Journal of Computational Physics 473 (2023) 111657

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Schwarz waveform relaxation-learning for advection-diffusion-reaction equations

Emmanuel Lorin [a,b,*], Xu Yang [c]

[a] School of Mathematics and Statistics, Carleton University, Ottawa, K1S 5B6, Canada
[b] Centre de Recherches Mathématiques, Université de Montréal, Montréal, H3T 1J4, Canada
[c] Department of Mathematics, University of California, Santa Barbara, CA 93106, USA

ARTICLE INFO

Article history:
Received 14 October 2021
Received in revised form 20 June 2022
Accepted 25 September 2022
Available online 18 October 2022

Keywords:
Physics-informed neural network
Schwarz waveform relaxation
Domain decomposition
Advection-diffusion-reaction equations

ABSTRACT

This paper develops a physics-informed neural network (PINN) combined with a Schwarz waveform relaxation (SWR) method for solving local and nonlocal advection-diffusion-reaction equations. Specifically, we derive the algorithm by constructing subdomain-dependent local solutions by minimizing local loss functions, allowing the decomposition of the training process in different domains in an embarrassingly parallel procedure. Provided the convergence of PINN, the overall proposed algorithm is convergent. By constructing local solutions, one can, in particular, adapt the depth of the deep neural networks, depending on the solution's spectral space and time complexity in each subdomain. One of the main advantages of using NN compared to standard solvers, is that the PINN algorithm introduces some learning in the SWR algorithm allowing for an acceleration of the overall algorithm, especially close to SWR convergence. We present some numerical experiments based on classical and Robin-SWR algorithms to illustrate the performance and comment on the convergence of the proposed method.

1. Introduction

This paper focuses on the derivation and analysis of a Neural-Network (NN) based Schwarz Waveform Relaxation (SWR) Domain Decomposition Method (DDM) for solving partial differential equations (PDE) in parallel. We will focus in this paper on a simple diffusion-advection-reaction equation. Still, the proposed strategy applies to any other evolution (in particular wave-like) equations, for which the convergence of SWR is proven [1–11]. More specifically, we derive a combined DDM-SWR and Physics-Informed Neural Network (PINN) method for solving local and nonlocal diffusion-advection-reaction equations. The latter was developed by Karniadakis et al. [12–14] and is a general strategy in scientific machine learning for solving PDE using deep neural networks via the minimization of well-designed loss functions. Notice that in [15], was also proposed a more direct DDM for solving PDE. Interestingly, both ([15] and the one presented here) methods could actually be combined; this was however not tested in this paper. Let us also mention a recent paper [16], where a combination of Schwarz DDM with NN-based solvers is proposed for stationary PDE. Beyond the derivation of the SWR-NN method, this paper's objective is to exhibit some fundamental properties that make this methodology promising. The general principle is to solve Initial Boundary Value Problems (IBVPs) by constructing local solutions (subdomain-dependent) obtained by minimizing local loss functions. The overall strategy is convergent (provided that the PINN method is convergent) and

* Corresponding author.
E-mail addresses: elorin@math.carleton.ca (E. Lorin), xuyang@math.ucsb.edu (X. Yang).

https://doi.org/10.1016/j.jcp.2022.111657
0021-9991/

allows, in particular, to locally decompose the training process in different subdomains within an embarrassingly parallel procedure. The construction of local solutions also allows to locally adapt the depth of the deep neural network, depending on the solution's spectral space and time complexity in each subdomain.

In this paper, we will primarily focus on the derivation and will not necessarily detail all the computational aspects, particularly regarding the selection of the training data. This will, however, be specified in the Numerical Experiment section. For convenience, we shall recall some basic aspects about PINNs, neural networks and SWR method for evolution equations, which shall be used in the paper later.

1.1. Basics on PINNs

Let us recall the principle of PINNs for solving, *e.g.*, an evolution PDE over $\Omega \times [0,T]$,

$$\left\{\begin{array}{lll} \partial_t u + Pu &= f & \text{in } \Omega \times [0,T] \\ Mu &= 0 & \text{in } \Gamma \times [0,T] \\ u(\cdot,0) &= u_0 & \text{in } \Omega \end{array}\right. , \tag{1}$$

where i) P is a differential operator in space, and M a differential or algebraic boundary operator over the domain boundary Γ; ii) are f and u_0 are imposed functions. The PINN approach, which generalizes the DE-solver from Lagaris [17] consists in parameterizing (by say $\boldsymbol{\theta}$) a NN, $N(\boldsymbol{\theta}, \boldsymbol{x}, t)$ approximating the solution to (1), by minimizing (a discrete version of) the following loss function

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) = &\|(\partial_t + P)N(\boldsymbol{\theta},\cdot,\cdot) - f\|_{L^2(\Omega\times(0,T))} + \lambda_1 \|MN(\boldsymbol{\theta},\cdot,\cdot)\|_{L^2(\Gamma\times(0,T))} \\ &+\lambda_2 \|N(\boldsymbol{\theta},\cdot,0) - u_0\|_{L^2(\Omega)} , \end{aligned}$$

where $\lambda_{1,2}$ are some free positive parameters and $\boldsymbol{\theta} \in \Theta \in \mathbb{R}^Q$ for some large Q, and where $\|\cdot\|_{L^2(\Omega\times(0,T))}$ (resp. $\|\cdot\|_{L^2(\Gamma\times(0,T))}$) denotes the L^2-norm over $\Omega \times (0,T)$ (resp. $\Gamma \times (0,T)$). Practically, the loss functions are constructed by estimating the values at a (very) large number of space & time-input data $\{(\boldsymbol{x}_j, t_n)\}_{j,n}$. Hence the L^2-norms are not exactly computed, but only approximated. Karniadakis and collaborators have developed numerous techniques to improve the efficiency and rate of convergence of PINN-algorithms for different types of PDE. We refer for instance to [12–14] for details. Ultimately, the PINN strategy is to provide more efficient solvers than standard methods (finite difference, finite-volume, finite-elements, spectral methods, pseudospectral methods, etc) for very high dimensional (stochastic or deterministic) PDEs. As far as we know, this is not clearly yet established, which justifies the active research in this field and the developed of new methods. Hereafter, let us summarize some general pros and cons of the PINN algorithms.

Pros.

- As with any machine learning algorithm, once the network is trained (and the neural network parameters evaluated), the solution can further be evaluated/tested at any space and time locations, without re-computing the full solution. This feature will actually be used as a learning tool in the SWR algorithm.
- The automatic differentiation avoids some crucial difficulties when approximating the evolution PDE equation, such as numerical diffusion and numerical stability. The latter is, in particular, the source of important issues when considering non-constant coefficients in the PDE.
- The PINN algorithm generally scales linearly, unlike most of the implicit evolution PDE solvers.
- The implementation of the PINN algorithm is independent of the PDE's structure, unlike the training efficiency.

Cons.

- The main (general) issue with neural network-based algorithms is that they usually do not provide global minima to loss functions. Hence there is no guarantee of the accuracy of the computed solution. However, if the network is accurately trained for simple problems (where it is possible to get solutions of reference), such networks can be used as an initial ansatz for more complex problems.
- Some general properties which must be satisfied by a solution may not easily be satisfied, requiring additional work on the structure of the NN: for instance, anti-symmetry of the wavefunction in quantum chemistry, conservation of L^2-norm in time for time-dependent Schrödinger or Dirac equation in quantum physics.
- Over or underfitting may occur if training is not properly performed. More generally, the overall accuracy and efficiency largely depend on the minimization algorithm, the loss function structure, and the search space dimension.
- The structure of neural networks limits the space of solutions.

Let us also mention that relatively similar NN-based techniques are used in quantum chemistry for tackling high-dimensional eigenvalue problems (computation of ground states for Hamiltonians of large molecules [18,19] which also motivates the combination with DDM methods.). The objective of this paper is not to discuss the overall efficiency of the

PINN method but its combination with DDM-SWR techniques. The main advantage of using NN in a DDM framework compared to standard solvers is the following: at a given Schwarz iteration k, the NN to be optimized can be initialized as the converged (in the sense of NN optimization) NN obtained at the previous Schwarz iteration $k-1$ (that is taking the latter as an ansatz at iteration k). In other words, unlike standard PDE solvers, which require a full computation of the evolution PDE in each subdomain starting from the same initial data, the computation of the PINN solution within each subdomain will be faster and faster while the Schwarz iteration increases (as the local initial parameters will be closer and closer to the converged ones, correspondng to the exact solution). Hence, the gain is not in terms of the number of Schwarz iterations but the acceleration of the optimization process.

1.2. Basics of neural networks

We here recall the basics of neural networks. We denote the neural network $N(\boldsymbol{\theta}, \boldsymbol{x})$ with $\boldsymbol{x} = (x_1, \cdots, x_d) \in \Omega \subseteq \mathbb{R}^d$ and where we denote by $\boldsymbol{\theta}$ the unknown parameters. Neural networks usually read (for 1 hidden layer, machine learning)

$$N(\boldsymbol{\theta}, \boldsymbol{x}) \; = \; \sum_{i=1}^{H} v_i \sigma_i\big(\sum_{j=1}^{d} w_{ij} x_j + u_i\big)\,, \tag{2}$$

where $\{\sigma_i\}_{1\leqslant i\leqslant H}$ are the sigmoid transfer functions, and H is the number of sigmoid units, $\{w_{ij}\}_{ij}$ are the weights and $\{u_i\}_i$ the bias. When considering several hidden layers (deep learning), we have then to compose functions of the form (2), [20]. That is

$$N = \mathcal{N}_p \circ \mathcal{N}_{p-1} \circ \cdots \mathcal{N}_1 \circ \mathcal{N}_0\,,$$

where for $0 \leqslant r \leqslant p$, $\mathcal{N}_r$ is defined from $\mathbb{R}^{a_r}$ (with $a_r \in \mathbb{N}$) to $\mathbb{R}^{a_{r+1}}$ by $\sigma_r(W_r X_r + b_r)$, σ_r is an activation function, $X_r \in \mathbb{R}^{a_r}$ and where $(a_0, \cdots, a_{p+1})$ where $p+1$ layers are considered. The layer $r=0$ is the input layer and $r=p+1$ is the output layer, such that $a_0 = a_{p+1} = m$. In fine, N from $X \in \mathbb{R}^m$ to $\mathbb{R}^m$.

