Journal of Computational Science 91 (2025) 102684

Contents lists available at ScienceDirect

Journal of Computational Science

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

An h-adaptive collocation method for Physics-Informed Neural Networks

Jan Trynda [a], Paweł Maczuga [a], Albert Oliver-Serra [b], Luis Emilio García-Castillo [c], Robert Schaefer [a], Maciej Woźniak [a,*]

[a] AGH University of Kraków Institute of Computer Science al. A. Mickiewicza 30, 30-059 Kraków, Poland
[b] University Institute of Intelligent Systems and Numeric Applications in Engineering (SIANI), University of Las Palmas de Gran Canaria (ULPGC), Spain
[c] Department of Signal Theory and Communications, Universidad Carlos III de Madrid, 28911 Madrid, Spain

ARTICLE INFO

Keywords:
Physics-Informed Neural Networks
Residual-based adaptive sampling
Advection-dominated problems
Almost-singular Poisson problems

ABSTRACT

Despite their flexibility and success in solving partial differential equations, Physics-Informed Neural Networks (PINNs) often suffer from convergence issues, even failing to converge, particularly in problems with steep gradients or localized features. Several remedies have been suggested to solve this problem, but one of the most promising is the dynamical adaptation of the collocation points. This paper explores a novel adaptive sampling method, of a stochastic nature, based on the Adaptive Mesh Refinement used in the Finite Element Method. The error estimates in our refinement algorithm are based on the value of the residual loss function. We tested our method against a variety of 1D and 2D benchmark problems that exhibit steep gradients near certain boundaries, with promising results.

1. Introduction

Physics-informed neural networks (PINNs) are a recent development in the machine learning community that provide a viable alternative to classical numerical methods to solve partial differential equations (PDEs) [1–10]. Instead of using an extensive database, as in data-driven deep learning approaches, PINNs use the problem's physics (the set of PDEs) as the neural network's loss function. In the 1990s, the potential of neural networks as universal function approximators to solve PDEs was already recognized [11,12]. However, it is only now, with the vast advances in computational capabilities, training algorithms [13] and automatic differentiation methods [14–16], that PINNs have become widely available.

Hence, the basic idea of PINNs is to train the neural network to minimize the error regarding the PDE residual and the boundary and initial conditions in a set of so-called collocation points.

Despite their success in a wide range of scientific applications, PINNs sometimes fail to converge to the correct solution. Various motives may explain this behavior. Some authors consider the problem related to the imposition of initial and or boundary conditions, and have modified the corresponding weights in the loss function [17,18] or have strongly included the boundary conditions in the PDE formulation [19]; others found that the problem lies in the stiffness of the gradient flow dynamics [20,21], and other authors have associated it with the location of the collocation points used during PINN training [22,23]. They note that the collocation points in the domain interior can only compute the PDE residual without considering the initial and boundary conditions. Therefore, the solution is not unique at these interior points, and the PINN may converge to a trivial solution. In this paper, we attack the problem from this angle and propose an adaptive sampling of collocation points to mitigate this issue

To improve the placement of collocation points in Physics-Informed Neural Networks (PINNs), researchers have proposed non-uniform and adaptive sampling strategies. In particular, in [7], the authors propose a non-uniform distribution that clusters the collocation points in regions where the solution exhibits steep gradients. Although effective, this approach requires *a priori* knowledge about the behavior of the solution and manual intervention to identify regions of interest. In contrast, adaptive strategies automate this process by dynamically concentrating the collocation points in areas where the loss function is relatively large. For example, in [24], Lu et al. introduce the *Residual-Based Adaptive Refinement* (RAR) technique, inspired by the Adaptive Mesh Refinement (AMR) method widely used in the Finite Element Method (FEM). An alternative adaptive strategy is presented in [25], where the collocation points are distributed according to a probability density function (PDF) that is proportional to the residual of the partial differential equation (PDE). A comparative study of these strategies, along with proposed refinements, is conducted in the recent work of Wu et al. [26].

Building on these adaptive approaches, in [27], the authors demonstrate that PINN training may not converge to the true solution due to

* Corresponding author.
E-mail addresses: pmaczuga@agh.edu.pl (P. Maczuga), albert.oliver@ulpgc.es (A. Oliver-Serra), legcasti@ing.uc3m.es (L.E. García-Castillo), schaefer@agh.edu.pl (R. Schaefer), macwozni@agh.edu.pl (M. Woźniak).

https://doi.org/10.1016/j.jocs.2025.102684
Received 22 February 2025; Received in revised form 2 July 2025; Accepted 16 July 2025
Available online 25 July 2025

the insufficient propagation of information from the boundary or initial conditions into the interior of the domain. This phenomenon results in narrow regions with large residuals surrounded by areas of otherwise small errors. To mitigate this issue, the authors propose the *Retain-Resample-Release* (R3) sampling algorithm. The R3 strategy incorporates three key mechanisms: (i) *Retaining* collocation points located in high-residual regions, (ii) *Resampling* a portion of points to ensure a uniform distribution across the domain, and (iii) *Releasing* those points where the residual has decreased and is no longer significant. For time-dependent problems, the authors introduce a causal variant, termed *Causal R3*, which prevents the progression of collocation points over time until the information from preceding states has been adequately propagated throughout the spatial domain.

This paper proposes a new adaptive strategy that adopts algorithms inspired by Adaptive Mesh Refinement in the FEM community [28]. Although one of the features of PINNs is their meshless nature, we propose starting with a coarse mesh of the domain and refining the elements where the residual is larger than a certain tolerance. This strategy conceptually resembles the idea of the R3 algorithm proposed in [27]. The elements with a high residual will be refined between training iterations (Retain), while the elements where the error decreases will not be refined again (Release). Additionally, the mesh structure ensures a uniform distribution over the domain (Resample). Our method combines two seemingly opposing strategies: a deterministic h-FEM inspired algorithm that refines the mesh in high-residual regions and a stochastic sampling approach that smooths out collocation points across the domain. We tested our method on a series of 1D and 2D problems that exhibit steep gradients near some of their boundaries. To validate our approach, for each problem, we compared the time and number of epochs required to train a Neural Network with identical hyperparameters, using our adaptive sampling strategy and a non-adaptive sampling strategy. To ensure reliable results, we trained the Neural Network multiple times for each strategy and evaluated the statistical measures. Specifically, we compare the mean and median training times, as well as the mean and median number of epochs, between the proposed strategy and a non-adaptive sampling strategy. The code is provided by the authors.[1]

2. The idea of PINN

Let us define the class of forward PDE problems that allows us to clearly pose the idea of the PINN method. This formulation can easily be extended to other similar problems. We denote by $\Omega \subset \mathbb{R}^n$ the bounded set with a positive measure and the Lipschitz boundary $\partial\Omega$ (e.g., a composition of $(n-1)$–dimensional smooth surfaces, without infinitely sharp junctions), so that $\overline{\Omega}$ is a compact set.

$$\text{Find } u \in C^m(\overline{\Omega});\, A(u(x)) = f(x)\ \forall x \in \Omega,\, B(u(x)) = b(x)\ \forall x \in \partial\Omega \tag{1}$$

where $A : C^m(\overline{\Omega}) \to C(\overline{\Omega})$, $f \in C(\overline{\Omega})$, $B : \{C^m(\overline{\Omega})|\partial\Omega \to C(\partial\Omega)\}$, $b \in C(\partial\Omega)$, and m is large enough to satisfy the requirements of the PDE operators A and B. The spaces $C^m(\overline{\Omega}), C(\overline{\Omega})$, and $C(\partial\Omega)$ are normed spaces equipped with the infinity norm denoted by $\|\cdot\|_{\infty,m,\Omega}$, $\|\cdot\|_{\infty,\Omega}$, and $\|\cdot\|_{\infty,\partial\Omega}$, respectively. (see, e.g. Schwartz Laurent; Analyze Mathematique, Hermann, Paris 1967 for details of the norm definitions [29]).

The residual of the problem (1) is a composition of two functions:

$$C^m(\overline{\Omega}) \ni v \to res_{\Omega}(v) = (A(v) - f) \in C(\Omega) \tag{2}$$

$$C^m(\overline{\Omega}) \ni v \to res_{\partial\Omega}(v) = (B(v) - b) \in C(\partial\Omega) \tag{3}$$

Of course, if $u \in C^m(\overline{\Omega})$ is a solution to (1), then $res_{\Omega}(u) = 0$ and $res_{\partial\Omega}(u) = 0$ are continuous functions in Ω and $\partial\Omega$, respectively. Because $C(\overline{\Omega})$ and $C(\partial\Omega)$ are subspaces of the Lebesgue spaces $L^2(\Omega)$, and $L^2(\partial\Omega)$, we can use their respective norms $\|res_{\Omega}(w)\|_{2,\Omega}$, and $\|res_{\partial\Omega}(w)\|_{2,\partial\Omega}$ to measure the error of the residual components (2), (3) for any approximate solution $w \in C^m(\overline{\Omega})$ to (1). We refer to [30] for details concerning function spaces, their topology, and PDEs.

Hypothesis 1. Let us assume that the operators A and B are continuous in adequate topologies, and the exact problem (1) possesses a unique solution $u \in C^m(\overline{\Omega})$.

The above hypothesis is satisfied for almost all PDE-BV problems well posed in the sense of Hadamard.

Now, we introduce the set of functions

$$G = \{g_\theta : \Omega \to \mathbb{R},\ \theta \in \mathbb{R}^{N_G}\} \tag{4}$$

being the realizations of a single predefined layered ANN architecture composed of a fixed number of layers. The input layer contains n neurons, and the output layer contains a single neuron. All neurons are equipped with the same activation function. The neurons are totally connected between two consecutive layers.

The realizations g_θ are indexed by their learnable parameters (weights, biases) denoted by $\theta \in \mathbb{R}^{N_G}$, where N_G is the number of such parameters of the architecture. (see, e.g. [31] for ANN's background).

Hypothesis 2. We assume that $G \subset C^m(\overline{\Omega})$, which allows us to compute the operators $res_{\Omega}, res_{\partial\Omega}$ for all PINN functions $g_\theta \in G$.

Hypothesis 3. The mapping $T : \Theta \ni \theta \to T(\theta) = g_\theta \in G$ returning the PINN instance of the assumed architecture is continuous from topology in $\mathbb{R}^{N_G}$ to the topology in $C^m(\overline{\Omega})$ imposed by the norm $\|\cdot\|_{\infty,m,\Omega}$.

The above Hypotheses 2 and 3 are satisfied for a broad class of ANN architectures in which the activation functions are at least m–times continuously differentiable.

Hypothesis 4. Learnable parameters θ of all realizations belong to the admissible compact set $\Theta \subset \mathbb{R}^{N_G}$.

Definition 2.1. The PINN approximation of (1) will be $g_{\hat{\theta}} \in G$ where $\hat{\theta} \in \Theta$ that satisfies

$$\hat{\theta} = \arg\min_{\theta\in\Theta}\{\|res_{\Omega}(g_\theta)\|^2_{2,\Omega} + \|res_{\partial\Omega}(g_\theta)\|^2_{2,\partial\Omega}\} \tag{5}$$

Remark 1. Given Hypotheses 1–4 the problem (5) admits at least one solution.

