ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230521

Adaptive Self-Supervision Algorithms for Physics-Informed Neural Networks

Shashank Subramanian[a;*], **Robert M. Kirby**[b], **Michael W. Mahoney**[a,c,d] **and Amir Gholami**[c]

[a]Lawrence Berkeley National Laboratory
[b]University of Utah
[c]University of California, Berkeley
[d]International Computer Science Institute

Abstract. Physics-informed neural networks (PINNs) incorporate physical knowledge from the problem domain as a soft constraint on the loss function, but recent work has shown that this can lead to optimization difficulties. Here, we study the impact of the location of the collocation points on the trainability of these models. We find that the vanilla PINN performance can be significantly boosted by adapting the location of the collocation points as training proceeds. Specifically, we propose a novel adaptive collocation scheme which progressively allocates more collocation points (without increasing their total number) to areas where the model is making higher errors (based on the gradient of the loss function in the domain). This, coupled with a judicious restarting of the training during any optimization stalls (by simply resampling the collocation points in order to adjust the loss landscape) leads to better estimates for the prediction error. We present results for several problems, including a 2D Poisson and diffusion-advection system with different forcing functions. We find that training vanilla PINNs for these problems can result in up to 70% prediction error in the solution, especially in the regime of low collocation points. In contrast, our adaptive schemes can achieve up to an order of magnitude smaller error, with similar computational complexity as the baseline. Furthermore, we find that the adaptive methods consistently perform on-par or slightly better than vanilla PINN method, even for large collocation point regimes. The code for all the experiments has been open sourced.

1 Introduction

A key aspect that distinguishes scientific ML (SciML) [10, 24, 29, 3, 16, 14] from other ML tasks is that scientists typically know a great deal about the underlying physical processes that generate their data. For example, while the Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs) used to simulate the physical phenomena may not capture every detail of a physical system, they often provide a reasonably good approximation. In some cases, we know that physical systems have to obey conservation laws (mass, energy, momentum, etc.). In other cases, we can learn these constraints, either exactly or approximately, from the data. In either case, *the main challenge in SciML lies in combining such scientific prior domain-driven knowledge with large-scale data-driven methods from ML in a principled manner*.

* Corresponding Author. Email: shashanksubramanian@lbl.gov

One popular method to incorporate scientific prior knowledge is to incorporate them as soft-constraints throughout training, as proposed with Physics Informed Neural Networks (PINNs) [17, 10, 12]. These models use penalty method techniques from optimization [2] and formulate the solution of the PDE as an unconstrained optimization problem that minimizes a self-supervision loss function that incorporates the domain physics (PDEs) as a penalty (regularization) term. Formulating the problem as a soft-constraint makes it very easy to use existing auto-differentiation frameworks for SciML tasks. Once trained, these PINNs can be promising alternatives to classical numerical methods and can offer significant computational advantages by circumventing prohibitive costs associated with solving large systems of equations and/or time integration (for spatiotemporal PDEs). However, training PINNs can be very difficult, and it is often challenging to solve the optimization problem [11, 26, 6]. This could be partly because the self-supervision term typically contains complex terms such as (higher-order) derivatives of spatial functions and other nonlinearities that cause the loss term to become ill-conditioned [11]. This is very different than unit ℓ_p ball or other such convex functions, more commonly used as regularization terms in ML. Several solutions such as loss scaling [26], curriculum or sequence-to-sequence learning [11], tuning of loss function weights [13], and novel network architectures [19] have been proposed to address this problem.

One overlooked, but very important, parameter of the training process is the way that the self-supervision is performed in PINNs, and in particular which data points in the domain are used for enforcing the physical constraints (commonly referred to as *collocation points* in numerical analysis). In the original work of [17], the collocation points are randomly sampled in the beginning of the training and kept constant throughout the learning process. However, we find that this is sub-optimal, and instead we propose adaptive collocation schemes. We show that these schemes can result in an order of magnitude better performance, with similar computational overhead.