1.3. Basics on SWR methods for evolution equations

In this subsection, we recall the principle of SWR-DDM for solving evolution PDE. Consider a d-dimensional first order in time evolution partial differential equation $\partial_t u + Pu = f$ in the spatial domain $\Omega \subseteq \mathbb{R}^d$, and time domain $(0, T)$, where P is a linear differential operator in space. The initial data is denoted by u_0, and we impose, say, null Dirichlet boundary conditions on $\Gamma_{\text{ext}} = \partial\Omega$. We present the method for 2 subdomains, although in practice an arbitrary number of subdomains is employed. We first split Ω into two open subdomains $\Omega_\varepsilon^\pm$, with or without overlap ($\Omega_\varepsilon^+ \cap \Omega_\varepsilon^- = \emptyset$ or $\Omega_\varepsilon^+ \cap \Omega_\varepsilon^- \neq \emptyset$), and $\varepsilon \geqslant 0$. The SWR algorithm consists in iteratively solving IBVPs in $\Omega_\varepsilon^\pm \times (0,T)$, using transmission conditions at the subdomain interfaces $\Gamma_\varepsilon^\pm := \partial\Omega_\varepsilon^\pm \cap \Omega_\varepsilon^\mp$. The imposed transmission conditions are established using the preceding Schwarz iteration data in the adjacent subdomain. That is, for $k \geqslant 1$, and denoting $u^\pm$ the solution in $\Omega_\varepsilon^\pm$, we consider

$$\left\{\begin{array}{lll} (\partial_t + P)u^{\pm,(k)} & = & f, \;\; \text{in } \Omega_\varepsilon^\pm \times (0,T), \\ u^{\pm,(k)}(\cdot,0) & = & u_0^\pm, \;\; \text{in } \Omega_\varepsilon^\pm, \\ \mathcal{T}_\pm u^{\pm,(k)} & = & \mathcal{T}_\pm u^{\mp,(k-1)}, \;\; \text{on } \Gamma_\varepsilon^\pm \times (0,T), \\ u^{\pm,(k)} & = & 0, \;\; \text{on } \Lambda_\varepsilon^\pm \times (0,T)\,, \end{array}\right. \tag{3}$$

with a given initial guess $u^{\pm,(0)}$, where $\mathcal{T}^\pm$ denotes a boundary/transmission operator and where $\Lambda_\varepsilon^\pm := \partial\Omega_\varepsilon^\pm \backslash \Gamma_\varepsilon^\pm$ are internal boundaries. Classical SWR (CSWR) method consists in taking $\mathcal{T}^\pm$ as the identity operator while Optimized-SWR (OSWR) method consists in taking $\mathcal{T}^\pm = \nabla_{\boldsymbol{n}^\pm} \pm \lambda_{\Gamma_\varepsilon}^\pm$ for some well chosen (optimized from the convergence rate point of view) $\lambda_{\Gamma_\varepsilon^\pm}^\pm \in \mathbb{R}^*$, and outward normal vector $\boldsymbol{n}^\pm$ to $\Gamma_\varepsilon^\pm$. The OSWR method is then a special case of Robin-SWR methods. In addition, the OSWR method is often convergent even without overlap. The latter is hence of crucial interest from the computational complexity point. We refer to [6,9–11,7] for details. The convergence criterion for the Schwarz DDM is typically given for any $0 < t \leqslant T$, by

$$\Big\| \|u^{+,(k)}(\cdot,\cdot) - u^{-,(k)}(\cdot,\cdot)\|_{\infty;\overline{\Omega_\varepsilon^+}\cap\overline{\Omega_\varepsilon^-}} \Big\|_{L^2(0,t)} \leqslant \delta^{\text{Sc}}, \tag{4}$$

with δ^{Sc} small enough. When the convergence of the full iterative algorithm is obtained at Schwarz iteration k^{cvg}, one gets the converged global solution $u^{\text{cvg}} := u^{(k^{\text{cvg}})}$ in Ω. The reconstructed solution u, is finally defined as $u_{|\Omega_\varepsilon^\pm} = u^{\pm,(k^{\text{cvg}})}$.

1.4. Advection-diffusion-reaction equation

Rather than considering a general situation, for which the rapid convergence of the SWR method and efficiency are not necessarily proven, we propose to focus on the advection-diffusion-reaction equation, for which both properties are established in [6] (see also [5,1–4] for the Schrödinger equation). Let us consider the following initial boundary-value problem: find the real function $u(\boldsymbol{x}, t)$ solution to the advection-diffusion-reaction equation on $\mathbb{R}^d$, $d \geqslant 1$,

$$\begin{cases} \partial_t u = \nu(\boldsymbol{x})\triangle u + \boldsymbol{a}(\boldsymbol{x})\cdot\nabla u + r(\boldsymbol{x})u,\ \boldsymbol{x}\in\mathbb{R}^d,\ t>0,\\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}),\ \boldsymbol{x}\in\mathbb{R}^d, \end{cases} \tag{5}$$

with initial condition u_0, and the real-valued space-dependent smooth reaction term r, advection vector $\boldsymbol{a}$ and diffusion ν. We recall from [6], that for considering $\Omega=\mathbb{R}$ with constant coefficients in (5), and for $u_0\in L^2(\Omega)$, $f\in L^2(0,T;L^2(\Omega))$, there exists a unique weak solution in $C(0,T;L^2(\Omega))\cap L^2(0,T;H^1(\Omega))$. Moreover, if $u_0\in H^2(\Omega)$ and $f\in L^2(0,T;L^2(\Omega))$, there exists a unique weak solution in $L^2(0,T;H^3(\Omega))\cap H^{3/2}(0,T;L^2(\Omega))$.

1.5. Organization of the paper

The rest of the paper is organized as follows. In Section 2, we derive the combined SWR-PINN method, and some properties are discussed in Sections 2.2 and 2.4. In particular we give strong arguments in favor of using SWR-PINN algorithms rather than a combination of SWR algorithms with standard evolution PDE solvers. Section 3 is devoted to some numerical experiments illustrating the convergence and properties of the overall SWR-PINN method. We make conclusive remarks in Section 4.

2. SWR-PINN method

In this section, we propose to combine PINN-based solvers with SWR-DDM to solve the advection-diffusion-reaction equation on a bounded domain $\Omega\subset\mathbb{R}^d$, imposing null Dirichlet boundary conditions at Γ. For the sake of simplicity of the presentation, the derivation is proposed for two subdomains; the extension to an arbitrary number of subdomains is straightforward.

2.1. Derivation of the SWR-PINN method

The standard SWR method for two subdomains consists in solving the IBVP using the following algorithm

$$\begin{cases} \partial_t u^{\pm,(k)} &= \nu(\boldsymbol{x})\triangle u^{\pm,(k)} + \boldsymbol{a}(\boldsymbol{x})\cdot\nabla u^{\pm,(k)} + r(\boldsymbol{x})u^{\pm,(k)},\ \text{in } \Omega_\varepsilon^\pm\times(0,T),\\ u^{\pm,(k)}(\cdot,0) &= u_0^\pm,\ \text{in } \Omega_\varepsilon^\pm,\\ \mathcal{T}_\pm u^{\pm,(k)} &= \mathcal{T}_\pm u^{\mp,(k-1)},\ \text{on } \Gamma_\varepsilon^\pm\times(0,T),\\ u^{\pm,(k)} &= 0,\ \text{on } \Lambda_\varepsilon^\pm\times(0,T), \end{cases} \tag{6}$$

where $\mathcal{T}_\pm$ is a boundary operator, and where we recall that $\Lambda_\varepsilon^\pm=\partial\Omega_\varepsilon^\pm\backslash\Gamma_\varepsilon^\pm$. The well-posedness and the convergence of this method and its rate of convergence for constant coefficients were established in [6] for different types of transmission conditions. SWR algorithms can actually be reformulated as a fixed point methods (FPM), and their rate of convergence is hence determined by a contraction factor of the FPM. More specifically, it is proven in [6] that for

$$\partial_t u + a\partial_x u - \nu\partial_{xx}u + ru = f,\ \ (x,t)\in\mathbb{R}^2\times(0,T),$$

where $f\in L^2(\Omega)$, $\nu>0$, a,b in $\mathbb{R}$, the CSWR method is convergent and has a convergence rate C_{CSWR} (contract factor), at least, given by

$$C_{\text{CSWR}} = \exp\Big(-\frac{\varepsilon}{\nu}(\sqrt{a^2+4\nu r})\Big).$$

In fact, this can be refined to a superlinear convergence rate $2/\sqrt{\pi}\int_{\varepsilon/\sqrt{\nu T}}^\infty e^{-s^2}ds$. For Robin-SWR methods, with transmission conditions $\partial_x\pm\lambda$, the convergence rate is actually improved

$$C_{\text{Robin}} = \sup_{\omega\in\mathbb{R}}\Big|\frac{\sqrt{a^2+4\nu(r+\mathrm{i}\omega)}-\lambda}{\sqrt{a^2+4\nu(r+\mathrm{i}\omega)}+\lambda}\Big|^2\exp\Big(-\frac{\varepsilon}{\nu}(\sqrt{a^2+4\nu(r+\mathrm{i}\omega)})\Big),$$

as $|\sqrt{a^2+4\nu(r+\mathrm{i}\omega)}-\lambda|/|\sqrt{a^2+4\nu(r+\mathrm{i}\omega)}+\lambda|<1$. We notice in particular, that a crucial element for the rate of convergence of SWR methods is the size of the overlapping zone. However, overlapping is not required for Robin-SWR methods to converge.

Rather than a standard approximation of (6) using a finite elements/difference or pseudospectral methods [4,5,3], we then propose to solve this system using a PINN method. We denote by $N(\boldsymbol{\theta},\boldsymbol{x},t)$ the generic NN to optimize, where $\boldsymbol{\theta}$ $(\in\Theta)$ denotes the unknown parameters. The SWR-NN hence consists in searching for an approximate solution to the SWR method by applying local PINN algorithms. That is, we now consider

$$\begin{cases} \partial_t N^{\pm,(k)} &= \nu(\boldsymbol{x})\triangle N^{\pm,(k)} + \boldsymbol{a}(\boldsymbol{x})\cdot\nabla N^{\pm,(k)} + r(\boldsymbol{x})N^{\pm,(k)},\ \text{in } \Omega_\varepsilon^\pm\times(0,T),\\ N^{\pm,(k)}(\cdot,0) &= u_0^\pm,\ \text{in } \Omega_\varepsilon^\pm,\\ \mathcal{T}_\pm N^{\pm,(k)} &= \mathcal{T}_\pm N^{\mp,(k-1)},\ \text{on } \Gamma_\varepsilon^\pm\times(0,T),\\ N^{\pm,(k)} &= 0,\ \text{on } \Lambda_\varepsilon^\pm\times(0,T). \end{cases} \tag{7}$$

Remark 2.1. For the CSWR method ($\mathcal{T}_\pm$ is the identity operator), it is proven in [6], among many other well-posed results, that for $f \in L^2(0,T;H^1(\Omega_\varepsilon^\pm))$ and $u_0 \in H^1(\Omega)$ with $u^{\pm,(0)} \in H^{3/4}(0,T)$ and some compatibility conditions, the algorithm (6) is well-posed in $L^2(0,T;H^2(\Omega)\cap H^1(0,T;L^2(\Omega))$.
Let us now denote $e^{\pm,(k)} := u^{\pm,(k)} - u$ and $\widetilde{e}^{\pm,(k)} := N^{\pm,(k)} - u$. In Theorem 3.3 from [6] it is stated that for any $k \geqslant 0$ and for some positive constant $C > 0$

$$\|(e^{+,(2k+1)}, e^{-,(2k+1)})\|_{\mathcal{H}_\varepsilon} \leqslant C \exp\Big(-k\frac{\varepsilon(\sqrt{a^2+4\nu r})}{\nu}\Big)\|(u(\varepsilon,\cdot),u(-\varepsilon/2,\cdot))\|_{({}_0H^{3/4}(0,T))^2}\,, \tag{8}$$

where we have denoted

$$\mathcal{H}_\varepsilon := L^2(0,T;H^2(\Omega_\varepsilon^+)\cap H^1(0,T;L^2(\Omega_\varepsilon^+)) \times L^2(0,T;H^2(\Omega_\varepsilon^+)\cap H^1(0,T;L^2(\Omega_\varepsilon^-))\,.$$

This leads to the following proposition.

