 \,,\end{aligned}$$

and, thanks to Proposition 3.4,

$$\|z_{2,m} - z_{1,m}\|^2 \leq \frac{\theta_m^2}{2^{2(m-1)}} \left(\|x_2 - x_1\|^2 + \|b_2 - b_1\|^2\right) .$$

Then, the following inequality allows us to quantify the Lipschitz properties of the neural network (6) with an error on b_n:

$$\|x_{1,m} - x_{2,m}\|^2 \leq \frac{\theta_m^2}{2^{2(m-1)}} \|x_2 - x_1\|^2 + \left(\frac{\theta_m^2}{2^{2(m-1)}} - \eta_{1,m}^2\right) \|b_2 - b_1\|^2 .$$

Two cases are of interest:

- If the network is initialized with a fixed signal, say $x_{1,0} = x_{2,0} = 0$, then

  $$\|x_{1,m} - x_{2,m}\|^2 \leq \left(\frac{\theta_m^2}{2^{2(m-1)}} - \eta_{1,m}^2\right) \|b_2 - b_1\|^2 .$$

  So, a Lipschitz constant with respect to the input data $T^* y^\delta$ is

  $$\vartheta_m = \sqrt{\frac{\theta_m^2}{2^{2(m-1)}} - \eta_{1,m}^2}. \tag{36}$$

- On the other hand, if the initialization is dependent on the observed image, i.e. $x_{1,0} = b_1$ and $x_{2,0} = b_2$,

  $$\|x_{1,m} - x_{2,m}\|^2 \leq \left(\frac{\theta_m^2}{2^{2m-3}} - \eta_{1,m}^2\right) \|b_2 - b_1\|^2 .$$

  So a higher Lipschitz constant value w.r.t. to the input data is obtained:

  $$\vartheta_m = \sqrt{\frac{\theta_m^2}{2^{2m-3}} - \eta_{1,m}^2}. \tag{37}$$

Remark 3.7. Let us go back to Model (6). We thus consider the virtual Model (11) without leakage factor, i.e., for every $n \in \{1, \ldots, m\}$, $\eta_n = 1$ and $\eta_{1,m} = 1$. Then, for every $n \in \{1, \ldots, m\}$,

$$z_n = \begin{pmatrix} x_n \\ b_n \end{pmatrix}, \quad z_n = Q_n(U_n z_{n-1}) , \quad \text{with} \quad \begin{cases} Q_n = \begin{pmatrix} R_n \\ \mathbb{1} \end{pmatrix} , \\ \\ U_n = \begin{pmatrix} W_n & V_n \\ 0 & \mathbb{1} \end{pmatrix} . \end{cases} \tag{38}$$

Assume that the virtual network in (38) is α-averaged. Then it is 1-Lipschitz. This is also consistent with (26) which implies that

$$\|U - 2^m(1-\alpha)\mathbb{1}\| - \|U\| + 2\theta_m \leq 2^m\alpha \quad \Rightarrow \quad \frac{\theta_m}{2^{m-1}} \leq 1 . \tag{39}$$

Then, according to (36), we would get $\vartheta_m = 0$, which would mean that the network delivers an output independent of the available data. If we except trivial cases for which $R_n = 0$ or $W_n = 0$ for some $n \in \{1, \dots, m\}$, this behavior is impossible. So this means that virtual network (38) cannot be α-averaged, hence Condition (32) is not met when, for every $n \in \{1, \dots, m\}$, $\eta_n = 1$.

This can be concluded more directly. For every $\alpha \in]0, 1[$, virtual network (38) cannot be α-averaged, since $R = (1 - 1/\alpha)\mathbb{1} + 1/\alpha S$ cannot be nonexpansive. Indeed suppose that $\|R(x_1, b_1) - R(x_2, b_2)\|^2 \leq \|x_1 - x_2\|^2 + \|b_1 - b_2\|^2$, for every (x_1, b_1) and (x_2, b_2) in $\mathscr{X} \times \mathscr{X}$. Since

$$\begin{aligned} \|R(x_1, b_1) - R(x_2, b_2)\|^2 &= \left\| (1 - 1/\alpha) \begin{pmatrix} x_1 - x_2 \\ b_1 - b_2 \end{pmatrix} + 1/\alpha \begin{pmatrix} x_{1,m} - x_{2,m} \\ b_1 - b_2 \end{pmatrix} \right\|^2 \\ &= \|(1 - 1/\alpha)(x_1 - x_2) + 1/\alpha(x_{1,m} - x_{2,m})\|^2 + \|b_1 - b_2\|^2 \\ &\leq \|x_1 - x_2\|^2 + \|b_1 - b_2\|^2 , \end{aligned}$$

we deduce that

$$\|(1 - 1/\alpha)(x_1 - x_2) + 1/\alpha(x_{1,m} - x_{2,m})\| \leq \|x_1 - x_2\| ,$$

which cannot stand since, for $b_1 \neq b_2$, $x_{1,m} - x_{2,m}$ can be nonzero when $x_1 = x_2$.

3.4 Link with the original neural network – use of a semi-norm

Remark 3.7 suggests that we need a finer strategy to evaluate the nonexpansiveness properties of Model (6). On the product space $\mathscr{X} \times \mathscr{X}$, we define the semi-norm which takes only into account the first component of the vectors:

$$z = (x, b) \mapsto |z| = \|x\|. \tag{40}$$

Let $L \colon \mathscr{X} \times \mathscr{X} \to \mathscr{X} \times \mathscr{X}$ be any bounded linear operator and, for every $z \in \mathscr{X} \times \mathscr{X}$, let $Lz = ((Lz)_{\mathrm{x}}, (Lz)_{\mathrm{b}})$. We define the associated operator semi-norm

$$|L| = \sup_{\|z\|=1} \|(Lz)_{\mathrm{x}}\| . \tag{41}$$

Lemma 3.8

Let $m \in \mathbb{N}\backslash\{0\}$. For every $n \in \{1, \dots, m\}$ and $i \in \{1, \dots, n\}$, the seminorm $|U_n \circ \cdots \circ U_i|$ is equal to $\sqrt{\overline{a}_{i,n}}$ with

$$\overline{a}_{i,n} = \sup_p \left(\beta_{i,n,p}^2 + \widetilde{\beta}_{i,n,p}^2 \right) . \tag{42}$$

Proof. The seminorm of $U_n \circ \cdots \circ U_i$ is the same as the norm of $U_n \circ \cdots \circ U_i$ where η_n has been set to 0. The result thus follows from Lemma 3.3 where $\eta_{i,n} = 0$. □

Proposition 3.9

Let $m \in \mathbb{N} \setminus \{0\}$. For every $i \in \{1, \dots, n\}$ and $n \in \{1, \dots, m-1\}$, let $a_{i,n}$ be defined by (27) and let $\overline{a}_{i,m}$ be given by (42). Set $\theta_0 = 1$ and define

$$(\forall n \in \{1, \dots, m-1\}) \quad \theta_n = \sum_{i=1}^{n} \theta_{i-1} \sqrt{a_{i,n}} \,, \tag{43}$$

$$\overline{\theta}_m = \sum_{i=1}^{m} \theta_{i-1} \sqrt{\overline{a}_{i,m}} \,. \tag{44}$$

Then the network in (12) with input (x_0, b_0) and output x_m is $\overline{\theta}_m / 2^{m-1}$-Lipschitz.

Proof. Network (12) can be expressed as $R_m \circ \overline{U}_m \circ Q_{m-1} \circ U_{m-1} \circ \cdots \circ Q_1 \circ U_1$ where

$$\overline{U}_m = D_{\mathrm{x}} \circ U_m \tag{45}$$

and D_{x} is the decimation operator

$$D_{\mathrm{x}} = [\mathbb{1} \quad 0]. \tag{46}$$

This network has the same Lipschitz properties as the network in (11) with $\eta_m = 0$. The result can thus be deduced from Proposition 3.4 by setting $\eta_{1,m} = 0$. □

To investigate averagedness properties, a first possibility is to consider a network from $\mathscr{X} \times \mathscr{X}$ to $\mathscr{X} \times \mathscr{X}$ with input (x_0, b_0) and output $(x_m, 0)$.