Background. We focus on scientific systems that have a PDE constraint of the following form:

$$\mathcal{F}(u(\boldsymbol{x})) = 0, \qquad \boldsymbol{x} \in \Omega \subset \mathbb{R}^d, \tag{1}$$

where $\mathcal{F}$ is a differential operator representing the PDE, $u(\boldsymbol{x})$ is the state variable (i.e., physical quantity of interest), Ω is the physical domain, and $\boldsymbol{x}$ represents spatial domain (2D in all of our results). To

Please check ArXiv or contact the authors for any appendices or supplementary material mentioned in the paper.

ensure existence and uniqueness of an analytical solution, there are additional constraints specified on the boundary, $d\Omega$, as well (such as periodic or Dirichlet boundary conditions). One possible approach to learn a representation for the solution is to incorporate the PDE as a hard constraint, and formulate a loss function that measures the prediction error on the boundary (where data points are available):

$$\min_{\theta} \mathcal{L}(u) \quad \text{s.t.} \quad \mathcal{F}(u) = 0, \tag{2}$$

where $\mathcal{L}(u)$ is typically a data mismatch term (this includes initial/boundary conditions but can also include observational data points), and where $\mathcal{F}$ is a constraint on the residual of the PDE system (i.e., $\mathcal{F}(u)$ is the residual) under consideration. Since constrained optimization is typically more difficult than unconstrained optimization [2], this constraint is typically relaxed and added as a penalty term to the loss function. This yields the following unconstrained optimization problem, namely the PINNs (soft constrained) optimization problem:

$$\min_{\theta} \mathcal{L}(u) + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}. \tag{3}$$

In this problem formulation, the regularization parameter, $\lambda_{\mathcal{F}}$, controls the weight given to the PDE constraints, as compared to the data misfit term; θ denotes the parameters of the model that predicts $u(\boldsymbol{x})$, which is often taken to be a neural network; and the PDE loss functional term, $\mathcal{L}_{\mathcal{F}}$, can be considered as a self-supervised loss, as all the information comes from the PDE system we are interested in simulating, instead of using direct observations. Typically a Euclidean loss function is used to measure the residual of the PDE for this loss function, $\|\mathcal{F}(u)\|_2^2$, where the ℓ_2 norm is computed at discrete points (collocation points) that are randomly sampled from Ω. This loss term is often the source of the training difficulty with PINNs [11, 6], which is the focus of our paper.

Main contributions. Unlike other work in the literature, which has focused on changing the training method or the neural network (NN) architecture, here we focus on the self-supervision component of PINNs, and specifically on the selection of the collocation points. In particular, we make the following contributions:

- We study the role of the collocation points for two PDE systems: steady state diffusion (Poisson); and diffusion-advection. We find that keeping the collocation points constant throughout training is a sub-optimal strategy and is an important source of the training difficulty with PINNs. This is particularly true for cases where the PDE problem exhibits local behaviour (e.g., in presence of sharp, or very localized, features).
- We propose an alternative strategy of resampling the collocation points when training stalls. Although this strategy is simple, it can lead to significantly better reconstruction (see Tab. 1 and Fig. 1). Importantly, this approach does not increase the computational complexity, and it is easy to implement.
- We propose to improve the basic resampling scheme with a gradient-based adaptive scheme. This adaptive scheme is designed to help to relocate the collocation points to areas with higher loss gradient, without increasing the total number of points (see Algorithm 1 for the algorithm). In particular, we progressively relocate the points to areas of high gradient as training proceeds. This is done through a cosine-annealing that gradually changes the sampling of collocation points from uniform to adaptive through training. We find that this scheme consistently achieves better performance than the basic resampling method, and it can lead to more than 10x improvements in the prediction error (see Tab. 1 and Fig. 1).
- We extensively test our adaptive schemes for the two PDE systems of Poisson and diffusion-advection while varying the number of collocation points for problems with both smooth or sharp features. While the resampling and adaptive schemes perform similarly in the large collocation point regime, the adaptive approach shows significant improvement when the number of collocation points is small and the forcing function is sharp (see Tab. 2).