Proposition 2.1. *Assume that the NN solution to (7) satisfies there exists $\eta(\overline{\boldsymbol{\theta}}^{\pm;(k)};\varepsilon)$ small enough so that (that is convergence of the PINN-method)*

$$\|N^{\pm,(k)} - u^{\pm,(k)}\|_{\mathcal{H}_\varepsilon} \leqslant \eta(\overline{\boldsymbol{\theta}}^{\pm;(k)};\varepsilon)\,, \tag{9}$$

then the PINN-CSWR and OSWR-PINN methods are convergent.

Proof. The proof is straightforward, since

$$\|\widetilde{e}^{\pm,(2k+1)}\|_{\mathcal{H}_\varepsilon} \leqslant \|e^{\pm,(2k+1)}\|_{\mathcal{H}_\varepsilon} + \|N^{\pm,(2k+1)} - u^{\pm,(2k+1)}\|_{\mathcal{H}_\varepsilon}\,.$$

Combining this inequality with (8) and (9) gives the convergence. Similar conclusions can be reached for the PINN-OSWR algorithms in $L^2(0,T;H^3(\Omega))\cap H^{3/2}(0,T;L^2(\Omega))$; see [6]. □

As explained above, we have two types of convergence in the PINN-SWR algorithms: convergence of a PINN algorithm at each SWR iteration and in each subdomain of the PINN algorithm. These two processes are performed one after the other. At each Schwarz iteration k, we perform

$$\boldsymbol{\theta}_{\ell+1}^{\pm,(k)} = \boldsymbol{\theta}_{\ell}^{\pm,(k)} - \nu\nabla\mathcal{L}^\pm(\boldsymbol{\theta}_\ell^{\pm,(k)})\,,$$

where ℓ is the gradient iteration index (backward pass) and ν the learning rate.

First, we use the standard technique to include the initial condition [17], by searching for a trial network (ansatz) in the form

$$T(\boldsymbol{\theta},\boldsymbol{x},t) = u_0(\boldsymbol{x}) + tN(\boldsymbol{\theta},\boldsymbol{x},t)\,.$$

For the sake of simplicity of the notations, we will yet denote by N the neural networks, which includes the contribution of the initial data. Notice that this step is not essential, but allows to simplify the loss functions. Hence at each Schwarz iteration, we solve (7), by minimizing the following *two* local "independent" loss functions $\mathcal{L}^\pm$, for some positive parameters $\lambda_{\text{Int}}^\pm$, $\lambda_{\text{Ext}}^\pm$ and where we have denoted $\boldsymbol{\theta} = (\boldsymbol{\theta}^-,\boldsymbol{\theta}^+)$. In particular, we benefit from local training processes (subdomain-dependent), which allows us to potentially avoid the use of the stochastic gradient method or/and improve its convergence. Typically, the mini-batches would actually correspond to training points for the local loss functions under consideration. At Schwarz iteration k, we hence minimize

$$\begin{aligned}\mathcal{L}^\pm(\boldsymbol{\theta}^{(k)}) \;=\; &\big\|\partial_t N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot) - \nu(\boldsymbol{x})\triangle N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot) - \boldsymbol{a}(\boldsymbol{x})\cdot\nabla N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot)\\
&-r(\boldsymbol{x})N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot)\big\|_{L^2(\Omega_\varepsilon^\pm\times(0,T))} + \lambda_{\text{Ext}}^\pm\big\|N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot)\big\|_{L^2(\Lambda_\varepsilon^\pm\times(0,T))}\\
&+\lambda_{\text{Int}}^\pm\big\|\mathcal{T}_\pm N^{\pm,(k)}(\boldsymbol{\theta}^{\pm,(k)},\cdot,\cdot) - \mathcal{T}_\pm N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\pm},\cdot,\cdot)\big\|_{L^2(\Gamma_\varepsilon^\pm\times(0,T))}\,,\end{aligned}$$

where $\overline{\boldsymbol{\theta}}^\pm = (\overline{\boldsymbol{\theta}}^{\pm,(k-1)})$ was computed at the Schwarz iteration $k-1$. Recall that practically the loss functions are numerically evaluated by approximating the norm using training points, typically randomly chosen in $\Omega_\varepsilon^\pm$. This method allows for a complete spatial decoupling of the problem over 2 (or arbitrary number of) subdomains. Finally, the reconstructed solution is hence defined as $N_{|\Omega_\varepsilon^\pm}(\cdot,t) = N^\pm(\cdot,t)$ for all $t \geqslant 0$. More specifically

$$\lim_{k\to+\infty}\Big\|\,\|N^+(\overline{\boldsymbol{\theta}}^{+,(k)},\cdot,\cdot) - N^-(\overline{\boldsymbol{\theta}}^{-,(k)},\cdot,\cdot)\|_{\infty,\overline{\Omega_\varepsilon^+}\cap\overline{\Omega_\varepsilon^-}}\Big\|_{L^2(0,T)} = 0\,, \tag{10}$$

and we define the solution to the advection-diffusion-reaction equation as

$$N = \begin{cases} N^{+,(k^{\text{cvg}})}(\overline{\boldsymbol{\theta}}^{+},\cdot,\cdot), & \text{in } \Omega_{\varepsilon}^{+}\times(0,T), \\ N^{-,(k^{\text{cvg}})}(\overline{\boldsymbol{\theta}}^{-},\cdot,\cdot), & \text{in } \Omega_{\varepsilon}^{-}\times(0,T), \end{cases}$$

where $\overline{\boldsymbol{\theta}}^{\pm} = \overline{\boldsymbol{\theta}}^{\pm,(k^{\text{cvg}})}$. Practically, in order to evaluate the loss functions, it is necessary to compute the equation at some very large $N_t N_{\boldsymbol{x}}^{\pm}$ randomly chosen input points $\{(\boldsymbol{x}_j^{\pm},t)\}_{n;j}$ in $\Omega_{\varepsilon}^{\pm}\times(0,T)$, as the L^2-norms are not exactly performed. From the optimization point of view, the method now requires minimizing two loss functions. Naturally, the two IBVPs are *totally* decoupled. As we are now considering two IBVPs on smaller spatial domains, we can locally adapt the depth of the local networks.

It is important to mention that, unlike SWR methods combined with standard numerical (finite-difference, -volume, -elements, pseudospectral) methods, for which convergence can be proven, the combination of SWR and PINN methods will not usually ensure convergence to zero of the residual history. This is due to the fact that from one Schwarz iteration to the next, the reconstructed solutions may slightly differ as the minima obtained by minimization of the local loss functions may a priori differ (even close to SWR convergence). This fact is actually inherent to the NN-based method. However, we expect the residual history to be small from a practical point of view and for loss functions sufficiently small. In addition to this argument, let us mention that the transmission condition is naturally not exactly satisfied if it is included in the loss function (and not encoded in the NN). A large enough weight can, for instance, be imposed on the transmission constraint to ensure that it is accurately satisfied.

2.2. About the interest of using SWR DDM for NN-based algorithms

In this section we discuss the computation complexity of the SWR-PINN solver in comparison with SWR algorithms for standard PDE solver. We start by recalling some basic facts about the computational complexity of PINN algorithms. For the sake of simplicity we assume that $N_{\boldsymbol{x}}$ and N_t respectively represent the number of space-time training data, and the number of space-time grid points. Similarly of we have 2 subdomains, we denote by $N_{\boldsymbol{x}}^{\pm}$ the number of spatial training points or grid points in $\Omega_{\varepsilon}^{\pm}$. Typically we can assume $N_{\boldsymbol{x}}^{\pm}\approx N_{\boldsymbol{x}}/2$

Complexity of the PINN algorithm. Generally speaking (independently of domain decomposition), the PINN algorithm requires the computation of the NN parameters obtained by minimizing the loss function (training process) as below:

- We first construct (usually refers as forward pass) the loss functions at some input data $\{(\boldsymbol{x}_j,t_n)\}_{(j,n)\in\mathbb{N}_{\boldsymbol{x}}^{(i)}\times\mathbb{N}_t}$ (resp. $\{(\boldsymbol{x}_j,t_n)\}_{(j,n)\in\mathbb{N}_{\boldsymbol{x}}^{(e)}\times\mathbb{N}_t}$) in the interior (resp. exterior) of the space-time domain. We denote by $N_{\boldsymbol{x}}^{(i)}$ (resp. $N_{\boldsymbol{x}}^{(e)}$ and N_t) the number of elements in the set $\mathbb{N}_{\boldsymbol{x}}^{(i)}$ (resp. $\mathbb{N}_{\boldsymbol{x}}^{(e)}$ and $\mathbb{N}_t$). For instance, with Dirichlet boundary conditions, the loss function (without normalization constraint) is numerically minimized in the form

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) \;=\; & \frac{1}{N_{\boldsymbol{x}}N_t}\sum\nolimits_{(j,n)\in\mathbb{N}_{\boldsymbol{x}}^{(i)}\times\mathbb{N}_t}\left|\big(\partial_t - P(\boldsymbol{x}_j,t_n)\big)\boldsymbol{N}(\boldsymbol{\theta},\boldsymbol{x}_j,t_n)\right|_2^2 \\ & + \frac{1}{N_{\boldsymbol{x}}^{(e)}N_t}\sum\nolimits_{(j,n)\in\mathbb{N}_{\boldsymbol{x}}^{(e)}\times\mathbb{N}_t}\left|\boldsymbol{N}(\boldsymbol{\theta},\boldsymbol{x}_j,t_n)\right|_2^2.\end{aligned}$$

  The parallelization of this process is straightforward.
- Minimization (backward pass) of the loss function by stochastic gradient descent (SGD) method [21–23]. The interest of SGD is to deal with (several) lower-dimensional optimization problems allowing to avoid the use of a deterministic gradient method on a high-dimensional search space. The latter requires to fix some parameters such as the size of the epochs (corresponding to the number of iterations in the model training) and a learning rate.

Practically, the forward and backward pass on mini-batches (stochastic gradient methods) are combined to reach a local minimum of the loss function. We denote by p_{G} the total number of epochs involved in the training, and by $N_{\boldsymbol{\theta}}$ the dimension of the search space, then

- The construction of the loss function (forward pass) requires $O(p_{\text{G}}N_{\boldsymbol{x}}N_tN_{\boldsymbol{\theta}})$ operations.
- The explicit gradient method update (backward pass) to minimize the loss function also requires $O(p_{\text{G}}N_{\boldsymbol{x}}N_tN_{\boldsymbol{\theta}})$ operations [21–23].

Complexity of standard evolution PDE algorithms. By standard advection-diffusion-reaction equation solvers, the main algorithmic cost is the loss function estimation and the computation of solutions to linear systems at each time iteration, involved in implicit or semi-implicit stable schemes [6]. The latter has a polynomial complexity so that, after N_t time steps, the overall complexity of a standard evolution PDE solver is $O(N_{\boldsymbol{x}}^{\alpha}N_t)$; where $1<\alpha\leqslant 3$ is typically dependent on the structure/sparsity of the matrix involved in the linear systems.

