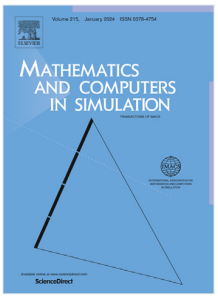
Journal Pre-proof



Data-driven fractional algebraic system solver

Emmanuel Lorin, Howl Nhan

PII: S0378-4754(25)00079-5
DOI: https://doi.org/10.1016/j.matcom.2025.03.004
Reference: MATCOM 6777

To appear in: *Mathematics and Computers in Simulation*

Received date : 26 September 2024
Revised date : 5 March 2025
Accepted date : 8 March 2025

Please cite this article as: E. Lorin and H. Nhan, Data-driven fractional algebraic system solver, *Mathematics and Computers in Simulation* (2025), doi: https://doi.org/10.1016/j.matcom.2025.03.004.

Revised manuscript (clean version)

Data-driven fractional algebraic system solver

Emmanuel LORIN[a,b,*], Howl NHAN[a]

[a] *School of Mathematics and Statistics, Carleton University, Ottawa, Canada, K1S 5B6*
[b] *Centre de Recherches Mathématiques, Université de Montréal, Montréal, Canada, H3T 1J4*

Abstract

In this paper, we explore a class of (data-driven/supervised) neural network-based algorithms for solving linear and fractional algebraic systems. The latter are reformulated as dynamical systems, and solved using neural networks. Some mathematical proprieties of the derived algorithms are proposed, as well as several illustrating numerical experiments.

Keywords: Fractional linear system; neural networks; scientific machine learning.

1. Introduction

We are interested in the numerical computation of fractional linear systems (FLS)

$$A^{\alpha}\mathbf{x} = \mathbf{b}, \tag{1}$$

for some $\alpha \in \mathbb{R}_+\backslash\{0\}$, where $A \in \mathrm{GL}(\mathbb{R}^n)$ and $\mathbf{b} \in \mathbb{C}^n$. We assume that A is a diagonalizable matrix in $\mathbb{C}\backslash\mathbb{R}_-$. The proposed approach will then be extended to generalized fractional linear systems $\sum_{i=1}^{N} A^{\alpha_i}\mathbf{x} = \mathbf{b}$ with $\alpha_i \in \mathbb{R}_+\backslash\{0\}$. This work, along with previous studies, is primarily driven by advancements in fractional partial differential equations, including the fractional Poisson equation, which can be approximated using fractional linear systems of the form $A^{\alpha}x = b$ [31]. Among fractional Laplacian-based models, the most extensively explored are time-dependent fractional heat equations and fractional Schrödinger equations (FSE). This work along with existing others is largely motivated by the development of fractional partial differential equations, including fractional Poisson equation which can be approximated by fractional linear systems $A^{\alpha}x = b$ [31]. The space-FSE was originally formulated by Laskin [30]. FSEs usually characterize nonlocal phenomena in quantum physics, offering deeper insights into quantum behaviors influenced by long-range interactions and time-dependent processes spanning multiple scales [1, 29]. Specifically, they depict quantum fluids of light [13], boson stars [7], and polariton condensates [36].
We develop here machine learning techniques and more specifically the optimization of data-driven neural networks to approximate the solution to (1), in particular by reformulating (1)

*Corresponding author
Email addresses: elorin@math.carleton.ca (Emmanuel LORIN), HowlNhan@cmail.carleton.ca (Howl NHAN)

as an "equivalent" dynamical system. The proposed approach is mainly based on Physics Informed Neural Networks (PINN) [37, 35, 44, 27, 2]. The objective of the work is to explore these new scientific machine learning techniques and to provide a proof-of-concept, rather than to compare the performance with standard techniques. A PINN-like solution to a differential system is a (possibly data-driven) neural network parameterized thanks to the minimization of the norm of the residual of the system, evaluated at some (randomly chosen) learning nodes. This approach has recently gained a huge interest in various scientific communities thanks to their simplicity, their nice mathematical and computational properties coming in particular from automatic differentiation and efficient stochastic gradient descent methods [10]. Although standard differential solvers are usually more efficient than neural network ones, in some research fields such as quantum chemistry (more specifically the computation of eigenvalues for high dimensional Hamiltonians) it was recently established that neural network-based techniques can out-perform traditional ones [11, 12].
The neural network-based ODE/PDE approach proposed in this work have potentially multiple interests:

- The differential reformulation of algebraic systems is not a new idea, and is known to be less efficient than standard iterative or direct ones for (sparse) linear and fractional systems. Indeed, explicit differential system solvers are usually constrained by stability conditions (implying potentially a large number of "time iterations"), while implicit solvers require the computation of intermediate linear systems. Automatic differentiation allows to avoid the (direct) discretization of differential operators.

- Efficient and highly parallelizable stochastic gradient descent methods which consist in successively minimizing an objective/loss function in high dimension, with successive computations in lower dimensional subspaces [10].

In this paper, we will also specifically focus on the data-driven training of the neural networks using solutions to (1) for a finite set of α's.

Let us here recall the basics of neural networks. The latter are parameterized functions constructed as the composition of i) discrete convoluted linear functions, and ii) nonlinear functions. In a d-dimensional space, we introduce $l \in \mathbb{N}$, $n_i \in \mathbb{N}$ for $i = 0, 1, \ldots, l$, with $n_0 = d$. We denote by $\boldsymbol{\theta} \in \prod_{i=1}^{l} \mathbb{R}^{n_i \times (n_{i-1}+1)}$ the neural network parameters such that: $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i\}_i$ with $\boldsymbol{\theta}_i = (\mathbf{w}^i, \mathbf{b}^i) \in \mathbb{R}^{n_i \times n_{i-1}} \times \mathbb{R}^{n_i}$ and $i = 1, \cdots, l$. A fully connected neural network $\widehat{\mathbf{f}}$ in $\mathbb{R}^d$ is a function from $\prod_{i=1}^{l} \mathbb{R}^{n_i \times (n_{i-1}+1)} \times \mathbb{R}^d$ to $\mathbb{R}^{n_l}$ such that

$$\widehat{\mathbf{f}}(\boldsymbol{\theta}; \tau) \quad = \mathbf{w}^l \sigma(\mathbf{w}^{l-1} \sigma(\mathbf{w}^2 \sigma(\mathbf{w}^1 \tau + \mathbf{b}^1) + \mathbf{b}^2) \cdots + \mathbf{b}^{l-1}) + \mathbf{b}^l, \tag{2}$$

where σ is a so-called activation function. We refer to [22, 3] for details on neural networks. In this manuscript, the activation functions will be taken smooth. The proposed algorithms largely rely on the following convergence results of neural network-based algorithms for differential equations, see [38] and more generally the celebrated Cybenko's Universal Approximation theorem [15].

The first part of this paper is devoted to the derivation of (data-driven) computational methods for solving fractional linear systems using neural networks. We then establish some mathematical properties. In the last part, we propose some numerical experiments to illustrate the methodology.

2. Neural network-based FLS solver

This section is devoted to the construction of approximate solutions to FLS for a continuous range of α's by data-driven neural networks (Machine Learning), also possibly combined with a neural network-based ODE solver (Scientific Machine Learning).

2.1. Linear systems

One of the key-principles of the proposed approach is to search for a neural network $\widehat{\mathbf{f}}(\boldsymbol{\theta};\tau)$ approximating the solution to (3), optimizing the parameters $\boldsymbol{\theta}$ by minimizing the residual of the linear system under consideration. We first present the algorithm for linear systems then for (generalized) fractional linear systems. The proposed approach for solving linear system (LS)

$$A\mathbf{x} = \mathbf{b}, \tag{3}$$

is based on its reformulation in the form of a dynamical system [14] of the form

$$\mathbf{y}'(\tau) = \mathbf{b} - A\mathbf{y}(\tau), \qquad \mathbf{y}(0) = \mathbf{b}\,, \tag{4}$$

where $A \in \mathrm{GL}_n(\mathbb{R}^n)$ is symmetric positive definite. It easy to see that $\mathbf{y}(\tau) \to_{\tau\to+\infty} A^{-1}\mathbf{b}$. The downside of this approach is that it potentially requires "large time" computations, in particular if the smallest eigenvalue is close to zero. In practice, it can be convenient to choose the neural network $\widehat{\mathbf{y}}(\boldsymbol{\theta};\tau)$ such that $\widehat{\mathbf{y}}(\boldsymbol{\theta};0) = \mathbf{b}$ by taking it in the form $f_{\tau_0}(\tau)\mathbf{b} + \widehat{\mathbf{y}}(\boldsymbol{\theta};\tau)$ with $f_{\tau_0}(0) = 1$ and $f_{\tau_0}(\tau) = 0$ for $\tau \geqslant \tau_0 > 0$. In the following we denote by $\widehat{\mathbf{y}}_\tau$, the derivative of $\widehat{\mathbf{y}}$ with respect to τ. The optimized parameters are obtained by solving the problem

$$\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Omega}}\mathcal{L}^{\mathrm{LS}}(\boldsymbol{\theta})^2\,, \tag{5}$$

where $\boldsymbol{\Omega}$ is a set of admissible parameters, and such that

$$\mathcal{L}^{\mathrm{LS}}(\boldsymbol{\theta})^2 \;=\; \lambda_1\|\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta};\cdot) + A\widehat{\mathbf{y}}(\boldsymbol{\theta};\cdot) - \mathbf{b}\|^2_{L^2([0,T];\mathbb{R}^n)} + \lambda_2|\widehat{\mathbf{y}}(\boldsymbol{\theta};0) - \mathbf{b}|_2^2, \tag{6}$$

for some positive hyper-parameters λ_1 and λ_2, and where we have denoted by $|\cdot|_2$ the ℓ_2-norm in $\mathbb{R}^n$. Practically, introducing $\{\tau_j\}_j$ a sequence of (randomly chosen or quadrature) positive real numbers, we implement a discrete version of (6) called *training* loss function:

$$\mathcal{L}_T^{\mathrm{LS}}(\boldsymbol{\theta})^2 \;=\; \frac{\lambda_1}{n_\tau}\sum_{j=1}^{n_\tau}|\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta};\tau_j) + A\widehat{\mathbf{y}}(\boldsymbol{\theta};\tau_j) - \mathbf{b}|_2^2 + \lambda_2|\widehat{\mathbf{y}}(\boldsymbol{\theta};0) - \mathbf{b}|_2^2\,, \tag{7}$$

which is minimized using a gradient descent-like method for $k \geqslant 0$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \nu_k \nabla \mathcal{L}_T^{\mathrm{LS}}(\boldsymbol{\theta}_k)^2 \,,$$

where $\boldsymbol{\theta}_0$ and the positive learning rate $\{\nu_k\}_k$ are given.
In summary (3) is transformed into a dynamical system itself solved using an optimization algorithm. Several levels of parallelization are hence possible (matrix-vector product, stochastic gradient descent).