The above Remark 1 is a direct consequence of the Weierstrass extreme value theorem (see, e.g., [29]) applied to the composition of continuous functions (PINN instantiation mapping T, PDE operators A and B, norms $\|\cdot\|_{2,\Omega}$, $\|\cdot\|_{2,\partial\Omega}$ and basic algebraic operations) that are continuous in the compact set Θ.

Remark 2. The solution of the problem (5) respecting Hypotheses 1–4 will be unique if the exact solution u belongs to G. Generally, the number of solutions $\hat{\theta}$ to (5) may be larger than one because G is not a convex set. Moreover $T(\hat{\theta})$ is not necessarily the best approximation of the exact solution u by the ANN realization belonging to G, which in the general case (omitting Hypothesis 4) does not exist (see [32] for details).

The solution of the above PINN problem will be obtained by a proper machine learning routine using loss functions, which represent values of both norms appearing in (5).

The collocation points in Ω and $\partial\Omega$ can be sampled using a specific probability distribution. Because in the implementation only a finite number of points from such sets is available (because of the restricted accuracy of real number representation), multiple sampling of a single point might occur.

[1] https://github.com/JanTry/PINN_HP.

This motivates us to use the populations (multisets) of collocation points that gather clones of elements from Ω and $\partial\Omega$. The population of the collocation points in Ω will be represented by the pair $P = (\Omega, \eta_P)$ where $\eta_P : \Omega \to \mathbb{N} \cup \{0\}$ so that $\eta_P(x)$ returns the number of clones of $x \in \Omega$ belonging to P. The number of clones contained in a multiset $P = (\Omega, \eta_P)$ can be calculated as follows $\sum_{x \in supp(\eta_P)} \eta_P(x)$ while $supp(\eta_P) = \overline{\{x \in \Omega; \eta_P \neq 0\}}$.

Analogously $P = (\partial\Omega, \eta_P), \eta_P : \partial\Omega \to \mathbb{N} \cup \{0\}$ will represent the population of boundary collocation points.

Let us now introduce two families of populations that gather a finite number of clones $\mu_\Omega, \mu_{\partial\Omega}$ of elements from Ω and $\partial\Omega$ respectively.

$$U_\Omega = \{P = (\Omega, \eta_P);\ \eta_P : \Omega \to \mathbb{Z}_+ \cup \{0\};\ \sum_{x \in supp(\eta_P)} \eta_P(x) = \mu_\Omega < +\infty\}, \tag{6}$$

$$U_{\partial\Omega} = \{P = (\partial\Omega, \eta_P);\ \eta_P : \partial\Omega \to \mathbb{Z}_+ \cup \{0\};\ \sum_{x \in supp(\eta_P)} \eta_P(x) = \mu_{\partial\Omega} < +\infty\} \tag{7}$$

For $P \in U_\Omega$, we set the following loss function:

$$\Theta \times U_\Omega \ni (\theta, P) \to l_\Omega(\theta, P) = \sum_{x \in supp(\eta_P)} \eta_P(x)(res_\Omega(g_\theta)(x))^2 \in \mathbb{R}_+, \tag{8}$$

while for $P \in U_{\partial\Omega}$ the loss function is

$$\Theta \times U_{\partial\Omega} \ni (\theta, P) \to l_{\partial\Omega}(\theta, P) = \sum_{x \in supp(\eta_P)} \eta_P(x)(res_{\partial\Omega}(g_\theta)(x))^2 \in \mathbb{R}_+. \tag{9}$$

Let us denote by $L_\Omega = \{l_\Omega(\cdot, P), P \in U_\Omega\}, L_{\partial\Omega} = \{l_{\partial\Omega}(\cdot, P), P \in U_{\partial\Omega}\}$ the families of all loss functions associated with the populations of internal and boundary points of fixed sizes μ_Ω and $\mu_{\partial\Omega}$, respectively. The quality of the residual approximation of PINN naturally depends on the choice of point populations used to construct the loss functions (8) and (9).

3. Learning PINN with a stochastic adaptation of collocation points

We will use the polymorphic symbol meas$(\cdot)$ of the Lebesgue measure of the subsets of $\mathbb{R}^n$ and $\mathbb{R}^{n-1}$. In particular, meas(Ω) denotes the "volume" of the exact solution's domain and meas$(\partial\Omega)$ the "surface" of its border.

Let us denote by $M(\Omega)$ and $M(\partial\Omega)$ the spaces of probabilistic measures on Ω and on $\partial\Omega$, respectively. Later, we will handle only measures possessing density functions $\rho_\Omega \in L^2(\Omega; \mathbb{R}_+), \rho_{\partial\Omega} \in L^2(\partial\Omega; \mathbb{R}_+)$.

The draft of the learning algorithm is shown in Listing 1.

```
1   BEGIN
2   Set $\mu_\Omega$ and $\mu_{\partial\Omega}$;
3   Sample $P_1 \in U_\Omega$ using the probability distribution from $M(\Omega)$
            with the uniform density $\rho_\Omega \equiv (\mathrm{meas}(\Omega))^{-1} \in L^2(\Omega)$;
4   Sample $P_2 \in U_{\partial\Omega}$ using the probability distribution from
            $M(\partial\Omega)$ with the uniform density $\rho_{\partial\Omega} \equiv (\mathrm{meas}(\partial\Omega))^{-1} \in L^2(\partial\Omega)$;
5   $t = 0$;
6   Learn PINN using loss function $l_\Omega(\cdot, P_1)+$
            $l_{\partial\Omega}(\cdot, P_2)$ getting final parameters $\hat{\theta} \in \Theta$;
7   Compute residual $res_\Omega(g_{\hat{\theta}})$;
8   Compute residual $res_{\partial\Omega}(g_{\hat{\theta}})$;
9   $P_1^t = P_1,\ P_2^t = P_2,\ \rho_\Omega^t = \rho_\Omega,\ \rho_{\partial\Omega}^t = \rho_{\partial\Omega},\ \hat{\theta}^t = \hat{\theta}$;
10  WHILE NOT $StoppingCondition(res_\Omega(g_{\hat{\theta}^t}), res_{\partial\Omega}(g_{\hat{\theta}^t}))$
11      $t = t + 1$
12      Compute new density functions $\rho_\Omega \in L^2(\Omega), \rho_{\partial\Omega} \in L^2(\partial\Omega)$
13      Sample $P_1 \in U_\Omega$ using $\rho_\Omega$ and $P_2 \in U_{\partial\Omega}$ using $\rho_{\partial\Omega}$
14      Learn PINN using loss function $l_\Omega(\cdot, P_1)+$
                $l_{\partial\Omega}(\cdot, P_2)$ getting final parameters $\hat{\theta} \in \Theta$;
15      Compute residual $res_\Omega(g_{\hat{\theta}})$;
16      Compute residual $res_{\partial\Omega}(g_{\hat{\theta}})$;
17      $P_1^t = P_1,\ P_2^t = P_2,\ \rho_\Omega^t = \rho_\Omega,\ \rho_{\partial\Omega}^t = \rho_{\partial\Omega},\ \hat{\theta}^t = \hat{\theta}$;
18  ENDWHILE
19  END
```

Listing 1 The algorithm of learning PINN.

The stopping condition of the entire learning procedure in Listing 1 (line 10) can be defined in multiple ways, depending on the convergence criteria adopted and the practical limitations. A typical and representative formulation, as employed in the numerical experiments presented, involves verifying whether the global sum of squared residual norms falls below a prescribed threshold. This criterion ensures a quantitatively controlled reduction in the approximation error. In practice, the local error within each mesh element is estimated by numerically approximating the integral of the residual norm using the trapezoidal rule, followed by normalization with respect to the element length. This yields an estimate of the mean residual norm in each element, facilitating a localized assessment of convergence. The process ends when all elements satisfy the designated error tolerance. Moreover, the algorithm must incorporate safeguards for scenarios in which further refinement of the finite element mesh is no longer feasible—either due to reaching a predefined local or global mesh density limit, denoted by $\mathcal{T}_{\mathrm{dense}}$. In such cases, the adaptation procedure is halted even if the residual-based criterion is not yet satisfied. The tolerance values used in the experimental setup were 10^{-4} for one-dimensional and 10^{-3} for two-dimensional simulations, respectively.

This strategy generates the sequence of tuples $\{(P_1^t, P_2^t\, \rho_\Omega^t,\ \rho_{\partial\Omega}^t,\ \hat{\theta}^t)\}$, $t \in \mathbb{N} \cup \{0\}$.

Next, we introduce algorithms for implementing pseudocode statements n Listing 1, which leads to an efficient minimization of both residual norms $\|res_\Omega(g_{\hat{\theta}^t})\|_{2,\Omega}$, $\|res_{\partial\Omega}(g_{\hat{\theta}^t})\|_{2,\partial\Omega}$ across consecutive iterations $t \in \mathbb{N} \cup \{0\}$.

4. Details of sampling measure adaptation strategy

4.1. *h*-FEM based densities

First, we introduce a family of density functions from class $L^2(\Omega)$ that will be used to sample the collocation points.

Let us consider the decomposition $T_{\mathrm{coarse}} = \{e_1, \dots, e_{N_{\mathrm{coarse}}}\}$ of Ω that satisfies the conventional conditions of the h -FEM meshes in $\mathbb{R}^n$. This decomposition T_{coarse} will be referred to as the *starting coarse mesh*. Next, we introduce the decomposition $T_{\mathrm{dense}} = \{e_1, \dots, e_{N_{\mathrm{dense}}}\}$, which is "nested" in T_{coarse}. This means that each element of T_{coarse} can be decomposed into several elements of T_{dense}. The T_{dense} decomposition might be reached by the h-adaptation strategy specific for the type of decomposition and dimension n (e.g. the simplistic Voronoi decomposition or the longest-edge refinement algorithm). The h-adaptation procedure produces a sequence of nested mesh decompositions starting from T_{coarse} and finishing at T_{dense}. Each element of such a chain will be nested in its predecessor and, of course, in T_{coarse}.

The set of intermediate decompositions is partially ordered by the h-adaptive algorithm, e.g. T_α preceded T_β if T_β can be obtained from T_α by several steps of the h-adaptation procedure. We denote $T_\alpha \leq T_\beta$ if T_α "is nested in" T_β.

We also introduce a boundary $\partial\Omega$ decomposition Tb, associated with the Ω decomposition T, such that $Tb = \{eb = e \cap \partial\Omega; e \in T, \mathrm{meas}(e \cap \partial\Omega) > 0\}$. We assume that the set of boundary decompositions associated with the nested family of domain decompositions is also partially ordered by the "nested in" relation and Tb_{dense} is nested in Tb_{coarse} and all intermediate boundary decompositions, so $T_\alpha \leq T_\beta \Rightarrow Tb_\alpha \leq Tb_\beta$. We refer to [33] for the necessary details concerning FE meshes, their topology, and adaptation (refinement and coarsening) methods.

Let $m_i = \mathrm{meas}(e_i), e_i \in T,\ i = 1, \dots, N_T$ for some intermediate mesh T.