Proposition 3.10

Let $m \in \mathbb{N} \setminus \{0, 1\}$. Let $\overline{a}_{1,m}$ be defined in Lemma 3.8 and $\overline{\theta}_m$ be defined in Proposition 3.9. Let $\alpha \in [1/2, 1]$. Define

$$\overline{b}_\alpha = \frac{1}{2} \sup_p \Big((\beta_{1,m,p} - \gamma_\alpha)^2 + \widetilde{\beta}_{1,m,p}^2 + \gamma_\alpha^2 + \\ + \sqrt{\left((\beta_{1,m,p} - \gamma_\alpha)^2 + \widetilde{\beta}_{1,m,p}^2 + \gamma_\alpha^2\right)^2 - 4(\beta_{1,m,p} - \gamma_\alpha)^2 \gamma_\alpha^2} \Big) \,, \tag{47}$$

with $\gamma_\alpha = 2^m(1-\alpha)$. If

$$\sqrt{\overline{b}_\alpha} - \sqrt{\overline{a}_{1,m}} \leq 2^m \alpha - 2\overline{\theta}_m \,, \tag{48}$$

then the network in (6) with input (x_0, b_0) and output $(x_m, 0)$ is α-averaged.

Proof. The network of interest is

$$Q_m \begin{bmatrix} \overline{U}_m \circ Q_{m-1} \circ U_{m-1} \circ Q_1 \circ U_1 \\ 0 \end{bmatrix} \,.$$

This network can be viewed as a special case of the network in (11) where $\eta_m = 0$, which implies that $\eta_{1,m} = 0$. The result is thus a consequence of Proposition 3.5. □

Another possibility for investigating averagedness properties consists of defining a network from $\mathscr{X}$ to $\mathscr{X}$. We will focus on two specific networks of the form

$$R_m \circ \overline{U}_m \circ Q_{m-1} \circ U_{m-1} \cdots Q_1 \circ \widehat{U}_1, \tag{49}$$

where $\overline{U}_m$ is given by (45).

i) The first one assumes that $x_0 = 0$ in (6). It is thus given by

$$\widehat{U}_1 = U_1 \begin{bmatrix} 0 \\ \mathbb{1} \end{bmatrix} . \tag{50}$$

ii) The second one assumes that $x_0 = b_0$ in (6). It is thus given by

$$\widehat{U}_1 = U_1 \begin{bmatrix} \mathbb{1} \\ \mathbb{1} \end{bmatrix} . \tag{51}$$

By proceeding similarly to the proof of Lemma 3.3 and Proposition 3.5, we obtain the following two results:

Lemma 3.11

Let $m \in \mathbb{N} \setminus \{0, 1\}$. For every $n \in \{1, \ldots, m-1\}$, the norm of $U_n \circ \cdots \circ U_2 \circ \widehat{U}_1$ is equal to $\sqrt{\widehat{a}_{1,n}}$ with

$$\widehat{a}_{1,n} = \begin{cases} \sup_p \widetilde{\beta}_{1,n,p}^2 + \eta_{1,n}^2 & \text{in case i)} \\ \sup_p \left((\beta_{1,n,p} + \widetilde{\beta}_{1,n,p})^2 \right) + \eta_{1,n}^2 & \text{in case ii)} \end{cases} \tag{52}$$

and the norm of $\overline{U}_m \circ \cdots \circ U_2 \circ \widehat{U}_1$ is equal to $\sqrt{\widehat{a}_{1,m}}$ with

$$\widehat{a}_{1,m} = \begin{cases} \sup_p \widetilde{\beta}_{1,m,p}^2 & \text{in case i)} \\ \sup_p (\beta_{1,m,p} + \widetilde{\beta}_{1,m,p})^2 & \text{in case ii).} \end{cases} \tag{53}$$

Proposition 3.12

Let $m \in \mathbb{N} \setminus \{0, 1\}$. For every $i \in \{2, \ldots, n\}$ and $n \in \{1, \ldots, m-1\}$, let $a_{i,n}$ be defined by (27) and let $\overline{a}_{i,m}$ be given by (42). For every $n \in \{1, \ldots, m\}$, let $\widehat{a}_{1,n}$ be defined by (52) and (53). Define $(\widehat{\theta}_n)_{1 \leq n \leq m}$ recursively by

$$(\forall n \in \{1, \ldots, m-1\}) \quad \widehat{\theta}_n = \sqrt{\widehat{a}_{1,n}} + \sum_{i=2}^{n} \widehat{\theta}_{i-1} \sqrt{a_{i,n}} \, , \tag{54}$$

$$\widehat{\theta}_m = \sqrt{\widehat{a}_{1,m}} + \sum_{i=2}^{m} \widehat{\theta}_{i-1} \sqrt{\overline{a}_{i,m}} \, . \tag{55}$$

Then network (49) is $\widehat{\theta}_m / 2^{m-1}$-Lipschitz.

The averagedness properties of network (49) in cases i) and ii) are consequences of these results.

Proposition 3.13

Let $m \in \mathbb{N} \setminus \{0,1\}$. Let $\widehat{a}_{1,m}$ be defined in Lemma 3.11 and $\overline{\theta}_m$ be defined in Proposition 3.12. Let $\alpha \in [1/2, 1]$. Define

$$\widehat{b}_\alpha = \begin{cases} \sup_p(\widetilde{\beta}_{1,m,p} - 2^m(1-\alpha))^2 & \text{in case i)} \\ \sup_p(\beta_{1,m,p} + \widetilde{\beta}_{1,m,p} - 2^m(1-\alpha))^2 & \text{in case ii).} \end{cases} \tag{56}$$

If

$$\sqrt{\widehat{b}_\alpha} - \sqrt{\widehat{a}_{1,m}} \leq 2^m\alpha - 2\widehat{\theta}_m \ , \tag{57}$$

then network (49) is α-averaged.

Proof. Let us calculate the operator norms of $\widehat{U} = \overline{U}_m \circ U_{m-1} \cdots \circ U_2 \circ \widehat{U}_1$. and $\widehat{U} - 2^m(1-\alpha)\mathbb{1}$. Applying Lemma 3.11 when $n = m$ yields

$$\|\widehat{U}\|^2 = \widehat{a}_{1,m} \ .$$

By following the same reasoning as in the proof of Proposition 3.5, we get

$$\|\widehat{U} - 2^m(1-\alpha)\mathbb{1}\|^2 = \widehat{b}_\alpha. \tag{58}$$

By applying now Proposition 3.2ii), we deduce that, when Condition (57) holds, network (49) is α-averaged. □

4 Numerical Examples

In this section, we present numerical tests carried out in the case of the class of Abel integral operators. We present in more details the architecture chosen to build the neural network. Several numerical examples are provided to illustrate the accuracy of the proposed method. The stability of the neural network is evaluated by computing its Lipschitz constant by relying upon the results of Section 3.

4.1 Problem formulation

To implement the neural network defined by (6), we focus on the Abel integral operator

$$\begin{array}{rcl|rcl} T & : & L^2(0,1) & \to & L^2(0,1) \\ & & x & \to & y(t) = \dfrac{1}{\Gamma(a)} \displaystyle\int_0^t (t-s)^{(a-1)} x(s) \, \mathrm{d}s \ , \end{array} \tag{59}$$

where $a > 0$ and Γ is the classical Gamma function, $\Gamma(a) = \int_0^{+\infty} t^{a-1} e^{-t} \, \mathrm{d}t$. The Abel operator T is injective, linear, and compact. The inverse problem linked to the Abel transform has been widely studied from a theoretical viewpoint, as in [41]. The range of T is a subset of $H^{-a}(0,1)$, the dual space of $H^a(0,1)$, and the problem is ill-posed of order a in the sense of [16].

