BTD-PINN: Boundary-to-Domain Adaptive Point Propagation in Physics-Informed Neural Networks

by

Dean Polimac

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday October 7th, 2025 at 12:00.

Student number:	5060699	
Project duration:	February 27th, 2025 – October 7th, 2025	
Thesis committee:	Dr. David M. J. Tax,	TU Delft, advisor
	Dr. Jing Sun,	TU Delft, daily supervisor
	Dr. Alexander Heinlein,	TU Delft, committee member

An electronic version of this thesis is available at http://repository.tudelft.nl/.

BTD-PINN: Boundary-to-Domain Adaptive Point Propagation in Physics-Informed Neural Networks

Dean Polimac

Abstract

Physics-Informed Neural Networks (PINNs) [1] have emerged as a promising mesh-free approach for approximating solutions of partial differential equations (PDEs). While their flexibility enables applications across diverse scientific domains, their performance is highly sensitive to the sampling of collocation points. Existing adaptive strategies, such as PACMANN [2], propagate points based on residual gradients but rely on uniform initialization within the domain, which may underutilize the critical role of boundary and initial conditions.

This paper introduces a boundary-to-domain (BTD) propagation strategy, where collocation points are initialized exclusively on the boundaries and subsequently propagated inward. By firstly focusing training on the boundary and initial conditions, the method aims to provide a more intuitive sampling progression that reflects the fundamental nature of PDEs. We combine this propagation scheme with optimizer alternation between Adam and L-BFGS for stable and efficient training.

Experimental evaluation on benchmark PDEs demonstrates that BTD propagation achieves competitive performance compared to classical PINNs and PACMANN [2] strategies, particularly on high-dimensional problems such as the 5D Poisson equation. While results on simpler PDEs, such as the 1D Burgers' and nonlinear Schrödinger equations, reveal limitations and sensitivity to hyperparameters, the approach highlights both the potential and challenges of boundary-focused sampling. Overall, this work provides new insights into the role of collocation point placement in PINNs and opens up directions for reducing variance, automating hyperparameter schedules, and further leveraging boundary information in adaptive sampling.

1 Introduction

Physics-Informed Neural Networks (PINNs) were first introduced by Raissi et al. [1] as a class of neural networks designed to approximate solutions of partial differential equations (PDEs). Partial differential equations are defined over a single or multiple spatial dimensions and often times a temporal one, which form the PDEs domain.

Prior to PINNs, PDEs were solved using numerical techniques such as finite-difference, finite-element, and spectral methods [3]. To this day, these methods are still commonly used, as they are supported by decades of research and are known to perform well. Despite this, there are several advantages that make PINNs more appealing. Firstly, all numerical techniques require a discrete mesh of points for which they can approximate the solution. This approach lacks versatility, as then the solution approximation is only available for the points which are on the mesh. Additionally, the number of points required to achieve a decent approximation is often times large and incurs high computational costs, and thus does not scale well for high-dimensional PDEs [1].

On the other hand, if one were to consider a standard data-driven neural network, instead of a PINN, they would notice more stable training and smoother convergence. Despite this, purely

data-driven neural networks struggle to find the true solution of PDE, that is, they lack the ability to extrapolate [1]. This is most often an issue for time-dependent PDEs where the network is trained on a short time interval, but is required to make approximations beyond the temporal horizon it was trained on.

PINNs provide an alternative by learning a continuous approximation of the PDE solution, whilst being able to extrapolate well enough beyond the training domain. A PINN takes as input the coordinates of a point within the domain and is trained to approximate the solution at that point for the given PDE. Unlike standard neural networks, which rely on supervised training, PINNs can be trained without any labeled data, due to the formulation of the loss function which they are trained to minimize. This is achieved by embedding the PDE, together with its initial and boundary conditions, in the loss function of the neural network. Since the PDE is formulated in such a way that it should always sum up to zero, the network adjusts its weight parameters so that the loss is minimized for any given point. It is important to note that, if available, empirical data can also be included in the loss function; however, in this work we do not focus on this data-driven approach, but instead center our attention on the previously described physics-driven approach.

Given their versatility and mesh-free formulation, PINNs have been applied in a wide range of scientific and engineering domains, including fluid dynamics [4, 5], heat transfer [6, 7], weather prediction [8], geoscience [9, 10], and molecular dynamics [11, 12, 13, 14]. These applications demonstrate the versatility of the method in modeling complex dynamical systems governed by PDEs.

Although the aforementioned aspects present advantages over numerical solvers and pure data-driven neural networks, they come at a cost. Training PINNs is computationally demanding, often requiring many epochs for convergence. Moreover, their performance is highly sensitive to the choice and distribution of collocation points, which must be carefully selected to ensure stable and accurate learning [15, 16, 17, 18]. Collocation points refer to the set of domain-specific coordinates at which the PDE residual is evaluated, and thus they determine how well the network learns to satisfy the physical laws across the domain. This raises an important question: How should these points be sampled to ensure that the network effectively learns the underlying solution of the PDE?

Currently, there exist two broad strategies for selecting collocation points, namely non-adaptive and adaptive sampling. In non-adaptive sampling, collocation points are chosen once at the start of training and remain fixed throughout the process. In contrast, adaptive sampling periodically re-samples or adjusts the collocation points as training progresses. Within the adaptive category, common approaches include random resampling and Latin hypercube resampling [15]. Despite their widespread use, these methods are relatively simple, and until recently little emphasis had been placed on developing more sophisticated sampling strategies. However, a growing body of work has shown that the way collocation points are selected can have a significant impact on the performance and stability of PINNs [2, 16, 19, 20].

Several adaptive sampling strategies have been proposed to improve PINN training, among them, the PACMANN method [2] is particularly notable. PACMANN propagates collocation points using different gradient ascent approaches to move the collocation points to regions where the PDE residual is high. This allows the points in the domain to shift toward regions where the approximation is poorest, while the boundary points remain fixed. Although effective, PACMANN begins by sampling points uniformly throughout the domain, which may under-utilize the critical role of boundary and initial conditions in shaping the solution. Since PDEs are fundamentally determined by these conditions, uniform initialization risks placing many collocation points in regions that provide little guidance during early training [19].

To address this limitation, we propose a new boundary-to-domain (BTD) propagation strategy. Our approach initializes collocation points exclusively on the boundaries and propagates them inward,

encouraging the network to learn the PDE solution in a manner consistent with its boundary and initial constraints. In practice, during training, we alternate between the Adam optimizer [21], and the L-BFGS optimizer [22] which is used for efficient convergence. The point propagation only takes place during the Adam phase, since L-BFGS relies on a fixed loss landscape to approximate second-order information reliably. This boundary-to-domain propagation approach is a more intuitive sampling strategy, since PDE solutions are fundamentally determined by their boundary and initial conditions. The motivation for adopting this perspective is supported by recent work that emphasizes the importance of temporal and spatial sampling when training PINNs [16, 20].

The structure of this article is as follows. In Section 2 we discuss previous and relevant work in the field of PINNs. Section 3 gives a brief overview of PINNs and introduces our proposed sampling method. The results are presented in Section 4, while they are analyzed in Section 5. Lastly, in Section 6 we summarize our findings and propose ideas for future work.

2 Related Work

In this Section we present the most common and relevant sampling approaches currently used. This is in no way an exhaustive list, and our main focus is to cover approaches tangential to ours.

To our knowledge, the first work proposing an adaptive sampling approach is by Lu *et al.* [23] that introduced residual-based adaptive refinement (RAR), where points are resampled according to the distribution of PDE residuals. This approach successfully identifies regions of poor approximation but requires evaluating residuals at a large number of candidate points, which can be computationally costly. Later extensions such as RAR-D and RAD [15] introduced nonlinear residual-based probability distributions or hybrid strategies, but these often lead to a continuously growing training set, which increases memory and training overhead.

Other methods build on this residual-based principle, such as DAS-PINNs by Tang et al. [24], where a generative model (KRNet [25]) is trained to approximate the residual distribution. While this avoids explicit uniform resampling, it requires training an additional neural network, adding computational overhead. Similarly, Mao and Meng [26] suggested splitting the sampling space into subdomains, and Zhou et al. [27] proposed weighted resampling based on sensor placement. These approaches can improve convergence but often require additional heuristics or domain knowledge to balance the sampling distribution.

A different perspective was introduced by Celaya et al. [28], who framed collocation selection as a fixed-budget optimization problem. By computing residuals at random points and applying spectral decomposition (SVD + QR-DEIM), they identify the most informative points. While this approach improves the efficiency of point selection compared to purely random sampling, it comes with several limitations. Reliance on SVD and QR decompositions adds significant computational overhead, which can become costly for large-scale or high-dimensional PDEs. Moreover, performance on higher-dimensional PDEs is not explored in the work, as it is only evaluated on benchmark 1D problems, namely the Burgers' and Allen-Cahn equations. This suggests that the limitations and practicality of this approach are not yet thoroughly investigated.