Complexity of SWR method for standard evolution PDE algorithms. Let us denote by k^{cvg} the number of Schwarz iterations necessary for the SWR algorithm to converge (with a given precision δ^{Sc}). It is easy to see that in each subdomain, the overall complexity $C_{\text{Standard + SWR}}$ for SWR algorithms combined with standard evolution PDE solvers is typically

$$C_{\text{Standard + SWR}} = O\left(k^{\text{cvg}}(N_{\boldsymbol{x}}^{\pm})^{\alpha} N_t\right).$$

Practically, it is required for k^{cvg} to be small enough. As it is well-known, the choice of the transmission conditions is a crucial element in minimizing k^{cvg}. Dirichlet transmission conditions are known to provide very slow convergence. At the opposite of the spectrum and for wave-like equations, (nonlocal) Dirichlet-to-Neumann like transmission conditions are known to provide extremely fast convergence, but are computationally complex to approximate. Another way to accelerate the convergence of the SWR algorithm consists in increasing the subdomain overlap (that is to increase ε). For the advection-diffusion-reaction equation, the optimized SWR method, based on optimized Robin-transmission conditions is a good compromise between convergence rate and computational complexity [5].

Complexity of SWR method for PINN algorithms. We denote by $N^{\pm}$ the local neural networks in $\Omega_{\varepsilon}^{\pm}$. Naturally, the larger the subdomain size, the deeper the depth of the searched local neural network associated to this subdomain. For two subdomains, the minimization step within the SWR-NN consists in solving *in parallel*

$$\overline{\boldsymbol{\theta}}^{\pm,(k)} = \text{argmin}_{\boldsymbol{\theta}\in\Theta^{\pm}} \mathcal{L}^{\pm}(\boldsymbol{\theta}^{(k)}),$$

rather than (for direct method)

$$\overline{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}\in\Theta} \mathcal{L}(\boldsymbol{\theta}),$$

where $N_{\boldsymbol{\theta}}^{\pm} := \#\Theta^{\pm} \leqslant N_{\boldsymbol{\theta}} := \#\Theta$. At a given training (forward and backward passes), the minimization of both NN hence requires $O(p_{\text{G}}^{\pm} N_{\boldsymbol{\theta}}^{\pm} N_{\boldsymbol{x};T}^{\pm})$ operations, where $p_{\text{G}}^{\pm}$ represents the number of optimization iterations in $\Omega_{\varepsilon}^{\pm}$. However, from a Schwarz iteration to the next, there is an important change compared to the standard SWR algorithm, as the PINN algorithm allows to improve the SWR algorithm by introducing some learning along the Schwarz process.
At the iteration $k+1$, we need to optimize the loss functions in $\Omega_{\varepsilon}^{\pm}$ taking into account the fact that the boundary values have been modified. Unlike standard SWR methods which require the computation of IBVP from scratch (but with updated BC), within the framework of SWR-PINN, we can initialize the NN at iteration $k+1$, using the parameters $\overline{\boldsymbol{\theta}}^{\pm,(k)}$ from the space-time approximation solution $N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)})$ in $\Omega_{\varepsilon}^{\pm}$, i.e., the optimization algorithm reads, at Schwarz iteration $k+1$ and for $\ell \geqslant 0$,

$$\boldsymbol{\theta}_{\ell+1}^{\pm,(k+1)} = \boldsymbol{\theta}_{\ell}^{\pm,(k+1)} - \nu \nabla \mathcal{L}^{\pm}(\boldsymbol{\theta}_{\ell}^{\pm,(k+1)}),$$

with $\boldsymbol{\theta}_0^{\pm,(k+1)} = \overline{\boldsymbol{\theta}}^{\pm,(k)}$.
In other words, we expect a faster convergence of the optimization algorithm as compared to the previous Schwarz iteration, at least, close to SWR convergence. By analogy, standard PDE solvers would consist in starting the optimization with random parameters $\boldsymbol{\theta}_0^{\pm,(k+1)}$. More specifically,

Proposition 2.2 *(SWR learning). We consider the algorithm SWR-PINN. Assume that at the iteration $k \geqslant 1$,*

$$\|N^{\pm,(k)} - N_{|\Omega_{\varepsilon}^{\pm}}\|_{L^2(\Omega_{\varepsilon}^{\pm}\times[0,T])} \leqslant C^k \|N^{\pm,(0)} - N_{|\Omega_{\varepsilon}^{\pm}}\|_{L^2(\Omega_{\varepsilon}^{\pm}\times[0,T])}, \tag{11}$$

for some constant $C < 1$. Then for smooth neural network $N^{\pm}$ and for a fixed precision, the number of optimization iterations $p_G^{\pm}$ is actually k-dependent (that is $p_G^{\pm}(k)$) and $p_G^{\pm}(k)$ goes to zero when k goes to infinity. More specifically, there exists a constant $D > 0$ such that

$$\left|\mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)}) - \mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)})\right| \leqslant DC^k \|N^{\pm,(0)} - N_{|\Omega_{\varepsilon}^{\pm}}\|_{L^2(\Omega_{\varepsilon}^{\pm}\times[0,T])}. \tag{12}$$

Moreover, the total computational complexity of the SWR-PINN algorithm is given by

$$C_{PINN+SWR} = O\left(N_{\boldsymbol{\theta}}^{\pm} N_{\boldsymbol{x}}^{\pm} N_t \sum_{k=1}^{k^{cvg}} p_G^{\pm}(k)\right).$$

Practically, we expect that, at least for a fixed k large enough, there exists $\ell > 1$ such that

$$p_{\text{G}}^{\pm}(k+\ell) \lesssim p_{\text{G}}^{\pm}(k).$$

Let us notice that (11) is a reasonable assumption in the standard analysis of convergence of SWR methods. Moreover, the value of C typically depends of the type of transmission conditions used at the subdomain interface. Practically, the gain in the SWR learning is also independent of the transmission condition.

Proof. Let us start by noticing that

$$N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)},\cdot,\cdot) \;=\; N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)},\cdot,\cdot) + \nabla_{\boldsymbol{\theta}} N^{\pm}(\boldsymbol{\xi}^{\pm,(k)},\cdot,\cdot)^T\big(\overline{\boldsymbol{\theta}}^{\pm,(k+1)} - \overline{\boldsymbol{\theta}}^{\pm,(k)}\big), \tag{13}$$

for some $\boldsymbol{\xi}^{\pm,(k)}$ and where the uppercase T refers to the transpose operator. The convergence acceleration then occurs as, the larger k the smaller $\|N(\overline{\boldsymbol{\theta}}^{\pm,(k+1)},\cdot,\cdot) - N(\overline{\boldsymbol{\theta}}^{\pm,(k+1)},\cdot,\cdot)\|_{L^2(\Omega_\varepsilon^{\pm}\times[0,T])}$. Moreover,

$$\mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)}) \;=\; \mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)}) + \nabla\mathcal{L}^{\pm}(\overline{\boldsymbol{\zeta}}^{\pm,(k)})^T(\overline{\boldsymbol{\theta}}^{\pm,(k+1)} - \overline{\boldsymbol{\theta}}^{\pm,(k)}),$$

for some $\overline{\boldsymbol{\zeta}}^{\pm,(k)}$. Therefore,

$$\big|\mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)}) - \mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)})\big| \;\le\; \big\|\nabla\mathcal{L}^{\pm}(\overline{\boldsymbol{\zeta}}^{\pm,(k)})\big\|\big\|\overline{\boldsymbol{\theta}}^{\pm,(k+1)} - \overline{\boldsymbol{\theta}}^{\pm,(k)}\big\|. \tag{14}$$

Then by using (13) in (14), we get

$$\begin{aligned}\big|\mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)}) - \mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)})\big| \;=\; &\big\|\nabla\mathcal{L}^{\pm}(\overline{\boldsymbol{\zeta}}^{\pm,(k)})\big\| \times \big\|\big[\nabla_{\boldsymbol{\theta}} N^{\pm}(\boldsymbol{\xi}^{\pm,(k)},\cdot,\cdot)\nabla_{\boldsymbol{\theta}} N^{\pm}(\boldsymbol{\xi}^{\pm,(k)},\cdot,\cdot)^T\big]^{-1}\\ &[N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)},\cdot,\cdot) - N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)},\cdot,\cdot)]\|_{L^2(\Omega_\varepsilon^{\pm}\times[0,T])}.\end{aligned} \tag{15}$$

Then using (11), there exist two positive constants D_1 and D_2, such that

$$\big|\mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k+1)}) - \mathcal{L}^{\pm}(\overline{\boldsymbol{\theta}}^{\pm,(k)})\big| \;\leqslant\; D_2 D_1^k \|N^{\pm,(0)} - N_{|\Omega_\varepsilon^{\pm}}\|_{L^2(\Omega_\varepsilon^{\pm}\times[0,T])}.$$

This concludes the proof. □

Hence, the smaller $\|\overline{\boldsymbol{\theta}}^{\pm,(k+1)} - \overline{\boldsymbol{\theta}}^{\pm,(k)}\|$, the faster the minimization of the loss function. Practically, we then expect the optimization process to be accelerated with increasing k. From the computational point of view, the SWR-PINN algorithm also allows i) to adapt the depth of (most of) local NNs compared to using one unique (global) NN, and ii) to estimate the local loss functions using local subdomain-dependent training data and potentially allows for using direct (none-stochastic) gradient methods for a sufficiently large number of subdomains. This step is the analog of the reduction of the size of the linear systems to be solved (scaling effect) within standard SWR when are applied with real space solvers [1–3].

2.3. Nonlocal operator

We have argued above that the use of SWR methods allows for an efficient parallel computation of the overall loss functions through the efficient estimation (using local training points) of local loss functions. We show below that whenever nonlocal terms are present in the equation, the SWR-PINN method remains unchanged. In the following, we assume that the equation contains a nonlocal operator F, typically defined as a convolution product:

- $F(u) = \partial_x^{\alpha} u$, with $\alpha > 0$, fractional derivative in space, modeling nonlocal effect [24]. The latter is actually defined as a convolution. We refer to [25] for details.
- $F(u) = \rho *_{\boldsymbol{x}} u$ where $*_{\boldsymbol{x}}$ denotes the spatial convolution product, a nonlocal potential for some given function ρ.