Recovering x from a noisy measurement $y^\delta = Tx + v^\delta$ is an inverse problem linked to a large variety of experimental contexts in physics. Indeed, the operator T allows to define derivatives of fractional order for $a < 1$ and integrals of arbitrary order for $a > 1$. The most common case is the semi-derivative, when $a = 1/2$. Typically, the inverse problem consists in searching a distribution of a two-dimensional or three dimensional object from measurements of the projection of this quantity onto an axis, in which case the radial distribution is linked to the values of these projections via the Abel transform (see plasmas and flames [14], tomography [42], or astrophysics [43]). In a different context, fractional calculus appears to be very convenient to describe properties of polymers [44] or surface-volume reaction problems [45]. Subsequently, a large number of physical applications have been documented in [41].

According to the theory in [46], the derivative operator is given as a power of the Laplacian denoted by B, defined on $\mathcal{D}(B)$:

$$\begin{cases} B = & -\Delta \\ \mathcal{D}(B) = & \left\{ x \in H^2(0,1) \mid x(1) = 0 \ , \ \ x'(0) = 0 \right\} \ . \end{cases} \tag{60}$$

Then, the continuous derivative operator D in (5) is chosen as $D = B^{r/2}$, with $r > 0$ characterizing the order of derivation. This choice ensures that the continuous operators T^*T and D^*D commute, since for $x \in L^2(0,1)$, we have $BT^*Tx = x$ and for $x \in \mathcal{D}(B)$, we have $T^*TBx = x$.

4.2 Discretization

We first describe the discretization choices to pass from our continuous framework to a numerical setting. Network (6) is made up of continuous operators. To carry out our experiments, we propose the following discretization. We suppose that the measured signal $y = Tx$ is acquired on a regular mesh of N points, $(t_i)_{0 \leq i \leq N-1}$ in the interval $[0,1]$, with $t_0 = 0$ and $t_N = 1$. The measured signal $y = (y_i)_{0 \leq i \leq N-1} = (y(t_i))_{0 \leq i \leq N-1}$ belongs to the space endowed with the finite element basis $(e_i)_{0 \leq i \leq N-1}$ associated to $(t_i)_{0 \leq i \leq N-1}$. However, instead of working only in the finite element basis, we also consider projection of the signal onto the span of the first K eigenvectors of the self-adjoint nonnegative operator T^*T. This choice is justified for two reasons. First, in such basis, the discretized forms of operators T^*T and D^*D respectively defined by (59) and (60) are diagonal and therefore commute. This is a prerequisite to apply Propositions 3.12 and 3.13. Second, the eigenvectors of T^*T denoted by $(u_k)_{k \in \mathbb{N}}$ are trigonometric polynomials and the retained discretization method is a spectral one, as defined in [47], or [48]. This discretization method can fully account for the regularity of the initial condition on x, under extra mild assumptions.

We denote by $(u_k, \beta_{T,k})_{k \in \mathbb{N}}$ the eigensystem of T^*T. Note that since T is a compact operator, the eigenvectors $(u_k)_{k \in \mathbb{N}}$ is a set of orthonormal eigenvectors, and $(\beta_{T,p})_{n \in \mathbb{N}}$ are strictly positive eigenvalues. The signal y is then discretized in the basis formed by the K first eigenvectors $(u_k)_{0 \leq k \leq K-1}$. Explicit values of $(u_k, \beta_{T,k})$ are given in [49]. As already stated, the operators T^*T and D^*D reduce to diagonal matrices, with the following eigenvalues on their

diagonal:

$$(\forall k \in \{0,\dots,K-1\}) \quad \beta_{T,k} = \left(\frac{4}{\pi^2(1+2k)^2}\right)^a \ , \quad \beta_{D,k} = \left(\frac{\pi^2(1+2k)^2}{4}\right)^r = \beta_{T,k}^{-r/a} \ . \tag{61}$$

Hereafter, we consider that $r = 1$, and $D = B^{1/2}$, with B defined by (60). We compute the change basis matrix denoted by $P = (P_{i,j})_{0\le i,j\le N-1}$ with, for every $i \in \{0,\dots,N-1\}$ and $j \in \{0,\dots,N-1\}$,

$$P_{i,j} = \frac{2\sqrt{2}}{\gamma_j}\cos\left(\frac{2i+1}{2N}\gamma_j\right)\sin\left(\frac{1}{2N}\gamma_j\right) \ , \quad \gamma_j = \sqrt{\beta_{T,i}} = \left(\frac{2}{\pi(1+2n)}\right)^a \ . \tag{62}$$

The operator T does not intervene in the neural network (6), as only T^*T and D^*D do. However, to generate synthetic data and the associated bias b_0, we need also a discretization for the operator T. Therefore, T is approximated by T_{elt} as a computation of an integral using the trapezoïdal rule, with stepsize $h = 1/(N-1)$, and, for $a \neq 1$, for $0 \le i < N$, $0 \le j < N$,

$$(T_{\text{elt}})_{i,j} = \begin{cases} \dfrac{1}{\Gamma(a)}\dfrac{h^a}{2a}\left((i-j+1)^a - (i-j-1)^a\right) & \text{if } j < i \ , \\ \dfrac{1}{\Gamma(a)}\dfrac{h^a}{2a}(i^a - (i-1)^a) & \text{if } j = 0,\ i \neq 0 \ , \\ \dfrac{1}{\Gamma(a)}\dfrac{h^a}{2a} & j = i,\ \text{if } i \neq 0 \ , \\ 0 & \text{if } i = j = 0, \text{ or } j > i. \end{cases} \tag{63}$$

Then, the operators T and T^* in the eigen basis are respectively approximated by $T_{\text{eig}} = PT_{\text{elt}}$ and $(T^*)_{\text{eig}} = PT_{\text{elt}}^\top$. Thus, on the one hand the synthetic data are calculated and stored in the basis of the finite elements, and, on the other hand, the algorithm operates in the basis of eigenvectors, except for the proximity operator, for which a change of basis is performed before and after.

To carry out the numerical experiments, we set $N = 2 \times 10^3$ and $K = 50$. Therefore, the regular signals x are approximated by their projection onto the space generated by the first K eigenvectors of T^*T.

4.3 Neural network architectures and characteristics

Structure The architecture that we propose here reflects the proposed Model (6), that unfolds the forward-backward algorithm for minimizing functional J defined by (5) over a finite number of iterations m. The main difference with the classical forward-backward algorithm lies in the fact that only a finite number of iterations m is performed instead of pursuing the iterations until convergence. Then, m corresponds to the number of layers of the neural network. Similarly

to the work proposed in [34], each layer of the neural network consists of a block made up of hidden layers which calculate the hyper-parameters and an iteration of the forward-backward algorithm. Here, the bias b_0 is taken as the discretization of $T^* y^\delta$, namely $P T_{\text{elt}}^\top y^\delta$. The hyper-parameters are defined independently across the network in order to provide more flexibility. The overall structure of the network is shown in Figure 1. The structure of its hidden layers is detailed in Figure 2.