Remark 2.1. *A more direct neural network-based method can be derived by simply minimizing* $\mathcal{L}^{DLS}(\boldsymbol{\theta})^2 = |A\widehat{\mathbf{y}}(\boldsymbol{\theta}) - \mathbf{b}|_2^2$. *This leads to a Richardson-like iterative solver (taking* $\widehat{\mathbf{y}}(\boldsymbol{\theta}) = \boldsymbol{\theta}$*) where* $\mathcal{L}^{DLS}(\boldsymbol{\theta})^2 = -2\mathbf{b}^T A\widehat{\mathbf{y}}(\boldsymbol{\theta}) - \widehat{\mathbf{y}}(\boldsymbol{\theta})^T A^T A\widehat{\mathbf{y}}(\boldsymbol{\theta}) + |\mathbf{b}|_2^2$, *with for* $k \geqslant 0$ *and* $\boldsymbol{\theta}_0$ *given, and* $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - 2\nu_k \big[A\widehat{\mathbf{y}}(\boldsymbol{\theta}_k) - \mathbf{b}\big]^T A \nabla_{\boldsymbol{\theta}} \widehat{\mathbf{y}}(\boldsymbol{\theta}_k)$.

2.2. Differential system-based FLS solver

We are now interested in the numerical computation of (1) assuming that $\alpha \in \mathbb{R}_+ \backslash \{0\}$, where $A \in \mathbb{R}^{n\times n}$ is a diagonalizable matrix with eigenvalues in $\mathbb{C}\backslash\mathbb{R}_-$, and $\mathbf{b} \in \mathbb{C}^n$ are given. Let us recall that A^α can be defined using the following Cauchy integral (see e.g. Theorem 6.2.28 from [23])

$$A^\alpha \;=\; (2\pi \mathtt{i})^{-1} A^\ell \int_{\Gamma_A} z^{\alpha-k} (zI - A)^{-1} dz \,, \tag{8}$$

for $\ell \geqslant 0$, where Γ_A is a contour in the complex plane *enclosing the spectrum of the matrix* A. It is easy to show to check that the n-dimensional differential system

$$\mathbf{y}'(\tau) = -\alpha(A - I)\big(I + \tau(A - I)\big)^{-1}\mathbf{y}(\tau), \qquad \mathbf{y}(0) = \mathbf{b} \,, \tag{9}$$

admits a solution $\mathbf{y}(\tau) = \big(I + \tau(A - I)\big)^{-\alpha}\mathbf{b}$ such that $\mathbf{y}(1) = A^{-\alpha}\mathbf{b}$, where I is the identity matrix in $\mathbb{R}^{n\times n}$ (see [20, 19, 21, 33]). We can then compute $\mathbf{x} = A^{-\alpha}\mathbf{b}$ by approximating (9). The approximation of (9) by standard numerical differential system solvers requires the computation of linear systems at each time step (see [4]) in order to evaluate $\mathbf{z}$ by solving $\big(I + \tau(A - I)\big)\mathbf{z}(\tau) = \mathbf{y}(\tau)$. In the PINN framework, we introduce 2 neural networks $\widehat{\mathbf{z}}$ and $\widehat{\mathbf{y}}$ with corresponding parameters $\boldsymbol{\theta}_\mathbf{z}$ and $\boldsymbol{\theta}_\mathbf{y}$ such that

$$\begin{array}{rcl} \widehat{\mathbf{y}}_\tau(\boldsymbol{\theta}_\mathbf{y};\tau) + \alpha(A - I)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z};\tau) & = & \mathbf{0} \,, \\ \widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y};\tau) - \big(I + \tau(A - I)\big)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z};\tau) & = & \mathbf{0} \,, \end{array}$$

where $\widehat{\mathbf{y}}_\tau$ denoted the partial derivative of $\widehat{\mathbf{y}}$ with respect to τ. Denoting $\boldsymbol{\Theta} = (\boldsymbol{\theta}_\mathbf{z}, \boldsymbol{\theta}_\mathbf{y})$, we then minimize the following loss function

$$\begin{array}{rcl} \mathcal{L}^{\mathrm{FLS}}(\boldsymbol{\Theta})^2 & = & \lambda_1 \|\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta}_\mathbf{y};\cdot) + \alpha(A - I)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z};\cdot)\|^2_{L^2([0,1];\mathbb{R}^n)} \\ & & +\lambda_2 \|\widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y};\cdot) - \big(I + \tau(A - I)\big)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z},\cdot)\|^2_{L^2([0,1];\mathbb{R}^n)} \\ & & +\lambda_3 |\widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y};0) - \mathbf{b}|_2^2 \,, \end{array} \tag{10}$$

where $\{\lambda\}_i$ with $i = 1, 2, 3$ are positive free parameters. Practically, we introduce $\{\tau_j\}_j$ a sequence of real numbers in $[0, 1]$ and define the training loss $\mathcal{L}_T^{\rm FLS}$ as follows:

$$\begin{aligned}\mathcal{L}_T^{\rm FLS}(\boldsymbol{\Theta})^2 \;=\; & \frac{\lambda_1}{n_\tau}\sum_{j=1}^{n_\tau}|\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta}_{\mathbf{y}};\tau_j)+\alpha(A-I)\widehat{\mathbf{z}}(\boldsymbol{\theta}_{\mathbf{z}};\tau_j)|_2^2 \\ & +\frac{\lambda_2}{n_\tau}\sum_{j=1}^{n_\tau}|\widehat{\mathbf{y}}(\boldsymbol{\theta}_{\mathbf{y}};\tau_j)+\big(I+\tau_j(A-I)\big)\widehat{\mathbf{z}}(\boldsymbol{\theta}_{\mathbf{z}};\tau_j)-\mathbf{b}|_2^2 \\ & +\lambda_3|\widehat{\mathbf{y}}(\boldsymbol{\theta}_{\mathbf{y}};0)-\mathbf{b}|_2^2\,.\end{aligned} \tag{11}$$

The FLS solver hence consists in minimizing (11), using a gradient descent algorithm reads

$$\boldsymbol{\Theta}_{k+1} = \boldsymbol{\Theta}_k - \nu_k \nabla \mathcal{L}_T^{\rm FLS}(\boldsymbol{\Theta}_k)^2\,,$$

with $\boldsymbol{\Theta}_0$ and $\{\nu_k\}_k$ given. Notice that the solution to standard linear systems $A\mathbf{x} = \mathbf{b}$ can also be obtained by solving (9) with $\alpha = 1$. Let us also mention that H^1-norms could also be used to define the loss functions.
This approach is developed for a fixed value of α. Below, we consider $\alpha-$dependent neural networks for approximating the solution to (1) for an interval in α.

2.3. Data-driven FLS solver

Standard machine learning is here discussed to compute the solution to algebraic linear systems for a continuous range of power α in (1), and supervised by a finite set $\{\mathbf{x}_1, \cdots, \mathbf{x}_p\}$ of $p > 1$ pre-computed (approximate) solutions to $A^{\alpha_m}\mathbf{x}_m = \mathbf{b}$ for $m = 1, \cdots, p$. More specifically, we intend to learn the vector-valued function $\mathbf{y}_\mathbf{b} : \alpha \in \Lambda := [\alpha_1, \alpha_p] \mapsto A^{-\alpha}\mathbf{b}$. In this goal we search for a vector valued neural network $\widehat{\mathbf{y}}_\mathbf{b}(\theta;\alpha)$ such that the data driven (DD) loss is defined by

$$\mathcal{L}_{\rm d}^{\rm DD}(\boldsymbol{\theta})^2 = \Delta\alpha\sum_{m=1}^{p}|\widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta};\alpha_m)-\mathbf{x}_m|^2_{\ell^2(\mathbb{R}^n)}\,, \tag{12}$$

where $\ell^2(\mathbb{R}^n)$ is the ℓ^2-norm on $\mathbb{R}^n$. Setting $\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}}\mathcal{L}_{\rm d}^{\rm DD}(\boldsymbol{\theta})^2$, we then define $\alpha \in \Lambda \mapsto \widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta}^*;\alpha)$ approximating $\mathbf{y}_\mathbf{b}$. The above data-driven approach can be improved as proposed below.
Improved data-driven solver. In order to accelerate and improve the training, we first define F from $\mathbb{R}$ to $\mathbb{R}^{n\times n}$

$$F : \alpha \mapsto A^{-\alpha}, \quad \frac{dF}{d\alpha} : \alpha \mapsto -\ln(A)A^{-\alpha}\,.$$

Assume given the following data $\{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_p, \mathbf{y}_p)\}$ solutions to $A^{\alpha_m}\mathbf{x}_m = \mathbf{b}$ and $\mathbf{y}_m = -\ln(A)\mathbf{x}_m$, for $m = 1, \cdots, p$. We search for a vector valued neural network $\widehat{\mathbf{y}}(\theta;\alpha)$ such that the data driven (DD) loss function is defined by

$$\mathcal{L}_{\rm i}^{\rm DD}(\boldsymbol{\theta})^2 = \Delta\alpha\sum_{m=1}^{p}\big\{|\widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta};\alpha_m)-\mathbf{x}_m|^2_{\ell^2(\mathbb{R}^n)} + |\partial_\alpha\widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta};\alpha_m)-\mathbf{y}_m|^2_{\ell^2(\mathbb{R}^n)}\big\}\,. \tag{13}$$

Setting $\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}}\mathcal{L}_{\rm i}^{\rm DD}(\boldsymbol{\theta})^2$, we then defined $\alpha \mapsto \widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta}^*;\alpha)$ approximating $\mathbf{y}_\mathbf{b}$.

2.4. Data-driven differential system-based FLS solver

The two independent methods developed in Subsections 2.2 and 2.3 can actually be combined leading to what is commonly called the PINN method [37, 35, 44, 27, 2]. It consists in searching a $(\tau,\alpha)-$ dependent neural network driven by data, and which is optimized by minimizing the residual of the parameterized differential system under consideration. Setting $\Lambda=[\alpha_1,\alpha_p]$ with $\alpha_1<\alpha_p$ and $D:=[0,1]\times[\alpha_1,\alpha]$, we search for $\mathbf{y}:(\tau,\alpha)\in D\mapsto\mathbf{y}(\tau,\alpha)$ solution to

$$\mathbf{y}'(\tau,\alpha)=-\alpha(A-I)\big(I+\tau(A-I)\big)^{-1}\mathbf{y}(\tau,\alpha),\qquad \mathbf{y}(0,\alpha)=\mathbf{b}\,. \tag{14}$$

In this goal, we search for $\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta}_\mathbf{y};\tau,\alpha)$ and $\widehat{\mathbf{z}}_\tau(\boldsymbol{\theta}_\mathbf{z};\tau,\alpha)$ for $(\tau,\alpha)\in[0,1]\times[\alpha_1,\alpha_p]$ using the following algorithms