Another compelling method is presented in [20], where the authors propose a causality-aware loss for time-dependent PDEs, ensuring that earlier states are learned before later ones. In essence, this causality-aware approach exploits the fact that most PDEs contain a time component that is used to sample the collocation points, thus preserving the physical causality inherent in dynamical systems. However, one drawback of this method is the additional complexity in the loss design, which may require careful tuning of weighting or scheduling strategies. Additionally, it is not clear how this approach could be used to approximate solutions of time-independent PDEs, since causality cannot

be exploited in those settings.

Building on this idea, Daw et al. [16] introduced the Retain–Resample–Release (R3) framework, where collocation points with high residuals are retained while low-residual ones are resampled, preventing PINNs from converging to trivial solutions. A causal extension further ensures that learning progresses is temporally consistent. While effective in mitigating propagation failures, R3 comes with its own limitations, mainly the frameworks dependence on several hyperparameters (e.g., thresholds and update intervals) whose optimal settings may vary depending on the problem. When it comes to the causal extension, it shares the same drawback as work done by [20], where it is not clear how this approach can be used on time-independent PDEs.

Lastly, the work by Visser *et al.* [2] tackles the sampling problem by performing gradient ascent to propagate the collocation points. The idea is to move the initially sampled points to regions with higher residuals instead of resampling them. Points that are sampled at the domain boundaries remain fixed throughout training, while points sampled within the domain are propagated. Authors explore the impact of different propagation algorithms on the performance of the network. Some of the propagation strategies include algorithms used by well-known optimizers such as RMSprop [29], Momentum [30], and Adam [21].

In summary, existing methods either rely on repeated residual evaluations, grow the training set indefinitely, or require uniform sampling over the entire domain. Our approach seeks to address these limitations by initially sampling points at the domain boundaries and propagating them inward. This boundary-to-domain strategy introduces a progression of collocation points, focusing the training effort on the regions most relevant to learning the PDE solution.

3 Methodology

First, this section will provide a brief contrast between numerical solvers and PINNs. Furthermore, we provide a more formal overview of Physics-Informed Neural Networks as defined by Raissi *et al.* [1]. We proceed by presenting our method for sampling and propagating collocation points during the training phase of a PINN. Lastly, we demonstrate the different network training approaches and their hyperparameter settings.

3.1 Physics-Informed Neural Networks

Classical approaches to solving partial differential equations (PDEs) rely on numerical solvers such as finite-difference, finite-element, finite-volue, and spectral methods [3]. These techniques approximate the solution on a discrete grid, which has two main drawbacks: (i) they provide solutions only at the predefined grid points, and (ii) the number of grid points grows rapidly with dimension, making high-dimensional PDEs computationally expensive.

Physics-Informed Neural Networks (PINNs) provide an alternative by learning a continuous approximation of the PDE solution. A PINN is a neural network that takes spatial and temporal coordinates (x, t) as input and outputs an estimate $\hat{u}(x, t)$ of the true solution $u(x, t)$. This inherently solves the issue that classical approaches have, where the solution can only be approximated at the predefined grid points. On the other hand, unlike standard supervised learning, which requires labeled data, PINNs can be trained without explicit solution data by enforcing the PDE itself as a learning constraint. This approach is referred to as the physics-driven approach – contrary to the data-driven approach which requires labeled data – and is the one we are taking in this research. This is because our focus lies in the propagation of collocation points, thus we cannot rely on preexisting fixed data.

We consider a time-dependent partial differential equation of the general form

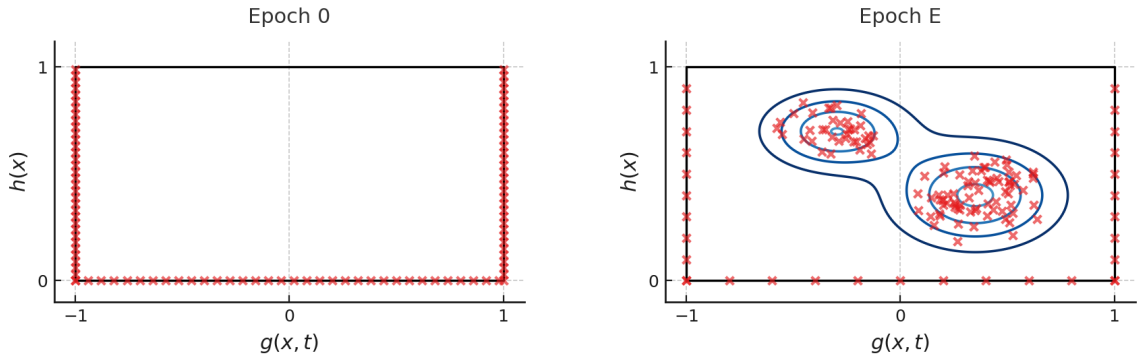
$$\frac{\partial u}{\partial t} + \mathcal{N}[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \tag{1}$$

where $u(x, t)$ is the true solution of the PDE, $\Omega \subset \mathbb{R}^D$ is the spatial domain, and T is the temporal horizon. We use $\mathcal{N}[\cdot]$ to denote the differential operator that is specific to the PDE, which when applied to u produces its spatial derivatives. Lastly, we use x to denote spatial coordinates and t for temporal coordinates.

To make the problem well-posed, we must also specify initial conditions (IC) describing the PDE's state at time $t = 0$, and boundary conditions (BC) describing its behavior on the spatial boundary $\partial\Omega$. Without these, the PDE would, in general, not have a unique solution. We define the initial and boundary conditions as follows:

$$\begin{aligned} u(x, 0) &= h(x) \\ \mathcal{B}[u](x^{bc}, t) &= g(x, t), \quad x^{bc} \in \partial\Omega, \quad t \in [0, T] \end{aligned} \tag{2}$$

Here, $h(x)$ defines the initial condition for a given spatial coordinate x, and $\mathcal{B}[\cdot]$ is the boundary operator describing the type of boundary condition imposed. For instance, in the heat equation we could impose a Dirichlet condition that fixes the temperature on the boundary (e.g., $u(x^{bc}, t) = 0$), or a Neumann condition that fixes its heat flux on the spatial boundary (e.g., $\frac{\partial u}{\partial x} = 0$ on $\partial\Omega$). Together with the initial condition $u(x, 0) = h(x)$, which specifies the starting temperature distribution inside the domain, these constraints ensure that the PDE problem is well-posed. Figure 1a depicts the initial setting of collocation points in a 2D domain, with the initial condition $h(x)$, and boundary condition $g(x, t)$, as well the propagation of points towards regions of higher residuals at a later epoch E.



(a) Epoch 0: Boundary initialization.

(b) Epoch E: Adaptive propagation.

Figure 1: Illustration of boundary-to-domain propagation. Collocation points are initialized along the boundaries of the domain (a), and later propagated into the domain towards high-residual regions (b).

During training, the PINN learns parameters θ such that $\hat{u}(x, t)$ satisfies the PDE, the initial condition, and the boundary conditions. This is achieved by minimizing the physics-informed loss function shown in Equation 3. The loss function consists of three terms, namely, the loss at the initial condition, the loss on the spatial boundaries, and the residual loss that enforces the PDE

dynamics inside the domain. The loss function can be rewritten into Equation 4, where $\mathcal{L}_{ic}$ is the initial condition term, $\mathcal{L}_{bc}$ is the boundary condition term, and $\mathcal{L}_{res}$ represents the residual loss.

$$\begin{aligned}\mathcal{L}(\theta) =& \frac{\lambda_{ic}}{N_{ic}} \sum_{i=1}^{N_{ic}} (\hat{u}(x_i^{ic}) - h(x_i^{ic}))^2 + \\ & \frac{\lambda_{bc}}{N_{bc}} \sum_{i=1}^{N_{bc}} (\mathcal{B}[\hat{u}](x_i^{bc}, t_i))^2 + \\ & \frac{\lambda_{res}}{N_{res}} \sum_{i=1}^{N_{res}} (\frac{\partial \hat{u}}{\partial t} + \mathcal{N}[\hat{u}](x_i, t_i))^2\end{aligned} \tag{3}$$

For each of the loss terms, we use a regularization term $\frac{\lambda}{N}$ to average the loss over the number of collocation points N, which basically gives us a combined MSE loss for each of the terms. The explicit form of the physics-informed loss can be seen in Equation 3.

$$\mathcal{L}(\theta) = \frac{\lambda_{ic}}{N_{ic}} \mathcal{L}_{ic} + \frac{\lambda_{bc}}{N_{bc}} \mathcal{L}_{bc} + \frac{\lambda_{res}}{N_{res}} \mathcal{L}_{res} \tag{4}$$

Here, N_{ic}, N_{bc}, N_{res} denote the number of points used for the initial condition, boundary condition, and residual terms, respectively, while the weights $\lambda_{ic}, \lambda_{bc}, \lambda_{res}$ balance out their contributions. The λ parameters are often predefined hyperparameters, but they can also be optimized as has been done in [31]. In our experiments, unless stated otherwise, all the weight terms λ are set to 1, while the number of points is varied depending on the PDE.