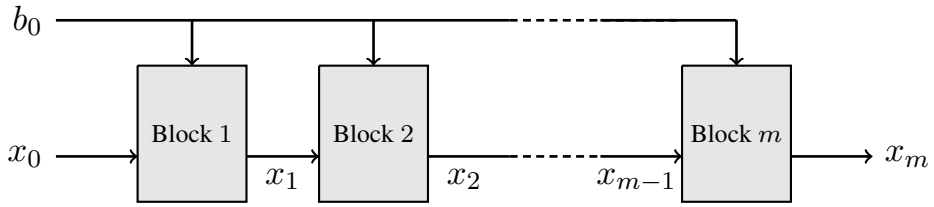

Figure 1: Global architecture of neural network (6)

The activation function, namely operator R_n in Figure 2, corresponds the proximal operator associated with g appearing in (5). We remind that, for an indicator function of a nonempty closed convex set, the proximity operator is a projection onto this set. However, such activation functions, especially in the case where one wishes to guarantee the positivity of the solution, may show bad properties during gradient back-propagation and training. These include vanishing gradient problems as shown in [50], [51] for the Rectified Linear Unit (ReLU) function. Then, we choose to consider instead a logarithmic barrier g to enable prior knowledge in the algorithm, as proposed in [34]. The activation is no more constant and depends on the gradient step λ_n and the barrier parameter $\mu_n > 0$ as

$$R_n = \text{prox}_{\lambda_n \mu_n g} \ .$$

Constraint and proximity operator More precisely, we experiment two possible choices for the function g in expression (5). As mentioned above, the prior knowledge on the constraint set C is thus embedded in the network through the logarithmic barrier function g. In both cases,

$$\begin{cases} C = \{x \in L^2(0,1) \mid c_i(x) \geq 0, \ 1 \leq i \leq p\} \ , \\ \\ \text{int } C = \{x \in L^2(0,1) \mid \ c_i(x) > 0, \ 1 \leq i \leq p\} \ , \\ \\ (\forall x \in L^2(0,1)) \quad g(x) = \begin{cases} -\sum_{i=1}^p \ln\left(c_i(x)\right) & \text{if } x \in \text{int } C \\ +\infty & \text{otherwise} \ , \end{cases} \end{cases} \tag{64}$$

where $(c_i)_{1 \leq i \leq p}$ are suitable functions allowing us to describe the constraint set. First, we consider that the signal x has a minimum value x_{min} and a maximum value x_{max}. Then C can be rewritten as

$$C = \left\{x \in L^2(0,1) \mid x \geq x_{\text{min}}, \ -x \geq -x_{\text{max}}\right\} \ . \tag{65}$$

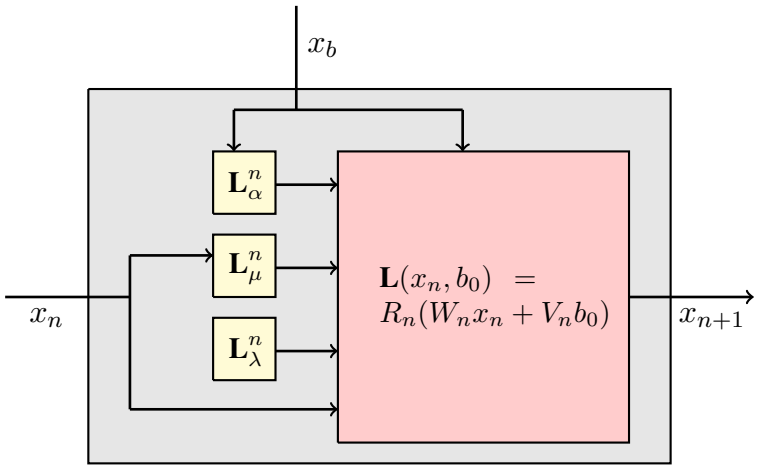

Figure 2: Architecture of one iteration - Block n.

This kind of constraint can be useful for example when the signal the experimenter wishes to recover corresponds to a positive and bounded physical quantity. Secondly, we consider an affine constraint such as, for $j > 0$,

$$C = \left\{ x \in L^2 \mid 0 \leq \int_0^1 t^j x(t) \, \mathrm{d}t \leq 1 \right\} . \tag{66}$$

This constraint reflects the fact that a physical quantity linked to the signal is bounded. For $j \in \{1, 2, 3\}$, the moment of order j involved in (66) represents the total mass of elements in a 1D, 2D, or 3D system, respectively.

The computation of the proximity operator associated to logarithmic barrier functions [2], after discretization, can be found in [34]. In our case, the barrier parameter in each layer n, denoted by μ_n is estimated with a convolutional neural network, which takes as input the output x_n of the $(n-1)$-th layer. The detailed architecture of $\mathbf{L}_\mu^n$ is depicted in Figure 3. Since the barrier parameter is positive, we enforce this constraint by an approximation of the ReLU activation function, namely Softplus [52], with $\beta > 0$,

$$\text{Softplus}(x, \beta) = \frac{1}{\beta} \ln \left(1 + e^{\beta x} \right) . \tag{67}$$

Other parameters We introduce between each layer a hidden layer responsible for computing the gradient step size λ_n, and the regularization parameter τ_n, respectively.

The gradient descent step $(\lambda_n)_{1 \leq n \leq m}$ only depends on the structure of the network. This parameter is then trained without any prior knowledge. Since its value is positive, we compute it as:

$$\lambda_n = \text{Softplus}(c_n), \tag{68}$$

where c_n is a scalar parameter of the network learned during training.

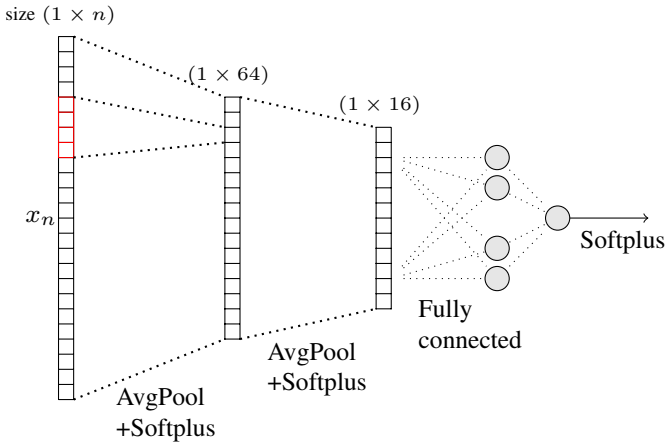

Figure 3: Architecture of one hidden layer $\mathbf{L}_\mu^n$ computing the barrier parameter.

The regularization parameters $(\tau_n)_{1\leq n\leq m}$ in neural network (6) should only depend on the bias b_0. Indeed, for a regularization of the generalized Tikhonov type, the regularization parameter theoretically depends on the regularity of the a priori and on the noise level. The theoretical optimal value of this parameter can be explicitly computed, as shown in [53] or [46],

$$\tau = c\left(\frac{\delta}{\rho}\right)^{\frac{2(a+r)}{a+q}}, \tag{69}$$

where $\rho = \|x\|_{L^q(0,1)}$, and $x \in H^q(0,1)$ represents the ideal signal, δ is the noise level in L^2 norm, a is the degree of ill-posedness of the inverse problem, r the level of regularization (or the order of the differential term in the regularization), and c is a constant. Since we do not have access to the noise level, we estimate it thanks to the Fourier transform of the signal. We assume here that the noise corresponds to the high frequency components of the signal. This assumption is only used to obtain an approximate value of the error. Subsequently, the algorithm makes it possible to search the optimal value without any assumption on the Fourier spectrum of the error. This is achieved by learning a constant d_n such that

$$\tau_n = \text{Softplus}(d_n)\left(\frac{\|b_0 - \text{FFT}_{f_{\max}}(b_0)\|_2}{\|\text{FFT}_{f_{\max}}(b_0)\|_q}\right)^{\frac{2(a+r)}{a+q}}, \tag{70}$$

where the operator $\text{FFT}_{f_{\max}}$ cuts the frequencies of the Fourier transform greater than $f_{\max}$, r is the order of derivation in the regularization term, and q is the order of regularity of the a priori, i.e. $x \in H^q(0,1)$. This form of regularization is theorically the best choice as long as $r \geq q/2 - a$, as shown in [53]. Moreover, this insures that the dependence of the τ_n parameter of the network on the bias b_0 is of second order and can be neglected while computing the Lipschitz constant.

4.4 Dataset and experimental settings

Synthetic Data To ensure the universality of the approach, we train the network on a wide variety of functions, without too strong a priori on their form or their properties. For example, we do not want to restrict our training to Gaussian-like functions which would be likely to be oversimplistic models. We are therefore looking for a sufficiently rich dictionary of functions sampled over N points. To create a diverse dataset of positively distributed functions, we found convenient to use histograms of color images from a standard image dataset. However, in order to properly reflect the a priori of regularity, the following processing is then carried out to these histograms. The functions are first smoothed using a Savitzky-Golay filter, with filter length 21 and polynomial order 5. Then, to ensure that such signals are in the range of T^*T, the outputs of the filter are padded at $t = 0$ by a constant value, and at $t = 1$ by zero. Finally, the signals are projected into the eigenvector basis described in Section 4.2. This process ensures that the obtained signal x in the training set belong to C^∞, as the eigenvectors do. In particular, x belongs to the space $\mathcal{D}(B)$ defined in (60), in which case the regularization is optimal in the sense of [16] (see [49]).