- We assume given $(\mathbf{x}_1,\mathbf{y}_1),\cdots,(\mathbf{x}_p,\mathbf{y}_p)$ (approximate) solutions to $A^{\alpha_m}\mathbf{x}_m=\mathbf{b}$ and $\mathbf{y}_m=-\ln(A)\mathbf{x}_m$ for $m=1,\cdots,p$.
- We minimize

$$\begin{array}{rcl}\mathcal{L}^{\mathrm{FLS}}(\boldsymbol{\Theta})^2 &=& \lambda_1\|\widehat{\mathbf{y}}_\tau(\boldsymbol{\theta}_\mathbf{y};\cdot,\cdot)+\alpha(A-I)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z};\cdot,\cdot)\|^2_{L^2(D;\mathbb{R}^n)}\\ &&+\lambda_2\|\widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y};\cdot,\cdot)-\big(I+\tau(A-I)\big)\widehat{\mathbf{z}}(\boldsymbol{\theta}_\mathbf{z};\cdot,\cdot)\|^2_{L^2(D;\mathbb{R}^n)}\\ &&+\lambda_3\|\widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y},0,\alpha)-\mathbf{b}\|^2_{L^2(\Lambda;\mathbb{R}^n)}\\ &&+\lambda_4\sum_{m=1}^p\left\{|\widehat{\mathbf{y}}(\boldsymbol{\theta}_\mathbf{y};1,\alpha_m)-\mathbf{x}_m|^2_{\ell^2(\mathbb{R}^n)}+|\partial_\alpha\widehat{\mathbf{y}}_\mathbf{b}(\boldsymbol{\theta}_\mathbf{y};1,\alpha_m)-\mathbf{y}_m|^2_{\ell^2(\mathbb{R}^n)}\right\}.\end{array} \tag{15}$$

Then denoting $\boldsymbol{\theta}^*=\mathrm{argmin}_{\boldsymbol{\theta}}\mathcal{L}^{\mathrm{DD}}(\boldsymbol{\theta})^2$, $\alpha\in\Lambda\mapsto\widehat{\mathbf{y}}(\boldsymbol{\theta}^*;1,\alpha)$ represents an approximate solution to $\alpha\in\Lambda\mapsto A^{-\alpha}\mathbf{b}$.

3. Generalized FLS

The ideas developed above can be actually be extended to generalized fractional linear systems (GFLS) $\sum_{i=1}^N A^{\alpha_i}\mathbf{x}=\mathbf{b}$ with $\alpha_i\in\mathbb{R}_+\backslash\{0\}$.

3.1. Mathematical setting

In this paper, we focus on systems of the form $(N=2)$:

$$(A^\alpha+A^\beta)\mathbf{x}=\mathbf{b}, \tag{16}$$

for some $\alpha,\,\beta\in\mathbb{R}_+\backslash\{0\}$, where $A\in\mathrm{GL}(\mathbb{R}^n)$ and $\mathbf{b}\in\mathbb{R}^n$, and assuming again that A is a diagonalizable matrix in $\mathbb{C}\backslash\mathbb{R}_-$. This very computational complex problem was studied in particular in [5]. Simple iterative (gradient) solvers for solving GFLS (18) typically reads for $k\geqslant 0$:

$$\mathbf{x}_{k+1}=\mathbf{x}_k+\rho_k(\mathbf{b}-A^\alpha\mathbf{x}_k-A^\beta\mathbf{x}_k)\,,$$

with some $\{\rho_k\}_k$ given. Observe that each gradient iteration requires computations similar to FLS. In order to solve this problem in a differential form, we first rewrite (16) as follows

$$A^\delta(A^\gamma+A^{-\gamma})\mathbf{x}=\mathbf{b},$$

with $\delta = (\alpha+\beta)/2$ and $\gamma = (\alpha-\beta)/2$. Hence the solution to (16) is solved in 2 steps: i) $A^{\delta}\mathbf{c} = \mathbf{b}$ (see Subsection 2.4), and ii) $(A^{\gamma} + A^{-\gamma})\mathbf{x} = \mathbf{c}$. Let us then focus in this subsection on the computation of

$$(A^{\gamma} + A^{-\gamma})\mathbf{x} = \mathbf{c}\,. \tag{17}$$

3.2. PDE-based Generalized FLS solver

It was proven in [5] that the solution to (17) can be obtained by solving a first order hyperbolic system.

Proposition 3.1. *For A diagonalizable in $\mathbb{R}_+$ and $\gamma \in \mathbb{R}_+$, the solution to* (17) *is given by* $\mathbf{z}(1/2, 1/2)$*, where* $\mathbf{z}$ *satisfies*

$$\partial_t \mathbf{z}(x,t) + A^{-\gamma}\partial_x \mathbf{z}(x,t) = \mathbf{0}\,, \tag{18}$$

for $(x,t) \in [0,1/2] \times [0,1/2]$ *with the following initial condition*

$$\mathbf{z}(x,0) = \frac{1}{2}((1-x)I + xA^{\gamma})^{-1}\mathbf{c}. \tag{19}$$

The explicit space/time discretization of (18) are restricted by stability conditions which deteriorate the overall efficiency of the computation to (17), due to the computation of a (large) number of intermediate FLS. We propose a neural network version for solving (18) by introducing 3 neural networks $\widehat{\mathbf{y}}(\boldsymbol{\theta}_{\mathbf{y}}; x, t)$, $\widehat{\mathbf{z}}(\boldsymbol{\theta}_{\mathbf{z}}; x, t, \tau)$ and $\mathbf{L}(\boldsymbol{\theta}_{\mathbf{L}}; x, t, \tau)$ taking their values in $\mathbb{R}^n$ and such that $\widehat{\mathbf{y}}(\boldsymbol{\theta}_{\mathbf{y}}, 1/2, 1/2)$ is the approximate solution to (18). *For the sake of notation readiness, hereafter we will omit the dependence in the parameters in the neural networks.* In order to evaluate the approximate solution $\widehat{\mathbf{y}}(1/2, 1/2)$ to (17), we then solve

$$\begin{aligned} \widehat{\mathbf{y}}_t(x,t) + \mathbf{L}(x,t,1) &= \mathbf{0}\,, \\ \mathbf{L}_{\tau}(x,t,\tau) + \gamma(A-I)\widehat{\mathbf{z}}(x,t,\tau) &= \mathbf{0}\,, \\ \mathbf{L}(x,t,\tau) - \big(I + \tau(A-I)\big)\widehat{\mathbf{z}}(x,t,\tau) &= \mathbf{0}\,, \end{aligned}$$

with $\mathbf{L}(x,t,0) = \widehat{\mathbf{y}}_x(x,t)$ and $\widehat{\mathbf{z}}(x,t,0) = \widehat{\mathbf{y}}_x(x,t)$, and where $\widehat{\mathbf{y}}_x$ denotes the partial derivative of $\widehat{\mathbf{y}}$. In addition, we need to initially impose

$$\widehat{\mathbf{y}}(x,0) = \frac{1}{2}((1-x)I + xA^{\gamma})^{-1}\mathbf{c}\,. \tag{20}$$

Condition (20) is encoded using 2 additional neural networks $\mathbf{P}$ and $\mathbf{Q}$

$$\begin{aligned} (1-x)\widehat{\mathbf{y}}(x,0) + x\mathbf{P}(x,1) &= \mathbf{c}/2\,, \\ \mathbf{P}_{\tau}(x,\tau) - \gamma(A-I)\mathbf{Q}(x,\tau) &= \mathbf{0}\,, \\ \mathbf{P}(x,\tau) - \big(I + \tau(A-I)\big)\mathbf{Q}(x,\tau) &= \mathbf{0}\,, \end{aligned}$$

with $\mathbf{P}(x,0) = \widehat{\mathbf{y}}(x,0)$ and $\mathbf{Q}(x,0) = \widehat{\mathbf{y}}(x,0)$. Notice in particular that $\mathbf{L}(x,t,1)$ provides an approximation to $A^{-\gamma}\widehat{\mathbf{y}}_x(x,t)$. Denoting $\boldsymbol{\Theta} = (\boldsymbol{\theta}_{\mathbf{L}}, \boldsymbol{\theta}_{\mathbf{z}}, \boldsymbol{\theta}_{\mathbf{y}}, \boldsymbol{\theta}_{\mathbf{P}}, \boldsymbol{\theta}_{\mathbf{Q}})$, $D_x = [0,1/2]$, $D_{x,t} =$

$[0,1/2]\times[0,1/2]$, $D_{x,\tau}=[0,1/2]\times[0,1]$ and $D_{x,t,\tau}=[0,1/2]\times[0,1/2]\times[0,1]$, the GFLS algorithm consists in minimizing the following loss function

$$\begin{aligned}\mathcal{L}^{\mathrm{GFLS}}(\boldsymbol{\Theta})^2 \;=\;& \lambda_1\|\widehat{\mathbf{y}}_t(\cdot,\cdot)+\mathbf{L}(\cdot,\cdot,1)\|^2_{L^2(D_{x,t};\mathbb{R}^n)}\\ &+\lambda_2\|\mathbf{L}_\tau(\cdot,\cdot,\cdot)+\gamma(A-I)\widehat{\mathbf{z}}(\cdot,\cdot,\cdot)\|^2_{L^2(D_{x,t,\tau};\mathbb{R}^n)}\\ &+\lambda_3\|\mathbf{L}(\cdot,\cdot,\cdot)-\big(I+\tau(A-I)\big)\widehat{\mathbf{z}}(\cdot,\cdot,\cdot)\|^2_{L^2(D_{x,t,\tau};\mathbb{R}^n)}\\ &+\lambda_4\big(\|\mathbf{L}(\cdot,\cdot,0)-\widehat{\mathbf{y}}_x(\cdot,\cdot)\|^2_{L^2(D_{x,t};\mathbb{R}^n)}+\|\widehat{\mathbf{z}}(\cdot,\cdot,0)-\widehat{\mathbf{y}}_x(\cdot,\cdot)\|^2_{L^2(D_{x,t};\mathbb{R}^n)}\big)\\ &+\lambda_5\|2(1-x)\widehat{\mathbf{y}}(\cdot,0)+2x\mathbf{P}(\cdot,1)-\mathbf{c}\|^2_{L^2(D_x;\mathbb{R}^n)}\\ &+\lambda_5\|\mathbf{P}_\tau(\cdot,\cdot)-\gamma(A-I)\mathbf{Q}(\cdot,\cdot)\|^2_{L^2(D_{x,\tau};\mathbb{R}^n)}\\ &+\lambda_6\|\mathbf{P}(\cdot,\cdot)-\big(I+\tau(A-I)\big)\mathbf{Q}(\cdot,\cdot)\|^2_{L^2(D_{x,\tau};\mathbb{R}^n)}\\ &+\lambda_7\big(\|\mathbf{P}(\cdot,0)-\widehat{\mathbf{y}}(\cdot,0)\|^2_{L^2(D_x;\mathbb{R}^n)}+\|\mathbf{Q}(\cdot,0)-\widehat{\mathbf{y}}(\cdot,0)\|^2_{L^2(D_x;\mathbb{R}^n)}\big)\,,\end{aligned}\tag{21}$$

where $\{\lambda_i\}_i$ are free positive hyper-parameters. Practically, we hence introduce sequence of randomly chosen points $\{x_j,\tau_j,t_k\}_{i,j,k}$ to get an discrete version of (11). The above approach may seem complex, however recall that the computation of GFLS is a far from trivial problem, see [5].