We consider the equation on a truncated domain $\Omega \subset \mathbb{R}^d$ with boundary Γ, as follows

$$\begin{cases}\partial_t u = \nu(\boldsymbol{x})\triangle u + F(u),\ \boldsymbol{x}\in\Omega,\ t>0,\\ u(\boldsymbol{x},0) = u_0(\boldsymbol{x}),\ \boldsymbol{x}\in\Omega,\\ u = 0,\ \boldsymbol{x}\in\Gamma,\end{cases} \tag{16}$$

and such that F is defined as a convolution product in space

$$F(u) = u *_{\boldsymbol{x}} \rho = \int_{\Omega} u(\boldsymbol{x}-\boldsymbol{y},t)\rho(\boldsymbol{y})d\boldsymbol{y}.$$

Then the SWR-PINN scheme reads

$$\begin{cases}\partial_t N^{\pm,(k)} &= \nu(\boldsymbol{x})\triangle N^{\pm,(k)} + \int_{\Omega_\varepsilon^{\pm}} N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}\\ &\quad + \int_{\Omega_\varepsilon^{\mp}} N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y},\ \text{ in } \Omega_\varepsilon^{\pm}\times(0,T),\\ N^{\pm,(k)}(\cdot,0) &= u_0^{\pm},\ \text{ in } \Omega_\varepsilon^{\pm},\\ \mathcal{T}_{\pm}N^{\pm,(k)} &= \mathcal{T}_{\pm}N^{\mp,(k-1)},\ \text{ on } \Gamma_\varepsilon^{\pm}\times(0,T),\\ N^{\pm,(k)} &= 0,\ \text{ on } \Lambda\backslash\Gamma_\varepsilon^{\pm}\times(0,T),\end{cases} \tag{17}$$

where $\overline{\boldsymbol{w}}^{\pm} \in \Theta_{\pm}$ was computed at the previous Schwarz iteration $(k-1)$, with some transmission operator $\mathcal{T}_{\pm}$. Hence in this case, we still have to minimize local loss functions

$$\begin{aligned}\mathcal{L}^{\pm}(\boldsymbol{\theta}^{(k)}) \;=\; & \big\| \partial_t N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot) - \nu(\boldsymbol{x})\triangle N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot) \\ & - \int_{\Omega_\varepsilon^{\pm}} N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y} \\ & - \int_{\Omega_\varepsilon^{\mp}} N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}\big\|_{L^2(\Omega_\varepsilon^{\pm}\times(0,T))} \\ & + \lambda_{\text{Int}}^{\pm}\big\| \mathcal{T}_{\pm}N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot) - \mathcal{T}_{\pm}N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\mp},\cdot,\cdot)\big\|_{L^2(\Gamma_\varepsilon^{\pm}\times(0,T))} \\ & + \lambda_{\text{Ext}}^{\pm}\big\| N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot)\big\|_{L^2(\Lambda_\varepsilon^{\pm}\times(0,T))}.\end{aligned}$$

Practically, we can approximate the convolution product as follows. Denoting by $\{\boldsymbol{x}_j\}_{j\in\mathcal{J}^{\pm}} \in \Omega^{\pm}$ the local spatial training points, for $\boldsymbol{x} \in \Omega_\varepsilon^{\pm}$, we approximate

$$\int_{\Omega_\varepsilon^{\pm}} N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y} + \int_{\Omega_\varepsilon^{\mp}} N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}$$

by

$$\sum_{j\in\mathcal{J}^{\pm}} c_j F\big(N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}_i-\boldsymbol{x}_j,t)\big) + \sum_{j\in\mathcal{J}^{\mp}} c_j F\big(N^{\mp,(k-1)}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x}_i-\boldsymbol{x}_j,t)\big),$$

for some weights $\{c_j\}_{j\in\mathcal{J}^{\pm}}$.

As it was discussed above the interest of using a DDM is to decompose the training and search of the local solution over smaller set of parameters. However, whenever the equation is *nonlocal*, it is necessary to extend the search of the parameters in the global computational domains. More specifically, for local equations, the local NN-solution in Ω_ε^{+} (resp. Ω_ε^{-}) only requires parameters in Θ_+ (resp. Θ_-). If the equation is nonlocal, in order to construct the solution in $\Omega_\varepsilon^{\pm}$, we have in principle to search the NN parameters in all Θ, containing both Θ_+ and Θ_-, as the solution values in $\Omega_\varepsilon^{\pm}$ depend on values of the solution in $\Omega_\varepsilon^{\mp}$. This problem would also occur to construct the loss function, using the direct PINNs method, within the term

$$\int_{\Omega} N(\boldsymbol{\theta},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}.$$

The SWR-PINN method allows to deal with this issue, as at Schwarz iteration k, the loss function in $\Omega_\varepsilon^{\pm}$ is evaluated through the solution in $\Omega_\varepsilon^{\pm}$ at the previous Schwarz iteration $(k-1)$ from $N^{\mp}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x},t)$ thanks to the previously evaluated parameter $\overline{\boldsymbol{\theta}}^{\mp}$.

2.4. How about a non-iterative domain decomposition in space?

The domain decomposition method which is derived in this paper is a SWR-in space method which is an iterative method allowing for the convergence of the decomposed solution towards the exact solution of the PDE under consideration. The main weakness of this DDM is the fact that the decoupled system has to be solved several times (iterative method). It is hence natural to ask if a "similar spatial domain decomposition", but non-iterative, is possible.
In this goal, we decompose the domain Ω as above: $\Omega_\varepsilon^{\pm}$, with or without overlap ($\Omega_\varepsilon^{+}\cap\Omega_\varepsilon^{-}=\emptyset$ or $\Omega_\varepsilon^{+}\cap\Omega_\varepsilon^{-}\neq\emptyset$), with $\varepsilon \geqslant 0$ and consider (16). That is we search for a solution of the form $N_{|\Omega_\varepsilon^{\pm}}(\cdot,t) = N^{\pm}(\overline{\boldsymbol{\theta}}^{\pm}\cdot,t)$ for all $t \geqslant 0$ such that

$$\left\{\begin{array}{ll}\partial_t N^{\pm} \;=\; \nu(\boldsymbol{x})\triangle N^{\pm} + \int_{\Omega_\varepsilon^{\pm}} N^{\pm}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y} & \\ \qquad\qquad + \int_{\Omega_\varepsilon^{\mp}} N^{\mp}(\boldsymbol{\theta}^{\mp},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}, \;\text{ in } \Omega_\varepsilon^{\pm}\times(0,T), & \\ N^{\pm} \;=\; 0, \;\text{ on } \Gamma\times(0,T), & \end{array}\right. \tag{18}$$

where $\overline{\boldsymbol{w}}^{\pm} \in \Theta_{\pm}$. The PINN-method then consists in solving

$$\overline{\boldsymbol{\theta}}^{\pm} = \text{argmin}_{\boldsymbol{\theta}\in\Theta^{\pm}}\mathcal{L}^{\pm}(\boldsymbol{\theta}),$$

where

$$\begin{aligned}\mathcal{L}^{\pm}(\boldsymbol{\theta}) \;=\; & \big\| \partial_t N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot) - \nu(\boldsymbol{x})\triangle N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot) \\ & - \int_{\Omega_\varepsilon^{\pm}} N^{\pm}(\boldsymbol{\theta}^{\pm},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y} \\ & - \int_{\Omega_\varepsilon^{\mp}} N^{\mp}(\boldsymbol{\theta}^{\mp},\boldsymbol{x}-\boldsymbol{y},t)\big)\rho(\boldsymbol{y})d\boldsymbol{y}\big\|_{L^2(\Omega_\varepsilon^{\pm}\times(0,T))} \\ & + \lambda_{\text{Ext}}^{\pm}\big\| N^{\pm,(k)}(\boldsymbol{\theta}^{\pm},\cdot,\cdot)\big\|_{L^2(\Gamma\times(0,T))}.\end{aligned}$$

Therefore, in this case, we still have to minimize local loss functions. However, there are two main issues:

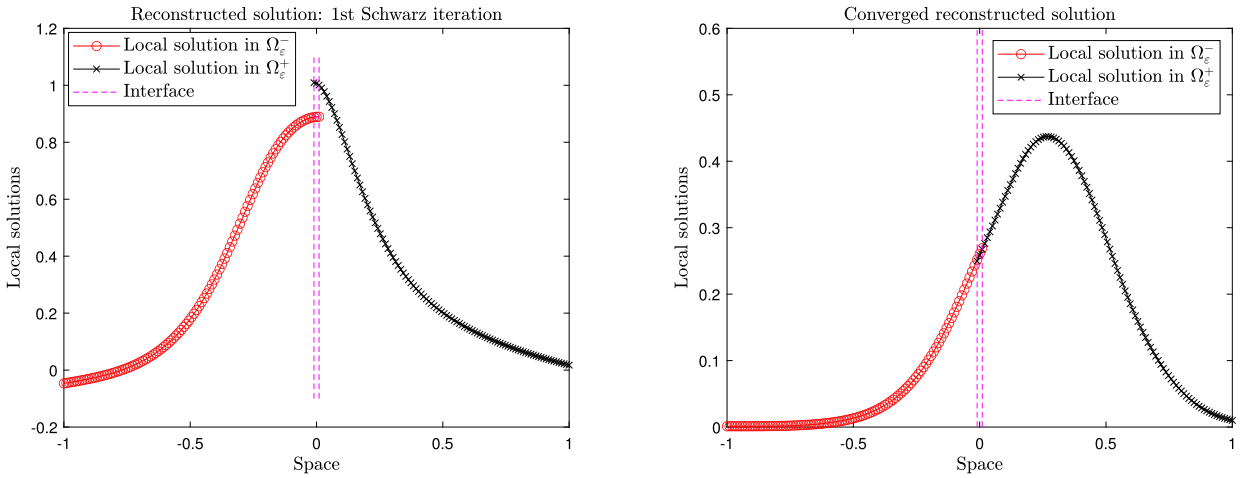

Fig. 1. Experiment 1. (Left) Reconstructed solutions after the first Schwarz iteration (Right) Reconstructed solutions at convergence of the SWR method.

- Even if ε is taken equal to zero, the decoupling of the solution in the two subdomains, naturally induces a discontinuity at the subdomain interfaces. It is possible to impose additional *compatibility conditions* (to be included in the loss function), in the form of continuity condition $N^+ = N^-$ at $\Gamma_\varepsilon^\pm$, differentiability, but the reconstructed global solution N (such that $N_{|\Omega_\varepsilon^\pm} = N^\pm$), will obviously not be an approximate solution to the equation under consideration. Moreover, the compatibility conditions will induce a re-coupling of the two systems in the spirit of the following item.
- The two systems, in N^+ and N^- are actually also coupled through the nonlocal term. This effect is similar to the addition of a compatibility condition described above. Hence, the computation of the loss functions $\mathcal{L}^\pm$ would not be embarrassingly parallel anymore. This is not an issue in the SWR framework; as in the latter case, say at Schwarz iteration k, the nonlocal term uses the approximate solution at the Schwarz iteration $k-1$, which is a known quantity.

Hence unlike the SWR-PINN method, for which (10) occurs

$$N^+_{\overline{\Omega}_\varepsilon^+ \cap \overline{\Omega}_\varepsilon^-}(\overline{\boldsymbol{\theta}}^+, \cdot, \cdot) \neq N^-_{\overline{\Omega}_\varepsilon^+ \cap \overline{\Omega}_\varepsilon^-}(\overline{\boldsymbol{\theta}}^-, \cdot, \cdot)\,. \tag{19}$$

3. Numerical experiments

In this section, we propose basic experiments in order to numerically illustrate the convergence of the overall method. The PINN-algorithm was implemented using deep-learning and optimization toolboxes from `matlab`, and `python` libraries `DeepXDE` [26] and `tensorflow` [27]. Although relatively simple, these experiments illustrate the proof of concept of the proposed strategy, but not to provide the best convergence possible (which will be the purpose of a future work).