In order to synthesize noisy signals y^δ, the discrete transformation T_{elt} defined by (63) is applied to the set of signal x created as aforementioned. Then, a zero-mean white Gaussian noise with a preset standard deviation δ is added.

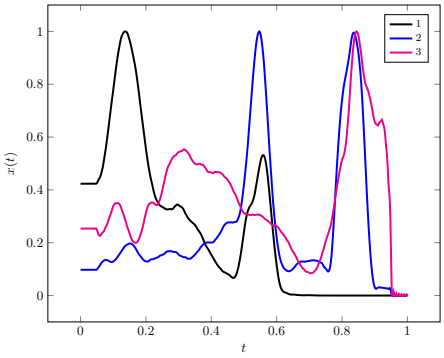

Figure 4: Example of three signals of synthetic data for the constraint (65). These signals are bimodal or almost bimodal, with variable peak widths. This dataset presents a great diversity of functions and demonstrates the agnostic nature of the model in order to represent a large panel of physical signals. The imposed constraints are the regularity of the signal (here $C^\infty(0, 1)$) and the boundary conditions at $t = 0$ and $t = 1$.

Figure 4 represents an example of three signals simulated by our method. In Figures 5 and 6, we display the image of those signals by the operator T defined by (1) with and without the presence of additive noise, respectively for $a = 1$ and $a = 1/2$. For $a = 1$, we recover the 1D integral operator. All our datasets and codes implemented in Pytorch are available online [1].

[1] https://github.com/ceciledellavalle/FBResNet

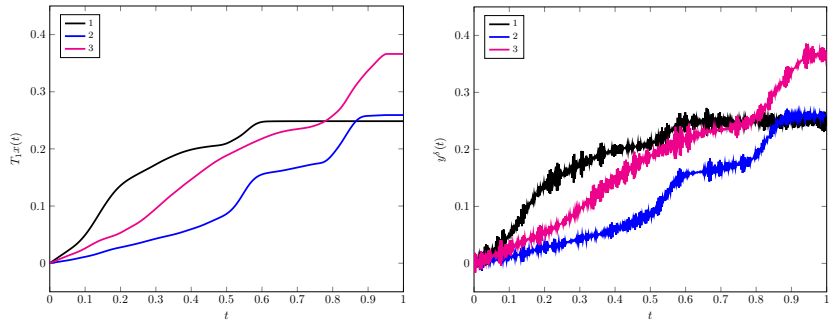

Figure 5: For the three examples displayed in Figure 4, we plot on the left their image by T defined by (1) for $a = 1$, and on the right the same signal after addition of noise with level $\delta = \|y^\delta - T_1 x\|_{L^2} = 0.05\|T_1 x\|_{L^2}$.

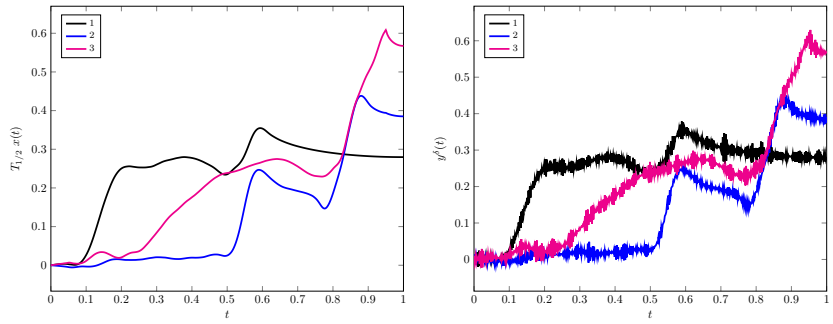

Figure 6: For the three examples displayed in Figure 4, we plot on the left their image by T defined by (1) for $a = 1/2$, and on the right the same signal after addition of noise with level $\delta = \|y^\delta - T_1 x\|_{L^2} = 0.05\|T_1 x\|_{L^2}$.

Training The network is classically trained in an end-to-end fashion. The gradient of the proximity operator is explicitly coded and inserted into the back-propagation according to the chain rule. We trained the network over 30 epochs with a learning rate of 10^{-3}, by using a training set of 400 signals. We use the Adam optimizer [54] to minimize the training loss, which is taken as the mean square error. We compute the validation loss at every epoch by using a set of 200 signals. The batchsize is equal to one. The training takes approximately three to four hours on an NVIDIA Titan Xp GPU, while the computational time required for testing one signal is only about 50ms on a 2.9 GHz 6-Core Intel Core i9.

Using Proposition 3.12, the Lipschitz constant is estimated at each epoch of training. One observes various behaviors, according to the initial values of the hyper-parameters. Either the Lipschitz constant increases until stabilizing, or it decreases.

4.5 Results and discussion

Results We display in Figures 7 and 8 the output of the neural network for respectively Gaussian signals and signals displayed in Figure 4, different values of a and two possible choices for set C defined either by (65) or by (66). We notice that the method performs well, and that under identical conditions, the obtained signal tends to be of lower quality when the order of the inverse problem a increases. The reason may be numerical. If we compare this performance to a classical gradient descent algorithm, when a increases the eigenvalues are greater, the gradient norm increases and, in order to compensate, the gradient stepsize decreases. Since the number of iteration is fixed, this could mean that the solution is away from the optimal solution of (3). The reason can also be theoretical: the convergence rate of the error with respect to the noise standard deviation decreases as a increases according to [53].

	(1)	(2)	(3)	(4)
	$a = 1$	$a = 1/2$	$a = 1$	$a = 1/2$
	$0 \leq x_i \leq 1$	$0 \leq x_i \leq 1$	$0 \leq h^2 \sum_i i x_i \leq 1$	$0 \leq h^2 \sum_i i x_i \leq 1$
(53) (i))	4.92×10^{-2}	4.91×10^{-2}	4.83×10^{-2}	4.83×10^{-2}
(53) (ii))	2.82×10^{-3}	7.00×10^{-3}	8.72×10^{-3}	3.37×10^{-3}

Table 1: Lipschitz constant of trained neural network (6) for various choices of order a and constraints, computed for an input $x_0 = b_0 = T^*(Tx + v^\delta) = T^* y^\delta$. We recall that h is the mesh stepsize equal to $1/N$.

Figure 9 shows the values of the hyper-parameters of the network after training for the constraint (65) and the integral operator, namely $a = 1$. We notice that the gradient step is smaller than $2/(\tau\beta_{D,K}) = 2\beta_{T,K}/\tau = 8 \times 10^{-4}$, which is theoretically the largest gradient step leading to a convergent forward-backward algorithm.

Lipschitz constant estimation Table 1 shows the Lipschitz constant obtained for the trained network under various conditions. A first remark is that, for all the studied problems, the Lipschitz constant does not vary much neither according to the choice of a nor of the constraint.