3.3. Data-driven GFLS solver

We propose to extend the principle presented in Section 2.3. We set

$$G:(\alpha,\beta)\mapsto(A^\alpha+A^\beta)^{-1}\,.$$

More specifically, we intend to approximate the vector-valued function $\mathbf{z}_\mathbf{b}:(\alpha,\beta)\mapsto(A^\alpha+A^\beta)^{-1}\mathbf{b}$ for $(\alpha,\beta)\in\Gamma:=[\alpha_1,\alpha_p]\times[\beta_1,\beta_{p'}]$ for $p,p'>1$.
We assume known a finite set of $p\times p'$ computed solutions $\{\mathbf{x}_{mm'}\}$ to $(A^{\alpha_m}+A^{\beta_{m'}})\mathbf{x}_{mm'}=\mathbf{b}$, for $m=1,\cdots,p$ and $m'=1,\cdots,p'$. We then search for a vector valued neural network $\widehat{\mathbf{z}}_\mathbf{b}(\theta;\alpha,\beta)$ such that the data driven (DD) loss is defined by

$$\mathcal{L}_3^{\mathrm{DD}}(\boldsymbol{\theta})^2=\Delta\alpha\Delta\beta\sum_{m=1}^{p}\sum_{m'=1}^{p'}|\widehat{\mathbf{z}}_\mathbf{b}(\boldsymbol{\theta};\alpha_m,\beta_{m'})-\mathbf{x}_{mm'}|^2_{\ell^2(\mathbb{R}^n)}\,.\tag{22}$$

Setting $\boldsymbol{\theta}^*=\mathrm{argmin}_{\boldsymbol{\theta}}\mathcal{L}_3^{\mathrm{DD}}(\boldsymbol{\theta})^2$, we then define $(\alpha,\beta)\mapsto\widehat{\mathbf{z}}_\mathbf{b}(\boldsymbol{\theta}^*;\alpha,\beta)$ the approximate solution $\mathbf{z}_\mathbf{b}$.
As proposed in Subsection 2.4, it is possible to develop a data-driven optimization using the PDE approach from Subsection 3.2.

3.4. Analysis

We provide some analytical properties of the presented algorithms more specifically the one based on the residual.
Computational complexity. We here discuss the computational complexity of the residual approach and compare it to standard solvers. We assume that i) $\widehat{\mathbf{y}}$ possesses $p=\pi_{i=1}^{l}n_i(n_{i-1}+1)$ scalar parameters to optimize for each component, ii) that $k_{\mathrm{GD}}^{(\mathrm{L})}$ (resp. $k_{\mathrm{GD}}^{(\mathrm{F})}$, resp. $k_{\mathrm{GD}}^{(\mathrm{G})}$) optimization iterations are necessary to minimize $\mathcal{L}^{(\mathrm{LS})}$ (resp. $\mathcal{L}_{\mathrm{d}}^{(\mathrm{FLS})}$, resp. $\mathcal{L}^{(\mathrm{GFLS})}$), and iii) we denote by N_t (resp. N_tN_τ) the total number of training points for LS and FLS (resp.

Solver	**CC-LS**	**CC-FLS**	**CC-GFLS**
Iterative	$\mathcal{O}(k_{\rm IT}^{(\rm L)} n^2)$	$\mathcal{O}(k_{\rm IT}^{(\rm F)} n^2)$	$\mathcal{O}(k_{\rm IT}^{(\rm F)} k_{\rm IT}^{(\rm L)} n^2)$
DE	$\mathcal{O}(N_t n^2)$	$\mathcal{O}(N_t n^3)$	$\mathcal{O}(N_t N_\tau n^3)$
N.N.	$\mathcal{O}(k_{\rm GD}^{(\rm L)}(pnN_t + n^2))$	$\mathcal{O}(k_{\rm GD}^{(\rm F)}(pnN_t + n^2))$	$\mathcal{O}(k_{\rm GD}^{(\rm G)}(pnN_t N_\tau + n^2))$

Table 1: Computational complexity for solving LS, FLS, GFLS.

GFLS). Standard approaches for solving (3) (resp. (1)) using (4) (resp. (9)) are assumed to require N_t time-iterations with an explicit solver. We also denote by N_τ the number of discretization points in τ for (18). Let us finally denote by $k_{\rm IT}^{(\rm L)}$ (resp. $k_{\rm IT}^{(\rm F)}$, resp. $k_{\rm IT}^{(\rm G)}$) the number of iterations for standard LS (resp. FLS, resp. GFLS) iterative solvers. We easily prove the following proposition.

Proposition 3.2. *Assume that $A \in GL_n(\mathbb{R})$ is a full matrix with spectrum in $\mathbb{R}_+\backslash\{0\}$. We summarize the computational complexity of the different solvers in the following Table.*

Practically we expect $k_{\rm IT}^{(\rm G)} > k_{\rm IT}^{(\rm F)} > k_{\rm IT}^{(\rm L)}$ and $k_{\rm GD}^{(\rm G)} > k_{\rm GD}^{(\rm F)} > k_{\rm GD}^{(\rm L)}$. For sparse matrices, the above complexity are naturally reduced. Large scale computations and comparisons will be proposed in a forthcoming paper.

Error analysis. In this paragraph we are interested in the analysis of error for FLS where α is considered as a variable (which hence generalizes what is presented in Subsection 2.2). Setting $\Lambda = [\alpha_1, \alpha]$ with $\alpha_1 < \alpha_2$ and $D := [0,1] \times [\alpha_1, \alpha]$, we search for the solution $\mathbf{y} : (\tau, \alpha) \in D \mapsto \mathbf{y}(\tau, \alpha)$ solution to (14). In this goal, we rewrite (14) in the following form

$$\begin{array}{rcll} \mathbf{y}'(\tau,\alpha) + \alpha(A-I)\mathbf{z}(\tau,\alpha) &=& 0, \qquad \mathbf{y}(0,\alpha) = \mathbf{b}\,, \\ \mathbf{y}(\tau,\alpha) - \big(I + \tau(A-I)\mathbf{z}(\tau,\alpha) &=& 0\,. \end{array} \tag{23}$$

Using [16], we search for a *two-layer* tanh*-neural networks* $\widehat{\mathbf{y}}^N(\boldsymbol{\theta}_{\mathbf{y}}; \tau, \alpha)$ and $\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}}; \tau, \alpha)$ (of arbitrary depth N) for $(\tau, \alpha) \in D$ by minimization of the following loss function

$$\begin{array}{rcl} \mathcal{L}_C^{\rm FLS}(\boldsymbol{\Theta})^2 &=& \lambda_1 \|\widehat{\mathbf{y}}_\tau^N(\boldsymbol{\theta}_{\mathbf{y}}; \cdot, \cdot) + \alpha(A-I)\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}}; \cdot, \cdot)\|^2_{L^2(D)} \\ && + \lambda_2 \|\widehat{\mathbf{y}}^N(\boldsymbol{\theta}_{\mathbf{y}}; \cdot, \cdot) - \big(I + \tau(A-I)\big)\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}}; \cdot, \cdot)\|^2_{L^2([0,1])} \\ && + \lambda_3 \|\widehat{\mathbf{y}}^N(\boldsymbol{\theta}_{\mathbf{y}}; 0, \cdot) - \mathbf{b}\|^2_{L^2(\Lambda)}\,. \end{array} \tag{24}$$

At the discrete level, we denote by $\mathcal{L}_T^{\rm FLS}(S_T; \boldsymbol{\Theta}; q)^2$ the training/discrete version of $\mathcal{L}_C^{\rm FLS}(\boldsymbol{\Theta})^2$ from an order $q \geqslant 1$ quadrature on S_T, a set of learning nodes on D. In practice, we search for

$$\boldsymbol{\Theta}_T^* = {\rm argmin}_{\boldsymbol{\Theta}} \mathcal{L}_T^{\rm FLS}(S_T; \boldsymbol{\Theta}; q)^2\,, \tag{25}$$

and we denote by $\mathbf{y}_T^{N^*}$ and $\mathbf{z}_T^{N^*}$ the corresponding optimized networks $\widehat{\mathbf{y}}^N(\boldsymbol{\theta}_{\mathbf{y}}^*; \cdot, \cdot)$ and $\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}}^*; \cdot, \cdot)$. *In the following, for the sake of readiness and by abuse of notation, we simply denote*

$\widehat{\mathbf{y}}^N(\boldsymbol{\theta}_{\mathbf{y}};\tau,\alpha)$, $\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}};\tau,\alpha)$ *by* $\widehat{\mathbf{y}}^N(\tau,\alpha)$, $\widehat{\mathbf{z}}^N(\tau,\alpha)$, *and we denote* $L^2(D;\mathbb{R}^n)$, $H^k(D;\mathbb{R}^n)$ *by* $L^2(D)$, $H^k(D)$.

The final objective is then to estimate the error e_T:

$$e_T = \|\widehat{\mathbf{y}}_T^{N^*}(1,\cdot) - \mathbf{y}(1,\cdot)\|_{L^2(\Lambda)}\,.$$

The error analysis can be established from [24, 9, 16, 17]. For this reason, we simply present a sketch of the proof. We mainly rely of the following approximability result for H^s-functions ($s \geqslant 3$) by two-layer *tanh*-neural networks.

Theorem 3.1. *(cf. [16]) Denoting* $D = \pi_{i=1}^d [a_i, b_i] \subset \mathbb{R}^d$ *with* $d \geqslant 2$, *we consider the Sobolev space* $H^s(Q)$ *for* $s \geqslant 3$. *For any* $f \in H^s(Q)$ *and any natural number* $N > 5$, *there exists a tanh-neural network denoted* $\widehat{f}^N$ *with 2 hidden layers such that for some* $C = C(s,k,f,d,Q) > 0$ *and all* $k \in \{0,\cdots,s-1\}$

$$\|f - \widehat{f}^N\|_{H^k(D)} \leqslant C(1+\ln^k N)N^{-s+k}\,. \tag{26}$$

Moreover the weights of $\widehat{f}^N$ *scale as* $O(N\ln N + N^\gamma)$ *for some* γ *dependent on* s,k,d.

The error analysis is decomposed in several steps and relies on the smoothness of $\mathbf{y}$ and $\mathbf{z}$ in (23), and follows the arguments from [16].