We can now introduce the set of densities spanned by the vectors

$$d_T = (d_1, \dots, d_{N_T});\ d_i \geq 0,\ i = 1, \dots, N_T\ ,\ \sum_{i=1,\dots,N_T} m_i\, d_i = 1 \tag{10}$$

Each density under consideration has the form

$$\rho_\Omega(x) = d_i \text{ if } x \in \mathrm{int}(e_i)\ ,\ \ e_i \in T \tag{11}$$

Such functions are step-wise constant belonging to $L^2(\Omega)$ and well defined for almost all $x \in \Omega$ except $x \in \partial\Omega$ and $x \in \partial e_i \in T_{\text{dense}}$, $i = 1, \dots, N_{\text{dense}}$.

Analogously, having a boundary decomposition measures

$$(mb_1, \dots, mb_{N_{Tb}});\ mb_i = \text{meas}(eb_i)\,,\ eb_i \in Tb \tag{12}$$

and a vector of boundary probability densities

$$db_{Tb} = (db_1, \dots, db_{N_{Tb}});\ db_i \geq 0,\ i = 1, \dots, N_{Tb}\,,\ \sum_{i=1,\dots,N_{Tb}} mb_i\, db_i = 1 \tag{13}$$

we can define the boundary density

$$\rho_{\partial\Omega}(x) = db_i \text{ if } x \in \text{int}(eb_i)\,,\ eb_i \in Tb \tag{14}$$

For an arbitrary elements $e \in T$ and $eb \in Tb$ the spaces $L^2(e), L^2(eb)$ with norms $\|\cdot\|_{2,e}, \|\cdot\|_{2,eb}$ can be used to evaluate residual on $e \subset \Omega$ and $eb \subset \partial\Omega$.

4.2. Adapting densities

The starting density vector associated with ρ^0_Ω will be set to

$$d_i = (m_i)^{-1},\ i = 1, \dots, N_{\text{dense}}\,, \tag{15}$$

while $\rho^0_{\partial\Omega}$ will be set to

$$db_i = (mb_i)^{-1},\ i = 1, \dots, Nb_{\text{dense}}\,. \tag{16}$$

The consecutive densities $\rho^t_\Omega, \rho^t_{\partial\Omega}$ will be based on the successive adaptations of the initial decomposition T_{coarse}. The Listing 2 presents the adaptation strategy.

```
BEGIN
$T = T_{coarse}$
compute $\|res_\Omega(g_\theta)|e\|_{2,e}$, $e \in T$
WHILE $\max\{\|res(g_\theta)|e\|_{2,e}, e \in T\} > threshold1$
    compute $T'; T_{dense} \geq T' > T$ by breaking all elements $e \in T$ for which $\|res(g_\theta)|e\|_{2,e} > threshold1$
    $T = T'$
    compute $\|res_\Omega(g_\theta)|e\|_{2,e}$, $e \in T$
ENDWHILE
compute $Tb$
set $\rho_\Omega$ according to the formula (11) with $d_i = \frac{\|res_\Omega(g_\theta)|e_i\|_{2,e_i}}{m_i}$, $i = 1, \dots, N_T$
compute $\rho_{\partial\Omega}$ according to (14) with $db_i = \frac{\|res_{\partial\Omega}(g_\theta)|eb_i\|_{2,eb_i}}{mb_i}$, $i = 1, \dots, N_{Tb}$
scale and normalize $\rho_\Omega$ and $\rho_{\partial\Omega}$
END
```

Listing 2: The adaptation algorithm.

The parameter $threshold1$ in Listing 2 is a user-defined tolerance used to guide mesh refinement. Any element with a normalized residual — computed via the trapezoidal rule and divided by element size — exceeding $threshold1$ is marked for refinement. In the numerical experiments, $threshold1$ was set equal to the global stopping tolerances, i.e., 10^{-4} for 1D, and 10^{-3} for 2D cases. If, in a given iteration, the maximum residual is already below this threshold, no refinement occurs, but this does not imply convergence. Final convergence is determined by the global residual norm after training. Overfitting is mitigated by residual normalization and the physics-driven nature of the loss function, even in cases with limited quadrature resolution.

4.3. Sampling collocation points

The simple two-phase sampling algorithm of members $P_1 \in U_\Omega$ according to the probability distribution from $M(\Omega)$ with a density ρ_Ω given by the vector d_T is presented in the Listing 3.

```
1   BEGIN
2   $P_1 = \emptyset$
3   FOR $e \in T$
4       $n_e = 0$
5   ENDFOR
6   FOR $i = 1, \mu_\Omega$
7       using roulette, sample $e \in T$ according to the probability distribution $(d_1 m_1, \dots, d_{N_T} m_{N_T})$
8       $n_e = n_e + 1$
9   ENDFOR
10  FOR $e \in T$
11      FOR $i = 1, n_e$
12          $p_i$ = sample point according to the uniform probability distribution on $e$
13          $P_1 = P_1 \cup \{p_i\}$
14      ENDFOR
15  ENDFOR
16  END
```

Listing 3: The algorithm for sampling collocation points.

Sampling $\mu_{\partial\Omega}$ clones to population $P_2 \in U_{\partial\Omega}$ according the density $\rho_{\partial\Omega}$ will be performed by the analogous algorithm. Additionally:

- In statement 2, P_1 denotes the empty multiset with the zero-occurrence function.
- The sampling strategy consists of two phases. In the first, we sample the number of points in each element of T (statements 6–9). In the second phase, we determine the location of each point within the given element.
- Sampling element e (statement 7) does not remove this element from the set T.
- The possible method of sampling the assumed number of points d with a uniform probability distribution over the element $e \in T$ (being a part of Ω) or on $\partial e \cap \partial\Omega$ (being a part of $\partial\Omega$) could be performed by the algorithm presented in the Listing 4.

```
BEGIN
wrap the element $e \in T$ into the ``brick'' $B = [a,b]^n$
FOR $k = 1, d$
    sample the point $x$ from $B$ with the uniform probability distribution on $B$
    WHILE $x \notin e$
        sample the point $x$ from $B$ with the uniform probability distribution on $B$
    ENDWHILE
ENDFOR
```

Listing 4: The algorithm of sampling points inside element.

Because of the finite accuracy implementation of real numbers, we should also allow sampling collocation points on the walls of the elements in T and in Tb, even if it rarely happens.

This procedure, in the case of internal decomposition T, starts with an arbitrary and unambiguous numbering of all elements $e \in T$. Let us consider sampling in an arbitrary element $\bar{e} \in T$. If the collocation point was sampled on any wall or vertex of $\bar{e}$, then it will be removed if it belongs to $\partial\Omega$ or it belongs to another $e \in T$ with a larger number than the ordering number of $\bar{e}$.

The procedure for boundary elements is similar, except that we will now only remove points sampled on the common wall with an element that has a higher ordering number.

5. The problem of optimal collocation points selection

If we accept all the principles of the proposed stochastic algorithm for PINN learning, we are able to formulate the associated problem of optimal selection of collocation points:

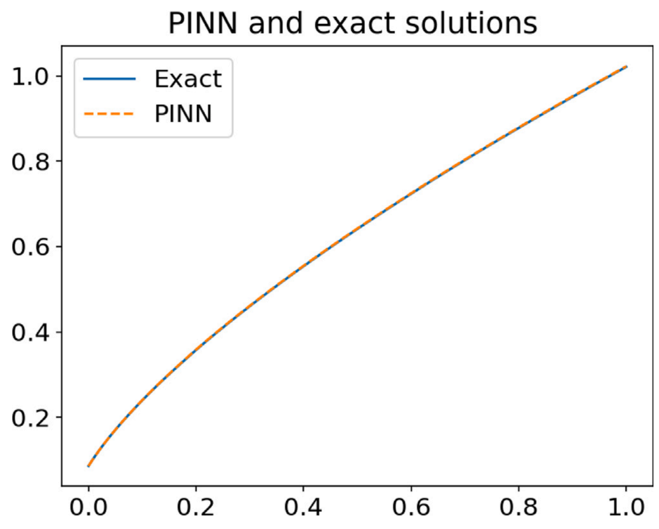


Fig. 1. Comparison of the exact solution and the PINN solution for benchmark B1.

Definition 5.1. Given $\mu_\Omega, \mu_{\partial\Omega}, T_{coarse}$ find $\hat{P}_1 \in U_\Omega, \hat{P}_2 \in U_{\partial\Omega}$ so that

$$\|res_\Omega(g_{\hat{\theta}})\|^2_{2,\Omega} + \|res_{\partial\Omega}(g_{\hat{\theta}})\|^2_{2,\partial\Omega} \leq \|res_\Omega(g_\theta)\|^2_{2,\Omega} + \|res_{\partial\Omega}(g_\theta)\|^2_{2,\partial\Omega} \quad (17)$$

where $g_{\hat{\theta}} \in G$ is learned by using loss function $l_\Omega(\cdot, \hat{P}_1) + l_{\partial\Omega}(\cdot, \hat{P}_2)$ and $g_\theta \in G$ are learned by $l_\Omega(\cdot, P_1) + l_{\partial\Omega}(\cdot, P_2)\ \forall P_1 \in U_\Omega, P_2 \in U_{\partial\Omega}$, see (8), (9).

Considering once more the implementation of all collocation points as a finite number of vectors (because of a restricted accuracy real number representation) contained in Ω and $\partial\Omega$, we can also handle the implementation of families of populations U_Ω and $U_{\partial\Omega}$ as finite sets (see e.g. [34]). The learning process mentioned in Definition 5.1 assigns the unique value of the objective (sum of the squares of the norms of both residuals) to each pair of populations. In consequence, this problem admits at least one global minimizer as a result of searching in a finite set.

6. Advantages of the proposed framework

The formal analysis of the proposed collocation point selection shows the following advantages:

1. Can be applied to an arbitrary dimension n of a computational domain $\Omega \subset \mathbb{R}^n$, assuming the existence of a proper type of FE grid and its adaptation strategy producing the sequence of nested meshes.
2. Is transparent with respect to the FE mesh adaptation policy, assuming only the existence of a starting "coarse" mesh and a final "dense" one, as well as the "is nested in" partial order of intermediate meshes; that is, each intermediate mesh T satisfies $T_{dense} \leq T \leq T_{coarse}$, where $\leq$ denotes the "is nested in" relation.
3. As far as the sampling procedure in the first phase is "proportional" to the residual, it allows scaling the density ρ_Ω (also dynamically), enlarging "selection pressure" towards elements with higher residual.
4. The number of collocation points μ_Ω is independent of the number of elements and the training epoch, so it can be dynamically changed.
5. The number of collocation points sampled on the boundary $\mu_{\partial\Omega}$ is independent of the number of points sampled in the interior and can be dynamically controlled during iterations.
6. The probability distribution and the method of sampling collocation points inside elements and on their boundaries can be changed without changing the overall strategy of assigning the number of points in each element or its boundary.
7. The existence of a T_{dense} nested in all possible meshes makes the natural constraints for the mesh adaptation policy applied. On the other hand, it allows for injective mapping of each density ρ_Ω obtained for the intermediate mesh $T \leq T_{dense}$ on the final mesh. This feature can be utilized by modeling the sampling procedure as the Markov process with a finite number of states and the later asymptotic analysis.
8. It is easy to observe that if $res_\Omega(g_\theta)(x) \neq 0\ \forall x \in \Omega$ and $res_{\partial\Omega}(g_\theta)(x) \neq 0\ \forall x \in \partial\Omega$, $\forall g_\theta \in G$, then both densities ρ_Ω and $\rho_{\partial\Omega}$ are strictly positive, so the arbitrary populations $P_1 \in U_\Omega$ and $P_2 \in U_{\partial\Omega}$ can be sampled. Because zero residual may occur rarely (practically only once if the exact solution $u \in G$), then the populations satisfying (17) might be sampled in one step with positive probability. So, the presented algorithm falls into the class of asymptotically correct stochastic global optimization strategies (see, e.g., [35]). As far as this feature does not provide the effective stopping condition of the process modifying the collocation points, we are sure that adjusting the sampling densities ρ_Ω and $\rho_{\partial\Omega}$ does not protect the sampling of any minimizer with probability one.