2 Related work

There has been a large body of work studying PINNs [18, 4, 9, 21, 32, 7, 20] and the challenges associated with their training [6, 28, 27, 26, 11, 25]. The work of [26] notes these challenges and proposes a loss scaling method to resolve the training difficulty. Similar to this approach, some works have treated the problem as a multi-objective optimization and tune the weights of the different loss terms [31, 1]. A more formal approach was suggested in [13] where the weights are learned by solving a minimax optimization problem that ascends in the loss weight space and descends in the model parameter space. This approach was extended in [15] to shift the focus of the weights from the loss terms to the training data points instead, and the minimax forces the optimization to pay attention to specific regions of the domain. However, minimax optimization problems are known to be hard to optimize and introduce additional complexity and computational costs. More recently, the work of [25] shows that incorporating causality in time can help training for time-dependent PDEs. There is also recent work that studies the role of the collocation points. For instance, [23] refines the collocation point set without learnable weights. They propose an auxiliary NN that acts as a generative model to sample new collocation points that mimic the PDE residual. However, the auxiliary network also has to be trained in tandem with the PINN. The work of [14] proposes an adaptive collocation scheme where the points are densely sampled uniformly and trained for some number of iterations. Then the set is extended by adding points in increasing rank order of PDE residuals to refine in certain locations (of sharp fronts, for example) and the model is retrained. However, this method can increase the computational overhead, as the number of collocation points is progressively increased. Furthermore, in [8] the authors show that the latter approach can lead to excessive clustering of points throughout training. To address this, instead they propose to add points based on an underlying density function defined by the PDE residual. Both these schemes keep the original collocation set (the low residual points) and increase the training dataset sizes as the optimization proceeds. Unlike the work of [14, 8], we focus on using gradient of the loss function, instead of the nominal loss value, as the proxy to guide the adaptive resampling of the collocation points. We show that this approach leads to better localization of the collocation points, especially for problems with sharp features. Furthermore, we incorporate a novel cosine-annealing scheme, which progressively incorporates adaptive sampling as training proceeds. Importantly, we note that we keep the number of collocation points the same and only resample them unlike previous works. Not only does this not increase or imbalance the computational overheard (number of data points is fixed) but it is easier to implement as well. Finally, we acknowledge two concurrent works that have appeared recently that also look at resampling strategies for PINNs [5, 30] that have suggested similar ideas of dynamically resampling points through training; however they also focus only on the PDE residual for sampling.

3 Methods

In PINNs, we use a feedforward NN, denoted $NN(\boldsymbol{x};\theta)$, that is parameterized by weights and biases, θ, takes as input values for coordinate points, $\boldsymbol{x}$, and outputs the solution value $u(\boldsymbol{x}) \in \mathbb{R}$ at these points. As described in §1, the model parameters θ are optimized through the loss function:

$$\min_{\theta} \mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}. \tag{4}$$

We focus on boundary-value (steady state) problems and define the two loss terms as:

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{n_b} \sum_{i=1}^{n_b} \|u(\boldsymbol{x}_b^i) - \hat{u}(\boldsymbol{x}_b^i)\|_2^2, \tag{5a}$$

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{n_c} \sum_{i=1}^{n_c} \|\mathcal{F}(u(\boldsymbol{x}_c^i)\|_2^2, \tag{5b}$$

where u is the model predicted solution, $\hat{u}$ is the true solution or data, $\boldsymbol{x}_b^i$ are points on the boundary, and $\boldsymbol{x}_c^i$ are collocation points uniformly sampled from the domain Ω. Here, n_b and n_c are the number of boundary and collocation points, respectively; and the boundary loss term $\mathcal{L}_{\mathcal{B}}$ implements a Dirichlet boundary condition, where we assume that the solution values are known on the boundary $d\Omega$. In PINNs [17], the collocation points used in Eq. 5b are randomly sampled with a uniform probability over the entire space Ω in the beginning of training and then kept constant afterwards (we refer to this approach as Baseline). While a uniformly distributed collocation point set may be sufficient for simple PDEs with smooth features, we find them to be sub-optimal when the problem exhibits sharp/local features, or even fail to train. To address this, we propose the following schemes.