1. We first show that the continuous loss function $\mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta})^2$ can be as small as wanted. From Theo. (3.1), there exist $\widehat{\mathbf{y}}^N$ and $\widehat{\mathbf{z}}^N$ such that for all $k \in \{0,\cdots,s-1\}$

   $$\|\mathbf{y} - \widehat{\mathbf{y}}^N\|_{H^k(D)} \leqslant C(1+\ln^k N)N^{-s+k}\,,$$

   and

   $$\|\mathbf{z} - \widehat{\mathbf{z}}^N\|_{H^k(D)} \leqslant C(1+\ln^k N)N^{-s+k}\,.$$

   Hence

   $$\|\partial_\tau \mathbf{y} - \partial_\tau \widehat{\mathbf{y}}^N\|_{H^k(D)} \leqslant C(1+\ln^k N)N^{-s+k+1}\,,$$

   for some constant $C > 0$. We then deduce that there exists 2 positive constants $C_1 > 0$ and $C_2 > 0$ such that (and as Λ is a bounded set) for all $k \in \{0,\cdots,s-1\}$

   $$\begin{aligned}\|\widehat{\mathbf{y}}_\tau^N(\cdot,\cdot) + \alpha(A-I)\widehat{\mathbf{z}}^N(\cdot,\cdot)\|_{L^2(D)}^2 \;\leqslant\; & C_1(1+\ln^k N)^2 N^{2(-s+k+1)} \\ & + C_2\rho(A-I)^2(1+\ln^k N)^2 N^{2(-s+k)}\,,\end{aligned}$$

   where $\rho(\cdot)$ denotes the spectral radius. Similarly, we have

   $$\begin{aligned}\|\widehat{\mathbf{y}}^N(\cdot,\cdot) - \big(I+\tau(A-I)\big)\widehat{\mathbf{z}}^N(\cdot,\cdot)\|_{L^2(D)}^2 \;\leqslant\; & C_1(1+\ln^k N)^2 N^{2(k-s)} \\ & + C_2(1+\rho(A-I))^2(1+\ln^k N)^2 N^{2(-s+k)}\,.\end{aligned}$$

We bound the contribution of the initial condition within the loss function. From Poincaré's inequality, there exists $C > 0$ such that for $k \in \{0, \cdots, s-1\}$

$$\begin{array}{rcl} \int_\Lambda |\widehat{\mathbf{y}}^N(0,\alpha) - \mathbf{b}|^2 d\alpha & = & \int_\Lambda |\widehat{\mathbf{y}}^N(0,\alpha) - \mathbf{y}(0,\alpha)|^2 d\alpha \\ & \leqslant & C \int_\Lambda \int_0^1 \left| \partial_\tau \widehat{\mathbf{y}}^N(\tau,\alpha) - \partial_\tau \mathbf{y}(\tau,\alpha) \right|^2 d\tau d\alpha \\ & \leqslant & \|\widehat{\mathbf{y}}^N - \mathbf{y}\|^2_{H^1(D)} \\ & \leqslant & C(1+\ln^k N)^2 N^{2(-s+k)} \,. \end{array}$$

We deduce that there exists $C > 0$, such that

$$\|\widehat{\mathbf{z}}^N(\boldsymbol{\theta}_{\mathbf{z}}; 0, \cdot) - \mathbf{b}\|^2_{L^2(\Lambda)} + \|\widehat{\mathbf{y}}^N(0,\cdot) - \mathbf{b}\|^2_{L^2([\Lambda)} \leqslant C(1+\ln^k N)^2 N^{2(-s+k)} \,.$$

Hence for any $\varepsilon > 0$, there exists $\boldsymbol{\Theta}^*_C = (\boldsymbol{\theta}^*_{\mathbf{y}}, \boldsymbol{\theta}^*_{\mathbf{z}})$ of depth N^* large enough, such that

$$\mathcal{L}^{\mathrm{FLS}}_C(\boldsymbol{\Theta}^*_C)^2 \leqslant \varepsilon^2 \,.$$

2. We now define the following residuals

$$\begin{array}{rcl} \widehat{\mathbf{R}}^N & = & -\mathbf{y}^N_\tau - \alpha(A-I)\widehat{\mathbf{z}}^N, \\ \widehat{\mathbf{S}}^N & = & -\widehat{\mathbf{y}}^N + \big(I + \tau(A-I)\big)\widehat{\mathbf{z}}^N \,. \end{array}$$

In this section, we bound the error

$$\Big(\int_\Lambda \|\mathbf{y} - \widehat{\mathbf{y}}^N\|^2_{L^2(D)} d\alpha \Big)^{1/2},$$

by the continuous loss function. We take the inner product in $\ell^2(\mathbb{R}^n)$ with $\widehat{\mathbf{y}}^N$ and $\widehat{\mathbf{z}}^N$ and integrate with respect to α

$$\begin{array}{rcl} \frac{1}{2}\frac{d}{d\tau}\|\widehat{\mathbf{y}}^N\|^2_{L^2(\Lambda)} & \leqslant & \int_\Lambda \alpha |\widehat{\mathbf{y}}^N \cdot (A-I)\widehat{\mathbf{z}}^N| + |\widehat{\mathbf{y}}^N \cdot \widehat{\mathbf{R}}^N| d\alpha \\ \int_\Lambda |\widehat{\mathbf{z}}^N \cdot (I + \tau(A-I))\widehat{\mathbf{z}}^N| d\alpha & \leqslant & \int_\Lambda |\widehat{\mathbf{z}}^N \cdot \widehat{\mathbf{y}}^N| + |\widehat{\mathbf{y}}^N \cdot \widehat{\mathbf{S}}^N| d\alpha \,. \end{array}$$

Notice that there exists $D(A) > 0$ (related to the eigenvalue of A with the smallest real part, which is supposed to be strictly positive) such that

$$\int_\Lambda |\widehat{\mathbf{z}}^N \cdot (I + \tau(A-I))\widehat{\mathbf{z}}^N| d\alpha \;\geqslant\; D(A)\|\widehat{\mathbf{z}}^N\|^2_{L^2(\Lambda)} \,.$$

Hence

$$\begin{array}{rcl} \frac{1}{2}\frac{d}{d\tau}\|\widehat{\mathbf{y}}^N\|^2_{L^2(\Lambda)} & \leqslant & C\|\widehat{\mathbf{y}}^N\|_{L^2(\Lambda)}\|\widehat{\mathbf{z}}^N\|_{L^2(\Lambda)} + \|\widehat{\mathbf{y}}^N\|_{L^2(\Lambda)}\|\widehat{\mathbf{R}}^N\|_{L^2(\Lambda)} \\ \|\widehat{\mathbf{z}}^N\|^2_{L^2(\Lambda)} & \leqslant & D(A)^{-1}\|\widehat{\mathbf{z}}^N\|_{L^2(\Lambda)}\|\widehat{\mathbf{y}}^N\|_{L^2(\Lambda)} + \|\widehat{\mathbf{y}}^N\|_{L^2(\Lambda)}\|\widehat{\mathbf{S}}^N\|_{L^2(\Lambda)} \,. \end{array}$$

We then deduce that there exists $C(A) > 0$

$$\frac{d}{d\tau}\|\widehat{\mathbf{y}}^N\|^2_{L^2(\Lambda)} \leqslant C(A)\|\widehat{\mathbf{y}}^N\|^2_{L^2(\Lambda)}\|\widehat{\mathbf{z}}^N\|_{L^2(\Lambda)} + \|\widehat{\mathbf{R}}^N\|^2_{L^2(\Lambda)} + \|\widehat{\mathbf{S}}^N\|^2_{L^2(\Lambda)}\,.$$

By Gronwall's inequality we get

$$\|\widehat{\mathbf{y}}^N(\tau,\cdot)\|^2_{L^2(\Lambda)} \leqslant \|\widehat{\mathbf{y}}^N(0,\cdot)\|^2_{L^2(\Lambda)}\exp(C(A)\tau) + \textstyle\int_0^\tau \big(\|\widehat{\mathbf{R}}^N(s,\cdot)\|^2_{L^2(\Lambda)} + \|\widehat{\mathbf{S}}^N(s,\cdot)\|^2_{L^2(\Lambda)}\big)ds\,.$$

Integrating w.r.t. to τ we then get

$$\begin{array}{rcl}\int_0^1 \|\widehat{\mathbf{y}}^N(\tau,\cdot)\|^2_{L^2(\Lambda)}d\tau & \leqslant & \|\widehat{\mathbf{y}}^N(0,\cdot)\|^2_{L^2(\Lambda)}\int_0^1\exp(C(A)\tau)d\tau \\ & & + \int_0^1 \big(\|\widehat{\mathbf{R}}^N(\tau,\cdot)\|^2_{L^2(\Lambda)} + \|\widehat{\mathbf{S}}^N(\tau,\cdot)\|^2_{L^2(\Lambda)}\big)d\tau\,.\end{array}$$

After some basics estimates we then deduce that for some $C(A) > 0$, we get

$$\|\mathbf{y} - \widehat{\mathbf{y}}^N\|^2_{L^2(D)} \leqslant C(A)\mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta})^2\,.$$

Next we bound $\|\mathbf{y}(1,\cdot) - \widehat{\mathbf{y}}^N(1,\cdot)\|_{L^2(\Lambda)}$, as follows

$$\begin{array}{rcl}\int_\Lambda \big|\mathbf{y}(1,\alpha) - \widehat{\mathbf{y}}^N(1,\alpha)\big|^2 d\alpha & = & \int_\Lambda \big|\mathbf{b} - \widehat{\mathbf{y}}^N(0,\alpha) + \int_0^1 \partial_s\mathbf{y}(s,\alpha) - \partial_s\widehat{\mathbf{y}}^N(s,\alpha)\big|^2 d\alpha \\ & \leqslant & 2\big(\|\mathbf{b} - \widehat{\mathbf{y}}^N(0,\cdot)\|^2_{L^2(\Lambda)} + \|\partial_\tau\mathbf{y} - \partial_\tau\widehat{\mathbf{y}}^N\|^2_{L^2(D)}\big)\,.\end{array}$$

Formally ther exists $C(A) > 0$

$$\begin{array}{rcl}\|\partial_\tau\mathbf{y} - \partial_\tau\widehat{\mathbf{y}}^N\|^2_{L^2(D)} & = & \|\alpha(A-I)(I+\tau(A-I))^{-1})\mathbf{y} - \widehat{\mathbf{y}}^N\|^2_{L^2(D)} \\ & \leqslant & C(A)\|\mathbf{y} - \widehat{\mathbf{y}}^N\|^2_{L^2(D)}\,.\end{array}$$

Hence

$$\|\mathbf{y}(1,\cdot) - \widehat{\mathbf{y}}^N(1,\cdot)\|^2_{L^2(\Lambda)} \leqslant C(A)\big(\|\mathbf{b} - \widehat{\mathbf{y}}^N(0,\cdot)\|^2_{L^2(\Lambda)} + \|\mathbf{y} - \widehat{\mathbf{y}}^N\|^2_{L^2(D)}\big)\,.$$

3. At the discrete level, we denote by $\mathcal{L}_T^{\mathrm{FLS}}(S_T;\boldsymbol{\Theta};q)^2$ the training/discrete version of $\mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta})^2$ where an order $q \geqslant 1$ quadrature on a set S_T of learning nodes on D. For any $\varepsilon > 0$, we assume that there exists a sufficently large sample set S_T such that

$$|\mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta})^2 - \mathcal{L}_T^{\mathrm{FLS}}(S_T;\boldsymbol{\Theta};q)^2| \leqslant \varepsilon^2\,.$$

4. According to Step 1. and 2. for any $\varepsilon > 0$, there exists $\boldsymbol{\Theta}_C^*$ and corresponding network is denoted $\widehat{\mathbf{y}}_C^{N^*}$ and is such that

$$\|\widehat{\mathbf{y}}_C^{N^*} - \mathbf{y}\|^2_{L^2(D)} \leqslant \mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta}_C^*)^2 \leqslant \varepsilon^2\,.$$

Moreover according to Step 3. there exists a data set S_T such that for any $\varepsilon > 0$, such that

$$|\mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta}_C^*)^2 - \mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}_C^*; q)^2| \leqslant \varepsilon^2 .$$

We finally denote by $\widehat{\mathbf{y}}_T^{N^*}$ the neural network obtained from a minimization of $\mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}; q)^2$ at the global minimum $\boldsymbol{\Theta}_T^*$ which is *assumed reached*, so that

$$\mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}_T^*; q)^2 \leqslant \mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}_C^*; q)^2 .$$

From Steps 2. and 3.

$$\begin{array}{rcl} \|\widehat{\mathbf{y}}_T^{N^*} - \mathbf{y}\|_{L^2(D)}^2 & \leqslant & \mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta}_T^*)^2 \\ & \leqslant & \mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}_T^*; q)^2 + \varepsilon^2/3 \\ & \leqslant & \mathcal{L}_T^{\mathrm{FLS}}(S_T; \boldsymbol{\Theta}_C^*; q)^2 + 2\varepsilon^2/3 \\ & \leqslant & \mathcal{L}_C^{\mathrm{FLS}}(\boldsymbol{\Theta}_C^*)^2 + \varepsilon^2 . \end{array}$$

Based on the above arguments, we can state the following theorem.