7. The schedule of experimental verification

Two benchmarks are used to evaluate the performance and accuracy of the proposed algorithms. They serve as a means of validation, allowing us to assess the accuracy of the algorithm. In addition, benchmarks test the robustness of algorithms against various difficulties, such as steep gradients. They also provide a standardized way to compare different algorithms. By applying these benchmarks, we gain insight into the behavior of algorithms under various conditions.

The selected benchmarks are elliptic problems and diffusion-convection problems formulated abstractly across one-dimensional (1D) and two-dimensional (2D) domains.

For the elliptic problems, the equation takes the form:

$$-\nabla\,(a(x)\nabla u(x)) = f(x), \quad x \in \Omega \subset \mathbb{R}^p, \quad p = \{1, 2\} \quad (18)$$

All benchmarks use boundary conditions that are either pure Dirichlet or a mix of Dirichlet and Neumann conditions. The problem involves finding $u \in C^2(\Omega)$ within a domain $\Omega \subset \mathbb{R}^p$, where $p = 1, 2$, that satisfies the proper boundary conditions on $\partial\Omega$. Given a right-hand side (RHS) function $f(x)$ in $C^0(\Omega)$ and a coefficient $a(x)$ in $C^1(\Omega)$, the problem is well-posed. The uniqueness and regularity of the solutions ensure that the benchmarks are reliable for testing the performance of numerical algorithms. In (18) where Ω is a domain, $u(x)$ is the manufactured solution, $a(x)$ is a tensor that characterizes the material properties, and $f(x)$ is the right-hand side.

In elliptic problems, steep gradients near domain edges, particularly in 2D cases, can lead to numerical instability. This case can be particularly problematic for the standard PINN algorithm (without collocation points adaptation), which may not detect such behavior and fail to converge. In our benchmarks with manufactured solutions, we shift the edge singularities slightly outside the domain boundaries, resulting in a problem without singularities. The gradient near the boundary remains very steep and significantly larger than within the domain, posing challenges for numerical algorithms to accurately capture these abrupt changes.

We select a C^2 function with a singularity as a candidate for the benchmark manufactured solution. Then, we choose a benchmark domain without singularities but with a boundary positioned close enough to the singularity to produce a locally high but finite solution gradient.

The second benchmark, the advection–diffusion problem, is formulated as follows:

$$\varepsilon \Delta u(x) - b(x)\nabla u(x) = f(x), \quad \forall x \in \Omega \subset \mathbb{R}^p, \quad p = \{1, 2\} \quad (19)$$

where $\varepsilon \in \mathbb{R}_+$ is the diffusion coefficient, $b \in C^0(\Omega)$ represents the advection vector, and $f \in C^0(\Omega)$ is the source term.

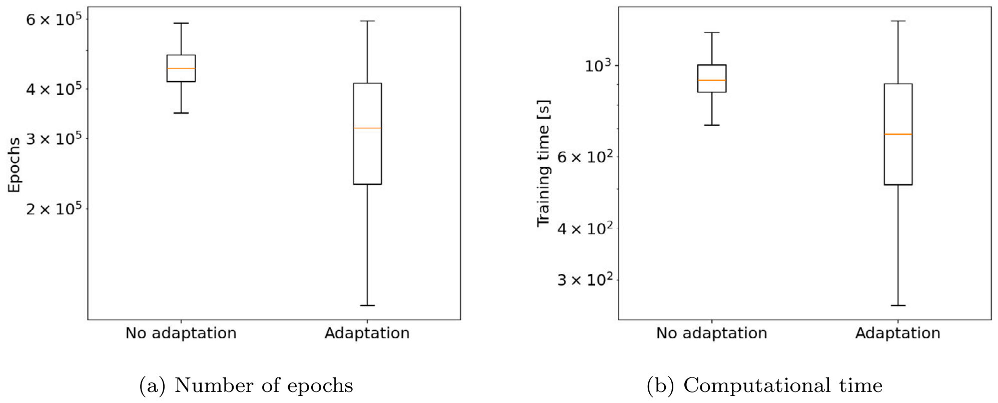


Fig. 2. Comparison of the number of epochs (left panel) and the computational time (right panel) required to reach given tolerance. 200 repeated runs were tested for PINN without adaptation and with middle point adaptation on an RTX 4070 GPU for benchmark B1.

Table 1
Poison benchmarks details.

	Ω	BC	Manufactured solution
B1	$(0,1)$	$g_D = (x+\mu)^{0.7}, x = 0$ $g_N = \frac{0.7}{(x+\mu)^{0.3}}, x = 1$	$(x+\mu)^{0.7}$
B2	$\left(0, \frac{\pi}{2}\right)$	$g_D = \tan(x-\mu), x = 0$ $g_N = \frac{1}{\cos^2(x-\mu)}, x = \frac{\pi}{2}$	$\tan(x-\mu)$
B3	$(0,1)^2$	$g_D = (x+\mu)^{0.7}(y+\mu)^{0.7}, x = 0 \vee y = 0;$ $g_N = \frac{0.7((x+\mu)+(y+\mu))}{((x+\mu)(y+\mu))^{0.3}}, x = 1 \vee y = 1$	$((x-\mu)(y-\mu))^{0.7}$
B4	$\left(0, \frac{\pi}{2}\right)^2$	$g_D = \tan(x-\mu)\tan(y-\mu), x = 0 \vee y = 0$ $g_N = \frac{\tan(y-\mu)}{\cos^2(x-\mu)} + \frac{\tan(x-\mu)}{\cos^2(y-\mu)}, x = \frac{\pi}{2} \vee y = \frac{\pi}{2}$	$\tan(x+\mu)\tan(y+\mu)$

Several difficulties arise when solving these benchmark problems, particularly for PINN algorithms.

In this problem, the solution exhibits steep gradients for small ε, making it challenging to capture the boundary layer without crashing.

7.1. Metrics

We employ a variety of metrics to assess the quality of algorithmic performance, appropriate for the stochastic nature of the algorithm. Since each run yields different performance results, multiple executions are necessary to obtain reliable measures.

The primary metrics we focus on are the average and median training time. They are critical in determining whether the algorithm performs statistically better across runs. When improvements are observed in both metrics, it can be inferred that the algorithm exhibits superior performance.

In addition to time-based metrics, we evaluated the average error in the best population of collocation points obtained during the run. We also analyze the regression of the $L^2(\Omega)$ error during the run and the spatial error distribution across the runs, which provides insights into the stability and robustness of the algorithm's performance. These error-based measurements allow us to better understand the accuracy and reliability of the algorithm beyond just its speed. Although these measurements are briefly formulated, the exact formulas will be explained later.

7.2. Benchmarks

We selected a set of benchmarks to rigorously evaluate the algorithm's performance, each with known analytical solutions, to provide a reliable basis for validation.

The Poisson equation with a manufactured solution allows us to test the algorithm's ability to accurately reproduce predefined behavior, while the advection–diffusion equation introduces complexity through coupled transport and diffusion processes. These equations are widely used in numerical analysis, making them well-suited for testing the capabilities of different approaches. The details of the benchmark problems B1-B4 are described in Table 1.

For each benchmark, we train a Physics-Informed Neural Network (PINN) with the same hyperparameters, using both the proposed adaptive sampling and a non-adaptive sampling algorithm. Then, the performance is evaluated using various metrics. This approach ensures a comprehensive assessment of the algorithms' accuracy and computational efficiency. We analyze all the metrics described in 7.1.

Let us define a set of manufactured solutions for both the 1D and 2D domains. These solutions provide known analytical forms, allowing the precise validation of the numerical results.

Problems B1, and B3 are designed to evaluate how well the algorithm can handle power-law behaviors with fractional exponents. The μ controls the steepness of the solution near the edge of the domain, forcing the algorithm to capture this variation accurately.

The benchmarks B2, and B4 are another type of difficult non-polynomial RHS. The solution has a steep gradient as $\mu \to 0$, introducing significant challenges to the algorithm, especially in capturing the behavior near the edge of the domain.

8. Benchmark results

We present an evaluation of our proposed numerical method for benchmarks B1-B4 and BA, which were defined in the previous section. A series of numerical experiments compared PINN with the adaptation of the initial guiding mesh T_{coarse} (Section 4.3) and without adaptation. Each experiment consists of repeatedly solving the benchmark problem, both with and without adaptation. We then perform a statistical analysis of the resulting metrics. We performed the experiments on a Linux workstation equipped with an RTX 4070 GPU with 8 GB of GPU VRAM,

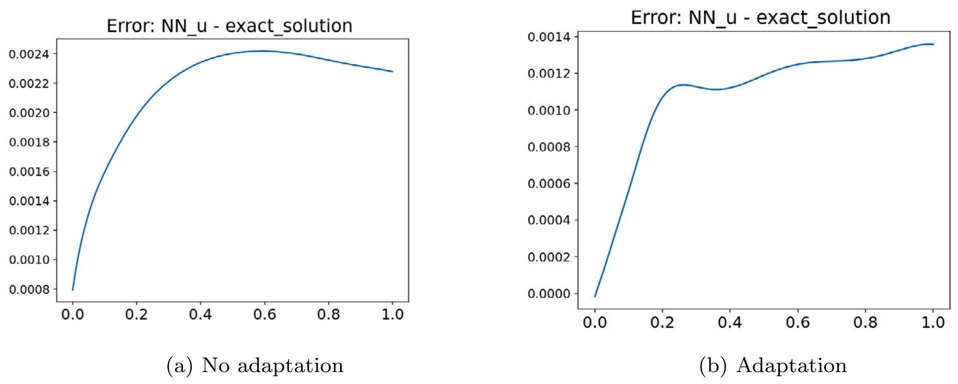


Fig. 3. Comparison of error for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B1. Computations were done on RTX 4070 GPU.