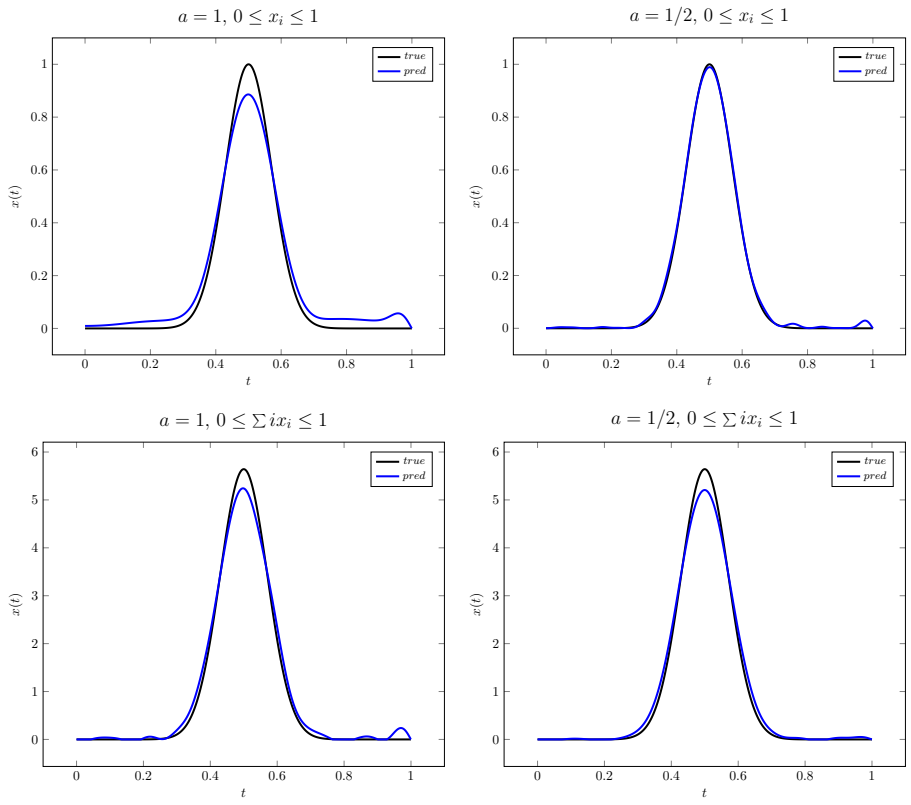

Figure 7: Output of the neural network for a Gaussian signal for various values of a and various constraints. The input of the neural network convolved signal Tx with an additive white noise of level $\delta = 0.05\|Tx\|$. The regularization prior is based on the derivative, namely $r = 1$ in (5), as a power of B defined in (60).

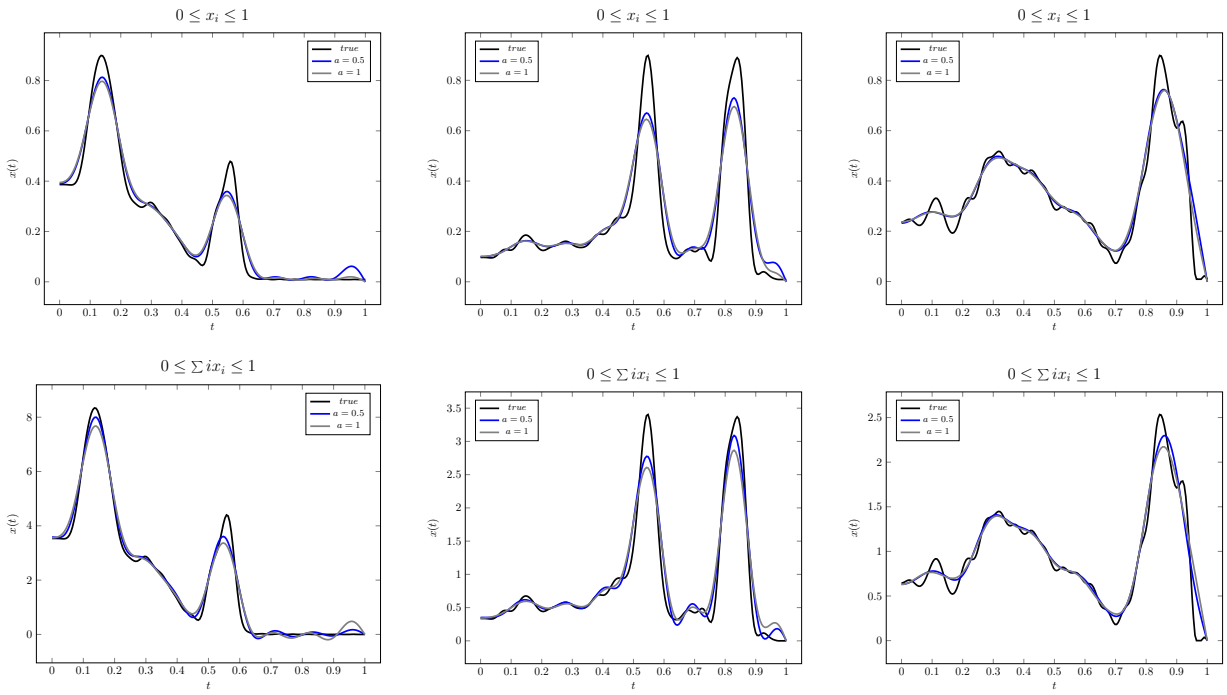

Figure 8: Example of outputs for three functions in the dataset. The measured signal is presented in Figures 5 and 6. The constraint $0 \leq \sum_i x_i \leq 1$ seems to give better result. We can also compare $a = 1$ and $a = 1/2$: when the order is smaller, the outputs look closer to the true signal.

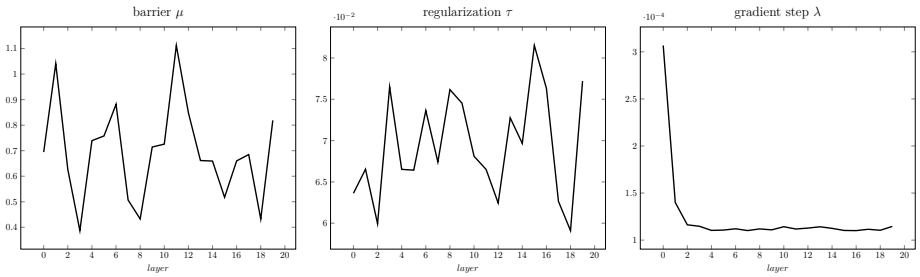

Figure 9: Hyper-parameters obtained after training for case (1) in Table 1.

A second remark is that the obtained Lipschitz constants are lower than 1. The algorithm tends to constrict the solutions. For $\delta = 0$, the norm $\|b_0\|$ is of the same order as $\|T^*Tx\|$. For noiseless data, the neural network (6) would act as the inverse operator $(T^*T)^{-1}$. Numerically, $1/\|T^*T\| = 1/\beta_{T,0} = 2.47$. This corresponds to the largest eigenvalue of T^*T. However, Lipschitz constants are smaller. A possible explanation would be the following, regular function x belong to a smaller vector space than noisy inputs b_0. When the regular signals are projected in the basis of eigenvectors of the compact operator T^*T, the coefficients decay extremely rapidly. Numerically, for a given function of the dataset, the coefficient corresponding to the tenth eigenvalues is in average lower than 10^{-3} times the first coefficient. However, the noisy input has non-zero coefficients over the entire spectrum. We can therefore expect the neural network to behave roughly as the following filter of high frequency on the spectrum:

$$\sum_{k=0}^{K-1} \frac{b_{0,k}}{\beta_{T,k} + \tau\beta_{D,k}} ,$$

where $b_{0,k} = \langle T^*Tx, u_k \rangle = \beta_{T,k} x_k$, and τ is a regularization parameter. Then, the Lipschitz constant is bounded by

$$\frac{1}{\beta_{T,0}} \max_{0 \le k \le K-1} \frac{\beta_{T,k}}{\beta_{T,k} + \tau\beta_{D,k}} .$$

As an example, for a Gaussian signal with mean value 0.5 and standard deviation $\sigma = 0.1$, and for τ of the order of 0.05 (as in Figure 9), we obtain $L \sim \sum_k x_k \pi^2/4(1+\tau\,(2k+1)^4) \sim 0.035$. This is the reason why we can expect the neural network Lipschitz constants to be roughly of the same order, namely around 10^{-2}.

Comparison For any value of a, there are many techniques to reverse T defined by (1). For $a = 1/2$, the Abel transform has been largely studied and three types of techniques are commonly used.

First we can mention interpolation techniques. Those consist in projecting the Abel operator into a basis whose properties reflect the regularity of the solution. The interpolation can be performed using Chebyshev polynomial as in [55] or [56] or a Gaussian function set as in [42]. They are fast, easy to implement and give good results for noiseless data. However, as exact-inverse methods, they have an extreme sensitivity to noise, and a preprocessing may be needed.