Theorem 3.2. *Consider* (1) *for* $\alpha \in \mathbb{R}^n \backslash \{0\}$ *with* A *a diagonalizable matrix in* $\mathbb{C}\backslash\mathbb{R}_-$. *Assume i) that the continuous loss function* $\mathcal{L}_{\mathcal{C}}$ (24) *can be approximated with arbitrary precision by a training loss* $\mathcal{L}_T$ (25) *thanks to a set of learning nodes* S_T, *and ii) that the optimization algorithm can reach the global minimimum* $\mathcal{L}_T$. *Then for any* $\varepsilon > 0$, *there exists a tanh−neural network with 2 hidden layers of depth* N^* *large enough corresponding to the global minimum of* $\mathcal{L}_T$, *such that*

$$\|\widehat{\mathbf{y}}_T^{N^*}(1, \cdot) - \mathbf{y}(1, \cdot)\|_{L^2(\Lambda; \mathbb{R}^n)} \leqslant \varepsilon .$$

Notice that the above theorem has a limited practical interest, as the approximate solution to (1) $\widehat{\mathbf{y}}_T^{N^*}(1, \cdot)$, is theoretically obtained from the minimization of the training loss $\mathcal{L}_T$ itself dependent on an unknown set of learning nodes S_T.

4. Numerics

This section is devoted to numerical experiments illustrating the fractional linear system solvers developed above. We will start with a small matrix exibiting the convergence per component, then we will consider higher dimensional tests. The numerical computations are performed thanks to the neural network library `jax`, see `https://jax.readthedocs.io`.

4.1. Data-driven solver

We consider $A \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ with $n = 100$ and $A_{i,j} = 1 + 2i\delta_{ij} + 0.1 \exp\big(-0.1(i^2 + j^2)\big)$ and $b_j = 1$.

Experiment 1. In this first experiment we construct a $\alpha-$dependent neural network as defined in Subsection 2.3 with 2 hidden layers and 4 neurons for solving $A^\alpha \mathbf{x} = \mathbf{b}$. Approximate solutions of reference to $A_m^\alpha \mathbf{x}_m = \mathbf{b}$ and $A_m^\alpha \mathbf{y}_m = -\ln(A)\mathbf{x}_m$ with $\alpha_m = 0.1m$ with

$m = 1, \cdots, p = 9$ are given to supervise the training. In this experiment we then optimize $\alpha \mapsto \widehat{\mathbf{y}}_{\mathbf{b}}^{(1)}(\boldsymbol{\theta}^*; \alpha)$ (referred as *direct training*) from

$$\boldsymbol{\theta}_1^* = \text{argmin}_{\boldsymbol{\theta}} \sum_{m=1}^{p} |\widehat{\mathbf{y}}_{\mathbf{b}}^{(1)}(\boldsymbol{\theta}; \alpha_m) - \mathbf{x}_m|^2_{\ell^2(\mathbb{R}^n)}\,, \tag{27}$$

and $\alpha \mapsto \widehat{\mathbf{y}}_{\mathbf{b}}^{(2)}(\boldsymbol{\theta}_2^*; \alpha)$ (referred as *improved training*) with

$$\boldsymbol{\theta}_2^* = \text{argmin}_{\boldsymbol{\theta}} \sum_{m=1}^{p} \left\{ |\widehat{\mathbf{y}}_{\mathbf{b}}^{(2)}(\boldsymbol{\theta}; \alpha_m) - \mathbf{x}_m|^2_{\ell^2(\mathbb{R}^n)} + |\partial_\alpha \widehat{\mathbf{y}}_{\mathbf{b}}^{(2)}(\boldsymbol{\theta}; \alpha_m) - \mathbf{y}_m|^2_{\ell^2(\mathbb{R}^n)} \right\}. \tag{28}$$

We report in Fig. 1 (Left), the loss function as a function of the number of optimization iterations (epochs) $\{(k, \mathcal{L}_\ell^{\text{DD}}(\boldsymbol{\theta}_k)^2) \,:\, k \geqslant 1\}$, $\ell = 1, 2$. We then report in Fig. 1 (Right), the L^2-norm errors between solutions of reference and data-driven optimized neural networks. For $\alpha_k = \alpha_1 + (k-1)\Delta\alpha$, for $k = 1, \cdots, K = 101$ and $\Delta\alpha = (\alpha_p - \alpha_1)/(K-1)$, we report

$$e_\Delta = \left\{ \left(\alpha_k, |\widehat{\mathbf{y}}_{\mathbf{b}}^{(\ell)}(\boldsymbol{\theta}_\ell^*; \alpha_k) - \mathbf{x}_k|_{\ell^2}\right) \,:\, k = 1, \cdots, K \right\}, \tag{29}$$

where $\mathbf{x}_k$ are solutions of reference to $A^{\alpha_k}\mathbf{x} = \mathbf{b}$, for $k = 1, \cdots, K$. This test illustrates the improvement of the training by including some information on the structure of the function $\mathbf{y}_{\mathbf{b}}$.

Experiment 1bis. We here propose to study the error of the data-driven approach as a

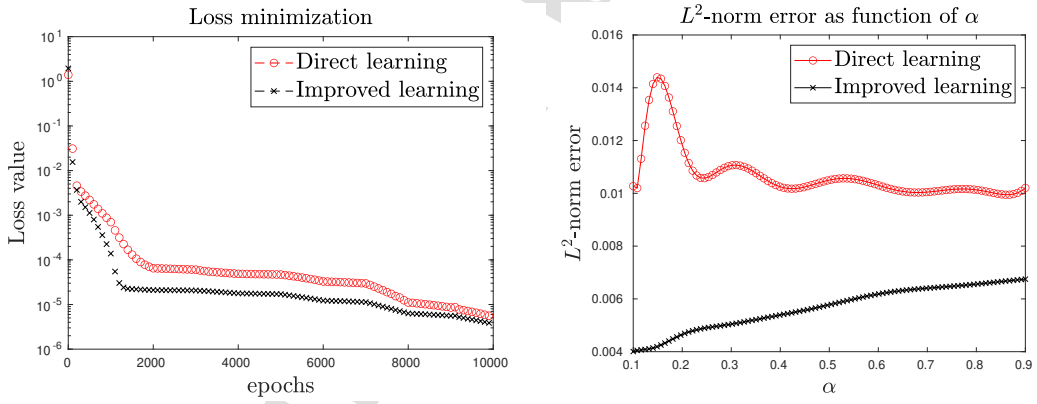


Figure 1: **Experiment 1.** For $\ell = 1, 2$. (Left) Loss functions $\{(k, \mathcal{L}_\ell^{\text{DD}}(\boldsymbol{\theta}_k)^2) \,:\, k \geqslant 1\}$ ($\ell = 1, 2$) (12) and (13). (Right) ℓ_2-norm error (29) with direct learning (27) and improved learning with (28).

function of d, the neural network *depth*, that is the largest number of neurons in the hidden layers. We report the global L^2-norm error (in α) as a function of d. The neural-network has 1 hidden layer, we make vary $N = 2, 4, 8, 16$, and report in Fig. 2 the corresponding

error, that is

$$\Big\{\Big(N, \Big(\Delta\alpha \sum_{m=1}^{p} |\widehat{\mathbf{y}}_{\mathbf{b}}^{N}(\boldsymbol{\theta}_{\nu}^{*}; \alpha_m) - \mathbf{x}_m|_{\ell^2}^2\Big)^{1/2}\Big) \ : \ N = 2, 4, 8, 16\Big\}\ , \tag{30}$$

with $\nu = \mathrm{d}, \mathrm{i}$ corresponding to the direct (d) and improved (i) learning.

Experiment 2. We here construct a $(\alpha, \beta)-$dependent neural network as defined in

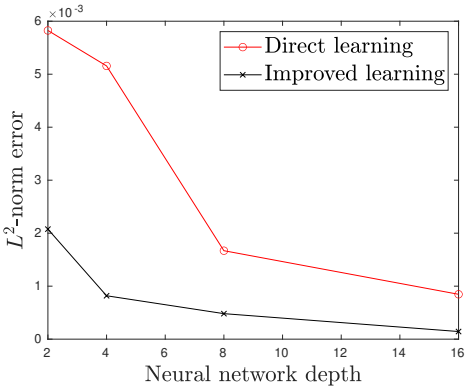


Figure 2: **Experiment 1bis.** ℓ^2-norm error as function of neural network depth (30) for direct and improved versions.