Resampling collocation points. Current PINNs are typically optimized with LBFGS. In our experiments, we found that in the baseline approach LBFGS often fails to find a descent direction and training stalls, even after hyperparamter tuning. This agrees with other results reported in the literature [26, 11, 6]. We find that this is partially due to the fact that the collocation points are kept constant and not changed. The simplest approach to address this is to resample the collocation points when LBFGS stalls. We refer to this approach as RESAMPLING. As we will discuss in the next section, we find this approach to be helpful for cases with moderate to large number of collocation points.

Adaptive sampling. While the RESAMPLING method is effective for large number of collocation points, we found it to be sub-optimal in the small collocation point regime, especially for problems with sharp/localized features. In this case, the RESAMPLING method still uses a uniform distribution with which to sample the new the collocation points; and, in the presence of a small number of collocation points and/or sharp/localized features, this is not an optimal allocation of the points. Ideally, we want to find a probability distribution, as a replacement for the uniform distribution, that can improve trainability of PINNs for a given number of collocation points and/or computational budget. There are several possibilities to define this distribution. The first intuitive approach would be to use the value of the PDE residual (Eq. 5b), and normalize it as a probability distribution. This could then be used to sample collocation points based on the loss values in the domain. That is, more points would be sampled in areas with with higher PDE residual, and vice versa. We refer to this approach as ADAPTIVE-R sampling (with R referring to the PDE residual). An alternative is to use the gradient of the loss function (PDE loss term $\mathcal{L}_{\mathcal{F}}$) w.r.t. the spatial grid, using that as the probability distribution with which to sample the collocation points. We refer to this approach as ADAPTIVE-G (with G referring to gradient of the loss). As we will show in the next section, when combined with a cosine annealing scheme (discussed next), both of these adaptive schemes consistently perform better than the RESAMPLING scheme.

Algorithm 1 Adaptive Sampling for Self-supervision in PINNs

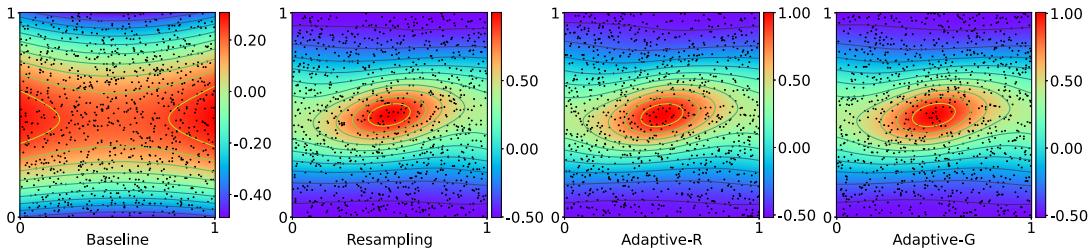
Require: Loss $\mathcal{L}$, NN model, number of collocation points n_c, PDE regularization $\lambda_{\mathcal{F}}$, T, s_w, momentum γ, max epochs i_{max}

$i \leftarrow 0$
while $i \leq i_{\text{max}}$ **do**
  Compute proxy function as the loss gradient (ADAPTIVE-G) or PDE residual (ADAPTIVE-R)
  Current proxy $\mathcal{P}_i \leftarrow \mathcal{P} + \gamma \mathcal{P}_{i-1}$
  $T_c \leftarrow i \bmod T$
  $\eta \leftarrow$ cosine-schedule(T_c, T)
  if $i \bmod e$ is true **then**
    Sample ηn_c points $\boldsymbol{x}_u$ uniformly
    Sample $(1-\eta) n_c$ points $\boldsymbol{x}_a$ using proxy function $\mathcal{P}_i$
  end if
  $\boldsymbol{x}_c \leftarrow \boldsymbol{x}_u \cup \boldsymbol{x}_a$
  Input $\boldsymbol{x} \leftarrow \boldsymbol{x}_b \cup \boldsymbol{x}_c$ where $\boldsymbol{x}_b$ are boundary points
  $u \leftarrow NN(\boldsymbol{x};\theta)$
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathcal{B}} + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$
  $\theta \leftarrow$ optimizer-update$(\theta, \mathcal{L})$
  if stopping-criterion is true **then**
    reset cosine scheduler $T_c \leftarrow 0$
  end if
  $i \leftarrow i + 1$
end while