Experiment 1. We consider the standard advection-diffusion-reaction equation

$$\partial_t u = a\partial_x u + \nu \partial_{xx} u + ru\,, \tag{20}$$

on $\Omega \times [0,T] = (-1,1) \times [0, 0.25]$ with Dirichlet boundary conditions at $x = \pm 1$ and such that $a = 0.5$, $\nu = 0.0$, $r = 0.1$ and the initial conditions are $u_0(x) = \exp(-30(x-0.1)^2)$. We decompose the domain in two subdomains $\Omega_\varepsilon^- = (-1, \varepsilon/2)$ and $\Omega_\varepsilon^+ = (-\varepsilon/2, 1)$ with $\varepsilon = 0.1$. We here use the Classical Schwarz Waveform Relaxation method, based on Dirichlet transmission conditions $N^{\pm,(k)}(\cdot, \pm\varepsilon/2, \cdot) = N^{\mp,(k-1)}(\cdot, \mp\varepsilon/2, \cdot)$, where $N^\pm$ are the two local NN defined in $\Omega_\varepsilon^\pm$. We consider the following data: the NN have both 5 layers, with 25 neurons each. We select 5000 internal collocation points. We also use a local SGM with 5 epochs and mini-batches size of 500. In the gradient method the learning rate with decay rate of 5×10^{-3} starting at 10^{-3}. We reconstruct the overall solution using a total of 5000 prediction points. Initially we take $N^{\pm,(0)} = 0$. We report the reconstructed solution after the first SWR iteration (resp. converged SWR algorithm) in Fig. 1 (Left) (resp. Right) from two the local solutions in $\Omega_\varepsilon^\pm$ at final time $T = 0.25$, and overlapping zone of size 1/99.

The SWR convergence rate is defined as the slope of the logarithm of the residual history according to the Schwarz iteration number, that is $\{(k, \log(\mathcal{E}^{(k)})) : k \in \mathbb{N}\}$, with (for 2 subdomains)

$$\mathcal{E}^{(k)} := \sum_{i=1}^{2} \Big\| \|N^{+,(k)} - N^{-,(k)}\|_{\infty;\overline{\Omega}_\varepsilon^+ \cap \overline{\Omega}_\varepsilon^-} \Big\|_{L^2(0,T)} \leqslant \delta^{\mathrm{Sc}}, \tag{21}$$

δ^{Sc} being a small parameter.

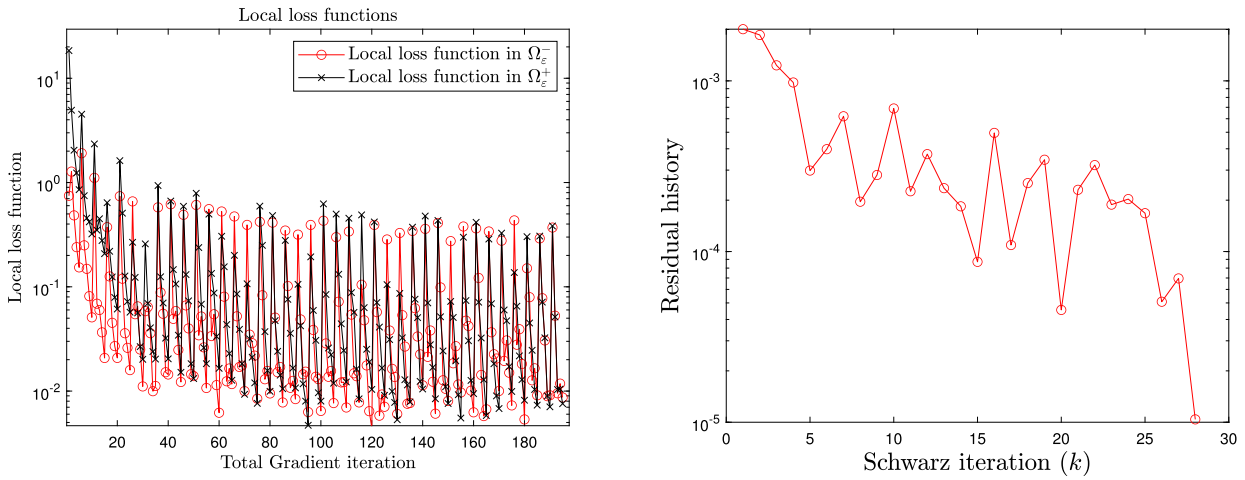

Fig. 2. Experiment 1. (Left) Convergence of stochastic gradient methods for each local loss functions. (Right) Convergence of SWR algorithm. Graph of SWR residual history.

We report in Fig. 2 (Left) the graph of convergence of the stochastic gradient methods applied to each local loss functions. Notice that each "oscillation" corresponds to a new Schwarz iteration. We report in Fig. 2 (Right) the graph of convergence of the SWR-method in the form of the residual history in the overlapping zone. The combination of convergent SWR methods with standard numerical (finite element, finite-difference) methods for which there is a uniform convergence to zero of the residual history as a function of the Schwarz iterations.

Notice that, unlike the converged SWR method combined with standard numerical methods where the residual goes to zero when k goes to infinity, the residual does not (exactly) go to zero. This is due to the fact that from one Schwarz iteration to the next, the (local) solution is obtained by constructing "new" local minima as the local loss functions are small but *not* null, and hence change from one iteration to the next.

Experiment 2. In the following experiment, we implement a Robin-SWR method for solving (20), which is expected to provide better convergence than CSWR at least at the continuous level, [5]. As it was discussed in [5] and recalled above, the optimized SWR (and more generally Robin-SWR) methods is convergent, even without overlap, that is when ε is null. We consider the same equation as above with $a = 1$, $r = 0$, $\nu = 5 \times 10^{-2}$ and $T = 0.5$. The initial condition is $u_0(x) = 10\exp(-4x^2)$. We decompose the domain in two subdomains $\Omega_\varepsilon^- = (-4, 0)$ and $\Omega_\varepsilon^+ = (0, 4)$, with hence $\varepsilon = 0$. The Robin transmission conditions $(\lambda \pm \partial_x)N^{\pm,(k)}(\cdot, 0, \cdot) = (\lambda + \partial_x)N^{\mp,(k-1)}(\cdot, 0, \cdot)$, where $N^\pm$ are the two local NN defined in $\Omega_\varepsilon^\pm$ and we have taken $\lambda = 5$. We consider the following data: the NN have both 8 layers, with 30 neurons each. We select 5000 internal collocation points. We also use local SGM with 5 epochs and mini-batches size of 500. In the gradient method the learning rate with decay rate of 10^{-3} starting at 10^{-3}. We reconstruct the overall solution using a total of 5000 prediction points. Initially we take $N^{\pm,(0)} = 0$. We report the reconstructed solution after the first SWR iteration (resp. converged SWR algorithm) in Fig. 3 (Left) (resp. 3 (Right)) from two the local solutions in $\Omega_\varepsilon^\pm$ at final time $T = 0.5$.

We next report in Fig. 4 (Left) the graph of convergence of the stochastic gradient methods applied to each local loss functions. We report in Fig. 4 (Right) the graph of convergence of the SWR-method in the form of the residual history in the overlapping zone.

Importantly and as expected, we observe that Robin-SWR-PINN still converges even if the two subdomains do not overlap.

Experiment 2bis. In the following we implement a non-overlapping 2-domain Robin-SWR experiment. We now consider that the diffusion coefficient is *space-dependent*: $\nu(x) = 0.1$ (resp. 2.5×10^{-2}) for $x \in \Omega_\varepsilon^+$ (resp. $x \in \Omega_\varepsilon^-$). The rest of the data are as follows: $a = 1$, $r = 0$, $\nu = 5 \times 10^{-2}$, $T = 0.1$, $\lambda = 5$. The initial condition is given by $u_0(x) = 10\exp(-3(x-1)^2)\cos(10(x-1)^2)$ and is such that the solution has a very different structure in the two subdomains. We want here to illustrate the ability of the derived approach to select different depths of the local neural networks, depending on the structure of the solution: in Ω_ε^- (resp. Ω_ε^+) the solution is mainly null (resp. oscillatory), except close to the interface. The two subdomains are $\Omega_\varepsilon^- = (-2, 0)$ and $\Omega_\varepsilon^+ = (0, 2)$, with hence $\varepsilon = 0$. The two local NN $N^\pm$, over $\Omega_\varepsilon^\pm$ have the following structure: N^- (resp. N^+) possesses 3 (resp. 10) layers and 10 (resp. 50) neurons. *The minimization process in N^- is much more efficiently performed than in N^+ with a relatively similar accuracy*. As above, we select 5000 internal collocation points. We also use local SGM with 5 epochs and mini-batches size of 5000. In the gradient method the learning rate with decay rate of 10^{-3} is starting at 10^{-3}. We reconstruct the overall solution using a total of 20000 prediction points. Initially we take $N^{\pm,(0)} = 0$. We report the reconstructed solution after the first SWR iteration (resp. converged SWR algorithm) in Fig. 5 (Top-Left) (resp. 5 (Top-Right)) from two the local solutions in $\Omega_\varepsilon^\pm$ at final time $T = 0.1$. We also zoom in (5, (Bottom-Left)), in the interface region to better observe the SWR convergence. The local loss functions are represented in Fig. 5

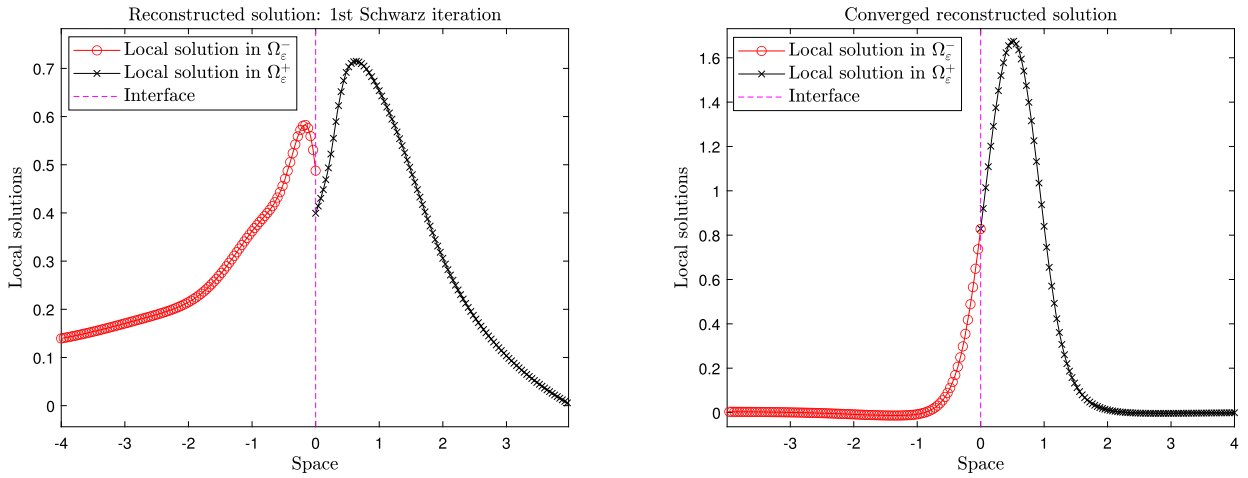

Fig. 3. Experiment 2. (Left) Reconstructed solutions after the first Schwarz iteration. (Right) Reconstructed solutions at convergence of the SWR method.