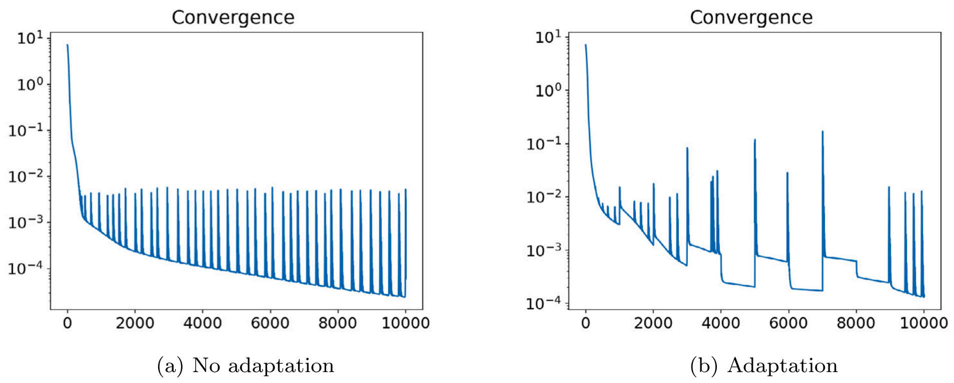


Fig. 4. Comparison of convergence for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B1. Computations were done on RTX 4070 GPU.

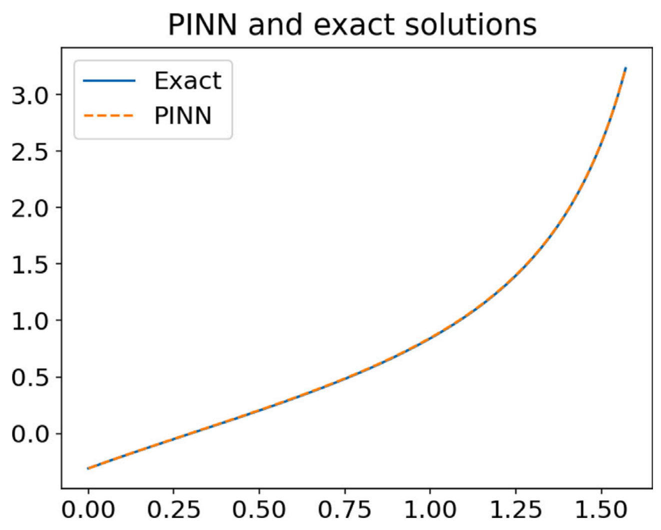


Fig. 5. Comparison of the exact solution and the PINN solution for benchmark B2.

AMD Ryzen 5 7600X CPU, 32 GB DDR5 6000 MHz CL36 RAM, running Ubuntu 20.04 Linux, and Python 3.10.

8.1. 1D Poison equation with manufactured solution of benchmark B1

We performed each experiment 50 times, both with and without adaptation, using a tolerance of 10^{-4} and running 1000 epochs per iteration, with a maximum of 1000 iterations per run. Both algorithms were tested for $\mu = 0.03$, and 200 collocation points.

Fig. 1 illustrates the comparison between an exemplary PINN solution and the exact solution.

Table 2
Average and median values for the number of epochs and the computational time for training PINN with and without adaptation, for benchmark B1.

	Number of epochs		Computational time	
	Average	Median	Average	Median
No adaptation	468 723	450 000	959.72 s	921.20 s
Adaptation	336 780	318 500	734.78 s	678.99 s

The results indicate that the average number of epochs required for training without adaptation was 468 723, whereas, with adaptation, this was reduced to 336 780. Similarly, the median number of epochs decreased from 450 000 without adaptation to 318 500 with adaptation. In terms of computational time, the average training duration dropped from 959.72 s to 734.78 s with adaptation, while the median time decreased from 921.20 s to 678.99 s. A summary of these findings is provided in Table 2, and Figs. 2(a) and 2(b) offer detailed comparisons of the number of epochs and training times.

Additionally, we analyze representative results for error and residual convergence in both versions of PINN. For these comparisons, we selected runs with exactly 8000 epochs. Fig. 3 shows the error for both algorithms. Fig. 4 illustrates the residual behavior, where the adapted method shows noticeable spikes in the loss function due to mesh regeneration. Fig. 3 reflects the fixed-budget behavior of the loss function, rather than the actual stopping point of the training. In practice, the loss often meets the target threshold much earlier, particularly in the adaptive case, and the method terminates accordingly.

8.2. 1D Poison equation with manufactured solution of benchmark B2

We performed each experiment 100 times, both with and without adaptation, using a tolerance of 10^{-4} and running 1000 epochs per iteration, with a maximum of 1000 iterations per run. Both algorithms were tested for $\mu = 0.3$ with 200 collocation points.

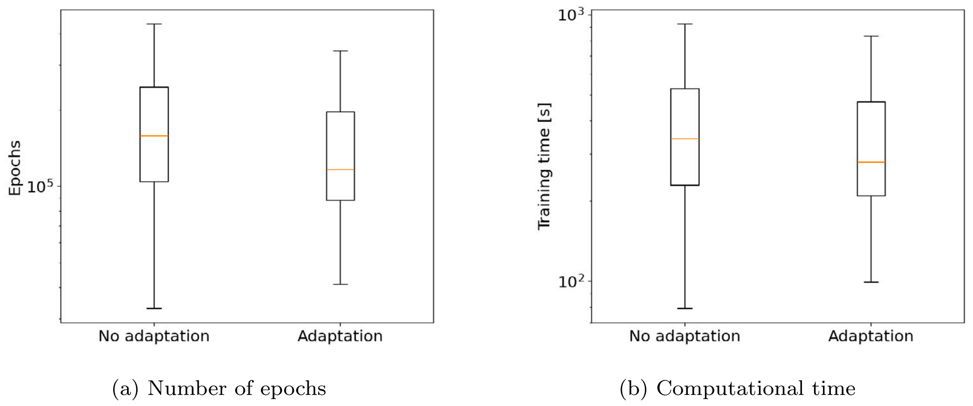


Fig. 6. Comparison of the number of epochs (left panel) and the computational time (right panel) required to reach given tolerance. 200 repeated runs were tested for PINN without adaptation and with middle point adaptation on an RTX 4070 GPU for benchmark B2.

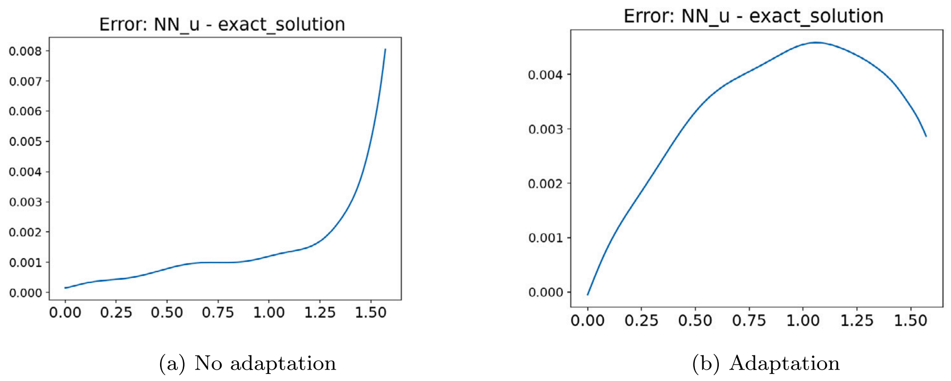


Fig. 7. Comparison of error for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B2. Computations were done on RTX 4070 GPU.

Table 3
Average and median values for the number of epochs and the computational time for training PINN with and without adaptation, for benchmark B2.

	Number of epochs		Computational time	
	Avergae	Median	Average	Median
No adaptation	241 030	158 000	517.35 s	342.93 s
Adaptation	152 640	116 500	368.03 s	280.70 s

Table 4
Average and median values for the number of epochs and the computational time for training PINN with and without adaptation for benchmark BA.

	Number of epochs		Computational time	
	Average	Median	Average	Median
No adaptation	106 338	35 000	240.51 s	81.89 s
Adaptation	18 660	17 000	49.36 s	45.21 s

Fig. 5 illustrates the comparison between an exemplary PINN solution and the exact solution.

The results indicate that the average number of epochs required for training without adaptation was 241 030, whereas, with adaptation, this was reduced to 152 640. Similarly, the median number of epochs decreased from 158 000 without adaptation to 116 500 with adaptation. In terms of computational time, the average training duration dropped from 517.35 s to 368.03 s with adaptation, while the median time decreased from 342.93 s to 280.70 s. A summary of these findings is provided in Table 3, and Figs. 6(a) and 6(b) offer detailed comparisons of the number of epochs and training times.

Additionally, we analyze representative results for error and residual convergence in both versions of PINN. For these comparisons, we selected runs with exactly 8000 epochs. Fig. 7 shows the error for both algorithms. Fig. 8 illustrates the residual behavior, where the adapted method shows noticeable spikes in the loss function due to mesh regeneration. As observed in Fig. 7, the previously pronounced local error near the right boundary has been effectively mitigated. Furthermore, the error distribution appears substantially more uniform throughout the domain.

8.3. 1D advection-dominated diffusion benchmark problem BA

We conducted each experiment 200 times, both with and without adaptation, setting a tolerance of 10^{-4} and running 1000 epochs per iteration, with a maximum of 1000 iterations per run. It is important to note that three cases failed to achieve the desired numerical error within 1000 iterations out of the 200 runs without adaptation. Both algorithms were tested for $\epsilon = 0.05$ with 200 collocation points.

Fig. 9 illustrates the comparison between an exemplary PINN solution and the exact solution.

The results show that the average number of epochs for training without adaptation was 106 338, compared to 18 660 for the adaptation. Similarly, the median number of epochs dropped from 35 000 without adaptation to 17 000 with adaptation. In terms of computational time, the adaptation reduced the average training time from 240.51 s to 49.36 s, and the median time from 81.89 s to 45.21 s. A summary of these results can be found in Table 4, while Figs. 10(a) and 10(b) display detailed comparisons of the number of epochs and training times, respectively.

Furthermore, we examine exemplary results for error and residual convergence in both versions of PINN. For these comparisons, we selected runs with exactly 8000 epochs. Fig. 11 presents the error for both

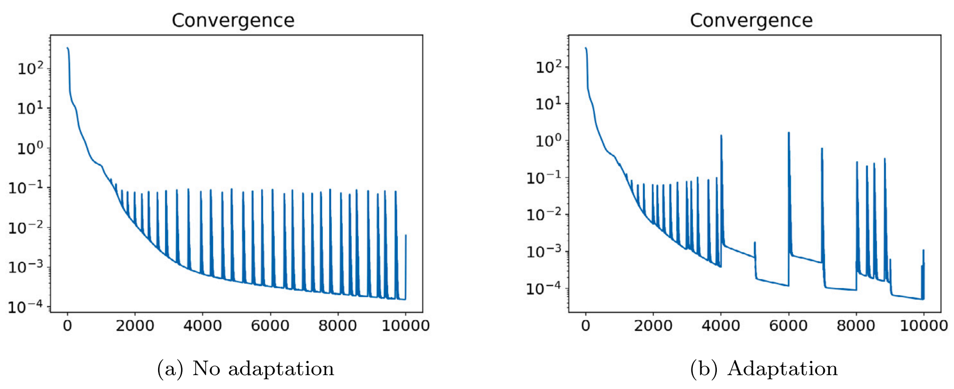


Fig. 8. Comparison of convergence for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B2. Computations were done on RTX 4070 GPU.