Nevertheless, they have shown good properties for sparse data [57], or for not evenly distributed measurements (see [55]).

Secondly, Fourier transform techniques, which consist in projecting the signal in the Fourier basis are presented in [58], [59] or similar techniques for any $a \leq 1$ in [60]. Those frequentist thresholding methods consist in reducing the weight of estimates on coefficients corresponding to smaller eigenvalues, for which the noise will overpower the signal. Those techniques show computational efficiency and good noise rejection capabilities, but suffer from some drawbacks as they are accurate only for certain types of input data that have a sparse representation in the Fourier domain as shown in [61] or more generally in [62].

Thirdly, we can also mention Kalman techniques for optimum least-squares estimation applied to the inversion from noisy data. Such techniques have been successfully applied to the Abel inverse transform in [63, 64].

Noise δ	Kalman		Neural Network		Fourier	
	$a=1$	$a=1/2$	$a=1$	$a=1/2$	$a=1$	$a=1/2$
0.1	0.436	0.383	0.280	0.126	0.237	0.148
0.05	0.293	0.260	0.177	0.089	0.186	0.142
0.01	0.126	0.125	0.095	0.075	0.177	0.140

Table 2: Averaged normalized error of the output $\|x^\delta - x\|/\|x\|$ obtained for different noise standard deviation values δ and different types of signals. The error for the Kalman Filter are of the order of $\sqrt{\delta}$ according to theory, but in practice finding the parameters allowing to reach such a precision is difficult. We compare the result with the neural network with the constraint defined by (65).

Two methods have been implemented, Fourier and Kalman, and tested over the same dataset of 50 signals x, different from the training set of the neural network. The averaged error on the outputs over the dataset for each methods are displayed in Table 2. The neural network (6) compares favorably with other techniques for a relative solution error in L^2. The Fourier method is more accurate as the noise level increases, since it works by filtering high frequencies. The Kalman method returns less regular solutions, and calibration of the regularization parameter raises an additional experimental difficulty.

5 Conclusion

In the continuity of the work of [34], the present paper proposes to unroll an algorithm obtained from a variational formulation of 1D integral inverse problems. This approach is versatile, since it allows to invert a broad family of integral or convolution operators, and it delivers a solution taking into account physical constraints of the problem. Indeed, with some existing methods, it may be difficult to enforce constraints, such as complying with some bounds or belonging to a given subspace. In particular, in many practical scenarios, solutions that do not fulfill basic constraints such as positivity may appear as irrelevant in terms of physical interpretation. The numerical solutions obtained for the case of the Abel operator indicate that the approach is easy to implement and computationally very attractive, since the training takes only a couple of hours

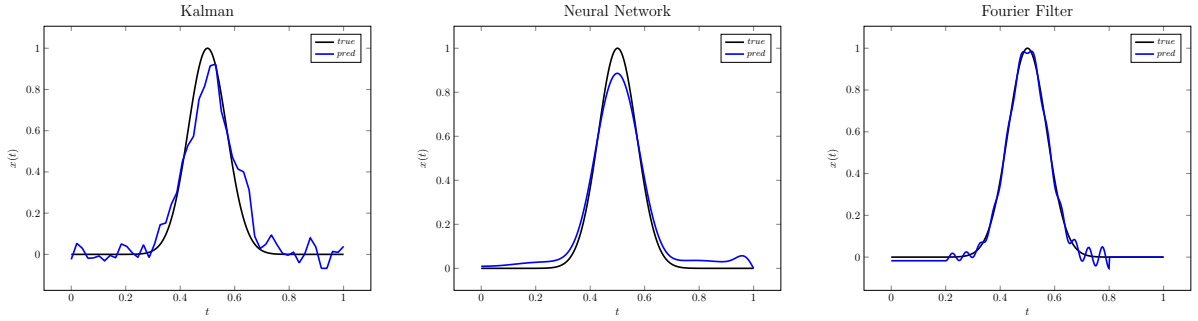

Figure 10: Output of the neural network for a Gaussian signal when $a = 1$, the white noise level is $\delta = 0.05\|Tx\|$, using three differents techniques : Kalman, neural network with constraint (65) and Fourier filtering.

and testing or prediction takes a few seconds on a regular CPU.
We additionally performed a theoretical analysis of robustness with respect to the observed data, which ensures the reliability of the proposed inverse method. In future work, more sophisticated neural network structures could be considered or additional parameters (such as the leakage factors we introduced in our analysis) could be learnt. Also, training sets which would better suited to specific applications could be employed within our framework. We think also that the α-averaged properties that we established pave the way for building recurrent networks in the spirit of [37].

References

[1] Mario Bertero, Patrizia Boccacci, Gabriele Desiderà, and Giuseppe Vicidomini. Image deblurring with Poisson data: from cells to galaxies. Inverse Problems, 25(12):123006, November 2009.

[2] Emilie Chouzenoux, Saïd Moussaoui, and Jérôme Idier. Majorize–minimize line-search for inversion methods involving barrier function optimization. Inverse Problems, 28(6):065011, May 2012.

[3] Zihui Wu, Yu Sun, Alex Matlock, Jiaming Liu, Lei Tian, and Ulugbek S. Kamilov. SIMBA: Scalable inversion in optical tomography using deep denoising priors. IEEE Journal of Selected Topics in Signal Processing, 14(6):1163–1175, October 2020.

[4] Louis-François Arsenault, Richard Neuberg, Lauren A. Hannah, and Andrew J. Millis. Projected regression method for solving fredholm integral equations arising in the analytic continuation problem of quantum physics. Inverse Problems, 33(11):115007, October 2017.

[5] Afef Cherni, Emilie Chouzenoux, and Marc-André Delsuc. PALMA, an improved algorithm for DOSY signal processing. The Analyst, 142(5):772–779, 2017.

[6] Eleonora Di Nezza, Giampiero Palatucci, and Enrico Valdinoci. Hitchhikers guide to the fractional Sobolev spaces. Bulletin des Sciences Mathématiques, 136(5):521–573, 2012.

[7] Michael Nussbaum. Asymptotic equivalence of density estimation and gaussian white noise. The Annals of Statistics, pages 2399–2430, 1996.

[8] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: LSTM cells and network architectures. Neural computation, 31(7):1235–1270, 2019.

[9] Heinz H. Bauschke and Patrick L. Combettes. Convex Analysis and Monotone Operator Theory in Hilbert Spaces, volume 408. Springer, New York, 2011.

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

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

[12] Markus Hegland. Variable Hilbert scales and their interpolation inequalities with applications to Tikhonov regularization. Applicable Analysis, 59(1-4):207–223, 1995.

[13] Bernd Hofmann and Masahiro Yamamoto. Convergence rates for Tikhonov regularization based on range inclusions. Inverse Problems, 21(3):805, 2005.

[14] Emil O. Åkesson and Kyle J. Daun. Parameter selection methods for axisymmetric flame tomography through Tikhonov regularization. Applied optics, 47(3):407–416, 2008.

[15] Kyle J. Daun, Kevin A. Thomson, Fengshan Liu, and Greg J. Smallwood. Deconvolution of axisymmetric flame properties using Tikhonov regularization. Applied optics, 45(19):4638–4646, 2006.

[16] Heinz W. Engl. Regularization of inverse problems. Kluwer Academic Publishers, Dordrecht Boston, 1996.

[17] Per Christian Hansen. The l-curve and its use in the numerical treatment of inverse problems. 1999.

[18] Stephan Antholzer, Markus Haltmeier, and Johannes Schwab. Deep learning for photoacoustic tomography from sparse data. Inverse problems in science and engineering, 27(7):987–1005, 2019.

[19] Andreas Kofler, Markus Haltmeier, Christoph Kolbitsch, Marc Kachelrieß, and Marc Dewey. A u-nets cascade for sparse view computed tomography. In International Workshop on Machine Learning for Medical Image Reconstruction, pages 91–99. Springer, 2018.