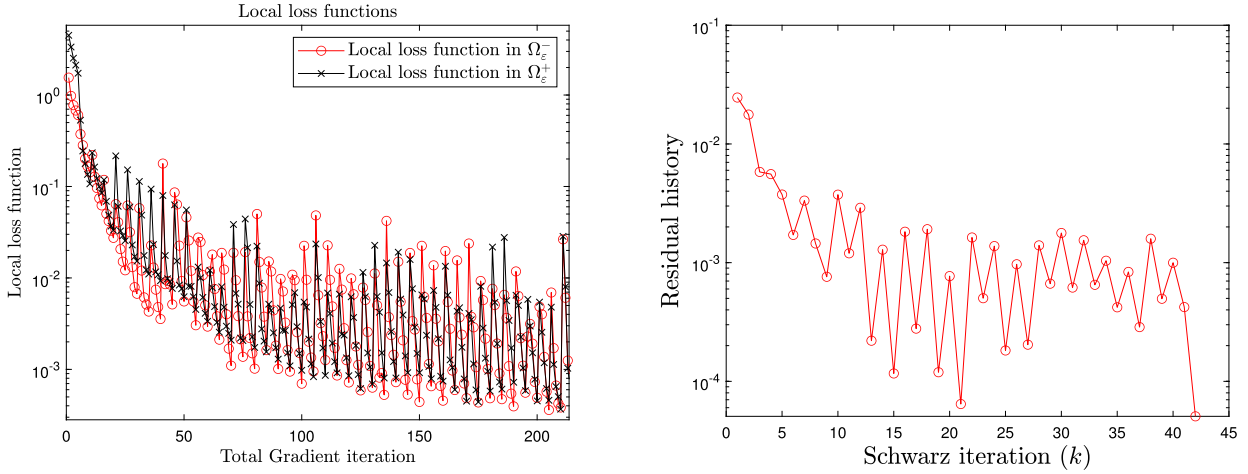

Fig. 4. Experiment 2. (Left) Convergence of stochastic gradient methods for each local loss functions. (Right) Convergence of SWR algorithm.

(Bottom-Right). We observe that roughly, the computational time to perform the solution in Ω_ε^+ was 2.5 times faster than in Ω_ε^-.

Experiment 3. In this last experiment, we consider a two-dimensional advection-diffusion equation on a square $[-1,1]^2 \times [0,T]$.

$$\partial_t u + \mathbf{a}\cdot\nabla u - \nu\triangle u = 0\,,$$

with $\mathbf{a} = (-0.5, 0)^T$ and $\nu = 0.1$ and $T = 2\pi/10$. The two subdomains are $\Omega_\varepsilon^+ = (-1,\varepsilon)\times(-1,1)$ and $\Omega_\varepsilon^+ = (-1,\varepsilon)\times(-1,1)$ where $\varepsilon = 0.1$; hence the interfaces are located at ± 0.1. The initial data is a Gaussian function $u_0(\boldsymbol{x}) = \exp(-5\|\boldsymbol{x}\|^2)$ and the final computational time is $T = 0.1$. A classical SWR algorithm is here combined with the PINN method. On the other subdomain boundaries, we impose null Dirichlet boundary conditions. The equation is solved using the library `DeepXDE` [26] combined with `tensorflow` [27]. In each subdomain is used a neural network with 3 layers and 10 neurons; Adam's optimizer is used (with learning rate 10^{-3}, and epoch size 10^3) along with tanh activation function. In Fig. 6 (Top), we report the initial data in $\Omega_\varepsilon^\pm$. In Fig. 6, we represent the solution at the end of the first Schwarz iteration (Left) and fully converged solution (Right) at final time T. In future works, we will propose more advanced simulations. The corresponding code will be made available on `GitHub`, where the interested reader could find all the relevant information regarding the code. One of the main interests of NN-based algorithms is that it allows to easily consider more complex PDEs without any additional fundamental stability or convergence issues (although this will increase the computational cost by the optimization problem). For instance, we consider below a modified quasi-linear (Burgers-like) equation

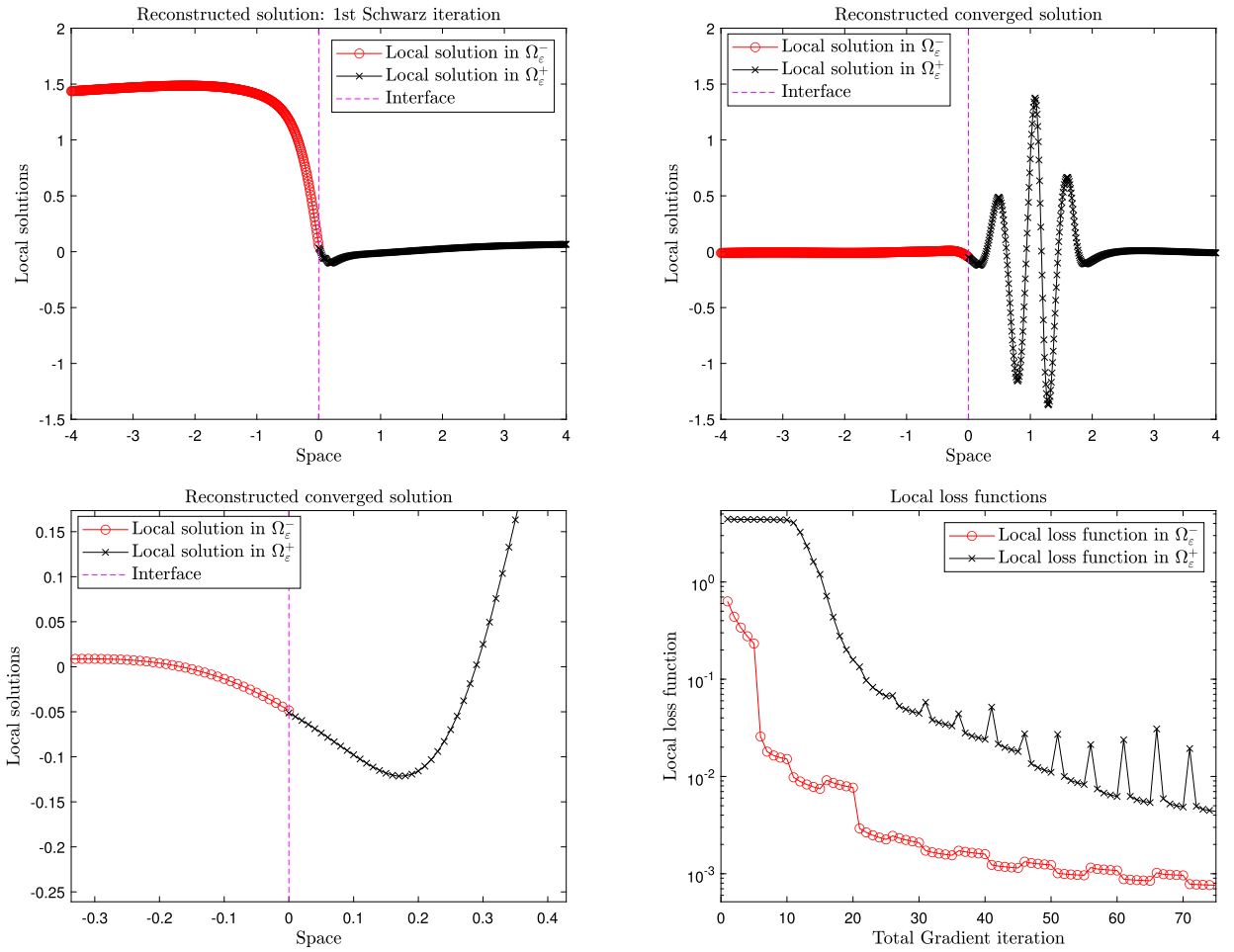

Fig. 5. Experiment 2bis. (Top-Left) Converged solutions after the first Schwarz iteration. (Top-Right) Reconstructed converged solution. (Bottom-Left) Zoom in of the converged solution in the interface region. (Bottom-Right) Local loss function convergence.

$$\partial_t u + \boldsymbol{a}(u)\cdot\nabla u - \nu\triangle u + r(\boldsymbol{x})u = 0\,,$$

with $r(\boldsymbol{x}) = 0.5\exp(-2\|\boldsymbol{x}\|^2)$, $\boldsymbol{a}(u) = (u,u)^T + (0.5,0.5)^T$. We have taken $u_0(\boldsymbol{x}) = \exp(-25\|\boldsymbol{x}\|^2)$ and $\nu = 0.05$. Notice that the algorithm remains identical. We show the convergence of the Schwarz algorithm and of the PINN algorithm in Fig. 7 (Top), which gives the ℓ_2-norm of $\|u^{+,(k)}(\cdot,T) - u^{-,(k)}(\cdot,T)\|_{L^2(\Omega_\varepsilon^-\cap\Omega_\varepsilon^+)}$ as a function of k. We present the reconstruction solution after two and eight Schwarz iterations in Fig. 7 (Bottom).

4. Conclusion

In this paper, we have derived a Schwarz Waveform Relaxed Physics Informed Neural Networks (SWR-PINN) method for solving advection-diffusion-reaction equations in parallel. Some preliminary illustrating experiments are presented to validate the approach.

4.1. Pros. and cons. of the SWR-PINN method

We summarize below the pros and cons of the proposed method.

Pros.

- Acceleration of the overall algorithm thanks to NN initialization (ansazt) from one Schwarz iteration to the next. We can even think about local training and testing closer and closer to the subdomain interfaces with increasing Schwarz iteration.
- Parallel construction of local neural networks with adaptive depth and complexity.

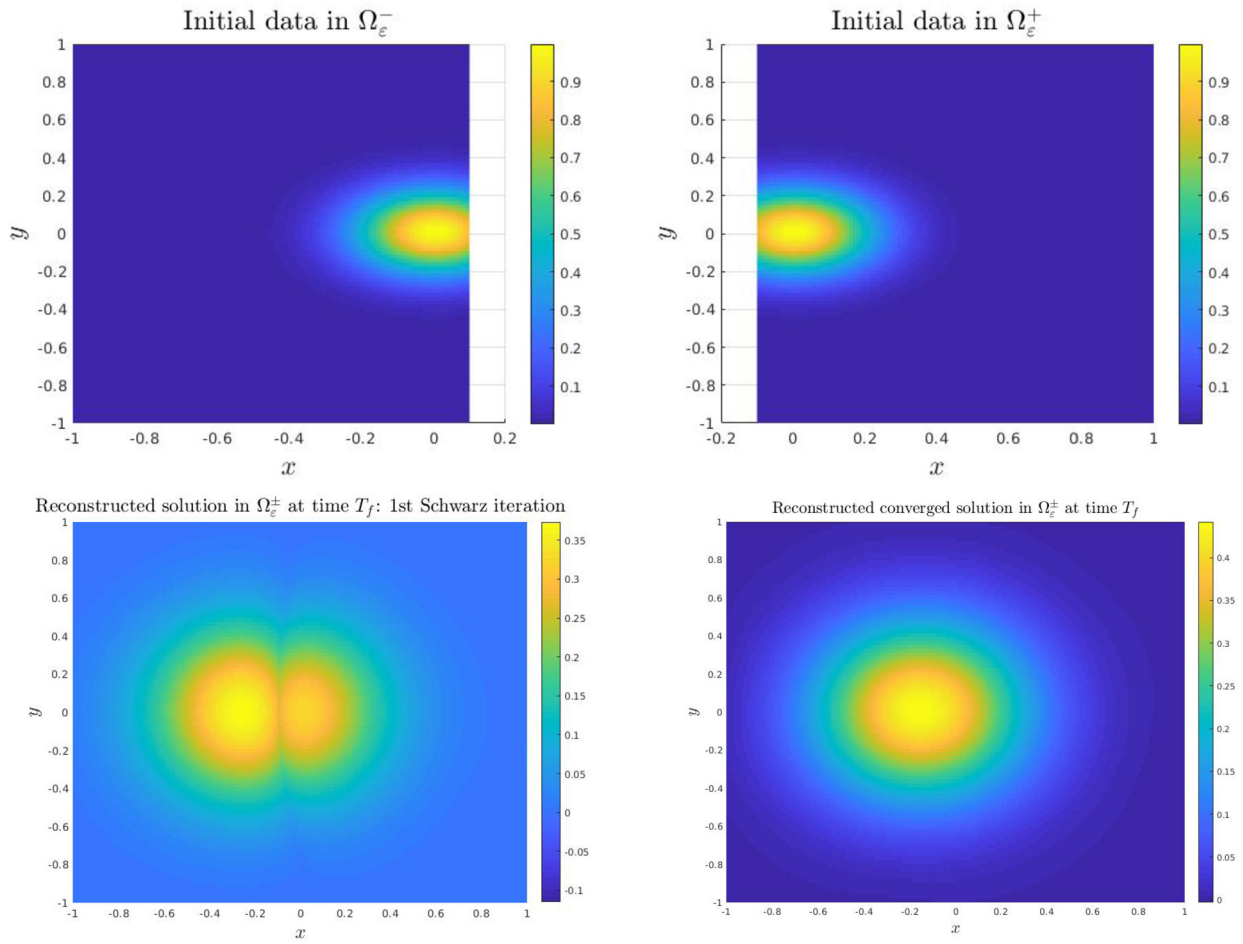

Fig. 6. Experiment 3. (Top) Initial data in $\Omega_\varepsilon^\pm$. (Bottom-Left) Solutions after the first Schwarz iteration at time T. (Bottom-Right) Reconstructed converged solution at time T.