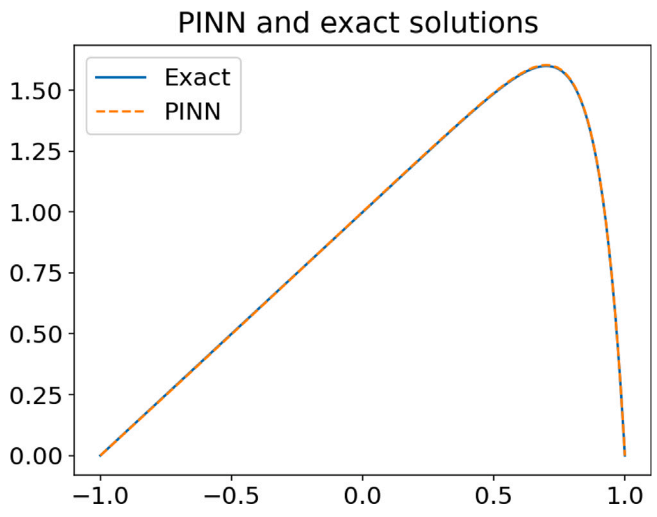


Fig. 9. Comparison of exact solution and PINN solution for benchmark BA.

Table 5
Average and median values for the number of epochs and the computational time for training PINN with and without adaptation, for benchmark B3.

	Number of epochs		Computational time	
	Average	Median	Average	Median
No adaptation	377 686	289 000	2472.87 s	1868.11 s
Adaptation	74 745	69 000	513.58 s	476.32 s

algorithms. With adaptation, the error remains uniformly low across the entire domain, oscillating around numerical zero, whereas without adaptation, significant errors accumulate near the domain singularity on the right. Fig. 12 shows the residual behavior, where the adapted method exhibits noticeable spikes in the loss function due to mesh regeneration.

8.4. 2D Poison equation with manufactured solution of benchmark B3

We performed each experiment 50 times, both with and without adaptation, using a tolerance of 10^{-3} and running 1000 epochs per iteration, with a maximum of 1000 iterations per run. Both algorithms were tested for $\mu = 0.1$ with 40 000 collocation points.

The results indicate that the average number of epochs required for training without adaptation was 377 686, whereas, with adaptation, this was reduced to 74 745. Similarly, the median number of epochs decreased from 289 000 without adaptation to 69 000 with adaptation. In terms of computational time, the average training duration dropped from 2472.87 s to 513.58 s with adaptation, while the median time decreased from 1868.11 s to 476.32 s. Out of 50 runs, training without adaptation has not reached the set tolerance five times, while training with adaptation has reached the expected tolerance goal every time. A summary of these findings is provided in Table 5, and Figs. 13(a) and 13(b) offer detailed comparisons of the number of epochs and training times.

Additionally, we analyze representative results for error and residual convergence in both versions of PINN. For these comparisons, we selected runs with exactly 20 000 epochs. Fig. 14 shows the error for both algorithms. Fig. 15 illustrates the residual behavior, where the adapted method shows noticeable spikes in the loss function due to mesh regeneration.

Table 6
Average and median values for the number of epochs and the computational time for training PINN with and without adaptation, for benchmark B4.

	Number of epochs		Computational time	
	Average	Median	Average	Median
No adaptation	316 000	242 000	4355.58 s	3350.71 s
Adaptation	143 457	105 000	2020.75 s	1493.72 s

8.5. 2D Poison equation with manufactured solution of benchmark B4

We performed each experiment 35 times, both with and without adaptation, using a tolerance of 10^{-3} and running 1000 epochs per iteration, with a maximum of 1000 iterations per run. Both algorithms were tested for $\mu = 0.5$ with 160 000 collocation points.

The results indicate that the average number of epochs required for training without adaptation was 316 000, whereas, with adaptation, this was reduced to 143 457. Similarly, the median number of epochs decreased from 242 000 without adaptation to 105 000 with adaptation. In terms of computational time, the average training duration dropped from 4355.58 s to 2020.75 s with adaptation, while the median time decreased from 3350.71 s to 1493.72 s. A summary of these findings is provided in Table 6, and Figs. 16(a) and 16(b) offer detailed comparisons of the number of epochs and training times.

Additionally, we analyze representative results for error and residual convergence in both versions of PINN. For these comparisons, we selected runs with exactly 20 000 epochs. Fig. 17 shows the error for both algorithms. Fig. 18 illustrates the residual behavior, where the adapted method shows noticeable spikes in the loss function due to mesh regeneration.

8.6. Additional comment on numerical results

The observed reduction in computational time in the adaptive setting, relative to the non-adaptive case, stems from the nature of the

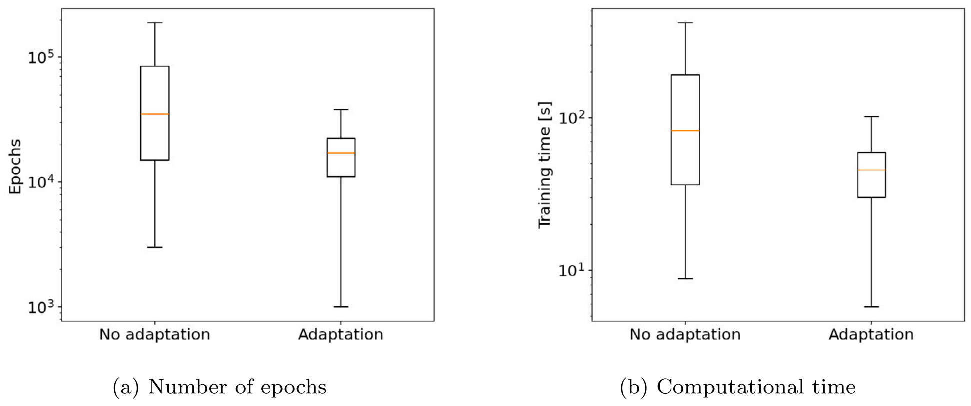


Fig. 10. Comparison of the number of epochs (left panel) and the computational time (right panel) required to reach given tolerance. 200 repeated runs were tested for PINN without adaptation and with middle point adaptation on an RTX 4070 GPU for benchmark BA.

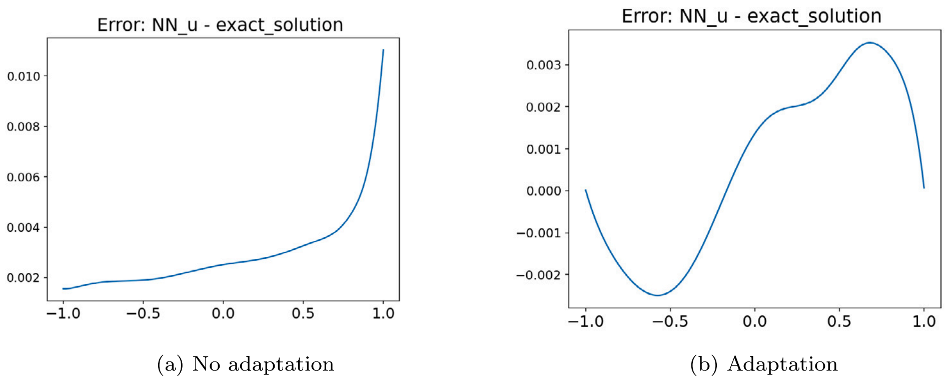


Fig. 11. Comparison of error for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark BA. Computations were done on RTX 4070 GPU.

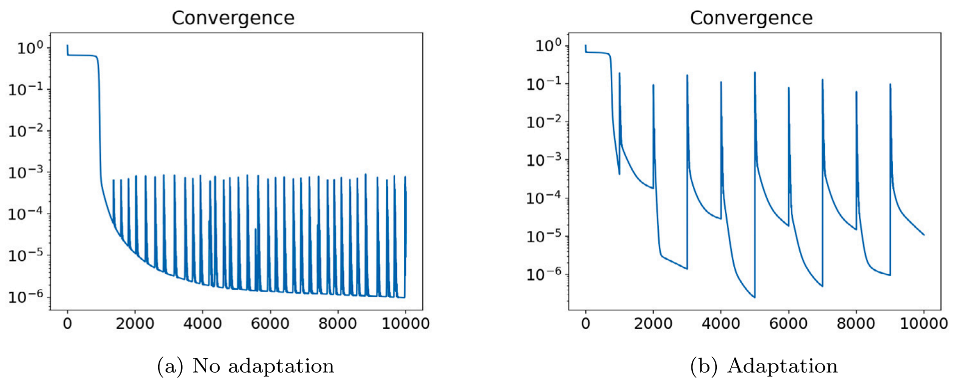


Fig. 12. Comparison of convergence for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark BA. Computations were done on RTX 4070 GPU.

stopping criterion employed in the numerical experiments. In all scenarios, the algorithm is designed to terminate after reaching a predefined accuracy threshold. The adaptive procedure, by dynamically refining the placement of collocation points, enables the algorithm to reach this accuracy more efficiently, thereby yielding a noticeable decrease in total run-time.

The stopping condition is evaluated on the basis of localized error estimates across mesh elements. For each element, we approximate the residual norm — computed using the trapezoidal rule — and normalize it by the size of the element. This normalization is particularly pertinent in the adaptive setting, where the mesh is non-uniform. The accuracy thresholds were set to 10^{-4} in one-dimensional and 10^{-3} in two-dimensional benchmarks.

The main stopping criterion is reaching a prescribed residual norm in the L^2 sense. The 1000 training epochs referenced in the numerical examples refer to a fixed number of epochs per adaptation step, not a global stopping limit. Overfitting is mitigated by residual-based adaptation: collocation points are concentrated in regions with high local residuals, ensuring a meaningful and robust reduction of the global residual norm across iterations.

It is important to emphasize that the presented numerical results were generated under a so-called final accuracy test regime, where comparisons are made based on computational cost required to achieve

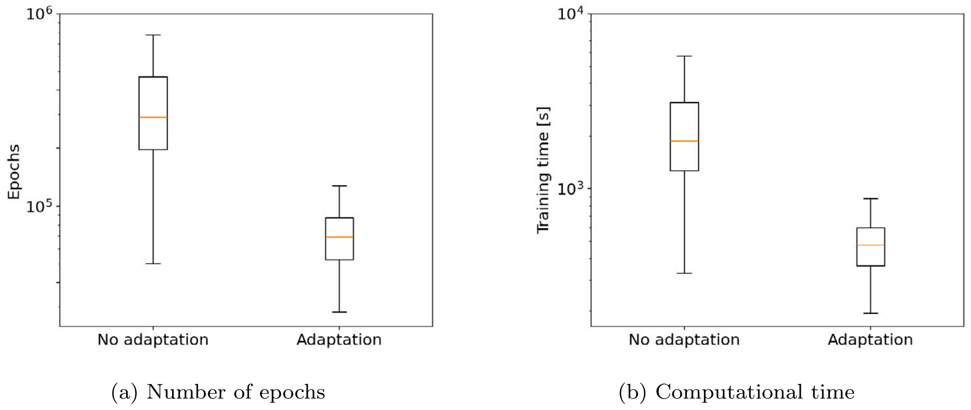


Fig. 13. Comparison of the number of epochs (left panel) and the computational time (right panel) required to reach given tolerance. 200 repeated runs were tested for PINN without adaptation and with middle point adaptation on an RTX 4070 GPU for benchmark B3.