Based on this formulation, we can see that the placement of collocation points strongly influences how well the network learns the solution, motivating the adaptive strategy explored in this paper.

3.2 Collocation Point Propagation

The main contribution of this work is a new adaptive strategy for propagating collocation points. The key idea is to start with points sampled on the spatial boundaries $\partial\Omega$ and the temporal boundary at $t = 0$, and iteratively propagate them towards the interior of the domain. In contrast, the PACMANN method [2] begins with points sampled uniformly throughout the domain and then propagates them. Our approach therefore differs by introducing a boundary-to-domain progression, which can be viewed as a more natural sampling strategy: information from the initial and boundary conditions is gradually extended into the domain.

The intuition behind this design is that boundary and initial conditions play a crucial role in determining the solution of a PDE, therefore, we focus on learning the PDEs behavior on the boundaries first. This stands in contrast to uniform initialization, where points in the interior may not yet carry meaningful residual information if the boundary behavior has not been sufficiently learned.

To ensure consistently low loss on the initial and boundary conditions, we maintain a fixed set of points along the boundaries. Without such anchors, collocation points would eventually need to be propagated back toward the boundaries, since the network may drift away from satisfying these conditions as it focuses disproportionately on minimizing the residual loss in the domain. Whether keeping a fixed set of boundary points is strictly necessary remains an open question.

Formally, let $\mathbf{z}_i = (\mathbf{x}_i, t_i)$ denote the spatial–temporal coordinate of the i-th collocation point. The propagation is performed using a gradient ascent update rule that moves points in the direction of increasing PDE residuals. Specifically, each point is updated according to the derivative of the

squared residual with respect to its coordinates as seen in Equation 5, where the PDE residual $r(\mathbf{z}_i)$ is defined in Equation 6.

$$\mathbf{z}_i^{t+1} = \mathbf{z}_i^t + \gamma \cdot \frac{\partial}{\partial \mathbf{z}_i}\left(r(\mathbf{z}_i)^2\right) \tag{5}$$

$$r(\mathbf{z}_i) = \frac{\partial \hat{u}(\mathbf{z}_i)}{\partial t} + \mathcal{N}[\hat{u}](\mathbf{z}_i) \tag{6}$$

Thus, the update of each point is given by the derivative of its squared residual contribution. In Equation 5, t denotes the training epoch and $\gamma > 0$ is the step-size parameter. Intuitively, this update shifts collocation points towards regions where the PDE residual is large, thereby focusing training on areas where the network's approximation is poorest, as displayed in Figure 1b. To preserve stability, the fixed boundary and initial points are never propagated, ensuring that the PDE constraints remain enforced at all times. In addition, if any propagated point leaves the domain $\Omega \times [0, T]$, it is uniformly resampled at random inside the domain. This prevents the effective number of collocation points from shrinking and maintains coverage of the solution space throughout training.

3.3 Network Training

The training framework for our method combines collocation point propagation with optimizer alternation. To ensure clarity, we first explain the main components before presenting the algorithms.

Collocation point schedule.

Collocation points are propagated every I epochs, after an initial warm-up period of W epochs. The warm-up period allows the network to first fit the initial and boundary conditions before points start moving into the domain. At each propagation event, every collocation point is updated for S successive steps, using Equation 5 with step-size γ. This multi-step propagation allows points to move further into regions of high residual, rather than being limited to a single incremental update. The same multi-step propagation approach is taken in [2]. After each propagation event, points that leave the domain $\Omega \times [0, T]$ are uniformly resampled inside the domain to maintain coverage. This approach is summarized in Algorithm 1, and is applied only during training with the Adam optimizer.

Propagation is not performed during the L-BFGS phase, since L-BFGS assumes a fixed loss landscape to reliably approximate second-order information. Allowing the collocation points to move during this stage would change the training set dynamically, undermining its stability and convergence. Instead, L-BFGS is used once Adam has already adapted the collocation points, providing efficient convergence on the resulting fixed configuration.

Optimizer alternation.

Training PINNs is notoriously challenging due to their non-convex loss landscape. Following the approach taken in [2], we alternate between two optimizers: Adam and L-BFGS. During training, P alternations take place, each time the network is trained using the Adam optimizer for E_A epochs, followed by E_L epochs using the L-BFGS optimizer. This training approach is summarized in Algorithm 2.

Other works, such as [32, 33], show that this alternating approach is beneficial to PINNs as the parameter space is often quite rigid. It allows us to exploit Adam's ability to avoid getting stuck at poorly selected saddle points, while using a second-order optimizer such as L-BFGS to

efficiently converge to a minimum once Adam finds a promising region. Alternating between these two optimizers multiple times gives Adam more leeway to escape possible poorly selected regions, as is also suggested in [34].

Algorithm 1 Collocation Point Propagation (given I, W, γ, S)

Initialize: sample collocation sets:
  $\{\mathbf{z}_i\}_{i=1}^{N_{res}}$ (collocation points);
  $\{\mathbf{z}_j^{ic}\}_{j=1}^{N_{ic}}$ (initial condition points);
  $\{\mathbf{z}_k^{bc}\}_{k=1}^{N_{bc}}$ (boundary condition points);
for *epoch* $= 1$ **to** max_epochs **do**
  Compute total loss $\mathcal{L}(\theta)$ and residual squared loss $r(\mathbf{z}_i)^2$;
  Update network parameters θ with the current optimizer;
  if *epoch* $\geq W$ **and** $rem(epoch, I) = 0$ **then** ▷ $rem(a, b)$ = remainder when dividing a by b
    for $s = 1$ **to** S **do** ▷ multi-step propagation
      for each residual point $\mathbf{z}_i$ **do**
        $\mathbf{z}_i^{,t} + \gamma \cdot \frac{\partial}{\partial \mathbf{z}_i}\big(r(\mathbf{z}_i)^2\big)$
      end for
      Resample out-of-domain points uniformly inside $\Omega \times [0, T]$;
    end for
  end if
end for

Algorithm 2 Network Training with Optimizer Alternation (given P, E_A, E_L)

for $p = 1$ **to** P **do** ▷ P alternation cycles
  for *epoch* $= 1$ **to** E_A **do**
    Train network parameters θ using Adam;
    Apply Algorithm 1 (collocation point propagation).
  end for
  for *epoch* $= 1$ **to** E_L **do**
    Train network parameters θ using L-BFGS;
    Note: no propagation is performed during L-BFGS optimization.
  end for
end for

4 Results

We begin this section by introducing the experimental setup used to evaluate the performance on different PDEs. Firstly, the metric used to measure the performance of said methods on different PDEs is defined. Furthermore, the general hyperparameter settings used across all PDEs are outlined. Lastly, the definitions and results obtained for each of the PDEs are presented, including their respective hyperparameter values.

4.1 Experimental Set-up

All experiments were carried out on the Delft AI Cluster [35], each run was repeated ten times to ensure statistical reliability. The implementation is based on the DeepXDE library [23] with PyTorch [36] as the back-end. The metric used to evaluate performance is the relative L^2 norm, defined as

$$\text{Relative } L^2 = \frac{\|u - \hat{u}\|_2}{\|u\|_2}, \tag{7}$$

where u denotes the exact solution and $\hat{u}$ the PINN approximation.

Unless otherwise stated, the Adam optimizer is used with a learning rate of 10^{-3}. The propagation step size γ and the number of propagation steps S are determined via hyperparameter tuning specific to the studied PDE. In the first stage of experiments, these settings are kept fixed while the warm-up period W and propagation interval I are optimized. Initially the γ and S hyperparameters are set to the ones used for the gradient ascent approach in [2]. Once suitable values for W and I are determined, γ and S are subsequently varied for each PDE to identify their optimal configuration. Training alternates between Adam and L-BFGS as described in Section 3.3, with specific schedules detailed in each experiment.

We evaluate the impact of varying four hyperparameters of our method: the warm-up period W, the propagation interval I, the propagation step size γ, and the number of propagation steps S. For demonstration purposes, this evaluation is conducted on three representative PDEs commonly used in the PINN literature and in particular by the PACMANN baseline [2], ensuring consistency in comparison. The 1D Burgers' equation, a nonlinear time-dependent PDE, is included as it is a standard test case in PINN studies. The 1D Allen–Cahn equation is considered as a complementary nonlinear PDE with different dynamics, allowing us to assess robustness across different PDE types. Finally, we extend the evaluation to more complex PDEs, including the 5D Poisson equation and the nonlinear Schrödinger equation, to investigate how the method performs in higher-dimensional and more challenging scenarios.