[20] Kerstin Hammernik, Teresa Klatzer, Erich Kobler, Michael P. Recht, Daniel K. Sodickson, Thomas Pock, and Florian Knoll. Learning a variational network for reconstruction of accelerated mri data. Magnetic resonance in medicine, 79(6):3055–3071, 2018.

[21] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. Inverse Problems, 33(12):124007, 2017.

[22] Hemant K. Aggarwal, Merry P. Mani, and Mathews Jacob. Modl: Model-based deep learning architecture for inverse problems. IEEE transactions on medical imaging, 38(2):394–405, 2018.

[23] Tim Meinhardt, Michael Moller, Caner Hazirbas, and Daniel Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In Proceedings of the IEEE International Conference on Computer Vision, pages 1781–1790, 2017.

[24] Jean-Christophe Pesquet, Audrey Repetti, Matthieu Terris, and Yves Wiaux. Learning maximally monotone operators for image recovery. arXiv preprint arXiv:2012.13247, 2020.

[25] Marzieh Hasannasab, Johannes Hertrich, Sebastian Neumayer, Gerlind Plonka, Simon Setzer, and Gabriele Steidl. Parseval proximal neural networks. Journal of Fourier Analysis and Applications, 26(4), Jul 2020.

[26] Mathilde Galinier, Mario Prato, Emilie Chouzenoux, and Jean-Christophe Pesquet. A hybrid interior point - deep learning approach for Poisson image deblurring. In 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), pages 1–6. IEEE, September 2020.

[27] Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. NETT: Solving inverse problems with deep neural networks. Inverse Problems, 36(6):065005, 2020.

[28] Tatiana A. Bubba, Mathilde Galinier, Matti Lassas, Marco Prato, Luca Ratti, and Samuli Siltanen. Deep neural networks for inverse problems with pseudodifferential operators: an application to limited-angle tomography. 2021.

[29] Davis Gilton, Greg Ongie, and Rebecca Willett. Neumann networks for linear inverse problems in imaging. IEEE Transactions on Computational Imaging, 6:328–343, 2019.

[30] 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.

[31] Yu Sun, Brendt Wohlberg, and Ulugbek S. Kamilov. An online plug-and-play algorithm for regularized image reconstruction. IEEE Transactions on Computational Imaging, 5(3):395–408, 2019.

[32] Mark Borgerding and Philip Schniter. Onsager-corrected deep learning for sparse linear inverse problems. In 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 227–231. IEEE, 2016.

[33] Kyong Hwan Jin, Michael T. McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. IEEE Transactions on Image Processing, 26(9):4509–4522, 2017.

[34] Carla Bertocchi, Emilie Chouzenoux, Marie-Caroline Corbineau, Jean-Christophe Pesquet, and Marco Prato. Deep unfolding of a proximal interior point method for image restoration. Inverse Problems, 36(3):034005, 2020.

[35] Suresh Kondati Natarajan and Miguel A. Caro. Particle swarm based hyper-parameter optimization for machine learned interatomic potentials. arXiv preprint arXiv:2101.00049, 2020.

[36] Martin Genzel, Jan Macdonald, and Maximilian März. Solving inverse problems with deep neural networks–robustness included? arXiv preprint arXiv:2011.04268, 2020.

[37] Patrick L. Combettes and Jean-Christophe Pesquet. Deep neural network structures solving variational inequalities. Set-Valued and Variational Analysis, pages 1–28, 2020.

[38] Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. SIAM Journal on Mathematics of Data Science, 2(2):529–557, 2020.

[39] Mathieu Serrurier, Franck Mamalet, Alberto González-Sanz, Thibaut Boissin, Jean-Michel Loubes, and Eustasio del Barrio. Achieving robustness in classification using optimal transport with hinge regularization. arXiv preprint arXiv:2006.06520, 2020.

[40] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In International Conference on Machine Learning, pages 854–863. PMLR, 2017.

[41] Rudolf Gorenflo and Francesco Mainardi. Fractional calculus. In Fractals and fractional calculus in continuum mechanics, pages 223–276. Springer, 1997.

[42] Vladimir Dribinski, Alexei Ossadtchi, Vladimir A. Mandelshtam, and Hanna Reisler. Reconstruction of Abel-transformable images: The Gaussian basis-set expansion Abel transform method. Review of Scientific Instruments, 73(7):2634–2642, July 2002.

[43] Sunil Kumar, Amit Kumar, Devendra Kumar, Jagdev Singh, and Arvind Singh. Analytical solution of Abel integral equation arising in astrophysics via laplace transform. Journal of the Egyptian Mathematical Society, 23(1):102–107, 2015.

[44] Igor Podlubny. Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications. Elsevier, 1998.

[45] Ryan M. Evans, Udita N. Katugampola, and David A. Edwards. Applications of fractional calculus in solving Abel-type integral equations: Surface–volume reaction problem. Computers & mathematics with applications, 73(6):1346–1362, 2017.

[46] Rudolf Gorenflo and Sergio Vessella. Abel integral equations, volume 1461. Springer, 1991.

[47] David Gottlieb and Steven A. Orszag. Numerical analysis of spectral methods: theory and applications. SIAM, 1977.

[48] John P. Boyd. Chebyshev and Fourier spectral methods. Courier Corporation, 2001.

[49] Rudolf Gorenflo and Masahiro Yamamoto. Operator theoretic treatment of linear Abel integral equations of first kind. Japan journal of industrial and applied mathematics, 16(1):137–161, 1999.

[50] Diganta Misra. Mish: A self regularized non-monotonic activation function, 2020.

[51] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on MNIST classification task. arXiv preprint arXiv:1804.02763, 2018.

[52] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In T. Leen, T. Dietterich, and V. Tresp, editors, Advances in Neural Information Processing Systems, volume 13, pages 472–478. MIT Press, 2001.

[53] Frank Natterer. Error bounds for Tikhonov regularization in Hilbert scales. Applicable Analysis, 18(1-2):29–37, 1984.

[54] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[55] Robert Piessens and Pierre Verbaeten. Numerical solution of the Abel integral equation. BIT Numerical Mathematics, 13(4):451–457, 1973.

[56] Rajesh K. Pandey, Suraj Suman, Koushlendra K. Singh, and Om P. Singh. An approximate method for Abel inversion using Chebyshev polynomials. Applied Mathematics and Computation, 237:120–132, 2014.

[57] Shuiliang Ma, Gming Hon Gao, Guangjun Zhang, and Lin Wu. A versatile analytical expression for the inverse Abel transform applied to experimental data with noise. Applied Spectroscopy, 62(6):701–707, June 2008.

[58] Shuiliang Ma, Hongming Gao, and Lin Wu. Modified fourier-hankel method based on analysis of errors in Abel inversion using fourier transform techniques. Applied optics, 47(9):1350–1357, 2008.

[59] Milan Kalal and Keith Nugent. Abel inversion using fast Fourier transforms. Applied optics, 27(10):1956–1959, 1988.

[60] Maarten V. de Hoop and Joonas Ilmavirta. Abel transforms with low regularity with applications to x-ray tomography on spherically symmetric manifolds. Inverse Problems, 33(12):124003, November 2017.

[61] Pankaj S. Kolhe and Ajay K. Agrawal. Abel inversion of deflectometric data: comparison of accuracy and noise propagation of existing techniques. Applied Optics, 48(20):3894–3902, Jul 2009.

[62] David L. Donoho. Nonlinear solution of linear inverse problems by wavelet–vaguelette decomposition. Applied and Computational Harmonic Analysis, 2(2):101–126, 1995.

[63] Eric W. Hansen and Phaih-Lan Law. Recursive methods for computing the Abel transform and its inverse. JOSA A, 2(4):510–520, 1985.

[64] Fernando Nunes, Jorge Santos, and Emilia M. Manso. Recursive algorithm for fast evaluation of the abel inversion integral in broadband reflectometry. Review of Scientific Instruments, 70(1):1047–1050, January 1999.