Subsection 3.3 with 3 hidden layers and 4 neurons per layer for solving

$$(A^{\alpha} + A^{\beta})\mathbf{x} = \mathbf{b},$$

Approximate solutions to $(A^{\alpha_m} + A^{\beta_{m'}})\mathbf{x}_{mm'} = \mathbf{b}$ with i) $\alpha_m = 0.25 + 0.1(m-1)$ with $m = 1, \cdots, 5$ and ii) $\beta_{m'} = 0.25 + 0.1(m'-1)$ with $m' = 1, \cdots, 5$ are provided to drive the training. In this experiment we then determine an optimal network $(\alpha, \beta) \mapsto \widehat{\mathbf{z}}_{\mathbf{b}}(\boldsymbol{\theta}^*; \alpha, \beta)$ referred as *direct training*

$$\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}} \sqrt{\Delta\alpha\Delta\beta} \sum_{m=1}^{5}\sum_{m'=1}^{5} |\widehat{\mathbf{z}}_{\mathbf{b}}(\boldsymbol{\theta}; \alpha_m, \beta_{m'}) - \mathbf{x}_{mm'}|_{\ell^2(\mathbb{R}^n)}^2\,. \tag{31}$$

We report in Fig. 3 (Left), the loss values as a function of the number of epochs $\{(k, \mathcal{L}^{\mathrm{DD}}(\boldsymbol{\theta}_k)^2) \ : \ k \geqslant 1\}$. We then report in Fig. 3 (Right), the L^2-norm error between solution of reference and the data-driven optimized neural network.

$$e = \big\{\big(\alpha, |\widehat{\mathbf{z}}_{\mathbf{b}}(\boldsymbol{\theta}^*; \alpha) - (A^{\alpha} + A^{\beta})^{-1}\mathbf{b}|_{\ell^2(\mathbb{R}^n)}\big) \ : \ (\alpha, \beta) \in [\alpha_1, \alpha_p] \times [\beta_1, \beta_p]\big\}\,.$$

We again present a discrete version of e for i) $\alpha_k = \alpha_1 + (k-1)\Delta\alpha$, for $k = 1, \cdots, K = 51$ and $\Delta\alpha = (\alpha_p - \alpha_1)/(K-1)$ and ii) $\beta_{k'} = \beta_1 + (k'-1)\Delta\beta$, for $k' = 1, \cdots, K = 51$ and $\Delta\beta = (\beta_p - \beta_1)/(K-1)$.

$$e_{\Delta} = \big\{\big(\alpha_k, \beta_{k'}, |\widehat{\mathbf{z}}_{\mathbf{b}}(\boldsymbol{\theta}_{\ell}^*; \alpha_k, \beta_{k'}) - \mathbf{x}_{kk'}|_{\ell^2}\big) \ : \ k = 1, \cdots, K,\ k' = 1, \cdots, K\big\}\,, \tag{32}$$

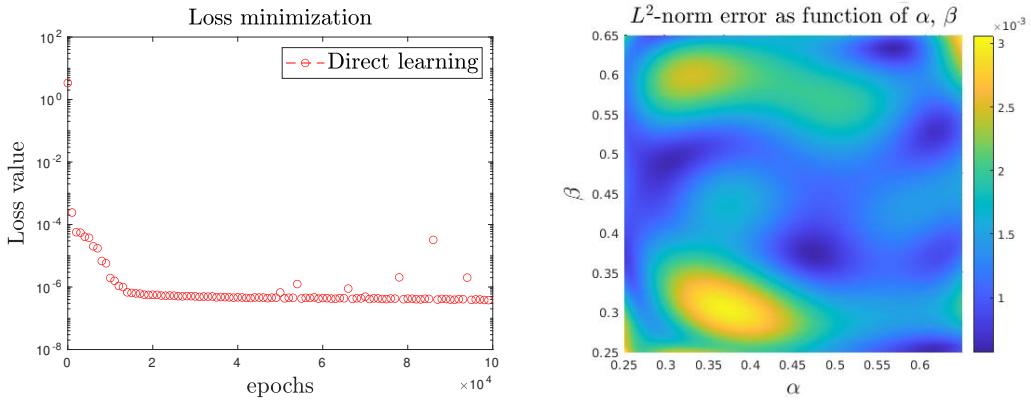


Figure 3: **Experiment 2.** For $\ell = 1, 2$. (Left) Loss functions $\{(k, \mathcal{L}^{\mathrm{DD}}(\boldsymbol{\theta}_k^2))\}$ (12). (Right) ℓ_2-norm error (32) with direct learning (31).

where $\mathbf{x}_{kk'}$ are solutions of reference of $(A^{\alpha_k} + A^{\beta_{k'}})\mathbf{x} = \mathbf{b}$. This test illustrates that solutions to GFLS can be accurately "learnt" thanks to data-driven neural networks.

4.2. Elementary tests of DE solver

In this subsection we consider a simple problem in order to illustrate the convergence of differential-based algorithms.

Experiment 3. We consider A and $\mathbf{b}$ defined as follows

$$A = \begin{pmatrix} 3 & 1 & 1 & 0 \\ -1 & 9 & 0 & 1 \\ 1 & 0 & 4 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \quad b = (1, 1, 1, 1)^T. \tag{33}$$

In the computations below, the neural network is constituted by a unique hidden layer and 10 neurons.

Linear systems. We here consider a linear system $A\mathbf{x} = \mathbf{b}$ as defined in (33). We report in Fig. 4 (Left) the approximate solution to (4) as well as the exact solution. The corresponding loss function is reported in Fig. 4 (Middle) illustrating the convergence of the algorithm. The approximate solution to the LS is then given by $\widehat{\mathbf{y}}(\boldsymbol{\theta}^*; T)$ and we report in Fig. 4 (Right) the ℓ_2-error $|\widehat{\mathbf{y}}(\boldsymbol{\theta}_k; T) - \mathbf{x}|_2$ as a function of k the number of epochs (i.e. gradient descent iterations), for $T = 5$. Naturally, the larger the smallest eigenvalue, the faster the convergence.

Experiment 4. Let us illustrate the above methodology for solving a FLS $A^{1/2}\mathbf{x} = \mathbf{b}$ with A and $\mathbf{b}$ defined in (33). We report in Fig. 5 (Left) the approximate solution to (9) as well as the exact solution. The corresponding loss function is reported in Fig. 5 (Middle) illustrating the convergence of the algorithm. The approximate solution to the FLS is then given by $\widehat{\mathbf{y}}(\boldsymbol{\theta}^*; 1)$ and we report in Fig. 5 (Right) the ℓ_2-error $|\widehat{\mathbf{y}}(\boldsymbol{\theta}_k; 1) - \mathbf{x}|_2$ as a function of

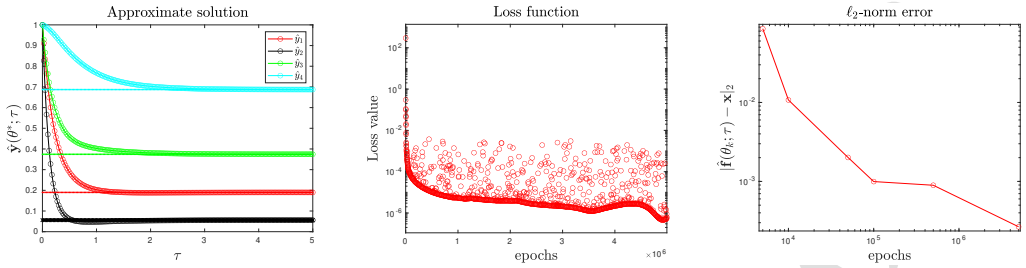


Figure 4: **Experiment 3.** (Left) Approximate solution $\widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau) = \big(\widehat{y}_1(\boldsymbol{\theta}^*;\tau), \widehat{y}_2(\boldsymbol{\theta}^*;\tau), \widehat{y}_3(\boldsymbol{\theta}^*;\tau), \widehat{y}_4(\boldsymbol{\theta}^*;\tau)\big)$ for $\tau \in [0,1]$. (Right) Loss function. (Right) ℓ_2-norm error.

k the number of epochs.

Experiment 5. We next consider a generalized fractional linear systems $(A^{1/2}+A^{-1/2})\mathbf{x} =$

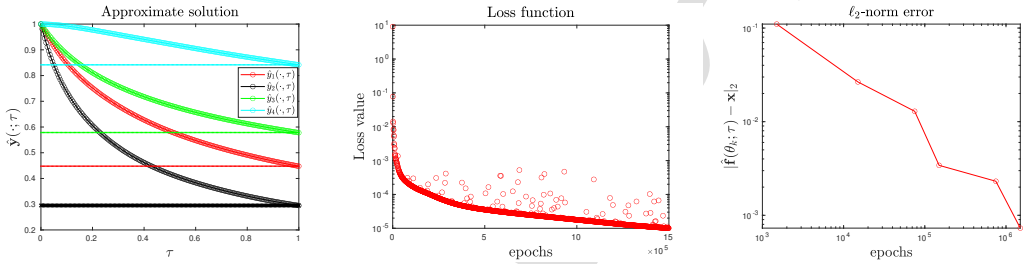


Figure 5: **Experiment 4.** (Left) Approximate solution $\widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau) = \big(\widehat{y}_1(\boldsymbol{\theta}^*;\tau), \widehat{y}_2(\boldsymbol{\theta}^*;\tau), \widehat{y}_3(\boldsymbol{\theta}^*;\tau), \widehat{y}_4(\boldsymbol{\theta}^*;\tau)\big)$ for $\tau \in [0,1]$. (Middle) Loss function. (Right) ℓ_2-norm error.

$\mathbf{b}$ with A and $\mathbf{b}$ defined in (33). We report in Fig. 6 (Left-Top) the approximate solution $\tau \in [0,1/2] \mapsto \widehat{\mathbf{y}}(1/2,\tau)$, and (Right-Top) $s \in [0,1/2] \mapsto \widehat{\mathbf{y}}(s,s)$ to (19) minimizing (21) as well as the exact solution. The corresponding loss function is reported in Fig. 6 (Bottom-Left) illustrating the convergence of the algorithm. The approximate solution to the GFLS is then given by $\widehat{\mathbf{y}}(1/2,1/2)$ and we report in Fig. 6 (Bottom-Right) the ℓ_2-error $|\widehat{\mathbf{y}}(1/2,1/2) - \mathbf{x}|_2$ as a function of the number of epochs k.

Experiment 6. In this section we consider the following matrix $A = \{a_{ij}\}_{i,j} \in \mathbb{R}^{50\times 50}$ and vector $\mathbf{b} \in \mathbb{R}^{50}$ defined as follows

$$a_{ij} = 2i\delta_i^j + 0.1\exp\big(-0.1(i^2+j^2)\big), \quad b_i = 1\,,$$

with δ_i^j is the Kronecker symbol. Although it is still a low dimensional problem, the convergence is maintained in higher dimension. We consider a neural network with one layer and 5 neurons. We first consider the linear system $A\mathbf{x} = \mathbf{b}$ solved using (4) with $T = 5/2$. We report in Fig. 7 (Left) the graph of convergence $\tau \in [0,T] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$.