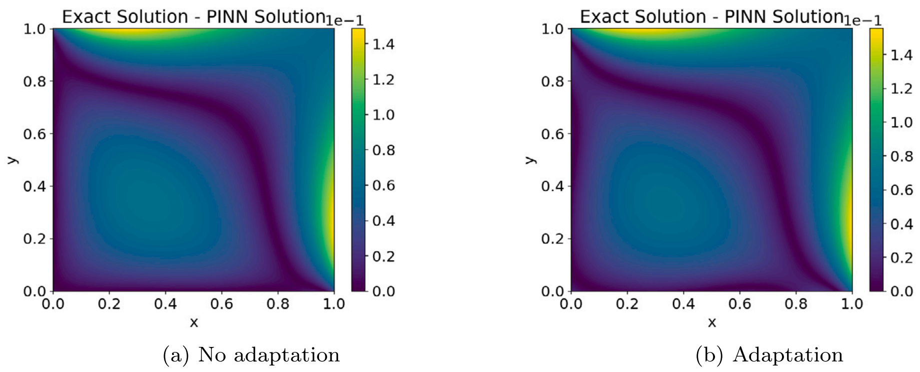


Fig. 14. Comparison of error for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B3. Computations were done on RTX 4070 GPU.

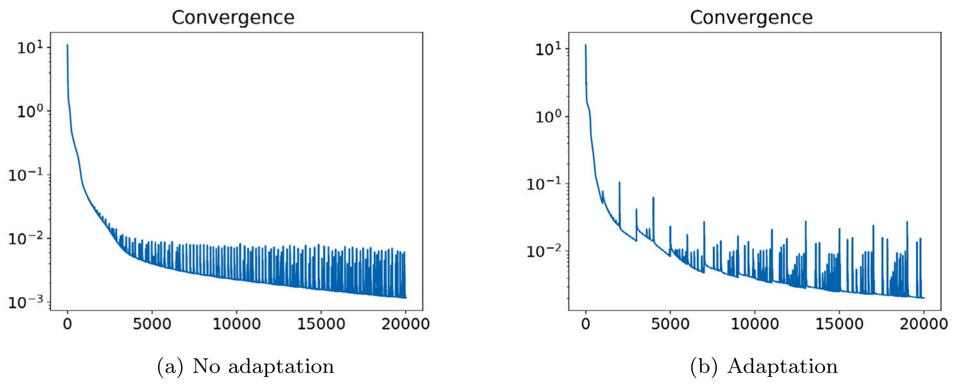


Fig. 15. Comparison of convergence for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B3. Computations were done on RTX 4070 GPU.

a common target accuracy, not on absolute error levels at a fixed computational budget. Accordingly, while the adaptive strategy does not necessarily produce markedly lower final errors compared to the non-adaptive method, it achieves the desired precision with significantly fewer computational resources and training epochs. This constitutes a critical practical advantage.

Nevertheless, it is possible to assess relative accuracy under an alternative regime—namely, a final budget test, wherein both adaptive and non-adaptive approaches are executed with an identical computational budget. In such a setting, one may directly compare the residual norms attained by both methods, providing further insight into their relative approximation quality. Although such a study is beyond the scope of the current manuscript, it could serve as a complementary analysis in future work.

In summary, the contribution of the present work lies primarily in the demonstrated computational efficiency and methodological adaptability of the proposed framework, which are especially relevant in large-scale or high-dimensional settings where computational resources are a limiting factor.

During all benchmark simulations, the weights and biases of the PINNs remained within finite bounds, typically in the range $[-20, 20]$ for weights and $[-10, 10]$ for biases. No numerical overflows occurred, which confirms that the training process remained within a compact parameter set $\Theta \subset \mathbb{R}^{N_G}$, thus supporting Hypothesis 4.

The benchmark PDEs are well-posed elliptic or convection–diffusion problems with smooth data and classical solutions $u \in C^2(\Omega)$, validating Hypothesis 1. Since network architectures use smooth activation functions (e.g. tanh), the realizations g_θ belong to $C^2(\Omega)$, satisfying

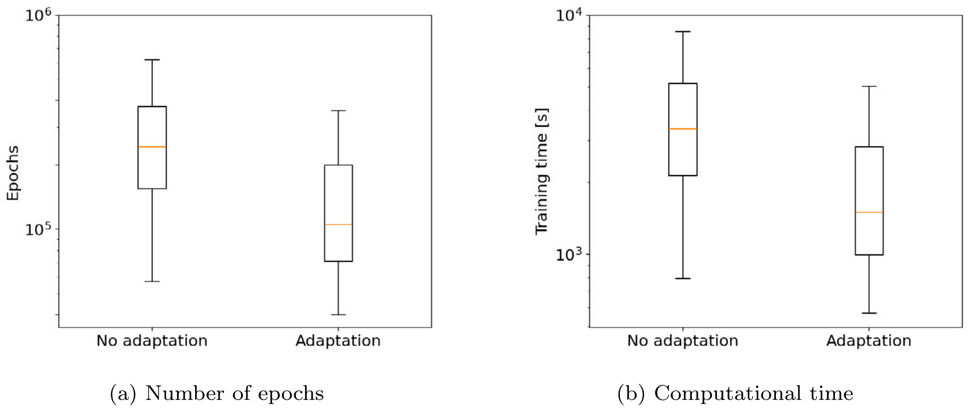


Fig. 16. Comparison of the number of epochs (left panel) and the computational time (right panel) required to reach given tolerance. 200 repeated runs were tested for PINN without adaptation and with middle point adaptation on an RTX 4070 GPU for benchmark B4.

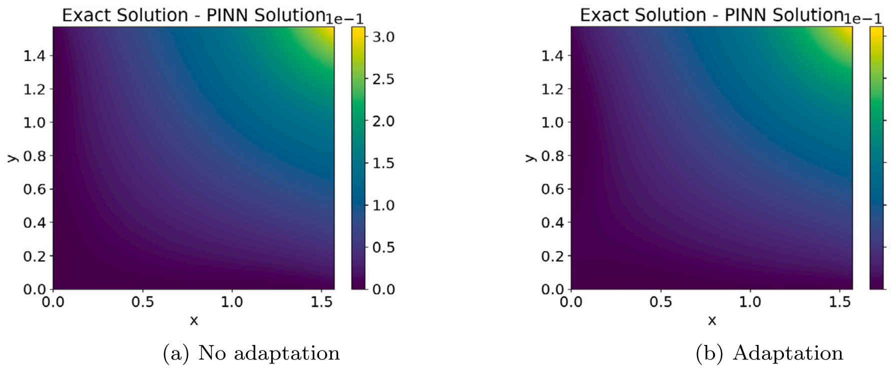


Fig. 17. Comparison of error for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B4. Computations were done on RTX 4070 GPU.

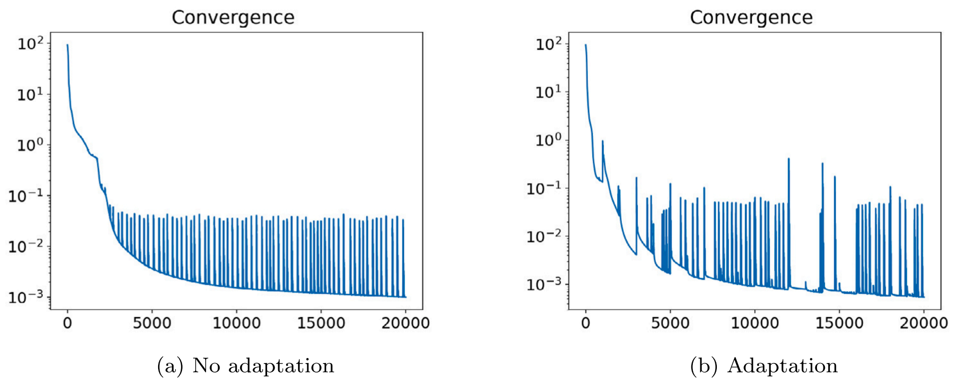


Fig. 18. Comparison of convergence for PINN without adaptation (left panel) and with adaptation (right panel) for benchmark B4. Computations were done on RTX 4070 GPU.

Hypothesis 2. The continuity of the mapping $\theta \mapsto g_\theta$ (Hypothesis 3) follows from standard results in neural networks with differentiable activations.

9. Conclusions and future work

This paper introduced an adaptive sampling strategy for Physics-Informed Neural Networks (PINNs) directly inspired by mesh refinement methods in the Finite Element Method community. Rather than fully relying on uniform sampling, the approach refines a coarse guiding mesh by subdividing elements where the residual is large, ensuring that the collocation points naturally concentrate in regions that require a steeper gradient. Comparisons on a range of 1D and 2D benchmark problems indicate faster training and a more robust convergence compared to the nonadaptive approach, particularly in the presence of steep gradients or boundary-layer phenomena.

This approach offers several core advantages. It naturally extends to arbitrary-dimensional problems when given a suitable guiding mesh and nested adaptation strategy. It remains agnostic to specific mesh refinement policies and ensures that collocation densities can be dynamically scaled in proportion to local residual. Crucially, the total number of collocation points and the number of boundary points are decoupled from the mesh size and can be adjusted at any stage. The probability distributions for sampling within elements or on their boundaries can also be flexibly changed.

The benchmark results confirm the synergy of two seemingly opposing aspects: a deterministic h-FEM inspired approach that refines collocation points in high-residual regions, balanced by a stochastic ‘blur’ that smooths out sampling. This strategy significantly reduces computational effort, especially compared to versions without adaptation. The stochastic ‘smoothing’ of the collocation points allocation brings additional advantage, protecting the strategy to omit any global minimizer with the probability 1 (see Section 6 item 8).

Future work will focus on aligning mesh refinement with temporal evolution in time-dependent problems (e.g., using causal sampling), scaling up to three-dimensional domains, and comparing performance against methods such as RAR, PDF-based refinement, and R3 sampling. Rigorous theoretical studies will also be essential for establishing stronger convergence guarantees.

The strategy can be extended in several ways. First, combining it with causal sampling [27] appears promising for time-dependent PDEs, ensuring that mesh refinement proceeds in sync with evolving solution features. Second, further testing on three-dimensional domains and more complex geometries would clarify how mesh-based sampling scales with problem dimensionality. Finally, a systematic comparison with other adaptive approaches such as RAR [24], PDF-based refinement [25], or R3 sampling [27] could pinpoint performance gains. At the same time, rigorous theoretical analyses would establish deeper convergence guarantees.

Finally, the proposed residual-driven adaptation mechanism could be naturally extended to variational formulations such as Variational Physics-Informed Neural Networks (VPINNs) [36], and Robust Variational Physics-Informed Neural Networks (RVPINNs) [37], where local error indicators and element-wise refinement strategies are inherently compatible with our mesh-guided collocation framework. This opens a promising direction for further methodological and theoretical development.

CRediT authorship contribution statement

Jan Trynda: Writing – original draft, Visualization, Software, Investigation. **Paweł Maczuga:** Software. **Albert Oliver-Serra:** Writing – original draft, Validation, Conceptualization. **Luis Emilio García-Castillo:** Writing – review & editing. **Robert Schaefer:** Writing – original draft, Formal analysis. **Maciej Woźniak:** Writing – review & editing, Writing – original draft, Validation, Supervision, Conceptualization.