To contextualize the results, we compare our propagation strategy against three baselines: the classical PINN approach without propagation or resampling, the PACMANN method that propagates points using gradient ascent, and the best performing PACMANN method where collocation points are propagated according to updates from the Adam optimizer [2].

4.2 1D Burgers' Equation

This section presents results obtained on the time-dependent 1D Burgers' equation on the domain $x \in [-1, 1], t \in [0, 1]$:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0, v = 0.01, \tag{8}$$

with boundary conditions

$$u(-1,t) = u(1,t) = 0,$$

and initial condition

$$u(x,0) = -sin(\pi x).$$

For training, the number of static boundary points is set to $N_{bc} = 40$ per boundary, and the number of points on the initial boundary is set to $N_{ic} = 160$.

Table 1: Hyperparameter ranges explored in the experiments.

Hyperparameters	**Values**
Warm-up period W	$\{0, 25, 50, 100, 200, 500\}$
Propagation interval I	$\{1, 10, 25, 50, 100, 200, 500\}$
Step size γ	$\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$
Propagation steps S	$\{1, 5, 10, 15\}$

We evaluate the effect of hyperparameter selection on Burgers' equation by varying the warm-up period W, propagation interval T, propagation step-size γ, and the number of propagation steps S. The values used for the four hyperparameters can be seen in Table 1.

Initially, $\gamma = 10^{-6}$ and $S = 1$ are set to the default values used for the gradient ascent approach in [2], after which these parameters are varied to determine their optimal settings for this PDE. We further vary the number of optimizer alternations $P \in \{1, 2, 3, 4, 5\}$. For all values of P, the total number of epochs trained with Adam and L-BFGS is fixed to 35,000 and 15,000, respectively. Consequently, the per-cycle epochs are set as

$$E_A = \frac{35{,}000}{P}, E_L = \frac{15{,}000}{P}.$$

This allows us to analyze the effect of optimizer alternation frequency while keeping the overall training budget constant.

Figure 2 presents the heatmap of the relative L^2 norm across different combinations of W and I. The Burgers' equation exhibits a less smooth error landscape, and the solution accuracy is highly sensitive to the choice of hyperparameters. In particular, frequent propagations with small values of I result in the best performance, while the warm-up period W generally has little influence.

Figure 3 illustrates how P influences performance given $I = 1$ when the total training budget is fixed at 35,000 Adam epochs and 15,000 L-BFGS epochs. The setting $P = 2$, $W = 50$, proves to perform poorly compared to other settings, and it is unclear why this is the case. The standard deviation for this setting is $\sigma^2 = 0.11$ which suggests consistent poor performance given these parameters across the ten independent runs. We can see that if $W = 0$ is selected, the number of alterations P does not significantly impact the performance. However, it is important to note that the selection of these two parameters can have a significant impact on the performance, making it as much as six times better or worse.

Table 2 reports the relative L^2 norm for the Burgers' equation over ten runs. The best metrics are highlighted in bold. While our approach achieves a very low best-case error, it also exhibits high variance across runs. By contrast, the Adam-based propagation variant achieves both strong mean performance and stability.

Figure 4 shows the relative L^2 error obtained across different combinations of hyperparameters γ and S. The results indicate that performance is sensitive to the choice of γ: larger step sizes

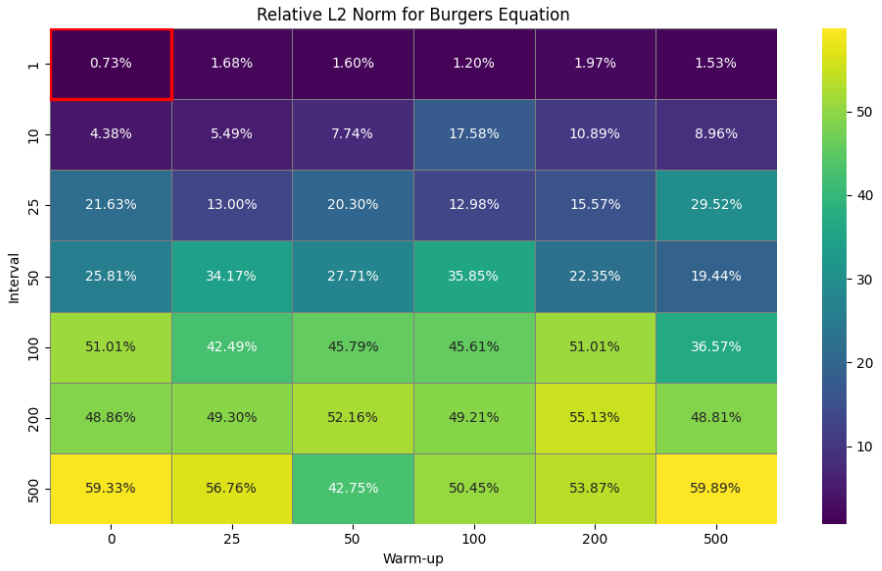


Figure 2: Relative L^2 norm for the 1D Burgers' equation across different warm-up periods W and propagation intervals I. Results were obtained using a step size of $\gamma = 10^{-6}, S = 1$ propagation steps, and $N_{res} = 2{,}500$ collocation points.

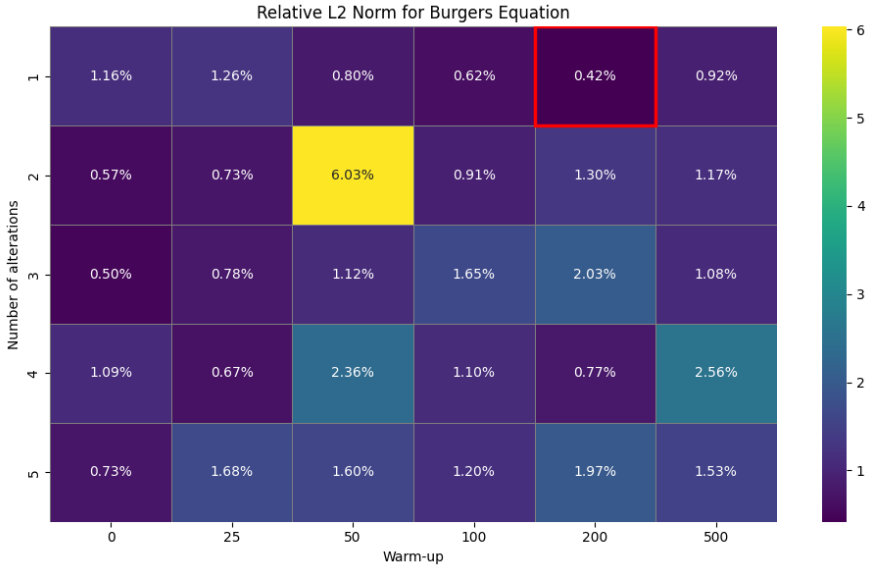


Figure 3: Relative L^2 norm for the 1D Burgers' equation across different warm-up periods W and the number of optimizer alterations P. Results were obtained using a step size of $\gamma = 10^{-6}, S = 1$ propagation steps, and $N_{res} = 2{,}500$ collocation points.

Table 2: Relative L^2 error on the 1D Burgers' equation for different sampling strategies (10 runs).

Method	**Mean Rel. L^2**	**Best Rel. L^2**	**Std. Dev.**
Boundary-to-Domain (Gradient Ascent)	0.73%	0.07%	0.56%
Classical PINN	0.31%	0.06%	0.18%
PACMANN (Gradient Ascent)	0.32%	0.05%	0.19%
PACMANN (Adam)	**0.11%**	**0.04%**	**0.05%**

($\gamma = 10^{-4}$) tend to destabilize training, while excessively small ones ($\gamma = 10^{-7}$ or below) lead to insufficient propagation and degraded accuracy. A moderate step size ($\gamma = 10^{-6}$) provides the best trade-off between stability and efficiency. Regarding the number of propagation steps, increasing S diminishes the networks performance in this setting. This suggests that a small number of propagation steps per event is sufficient to guide the collocation points towards informative regions of the domain, while further increasing S has a detrimental effect on the network's performance in this setting.