- For convergent PINN algorithms, the SWR-PINN is convergent.

Cons.

- As a fixed point method, SWR methods require several iterations.
- The transmission conditions must be accurately satisfied through a penalization term in the loss function in order to accurately implement the SWR algorithm. Ideally, we should directly include the transmission within the definition in the NN. This is possible, considering the CSWR (Dirichlet-based transmission conditions) method and the following NN, $T^\pm$

$$T^{\pm;(k)}(\boldsymbol{\theta},\boldsymbol{x},t) = N^{\mp;(k-1)}_{|\Gamma_\varepsilon^\pm\times(0,T)}(\overline{\boldsymbol{\theta}}^{\mp},\boldsymbol{x},t) + \big(\mathbf{1}_{\Gamma_\varepsilon^\pm\times(0,T)}(\boldsymbol{x},t)-1\big)N^{\pm;(k)}(\boldsymbol{\theta},\boldsymbol{x},t)\,.$$

- Convergence or high precision of the overall algorithm can be hard to reach if the PINN algorithm is not used with sufficiently high precision. Instable numerical behavior can also be observed with the CSWR method.

4.2. Concluding remarks and future investigations

As far as we know, this paper is the first attempt to combine the SWR and PINN methods. Although the theory of SWR-DDM is now well developed in terms of convergence and convergence rate for different types of evolution PDE and their approximation with finite difference and finite element methods, the theory of convergence of PINN is not yet complete. Consequently, the convergence of the overall SWR-PINN method is still subject to the proof of convergence of the latter, which is largely empirically established. In future works, we plan to study "real-life" applications, focusing on the learning of SWR using NN-based algorithms.

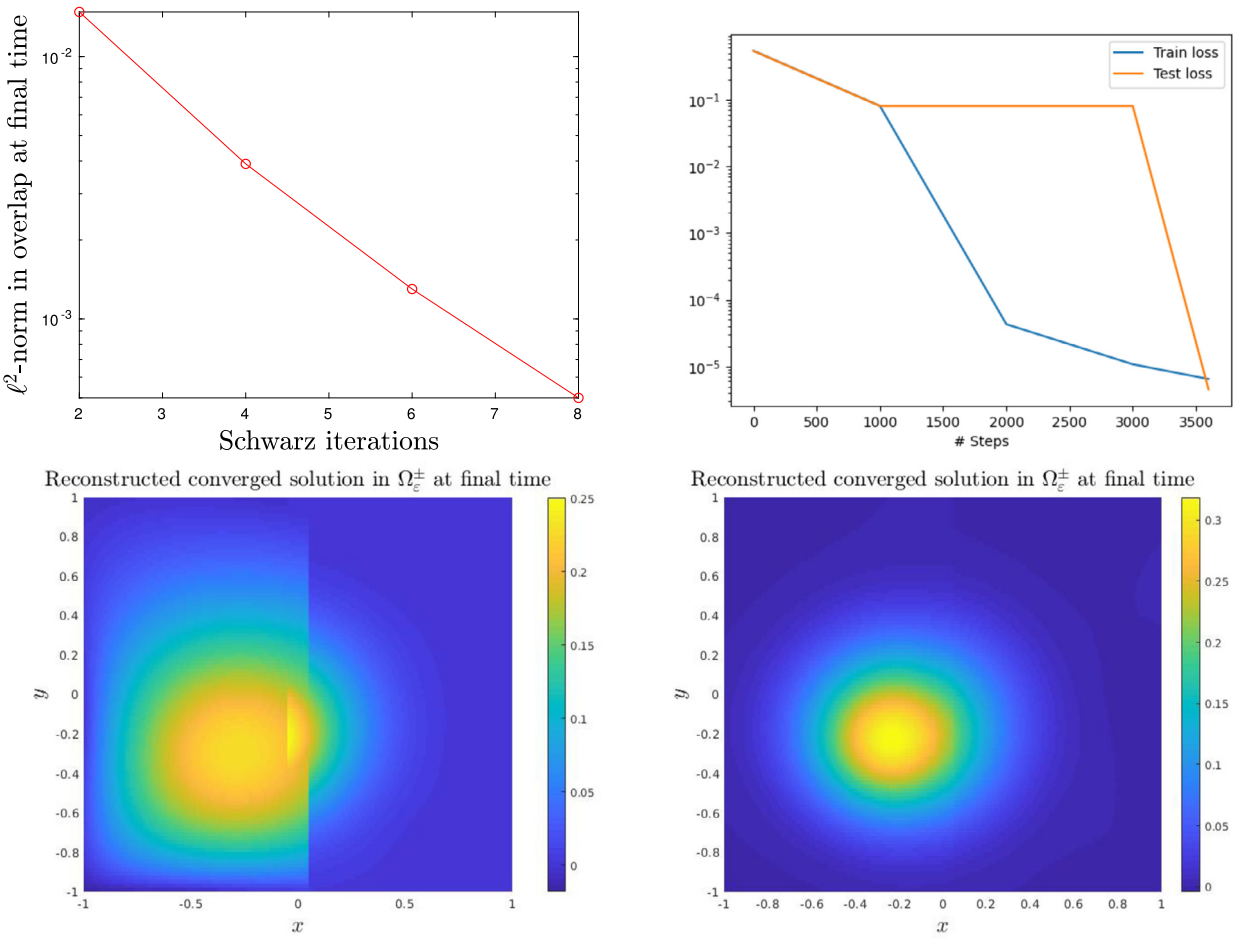

Fig. 7. Experiment 3. (Top-Left) Schwatz algorithm convergence graph. (Top-Right) Train and test loss. (Bottom-Left) Reconstructed solution after 2 Schwarz iterations at final time. (Bottom-Right) Reconstructed solution after 8 Schwarz iterations at time T.

CRediT authorship contribution statement

Both authors were equally involved in all the aspects of this work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

X.Y. was partially supported by the NSF grants DMS-1818592 and DMS-2109116. E.L. was partially supported by NSERC through the Discovery Grant program RGPIN-2018-05321. E.L. would also like to thank the Department of Mathematics at the University of California Santa Barbara and the Institute of Pure and Applied Mathematics from the University of California Los Angeles for their financial and administrative supports.

References

[1] X. Antoine, E. Lorin, Multilevel preconditioning technique for Schwarz waveform relaxation domain decomposition method for real- and imaginary-time nonlinear Schrödinger equation, Appl. Math. Comput. 336 (2018) 403–417.

[2] X. Antoine, E. Lorin, An analysis of Schwarz waveform relaxation domain decomposition methods for the imaginary-time linear Schrödinger and Gross-Pitaevskii equations, Numer. Math. 137 (4) (2017) 923–958.
[3] X. Antoine, F. Hou, E. Lorin, Asymptotic estimates of the convergence of classical Schwarz waveform relaxation domain decomposition methods for two-dimensional stationary quantum waves, ESAIM: Math. Model. Numer. Anal. 52 (4) (2018) 1569–1596.
[4] X. Antoine, E. Lorin, On the rate of convergence of Schwarz waveform relaxation methods for the time-dependent Schrödinger equation, J. Comput. Appl. Math. 354 (2019) 15–30.
[5] L. Halpern, J. Szeftel, Optimized and quasi-optimal Schwarz waveform relaxation for the one-dimensional Schrödinger equation, Math. Models Methods Appl. Sci. 20 (12) (2010) 2167–2199.
[6] M. Gander, L. Halpern, Optimized Schwarz waveform relaxation methods for advection reaction diffusion problems, SIAM J. Numer. Anal. 45 (2) (2007).
[7] M.J. Gander, L. Halpern, F. Nataf, Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation, in: Proceedings of the 11th International Conference on Domain Decomposition, 1999, pp. 27–36.
[8] X. Antoine, E. Lorin, Asymptotic convergence rates of Schwarz waveform relaxation algorithms for Schrödinger equations with an arbitrary number of subdomains, Multiscale Sci. Eng. 1 (1) (2019) 34–46.
[9] M.J. Gander, Optimal Schwarz waveform relaxation methods for the one-dimensional wave equation, SIAM J. Numer. Anal. 41 (2003) 1643–1681.
[10] M.J. Gander, Optimized Schwarz methods, SIAM J. Numer. Anal. 44 (2006) 699–731.
[11] V. Dolean, P. Jolivet, F. Nataf, An Introduction to Domain Decomposition Methods: Theory and Parallel Implementation, 2015.
[12] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[13] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (4) (2019) A2603–A2626.
[14] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, SIAM J. Sci. Comput. 42 (1) (2020) A292–A317.
[15] A.D. Jagtap, G.E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041.
[16] A. Heinlein, A. Klawonn, M. Lanser, J. Weber, Combining machine learning and domain decomposition methods for the solution of partial differential equations—a review, GAMM-Mitt. 44 (1) (2021).
[17] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[18] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. 91 (Dec 2019) 045002.
[19] G. Carleo, M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science 355 (6325) (2017) 602–606.
[20] B. Després, Analyse numérique et neural networks, Technical report, Université de Paris, 2021, https://www.ljll.math.upmc.fr/despres.
[21] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (1951) 400–407.
[22] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Physica-Verlag/Springer, Heidelberg, 2010, pp. 177–186.
[23] S. Sun, Z. Cao, H. Zhu, J. Zhao, A survey of optimization methods from a machine learning perspective, IEEE Trans. Cybern. 50 (8) (2020) 3668–3681.
[24] A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, Z. Mao, W. Cai, M.M. Meerschaert, M. Ainsworth, G.E. Karniadakis, What is the fractional Laplacian? A comparative review with new results, J. Comput. Phys. 404 (2020).
[25] E.C. De Oliveira, J.A. Tenreiro Machado, A review of definitions for fractional derivatives and integral, Math. Probl. Eng. (2014) 2014.
[26] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, Deepxde: a deep learning library for solving differential equations, 2020.
[27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, TensorFlow: large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org.