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.

Acknowledgments

The authors are grateful for the support from the funds that the Polish Ministry of Science and Higher Education assigned to the AGH University of Krakow. The work was partially supported by the “Excellence initiative - research university” for the AGH University of Krakow. Albert Oliver-Serra is supported by the “Ayudas para la recualificación del sistema universitario español” (grant funded by the ULPGC, the Ministry of Universities by Order UNI/501/2021 of 26 May, and the European Union-Next Generation EU Funds) and by the grant contract “PRECOMP02 SD-24/03” (awarded by the Ministry of Universities, Science, Innovation, and Culture of the Government of the Canary Islands to the University of Las Palmas de Gran Canaria). Luis E. Garcia-Castillo has been supported by the Spanish Government throughout the project PDC2023-145929-C31 and by the Regional Government of Madrid throughout the project DISCO6G-CM.

Data availability

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

References

[1] Z. Aldirany, R. Cottereau, M. Laforest, S. Prudhomme, Multi-level neural networks for accurate solutions of boundary-value problems, Comput. Methods Appl. Mech. Engrg. 419 (2024) 116666, http://dx.doi.org/10.1016/j.cma.2023.116666.

[2] A. Bihlo, R.O. Popovych, Physics-informed neural networks for the shallow-water equations on the sphere, J. Comput. Phys. 456 (2022) 111024, http://dx.doi.org/10.1016/j.jcp.2022.111024.

[3] S. Cai, Z. Mao, Z. Wang, M. Yin, G.E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: a review, Acta Mech. Sin. 37 (12) (2021) 1727–1738, http://dx.doi.org/10.1007/s10409-021-01148-1.

[4] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951, http://dx.doi.org/10.1016/j.jcp.2020.109951.

[5] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440, http://dx.doi.org/10.1038/s42254-021-00314-5.

[6] P. Maczuga, M. Sikora, M. Skoczeń, P. Rożnawski, F. Tłuszcz, M. Szubert, M. Łoś, W. Dzwinel, K. Pingali, M. Paszyński, Physics informed neural network code for 2D transient problems (PINN-2DT) compatible with google colab, 2024, http://dx.doi.org/10.48550/arXiv.2310.03755, arXiv:2310.03755.

[7] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789, http://dx.doi.org/10.1016/j.cma.2019.112789.

[8] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, IMA J. Numer. Anal. 42 (2) (2021) 981–1022, http://dx.doi.org/10.1093/imanum/drab032.

[9] M. Raissi, P. Perdikaris, G. 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, http://dx.doi.org/10.1016/j.jcp.2018.10.045.

[10] M. Rasht-Behesht, C. Huber, K. Shukla, G.E. Karniadakis, Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions, J. Geophys. Res.: Solid Earth 127 (5) (2022) http://dx.doi.org/10.1029/2021jb023120.

[11] M.W.M.G. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (3) (1994) 195–201, http://dx.doi.org/10.1002/cnm.1640100303.

[12] I. Lagaris, A. Likas, D. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000, http://dx.doi.org/10.1109/72.712178.

[13] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, http://dx.doi.org/10.48550/arXiv.1412.6980.

[14] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (153) (2018) 1–43, URL http://jmlr.org/papers/v18/17-468.html.

[15] C.C. Margossian, A review of automatic differentiation and its efficient implementation, WIREs Data Min. Knowl. Discov. 9 (4) (2019) http://dx.doi.org/10.1002/widm.1305.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.

[17] L.D. McClenny, U.M. Braga-Neto, Self-adaptive physics-informed neural networks, J. Comput. Phys. 474 (2023) 111722, http://dx.doi.org/10.1016/j.jcp.2022.111722.

[18] R. van der Meer, C.W. Oosterlee, A. Borovykh, Optimally weighted loss functions for solving PDEs with neural networks, J. Comput. Appl. Math. 405 (2022) 113887, http://dx.doi.org/10.1016/j.cam.2021.113887.

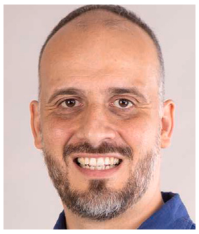
[19] M. Łoś, M. Paszyński, Robust physics informed neural networks, 2024, http://dx.doi.org/10.48550/arXiv.2401.02300, arXiv:2401.02300.

[20] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081, http://dx.doi.org/10.1137/20m1318043.

[21] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: A neural tangent kernel perspective, J. Comput. Phys. 449 (2022) 110768, http://dx.doi.org/10.1016/j.jcp.2021.110768.
[22] J. Cheng Wong, C. Ooi, A. Gupta, Y.-S. Ong, Learning in sinusoidal spaces with physics-informed neural networks, IEEE Trans. Artif. Intell. (2024) 1–15, http://dx.doi.org/10.1109/tai.2022.3192362.
[23] F.M. Rohrhofer, S. Posch, C. Gößnitzer, B.C. Geiger, On the role of fixed points of dynamical systems in training physics-informed neural networks, 2023, http://dx.doi.org/10.48550/arXiv.2203.13648, arXiv:2203.13648.
[24] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228, http://dx.doi.org/10.1137/19m1274067.
[25] M.A. Nabian, R.J. Gladstone, H. Meidani, Efficient training of physics-informed neural networks via importance sampling, Comput.-Aided Civ. Infrastruct. Eng. 36 (8) (2021) 962–977, http://dx.doi.org/10.1111/mice.12685.
[26] C. Wu, M. Zhu, Q. Tan, Y. Kartha, L. Lu, A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 403 (2023) 115671, http://dx.doi.org/10.1016/j.cma.2022.115671.
[27] A. Daw, J. Bu, S. Wang, P. Perdikaris, A. Karpatne, Mitigating propagation failures in physics-informed neural networks using retain-resample-release (R3) sampling, 2023, http://dx.doi.org/10.48550/arXiv.2207.02338, arXiv:2207.02338.
[28] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (3) (1984) 484–512, http://dx.doi.org/10.1016/0021-9991(84)90073-1.
[29] L. Schwartz, Analyse Mathematique, Hermann, Paris, 1967.
[30] H. Brezis, Functional Analysis, Sobolev Spaces and Partial Differential Equations, Springer New York, 2011, http://dx.doi.org/10.1007/978-0-387-70914-7.
[31] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.
[32] P. Petersen, M. Raslan, F. Voigtlaender, Topological properties of the set of functions generated by neural networks of fixed size, Found. Comput. Math. (2021) 375–444, http://dx.doi.org/10.1007/s10208-020-09461-0.
[33] P. Frey, P.L. George, Mesh Generation, second ed., Wiley-ISTE, 2013.
[34] M.D. Vose, The Simple Genetic Algorithm: Foundations and Theory, MIT Press, 1999, http://dx.doi.org/10.7551/mitpress/6229.001.0001.
[35] A.H.G. Rinnoy Kan, G.T. Timmer, Stochastic global optimization methods, Math. Program. 39 (1987) 27–56.
[36] C. Uriarte, M. Bastidas, D. Pardo, J.M. Taylor, S. Rojas, Optimizing variational physics-informed neural networks using least squares, Comput. Math. Appl. 185 (2025) 76–93, http://dx.doi.org/10.1016/j.camwa.2025.02.022, URL https://www.sciencedirect.com/science/article/pii/S0898122125000811.
[37] S. Rojas, P. Maczuga, J. Muñoz-Matute, D. Pardo, M. Paszyński, Robust variational physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 425 (2024) 116904, http://dx.doi.org/10.1016/j.cma.2024.116904, URL https://www.sciencedirect.com/science/article/pii/S0045782524001609.

Jan Trynda graduated in 2022 with engineering degree and then got his master degree in 2023 at Computer Science at AGH University of Science and Technology, Kraków, Poland. His research interest includes physics informed neural networks as well as algorithms for different architectures of parallel machines.

Paweł Maczuga received Master's degree in Computer Science from AGH University of Kraków (Poland) in 2021. He is currently pursuing PhD degree in the same university. His work centers around graph grammars (with applications in mesh generation) and simulations using Finite Element Method as well as Physics-Informed Neural Networks.



Albert Oliver-Serra is an associate professor at the University of Las Palmas de Gran Canaria, Spain. His research focuses on applying numerical methods to environmental problems. He has participated in seven research projects studying wind fields, solar radiation, and air quality, leading to the development of two registered software programs: “MapSol” and “Wind3D”, with the latter generating technology transfer contracts. His research interests also include mesh generation for environmental modeling and simulation. His recent research, in the framework of the project "Machine Learning Methods for Environmental Problems" that he's leading, has expanded into Physics- Informed Neural Networks (PINNs). He has a strong international collaboration with Professor Maciej Paszyski's research group at AGH University.

Luis Emilio Garcia-Castillo was born in Madrid, Spain, in 1967. He received the Ingeniero de Telecomunicación degree and the Ph.D. degree from the Universidad Politócnica de Madrid, Madrid, in 1992 and 1998, respectively. From 1997 to 2000, he was an Associate Professor with the Universidad Politócnica de Madrid. From 2000 to 2005, he was an Associate Professor with the Universidad de Alcaló, Madrid. Since 2005, he has been with the Department of Signal Theory and Communications, Universidad Carlos III de Madrid, Madrid. He has been the Principal Investigator of five projects of the National Plan of Research, Spain, one of the Regional Plan of Research, Madrid, and one with the American Air Force Office of Scientific Research, Arlington County, VA, USA. He has also participated in a number of projects and contracts, financed by international, European, and national institutions and companies. He has authored one book, five contributions for chapters and articles in books, over 58 articles in international journals, and over 100 papers in international conferences, symposiums, and workshops, plus a number of national publications and reports. His current research interests include the application of numerical methods to high-performance computational electromagnetics including finite elements, hp-adaptivity, hybrid methods, and domain decomposition methods. Dr. Garcia-Castillo received two prizes for his Ph.D. thesis from the Colegio Oficial Ingenieros de Telecomunicación, Madrid, and the Universidad Politócnica de Madrid.

Robert Schaefer is a Full professor at the Faculty of Computer Science, AGH University of Science and Technology. Author and co-author of about 200 scientific books, papers, and conference contributions, in particular author of the book "Foundation of Genetic Global Optimization" (Springer 2007). General chair of the PPSN 2010 Conference and Steering Committee Member of the PPSN Series. PC member and cochair of more than 100 scientific conferences in computational sciences and artificial intelligence. Recent research areas: memetic adaptive algorithms solving forward and inverse problems for PDEs—application to oil and gas surveying, theory of stochastic population-based algorithms, computing multi-agent systems. Former research: modeling of the blood flow in arteries, modeling of nonlinear flow in porous media.

Maciej Wozniak is an associated professor on Institute of Computer Science in AGH University of Science and Technology, Kraków, Poland. He received his master degree in Computer Science in 2013. He received his bachelor degree in Business Management in 2015. In 2017 he received his Ph.D. in computational science from AGH-UST. Since 2012 he is a member of Prof. Maciej Paszynski research group, working primarily on fast parallel direct solvers for isogeometric finite element methods targeting different parallel architectures.