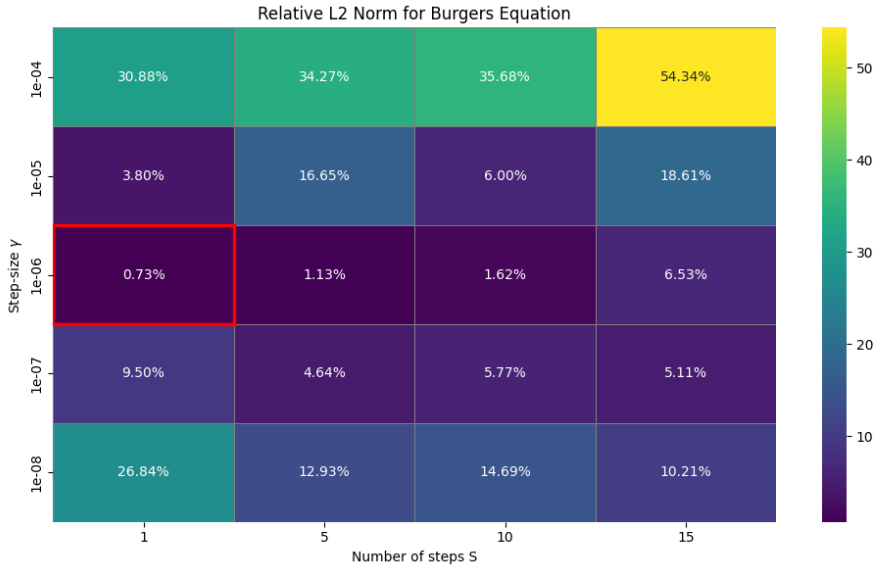


Figure 4: Relative L^2 norm for the 1D Burgers' equation across different step-sizes γ and different numbers of propagation steps S. Results were obtained using warm-up $W = 0$, propagation interval $I = 1$, $P = 5$, and $N_{res} = 2{,}500$ collocation points.

4.3 1D Allen-Cahn Equation

This section presents results obtained on the 1D Allen–Cahn equation on the domain $x \in [-1, 1]$, $t \in [0, 1]$:

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} + u^3 - u = 0, \qquad \nu = 10^{-4}, \tag{9}$$

with boundary conditions

$$u(-1, t) = u(1, t) = 0,$$

and initial condition

$$u(x, 0) = x^2 \cos(\pi x).$$

For training, we use $N_{bc} = 40$ static boundary points per boundary and $N_{ic} = 160$ points on the initial temporal boundary. The experiments are set up in the same way as for the Burgers' equation: we first vary the warm-up period W and the propagation interval I, as shown in Figure 5. In this stage, the step size is fixed to $\gamma = 10^{-8}$ and the number of propagation steps to $S = 5$.

To ensure consistency with the benchmarks, we fix the number of optimizer alternations to $P = 5$, following [2], so that the results are directly comparable. Table 3 summarizes the performance across different sampling strategies when using 2,500 collocation points. Interestingly, the classical PINN achieves strong performance at very low cost, since it requires no propagation of points. More generally, under this configuration, all methods perform competitively on the Allen–Cahn equation.

Figure 5 depicts the optimal setting of W and I for the Allen-Cahn equation, when setting $\gamma = 10^{-8}$ and $S = 5$, as done for the gradient ascent approach in [2]. We notice that for smaller values of W and I the network seems to perform well on average, whilst there is a drop in performance as both values of W and I tend to increase.

Figure 6 presents the relative L^2 error for different combinations of hyperparameters γ and S. The results show that the network is highly sensitive to γ: large values such as $\gamma = 10^{-4}$ hinder training, while very small values ($\gamma \leq 10^{-8}$) limit the propagation effect and lead to higher errors. The best performance is achieved at a moderate step size of $\gamma = 10^{-6}$, in line with the trends observed for the Burgers' equation.

With respect to the number of propagation steps S, the overall pattern indicates that a small number of steps is sufficient. Increasing S beyond 5 does not improve performance and in some cases even leads to deterioration, suggesting that additional propagation steps may move points too aggressively, away from informative regions. This again highlights the importance of balancing stability and adaptivity in the propagation process.

An important observation is that there seems to be a pattern where the network performs well when the sizes of the parameters are inversely proportional. For large values of γ, lower values of S result in significantly better performance. Conversely, for large values of S, the network prefers a smaller step size γ. This is not completely in line with the findings made on the 1D Burgers' equation, where the network appeared to be much more sensitive to the step size γ, while the number of steps S had little impact provided that γ was set appropriately.

Table 3: Relative L^2 error on the 1D Allen-Cahn equation for different sampling strategies (10 runs).

Method	**Mean Rel. L^2**	**Best Rel. L^2**	**Std. Dev.**
Boundary-to-Domain (Gradient Ascent)	0.41%	0.13%	0.29%
Classical PINN	0.35%	**0.08%**	0.23%
PACMANN (Gradient Ascent)	0.50%	0.25%	0.14%
PACMANN (Adam)	**0.29%**	0.08%	**0.15%**

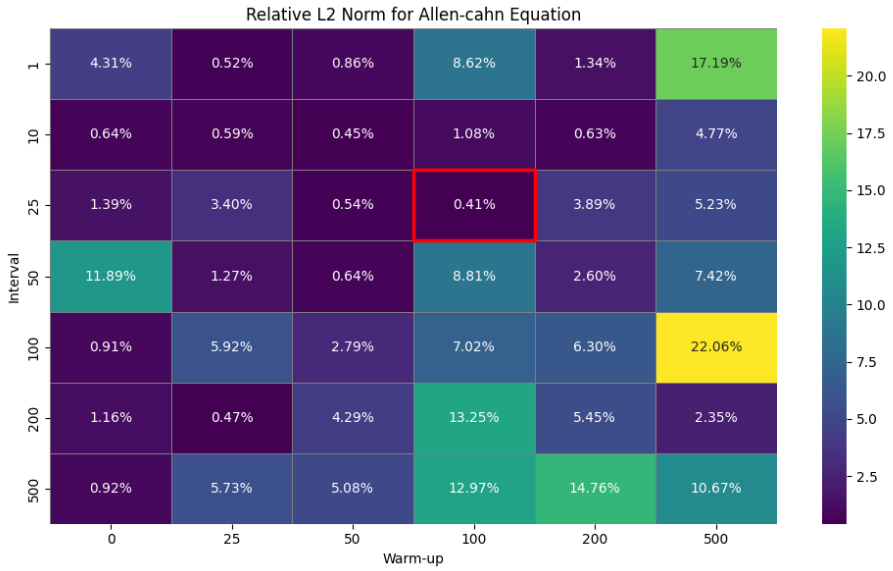


Figure 5: Relative L^2 norm for the 1D Allen-Cahn equation across different warm-up periods W and propagation intervals I. Results were obtained using a step size of $\gamma = 10^{-8}$, $S = 5$ propagation steps, and $N_{res} = 2{,}500$ collocation points.

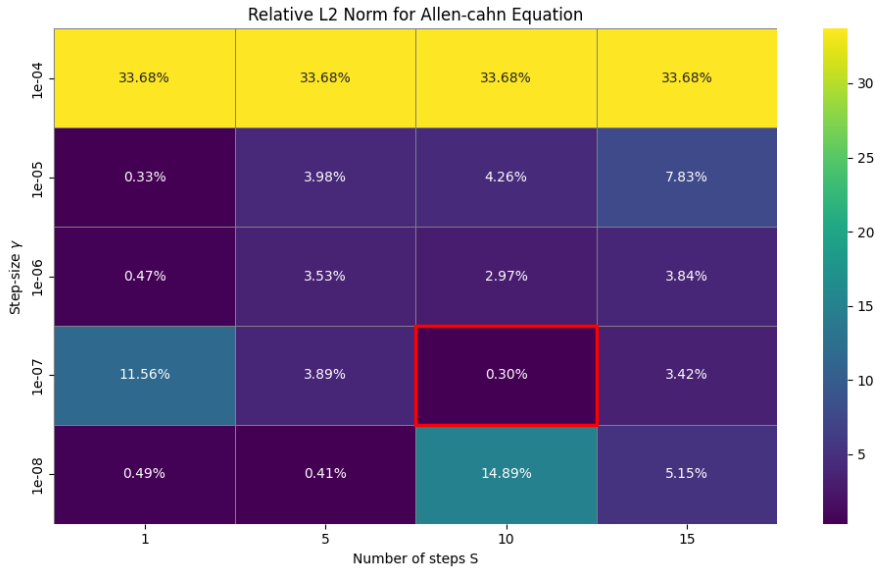


Figure 6: Relative L^2 norm for the 1D Allen-Cahn equation across different step-sizes γ and different numbers of propagation steps S. Results were obtained using warm-up $W = 100$, propagation interval $I = 25$, $P = 5$, and $N_{res} = 2{,}500$ collocation points.

4.4 5D Poisson Equation

This section presents results obtained on the 5D Poisson equation on the domain $x \in [-1, 1]^5$:

$$-\Delta v(x) = f(x), \tag{10}$$

with Dirichlet boundary conditions

$$v(x) = 0, \qquad x \in \partial\Omega,$$

where $\Omega = [-1, 1]^5$. The exact analytical solution is given by

$$v(x) = \prod_{i=1}^{5} \sin(\pi x_i).$$

The Poisson equation in higher dimensions is a widely used benchmark for evaluating PINN methods, including [2], and is particularly well suited to test the scalability of sampling strategies. Its high dimensionality makes it challenging for PINNs, since the curse of dimensionality significantly increases the difficulty of covering the solution space with collocation points.