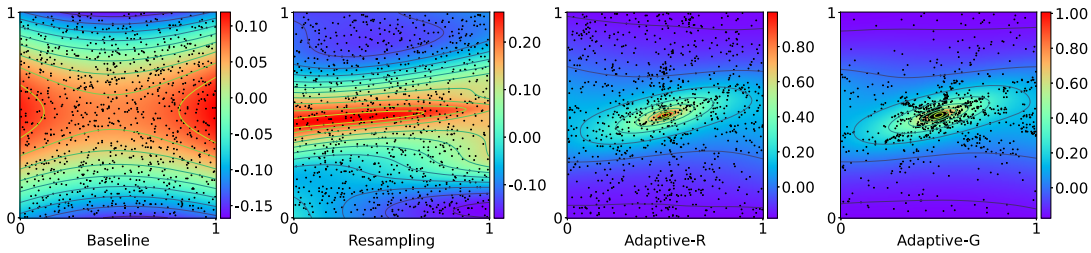
Progressive adaptive sampling. Given the non-convex nature of the problem, it might be important to not overly constrain the NN to be too locally focused, as the optimization procedure could get stuck in a local minima. However, this can be addressed by progressively incorporating adaptive sampling as training proceeds. In particular, we use a cosine annealing strategy. This allows the NN to periodically alternate between focusing on regions of high error as well as uniformly covering larger parts of the sample space, over a period of iterations. Specifically, in each annealing period, we start with a full uniform sampling, and progressively incorporate adaptively sampled points using a cosine schedule rule of $\eta = 1/2(1 + \cos \pi T_c/T)$, where η is the fraction of points uniformly sampled, T_c is the number of epochs since the last restart, and T is the length of the cosine schedule. We use this schedule to resample the collocation every e epochs. Given the periodic nature of the cosine annealing, that approach balances local adaptivity without losing global information. We outline the adaptive sampling Algorithm 1.

4 Experiments

In this section, we show examples that highlight the prediction error improvement using our adaptive self-supervision method on two PDE systems: Poisson's equation and (steady state) diffusion-advection. We open-source our code and appendix at [22].



(a) $TC1$



(b) $TC2$

Figure 1: *We visualize the predicted solution at all testing points for both test-cases TC1 (top; smooth source function with $\sigma_f = 0.1$) and TC2 (bottom; sharp source function with $\sigma_f = 0.01$). We also visualize the location of the collocation points for the final model. We observe that for the smooth source, the resampling helps predict a good solution, while the sharp source functions still cause the model to fail. The adaptive schemes are able to capture the correct solution. We also observe that the gradient boosted sampler* ADAPTIVE-G *is more aggressive in localizing the points and shows the best prediction errors of less than 5% for TC1 and TC2.*

Problem setup and metrics. We use the same problem set up as in [17, 26, 11] for the NN model, which is a feed forward model with hyperbolic tangent activation function, trained with LBFGS optimizer. We focus on 2D spatial domains in $\Omega = [0, 1]^2$. The training data set contains points that are randomly sampled from a uniform 256^2 mesh on Ω. The testing data set is the set of all 256^2 points, along with the true solution of the PDE $\hat{u}(\boldsymbol{x})$ at these points. Furthermore, we use a constant regularization parameter of $\lambda_{\mathcal{F}} = 1\text{E-}4$. This hyperparameter is difficult to tune since it is part of the loss function. Furthermore, we also do not have access to the true solution (and hence solution error) during validation. We empirically choose this value such that the boundary loss and PDE residual loss are roughly balanced. To ensure a fair comparison, we tune the rest of the hyperparameters both for the baseline model as well as for the adaptive schemes (see appendix for hyperparameter list and values). For tuning the parameters, we use the validation loss computed by calculating the total loss over randomly sampled points in the domain (30K points for all the experiments). Note that we do not use any signal from the analytical solution, and instead only use the loss in Eq. 4. We train the optimizer for 5000 epochs with a stall criterion when the loss does not change for 10 epochs. We keep track of the minimum validation loss in order to get the final model parameters for testing. We compute our prediction error using two metrics: the ℓ_2 relative error, $\mu_1 = \|u - \hat{u}\|_2 / \|\hat{u}\|$ and the ℓ_1 error $\mu_2 = \|u - \hat{u}\|_1$, where u is the model prediction solution from the best validation loss epoch and $\hat{u}$ is the true solution. All runs are trained on an A100 NVIDIA GPU on the Perlmutter supercomputer.