We next consider the fractional system $A^{3/4}\mathbf{x} = \mathbf{b}$ solved using (9). We report in Fig. 7

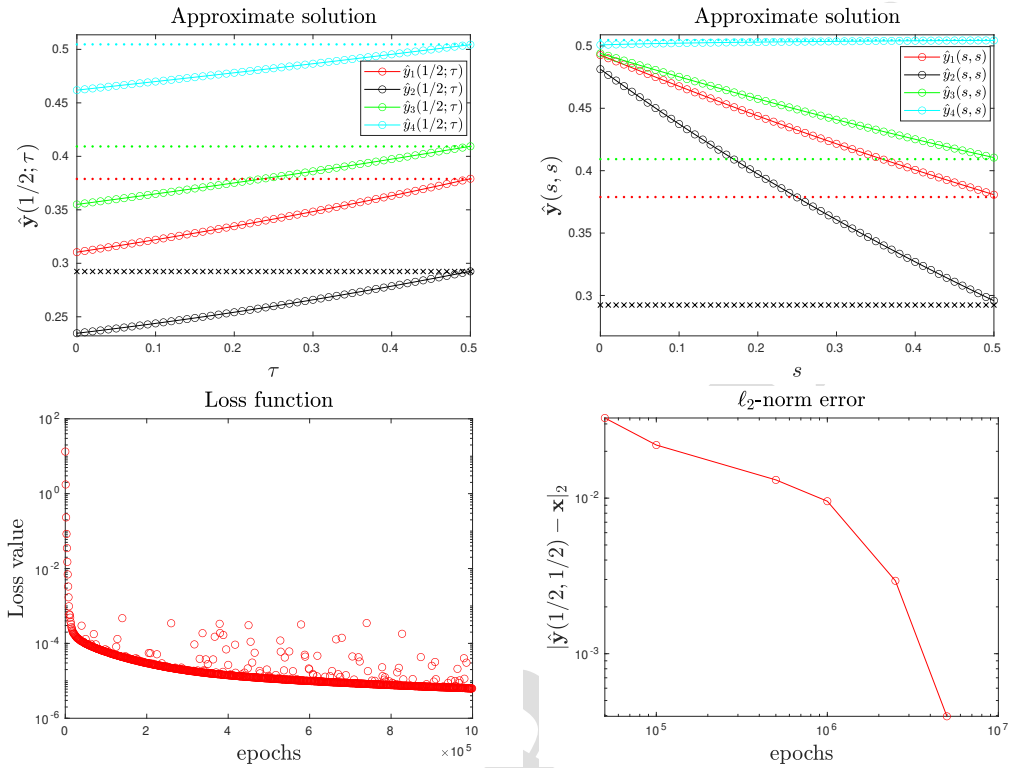


Figure 6: **Experiment 5.** (Top-Left) Approximate solution $\widehat{\mathbf{y}}(1/2,\tau) = \big(\widehat{y}_1(1/2,\tau),\widehat{y}_2(1/2,\tau),\widehat{y}_3(1/2,\tau),\widehat{y}_4(1/2,\tau)\big)$ for $\tau \in [0,1/2]$. (Top-Right) $\widehat{\mathbf{y}}(s,s) = \big(\widehat{y}_1(s,s),\widehat{y}_2(s,s),\widehat{y}_3(s,s),\widehat{y}_4(s,s)\big)$ for $s \in [0,1/2]$. (Bottom-Left) Loss function. (Bottom-Right) ℓ_2-norm error as function of epochs.

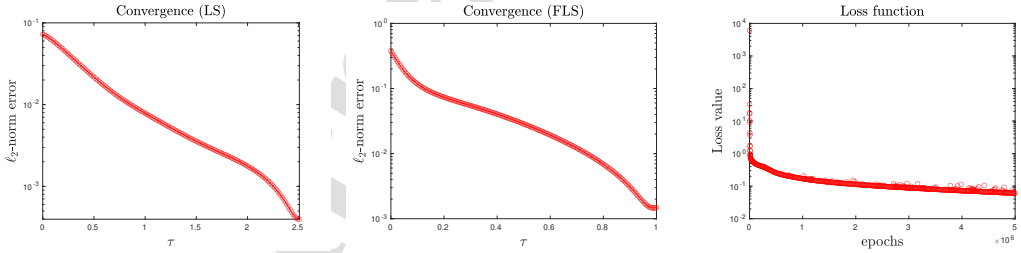


Figure 7: **Experiment 6.** (Left) ℓ_2-norm error for LS: $\tau \in [0,T] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$. (Middle) ℓ_2-norm error for FLS: $\tau \in [0,1] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$. (Right) Loss function (FLS).

(Middle) the graph of convergence $\tau \in [0,1] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$, and the graph of the loss function in Fig. 7 (Right).

Experiment 6bis. We now consider $A = \{a_{ij}\}_{i,j} \in \mathbb{R}^{10^3 \times 10^3}$ and vector $\mathbf{b} \in \mathbb{R}^{10^3}$ for $\alpha = 0,25, 0,5, 0.7$. The network possesses with 2 hidden layers with 5 neurons each and output layer with 10^3 neurons. We take 20 learning nodes $\{t_i\}_i$ randomly chosen between 0 ans 1. We report in Fig. 8 (Left) the graph of convergence $\tau \in [0,1] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$, and the graph of the loss function in Fig. 8 (Right).

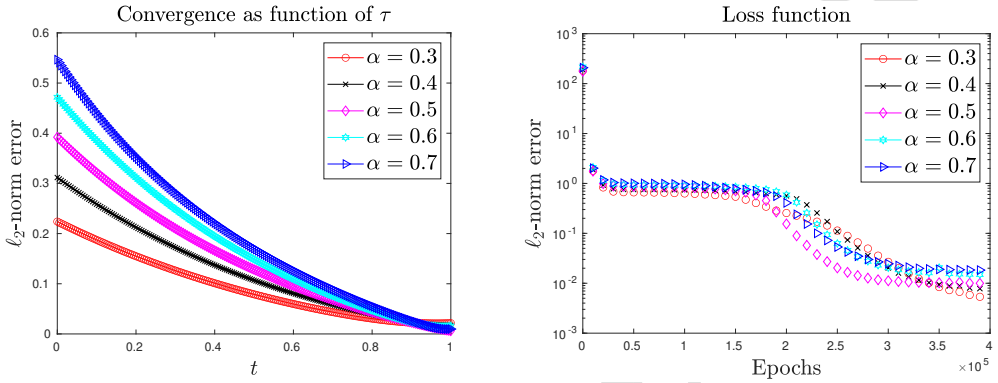


Figure 8: **Experiment 6bis.** (Left) ℓ_2-norm error for LS: $\tau \in [0,T] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$. (Middle) ℓ_2-norm error for FLS: $\tau \in [0,1] \mapsto |\mathbf{x} - \widehat{\mathbf{y}}(\boldsymbol{\theta}^*;\tau)|_2$. (Right) Loss function (FLS).

These experiments illustrate the proof-of-concept of the proposed algorithms, although they would require a lot of improvement to be competitive in comparison with traditional linear algebra solvers.

5. Conclusion

In addition to concluding remarks, we here propose a possible extension/application of the developed approach to fractional differential equations.

5.1. Application to fractional Poisson equation

Let us mention that the approach can be used to numerically solve the fractional Poisson equation on a bounded domain [31]

$$\begin{array}{rcll} (-\triangle)^{\alpha/2} u & = & f(x), & \text{in } \Omega, \\ u(x) & = & 0, & \text{in } \Omega^c, \end{array} \tag{34}$$

where Ω is a bounded domain, f is a given function, and $\alpha \in (0,2)$. Let us denote by $A_h \in \mathbb{R}^{n \times n}$ (resp. $f_h \in \mathbb{R}^n$) a finite difference approximation of the Laplace operator with null Dirichlet boundary condition (resp. f). It can be shown (see [31] for discussion and references) that a finite difference approximation of (34) reads

$$A_h^{\alpha/2} u_h = f_h,$$

where u_h is the finite difference approximation of u solution to (34). This FLS can then be solved using the approach discussed in this paper. Extension to fractional heat or Schrödinger equations are also natural extensions.

5.2. Concluding remarks

In this paper, we have proposed and validated a neural network-based methodology for solving linear and fractional algebraic systems thanks to differential reformulation. Although the developed solvers can generally speaking, still hardly compete with standard solvers in term of efficiency, some recent works in quantum chemistry for the electronic structure calculation (see [11, 12]) show that in very high dimension, neural network methods can out-perform. In future works, we plan to develop neural network-based algorithms for preconditioning standard large linear system solvers [33], which seems to be a promising approach from the efficiency point of view.

References

[1] B. N. Achar, B. T. Yale, and J. W. Hanneken. Time fractional Schrödinger equation revisited. *Adv. Math. Phys.*, 2013:290216, 2013.

[2] T. Alt, K. Schrader, M. Augustin, P. Peter, and J. Weickert. Connections between numerical algorithms for pdes and neural networks. *Journal of Mathematical Imaging and Vision*, 65(1):185–208, 2023.

[3] M. Anthony and P. L. Bartlett. *Neural network learning: theoretical foundations*. Cambridge University Press, Cambridge, 1999.

[4] X. Antoine and E. Lorin. ODE-based double-preconditioning for solving linear systems $A^\alpha x = b$ and $f(A)x = b$. *Numer. Linear Algebra Appl.*, 28(6):Paper No. e2399, 22, 2021.

[5] X. Antoine and E. Lorin. Generalized fractional algebraic linear system solvers. *J. Sci. Comput.*, 91(1):Paper No. 25, 30, 2022.

[6] W. Bao and X. Dong. Numerical methods for computing ground state and dynamics of nonlinear relativistic Hartree equation for boson stars. *J. Comput. Phys.*, 230:5449–5469, 2011.

[7] A. Biswas, J. Tian, and S. Ulusoy. Error estimates for deep learning methods in fluid dynamics. *Numerische Mathematik*, 151(3):753–777, 2022.

[8] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018.

[9] G Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, Dec 2019.

[10] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[11] I. Carusotto and C. Ciuti. Quantum fluids of light. *Rev. Mod. Phys.*, 85:299–366, 2013.

[12] J.-P. Chehab and J. Laminie. Differential equations and solution of linear systems. *Numer. Algorithms*, 40(2):103–124, 2005.

[13] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2(4):303–314, 1989.

[14] T. De Ryck, A. D. Jagtap, and S. Mishra. Error estimates for physics-informed neural networks approximating the navier–stokes equations. *IMA Journal of Numerical Analysis*, 44(1):83–119, 2024.

[15] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

[16] N. Hale, N. J. Higham, and L. N. Trefethen. Computing A^α, $\log(A)$, and related matrix functions by contour integrals. *SIAM J. Numer. Anal.*, 46(5):2505–2523, 2008.

[17] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation.

[18] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation.

[19] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci. U.S.A.*, 79(8):2554–2558, 1982.

[20] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[21] R. Hu, Q. Lin, A. Raydan, and S. Tang. Higher-order error estimates for physics-informed neural networks approximating the primitive equations. *Partial Differential Equations and Applications*, 4(4):34, 2023.

[22] I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

[23] N. Laskin. Fractional quantum mechanics. *Phys. Rev. E*, 62:3135–3145, 2000.

[24] N. Laskin. Fractional quantum mechanics and Lévy path integrals. *Phys. Lett. A*, 268:298–304, 2000.

[25] A. Lischke, G. Pang, M. Gulian, and et al. What is the fractional Laplacian? A comparative review with new results. *J. Comput. Phys.*, 404:109009, 62, 2020.

[26] E. Lorin and S. Tian. A numerical study of fractional linear algebraic systems. *Math. Comput. Simulation*, 182:495–513, 2021.

[27] G. Pang, L. Lu, and G.E. Karniadakis. fPINNs: fractional physics-informed neural networks. *SIAM J. Sci. Comput.*, 41(4):A2603–A2626, 2019.

[28] F. Pinsker, W. Bao, Y. Zhang, H. Ohadi, A. Dreismann, and J. J. Baumberg. Fractional quantum mechanics in polariton condensates with velocity-dependent mass. *Phys. Rev. B*, 92:195310, 2015.

[29] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[30] Y. Shin, J. Darbon, and G. Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *Commun. in Comput. Phys.*, 28(5):2042–2074, 2020.

[31] L. Yang, D. Zhang, and G. E. Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM J. Sci. Comput.*, 42(1):A292–A317, 2020.