For this experiment, the step size is fixed at $\gamma = 10^{-2}$ and the number of propagation steps at $S = 5$. These values are based on the optimal parameters reported in [2], specifically for the case where collocation points are propagated using the Adam algorithm. The warm-up period is set to $W = 100$ and the propagation interval to $I = 10$, chosen as reasonable defaults. The motivation for $W = 100$ is to allow the network sufficient time to learn the boundary behavior before propagation begins. In addition, setting $I = 10$ encourages more frequent propagation in order to better explore

the high-dimensional domain. A total of 750 collocation points and 750 fixed boundary points are used, consistent with the setup in [2].

Table 4: Relative L^2 error on the 5D Poisson equation for different sampling strategies (10 runs).

Method	**Mean Rel. L^2**	**Best Rel. L^2**	**Std. Dev.**
Boundary-to-Domain (Gradient Ascent)	**8.99%**	**7.00%**	1.11%
Classical PINN	80.98%	67.93%	2.59%
PACMANN (Gradient Ascent)	15.09%	12.05%	1.93%
PACMANN (Adam)	9.02%	7.70%	**0.96%**

Table 4 summarizes the results across different sampling strategies. The classical PINN baseline performs very poorly, with a mean relative L^2 error of nearly 81%. PACMANN with gradient ascent achieves significantly better performance at around 15%, but still lags behind the other methods. Our approach and PACMANN with Adam-based propagation achieve the lowest relative L^2 norms, both around 9%, with our method producing the overall best result in terms of performance.

4.5 1D Non-Linear Schrödinger Equation

This section presents results obtained on the nonlinear Schrödinger equation as defined in [1] on the domain $x \in [-5, 5]$, $t \in [0, \pi/2]$:

$$i\frac{\partial h}{\partial t} + 0.5\frac{\partial^2 h}{\partial x^2} + |h|^2 h = 0, \tag{11}$$

with the initial condition

$$h(0, x) = 2\,\mathrm{sech}(x),$$

and periodic boundary conditions

$$h(t, -5) = h(t, 5), \qquad \frac{\partial h}{\partial x}(t, -5) = \frac{\partial h}{\partial x}(t, 5),$$

where $h(t, x)$ is the complex-valued solution. Following [1], we define the nonlinear operator

$$f(t, x) := i\frac{\partial h}{\partial t} + 0.5\frac{\partial^2 h}{\partial x^2} + |h|^2 h.$$

This equation is a standard benchmark for PINNs as it involves a complex-valued solution, nonlinear interactions, and periodic boundary conditions. It allows us to test whether the proposed propagation strategy can capture oscillatory and nonlinear behavior in time-dependent PDEs.

We use $N_{res} = 7000$ collocation points, $N_{bc} = 800$ fixed points on the spatial periodic boundaries, and $N_{ic} = 160$ points on the initial temporal boundary. For the PACMANN baselines we follow the propagation settings with step size $\gamma = 10^{-8}$ and number of propagation steps $S = 5$. For our approach, we adopt a more balanced configuration, in line with the Allen-Cahn experiments, using $\gamma = 10^{-8}$ and $S = 5$. For fair comparison, we opt out of using a warm-up period, hence $W = 0$, and our propagation interval I is set to $I = 50$, since both Adam, and gradient ascent in [2] use the said interval.

Table 5 reports the relative L^2 error over ten runs for the classical PINN and PACMANN with gradient ascent (GA). The classical approach achieves the best performance across all three metrics using 7800 points overall, while PACMANN Adam has comparable performance in terms of relative L^2 error, as highlighted.

Table 5: Relative L^2 error on the nonlinear Schrödinger equation (10 runs).

Method	**Mean Rel. L^2**	**Best Rel. L^2**	**Std. Dev.**
Boundary-to-Domain (Gradient Ascent)	0.21%	0.19%	0.019%
Classical PINN	**0.18%**	**0.16%**	**0.009%**
PACMANN (Gradient Ascent)	0.19%	0.17%	0.014%
PACMANN (Adam)	**0.18%**	0.17%	0.006%

5 Discussion

The results presented in Section 4 highlight both the advantages and limitations of boundary-to-domain collocation point propagation. By initially sampling points at the domain boundaries, the method is designed to first learn the behavior on the domain boundaries, allowing information from the initial and boundary conditions to flow naturally toward the interior solution. This stands in contrast to uniform initialization, where collocation points placed in the interior may fall in regions where the residual signal is not yet informative. Anchoring the training process at the boundaries encourages the network to first learn parameters θ that faithfully capture the governing constraints of the PDE, which in turn provides a stronger foundation for making meaningful predictions once propagation into the domain begins.

Our approach struggles on the 1D Burgers' equation compared to others. The performance on the 1D Burgers' equation is also noticeably impacted by parameter P, where in Figure 3 we can see that for $P = 1$, $W = 200$, our approach performs significantly better, and achieves somewhat comparable results to other approaches. Additionally, due to limitations in time and computing power, we made the strong assumption that W and I are independent of γ and S, thus they were optimized separately. In order to remain consistent with the work done in [2], we selected an alteration period of $P = 5$, despite showing that for our case $P = 1$ would have been more beneficial for this PDE.

On the other hand, our approach has proven to work well on the Allen–Cahn equation, where it achieved performance comparable to the PACMANN approach, and in some cases obtained lower best-case relative errors. In this setting, it is interesting to see that our method outperformed the classical gradient ascent approach, despite using the same parameters. This suggests that propagating points from boundaries towards the domain might be more beneficial in some settings, as opposed to sampling them randomly from within the domain. It is also interesting that for the boundary-to-domain approach, a larger step size $\gamma = 10^{-6}$ is preferential, in contrast to $\gamma = 10^{-8}$ which is reported to perform best for the gradient ascent approach in [2].

However, in general, these improvements came at the cost of much higher variance across runs, which is seen on both Burgers' and the Allen–Cahn equations. A likely explanation for this is that, depending on the weight initialization, fewer or more points might move outside of the domain when propagation starts. This would cause them to be uniformly resampled within the domain, leading to an increase in variance.

Another conclusion which can be drawn, is the network's high sensitivity to the propagation step size γ, and a somewhat lower sensitivity towards the number propagation steps S. As suggested in Figure 4, we can see that when selecting $\gamma = 10^{-6}$, selecting a different value for S does not lead to significant improvements or deterioration in performance.

On the other hand, for the 1D Allen–Cahn equation, setting both γ and S proved to be very important, as suggested in Figure 6. Here, we see that there appears to be an inversely proportional relationship between the two parameters, which yields the best performance. This also shows us that when using boundary-to-domain propagation, the impact of hyperparameters plays an important

role and is problem specific.

The advantage of the proposed approach comes to light on high-dimensional PDEs, as demonstrated on the 5D Poisson equation. Here, classical PINNs performed poorly due to the curse of dimensionality, while boundary-to-domain propagation and PACMANN-based strategies performed much better. In this setting, our approach has a slight advantage over both the gradient ascent and the Adam variation of PACMANN [2]. Given that all three adaptive approaches share the same hyperparameter values, the most likely explanation for the difference in performance between the PACMANN gradient ascent approach, and our boundary-to-domain approach, is that in this setting, the distribution of points has a significant impact on the performance. Although the BTD approach resamples points that are propagated outside of the domain boundaries, some points will still remain near the boundaries, thus, the network will be given time to learn the parameters which allow it to make sufficiently good predictions for that region, before propagating the points inward. On the other hand, when using the PACMANN gradient ascent approach [2], one relies on the uniform sampling to position points near the boundaries initially; otherwise, they need to be propagated towards those regions. In case very few points are initially sampled in regions near boundaries, and given that the residual inside the domain is high at the start of the network training, most points will remain in the inner regions of the domain until the network adjusts its parameters. Since the dimensionality is high, it will take longer for points to move towards the boundaries, and along the way some point's direction of propagation might change. Essentially, the results suggest that taking a boundary-to-domain approach is more beneficial than taking a domain-to-boundary, raising an important question of whether this is a general occurrence in higher-dimensional PDEs, and it opens up avenues for further research.

Lastly, when considering the 1D non-linear Schrödinger equation, all approaches perform similarly well as shown in Table 5. In this setting, a simpler approach such as non-adaptive uniform sampling appears to be sufficient, and performing collocation point propagation does not yield any improvements in performance. This might also suggest that in certain settings, a simpler approach is more beneficial, as using adaptive sampling approaches might not lead to an improvement in performance, while still incurring higher computational costs.