4.1 2D Poisson's equation

Problem formulation. We consider a prototypical elliptic system defined by the Poisson's equation with source function $f(\boldsymbol{x})$. This system represents a steady state diffusion equation:

$$-\text{div}\, \boldsymbol{K} \nabla u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{6}$$

where $\boldsymbol{K}$ denotes the diffusion tensor. For homogeneous diffusion tensors, the solution of the poisson's equation with doubly-periodic boundary conditions can be computed using the fast Fourier transform on a discrete grid as:

$$\hat{u} = F^{-1}\Big(\frac{-1}{-(k_x^2 k_{11} + k_y^2 k_{22} + 2k_x k_y k_{12})} F(f(\boldsymbol{x}))\Big), \tag{7}$$

where F is the Fourier transform, k_x, k_y are the frequencies in the Fourier domain, and k_{11}, k_{22}, k_{12} are the diagonal and off-diagonal coefficients of the diffusion tensor $\boldsymbol{K}$. We enforce the boundary conditions as Dirichlet boundary conditions using the true solution to circumvent the ill-posedness of the doubly-periodic Poisson's equation.[1] We use a Gaussian function as the source with standard deviation σ_f and consider the diffusion tensor $k_{11} = 1, k_{22} = 8, k_{12} = 4$ to make the problem anisotropic. We consider two test-cases (TC):

- TC1: smooth source function with $\sigma_f = 0.1$
- TC2: sharp source function with $\sigma_f = 0.01$.

We show these source functions and the corresponding target solutions in the appendix. For any experiment, we sweep over hyperparameters and select the ones that show the lowest validation loss.

Observations. We start by examining the performance of the difference methods for the relatively small number of collocation points of $n_c = 1,000$ (which corresponds to 1.5% of the 256×256 domain Ω). We report the errors μ_1 and μ_2 for the different schemes in Tab. 1. We also visualize the predicted solution for each method in Fig. 1, along with the location of the collocation points overlayed for the final model. We can clearly see that the baseline model does not converge (despite hyperparameter tuning) due to the optimizer stalls. However, RESAMPLING achieves significantly better errors, especially for the

[1] Note that the NN reaches suboptimal solutions with periodic boundaries due to the forward problem ill-posedness.

Table 1: *We report the relative and absolute errors for the predicted solution for the two test-cases* TC1 *and* TC2 *for the four different schemes. The vanilla PINN training (baseline) fails to converge to a good solution for small number of collocation points (n_c), despite tuning the hyperparameter. The* RESAMPLING *method however, achieves significantly better results for the smooth testing case (*TC1*), but incurs high errors for the second test case which exhibits sharper features (*TC2*). To the contrary, the two adaptive schemes of* ADAPTIVE-R *and* ADAPTIVE-G *consistently achieve better (or comparable) results than both baseline and* RESAMPLING *for both test cases.*

Test-case	Errors	Baseline	RESAMPLING	ADAPTIVE-R	ADAPTIVE-G
TC1	μ_1	5.34E-1	**1.61E-2**	2.56E-2	1.80E-2
	μ_2	7.84E+0	1.68E-1	2.57E-1	**1.43E-1**
TC2	μ_1	7.09E-1	4.83E-1	6.03E-2	**4.08E-2**
	μ_2	15.0E+1	9.51E+0	7.20E-1	**5.04E-1**