At the same time, several limitations of the boundary-to-domain method must be acknowledged. First, the approach introduces a number of hyperparameters: warm-up period, propagation interval, step size, number of alterations, and number of steps, all of which interact in nontrivial ways. This increases the burden of hyperparameter tuning, which is already a major challenge in PINN training [33]. Further findings presented in Appendix B show that the previously made assumption of optimizing W and I first independently of γ and S does not always hold and portrays the sensitivity of our BTD approach to hyperparameter settings. Second, while the method reduces the need for repeated global resampling, it does not eliminate the overhead associated with updating collocation points; particularly in higher dimensions, the cost of computing residual gradients can become significant. An important observation is that the boundary-to-domain approach still depends on uniformly resampling points that have been propagated outside the domain, indicating that boundary propagation alone is insufficient. Indeed, propagating points exclusively from the boundaries performs worse than uniform resampling, as demonstrated in Appendix A, where both methods were evaluated on the 1D Burgers' equation. Nevertheless, results on the 5D Poisson equation suggest that restricting propagation to the boundary-to-domain scheme, while resampling around the boundaries, may be advantageous for training in higher-dimensional PDEs. Finally, the scope of the evaluation was limited to a small set of benchmark PDEs. While these are widely used in the literature, they may not capture the full range of difficulties encountered in real-world scientific and engineering applications.

6 Conclusion

This study introduced a boundary-to-domain (BTD) propagation strategy for sampling collocation points in Physics-Informed Neural Networks. Unlike existing approaches such as PACMANN, which begin from uniform sampling across the domain, our method samples collocation points at the initial and boundary conditions and propagates inward. This perspective emphasizes the fundamental role of boundaries in determining PDE solutions and provides a more intuitive progression of sampling.

Across benchmark problems, the results highlight both the strengths and limitations of the approach. On the 1D Allen–Cahn and 5D Poisson equations, BTD propagation achieved competitive performance compared to the classical, and PACMANN [2] strategies, suggesting particular benefits in higher-dimensional or more complex PDEs. For the 1D Burgers' equation, however, performance was more sensitive to hyperparameters and showed higher variance across runs. In the case of the nonlinear Schrödinger equation, adaptive propagation provided little benefit over classical PINNs, indicating that the utility of propagation is problem dependent.

A key finding is the method's strong sensitivity to the propagation hyperparameters, especially the step size γ, and number of propagation steps S. While this highlights the flexibility of the approach, it also increases the burden of tuning, which remains a significant challenge in training PINNs. Moreover, although sampling collocation points at the boundaries appears to improve training in some settings, it is not universally advantageous.

In summary, boundary-to-domain propagation offers a promising alternative to existing adaptive sampling strategies, particularly for high-dimensional PDEs where traditional PINNs struggle. At the same time, it introduces open questions regarding variance reduction, automatic hyperparameter selection, and the necessity of fixed boundary points. These findings open up several avenues for future work, for example, developing adaptive schedules for γ and S by decreasing the step size as training progresses, analogous to learning rate decay used in different optimization algorithms. Another interesting question is whether propagating points using the Adam algorithm instead of gradient ascent yields better performance when taking the boundary-to-domain approach. Lastly, it remains an open question whether strictly following the boundary-to-domain scheme—without uniform resampling and instead resampling only around the boundaries—constitutes a generally better strategy for higher-dimensional PDEs.

To conclude, this work shows that while boundary-to-domain propagation does not universally outperform existing strategies, it offers meaningful improvements in settings where residuals are difficult to approximate or where dimensionality poses a major challenge.

References

[1] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational physics* 378 (2019), pp. 686–707.

[2] Coen Visser, Alexander Heinlein, and Bianca Giovanardi. "PACMANN: Point Adaptive Collocation Method for Artificial Neural Networks". In: *arXiv preprint arXiv:2411.19632* (2024).

[3] Eitan Tadmor. "A review of numerical methods for nonlinear partial differential equations". In: *Bulletin of the American Mathematical Society* 49.4 (2012), pp. 507–554.

[4] Xiaowei Jin et al. "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations". In: *Journal of Computational Physics* 426 (2021), p. 109951.

[5] Ryno Laubscher. "Simulation of multi-species flow and heat transfer using physics-informed neural networks". In: *Physics of Fluids* 33.8 (2021).

[6] Shengze Cai et al. "Physics-informed neural networks for heat transfer problems". In: *Journal of Heat Transfer* 143.6 (2021), p. 060801.

[7] Darioush Jalili et al. "Physics-informed neural networks for heat transfer prediction in two-phase flows". In: *International Journal of Heat and Mass Transfer* 221 (2024), p. 125089.

[8] Jaideep Pathak et al. "Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators". In: *arXiv preprint arXiv:2202.11214* (2022).

[9] Ehsan Haghighat and Ruben Juanes. "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113552.

[10] Xiaoping Zhang et al. "GW-PINN: A deep learning algorithm for solving groundwater flow equations". In: *Advances in Water Resources* 165 (2022), p. 104243.

[11] Mahmudul Islam et al. "Extraction of material properties through multi-fidelity deep learning from molecular dynamics simulation". In: *Computational Materials Science* 188 (2021), p. 110187.

[12] Thomas Stielow and Stefan Scheel. "Reconstruction of nanoscale particles from single-shot wide-angle free-electron-laser diffraction patterns with physics-informed neural networks". In: *Physical Review E* 103.5 (2021), p. 053312.

[13] Chensen Lin et al. "Operator learning for predicting multiscale bubble growth dynamics". In: *The Journal of Chemical Physics* 154.10 (2021).

[14] Chensen Lin et al. "A seamless multiscale operator neural network for inferring bubble dynamics". In: *Journal of Fluid Mechanics* 929 (2021), A18.

[15] Chenxi Wu et al. "A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks". In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115671.

[16] Arka Daw et al. "Rethinking the importance of sampling in physics-informed neural networks". In: *arXiv preprint arXiv:2207.02338* (2022).

[17] Marcus Münzer and Chris Bard. "A curriculum-training-based strategy for distributing collocation points during physics-informed neural network training". In: *arXiv preprint arXiv:2211.11396* (2022).

[18] Vittorio Bauduin, Salvatore Cuomo, and Vincenzo Schiano Di Cola. "Impact of collocation point sampling techniques on PINN performance in groundwater flow predictions". In: *Journal of Computational Mathematics and Data Science* 14 (2025), p. 100107.

[19] Arka Daw et al. "Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling". In: *arXiv preprint arXiv:2207.02338* (2022).

[20] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. *Respecting causality is all you need for training physics-informed neural networks*. 2022. arXiv: 2203.07404 [cs.LG]. URL: https://arxiv.org/abs/2203.07404.

[21] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[22] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[23] Lu Lu et al. "DeepXDE: A deep learning library for solving differential equations". In: *SIAM review* 63.1 (2021), pp. 208–228.

[24] Kejun Tang, Xiaoliang Wan, and Chao Yang. "DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations". In: *Journal of Computational Physics* 476 (2023), p. 111868.

[25] Keju Tang, Xiaoliang Wan, and Qifeng Liao. "Deep density estimation via invertible block-triangular mapping". In: *Theoretical and Applied Mechanics Letters* 10.3 (2020), pp. 143–148.

[26] Zhiping Mao and Xuhui Meng. "Physics-informed neural networks with residual/gradient-based adaptive sampling methods for solving partial differential equations with sharp solutions". In: *Applied Mathematics and Mechanics* 44.7 (2023), pp. 1069–1084.

[27] Chenhong Zhou et al. "Enhanced Physics-Informed Neural Networks with Optimized Sensor Placement via Multi-Criteria Adaptive Sampling". In: *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2024, pp. 1–8.

[28] Adrian Celaya, David Fuentes, and Beatrice Riviere. "An Adaptive Collocation Point Strategy For Physics Informed Neural Networks via the QR Discrete Empirical Interpolation Method". In: *arXiv preprint arXiv:2501.07700* (2025).

[29] T. Tieleman. *Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude*. 2012. URL: https://cir.nii.ac.jp/crid/1370017282431050757.

[30] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1139–1147. URL: https://proceedings.mlr.press/v28/sutskever13.html.

[31] Levi McClenny Ulisses Braga-Neto. "Self-Adaptive Physics-Informed Neural Networks using a Soft Attention Mechanism". In: (2021).

[32] Stefano Markidis. "The old and the new: Can physics-informed deep-learning replace traditional linear solvers?" In: *Frontiers in big Data* 4 (2021), p. 669097.

[33] Pratik Rathore et al. "Challenges in training pinns: A loss landscape perspective". In: *arXiv preprint arXiv:2402.01868* (2024).

[34] Seifallah Elfetni and Reza Darvishi Kamachali. "PINNs-MPF: A physics-informed neural network framework for multi-phase-field simulation of interface dynamics". In: *Engineering Analysis with Boundary Elements* 176 (2025), p. 106200.

[35] Delft AI Cluster (DAIC). *The Delft AI Cluster (DAIC), RRID:SCR_025091*. 2024. DOI: 10.4233/rrid:scr_025091. URL: https://doc.daic.tudelft.nl/.

[36] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

A Resampling Strategies

Figures 7 and 8 depict the relative L^2 norm for the 1D Burgers' equation using two different resampling strategies, uniform and Gaussian, respectively. In Gaussian resampling, points that exit the domain are relocated to a region just inside the boundary they have crossed. This is achieved by centering a half-Gaussian distribution at the boundary and sampling new points from it, ensuring they remain within the domain. However, as the figures indicate, this method performs worse than uniform resampling and additionally introduces a hyperparameter, σ^2, which serves as the variance of the half-Gaussian distribution. This further complicates the already difficult process of hyperparameter optimization without any apparent improvement.

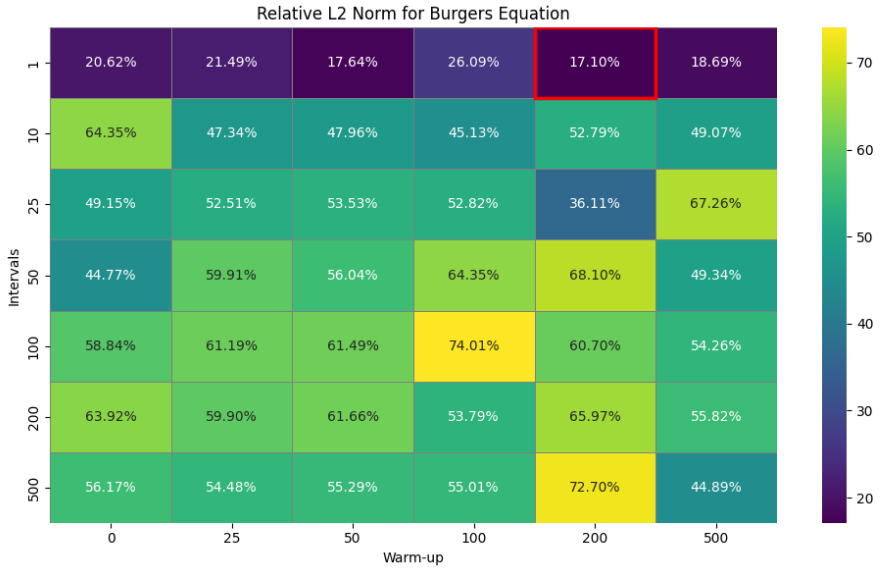


Figure 7: Relative L^2 norm for the 1D Burgers' equation for different combinations of warm-up W and propagation periods I. Results were obtained using warm-up $\gamma = 1e^{-5}$, $P = 5$, and $N_{res} = 1{,}000$ with uniform resampling.

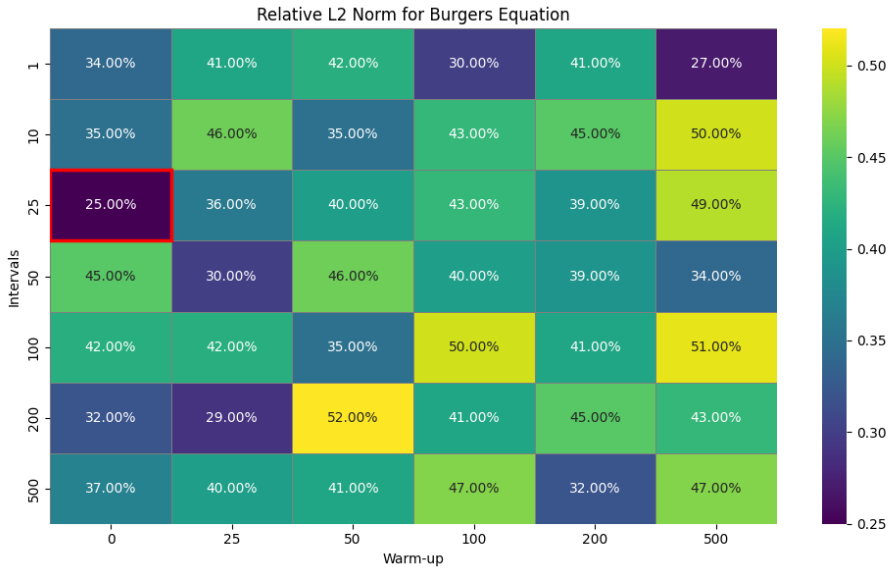


Figure 8: Relative L^2 norm for the 1D Burgers' equation for different combinations of warm-up W and propagation periods I. Results were obtained using warm-up $\gamma = 1e^{-5}$, $P = 5$, and $N_{res} = 1{,}000$ with Gaussian resampling ($\sigma^2 = 0.75$).

B Hyperparameter Optimization

To obtain the results in Section 4, a strong assumption is made that warm-up W and propagation I can be optimized independently of hyperparameters γ and S. To outline the limitations of this approach, and further stress the importance of hyperparameter optimization when relying on the BTD approach, Figure 10 displays a setting where – despite using suboptimal W and I according to Figure 5 – better performance is achieved. On the other hand, Figure 9 suggests that in this setting – where again W and I are suboptimal according to the findings depicted in Figure 2 – a somewhat more stable performance is achieved for $\gamma = 10^{-6}$, and the network is less sensitive to hyperparameter S. This further displays the sensitivity of the BTD approach to hyperparameter settings and presents an important consideration when training the network.

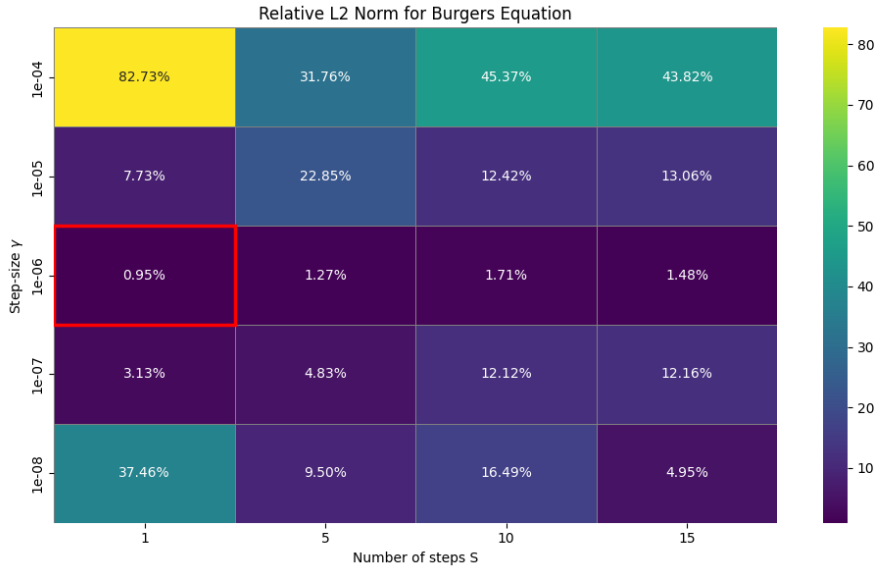


Figure 9: Relative L^2 norm for the 1D Burgers' equation across different step-sizes γ and different numbers of propagation steps S. Results were obtained using warm-up $W = 200$, propagation interval $I = 1$, $P = 5$, and $N_{res} = 2{,}500$ collocation points.

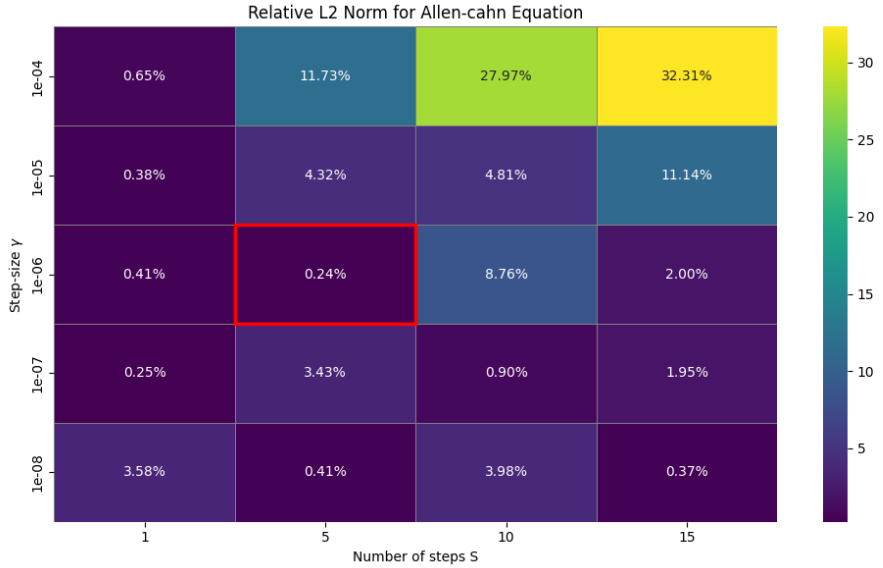


Figure 10: Relative L^2 norm for the 1D Allen-Cahn equation across different step-sizes γ and different numbers of propagation steps S. Results were obtained using warm-up $W = 25$, propagation interval $I = 100$, $P = 5$, and $N_{res} = 2{,}500$ collocation points.