Table 2: *We report relative errors for the Baseline,* RESAMPLING *and* ADAPTIVE-G *schemes as a function of number of collocation points n_c for five different values in $\{500, 1000, 2000, 4000, 8000\}$. We observe that the* RESAMPLING *method performs well at the larger collocation points regime and for the test-cases with smooth functions.* ADAPTIVE-G *shows a consistent performance on-par or better than the* RESAMPLING *across all n_c values.*

Test-case	Method	$n_c = 500$	$n_c = 1000$	$n_c = 2000$	$n_c = 4000$	$n_c = 8000$
TC1	Baseline	4.41E-1	5.34E-1	2.73E-2	4.70E-2	5.39E-1
	RESAMPLING	2.53E-2	**1.61E-2**	**2.14E-2**	**2.10E-2**	1.98E-2
	ADAPTIVE-G	**1.94E-2**	1.80E-2	2.17E-2	2.59E-2	**1.79E-2**
TC2	Baseline	7.64E-1	7.09E-1	7.34E-1	6.93E-1	5.44E-1
	RESAMPLING	3.85E-1	4.83E-1	6.74E-2	4.68E-2	5.85E-2
	ADAPTIVE-G	**4.86E-2**	**4.08E-2**	**4.43E-2**	**3.55E-2**	**3.91E-2**

smooth source function setup (TC1). However, for the sharp source experiment (TC2), the resampling does not help and shows an error of about 50%. Overall, the adaptive schemes show much better (or comparable) prediction errors for both test-cases. In particular, ADAPTIVE-R and ADAPTIVE-G achieve about 2–5% relative error (μ_1) for both TC1 and TC2. The visual reconstruction shown in Fig. 1 also shows the clearly improved prediction with the adaptive schemes (last two columns), as compared to the baseline with/without resampling (first two columns). Note that the ADAPTIVE-G method assigns more collocation points around the sharp features. This is due to the fact that it uses the gradient information for its probability distribution, instead of the residual of the PDE which is used in ADAPTIVE-R method. To show the effect of the scheduler, we show an ablation study with the cosine-annealing scheduler in the appendix.

Finally, we note that while computing the sampling proxies incurs additional costs, this is done only every several epochs (typically O(100), dictated by the scheduler) and hence the overall cost is the same as the baseline. For instance, the training time per epoch for TC2 is about $t_r = 0.05$ seconds, the validation time is $O(10)t_r$ (because we use around 10 times the collocation points for validation, assuming $n_c = 1000$), and the time for resampling is roughly $O(10)t_r$ (because we once again use about 10 times the points for computing the proxy function and the resampling itself is negligible cost). For ADAPTIVE-G, the resampling cost is further increased by 2x due to the fact that a backward pass is performed to compute the loss gradient (for the sampling proxy). However, we note the following: the resampling (and computation of proxy) is only done every e epochs (see line 7, Algorithm 1 and e is a tunable hyperparameter). In our experiments, $e = O(100)$. Hence, our *amortized* cost (over 5000 epochs) or wall-clock time-to-solution is roughly the same irrespective of the adaptation (Baseline $\sim$ 45 minutes vs Adapted $\sim$ 47 minutes). We do note that if e is small, then the compute cost can increase significantly. However, constantly adapting the points is detrimental for learning, as it does not allow the neural network to learn sufficiently over the sampled points and hence e is typically larger (as observed in hyperparameter tuning).

We then repeat the same experiment, but now varying the number of collocation points, n_c, from 500 to $8K$, and we report the relative error (μ_1) in Tab. 2. We observe a consistent trend, where the Baseline (PINN training without resampling) does not converge to a good solution, whereas ADAPTIVE-G consistently achieves up to an order of magnitude better error. Also, note that the performance of the RESAMPLING method significantly improves and becomes on par with ADAPTIVE-G as we increase n_c. This is somewhat expected since at large number of collocation points resampling will have the chance to sample points near the areas with sharp features. In §4.3, we show the errors for every test-case as a function of number of collocation points using 10 different random seed values to quantify the variance in the different methods. We observe that the adaptive schemes additionally show smaller variances across seeds, especially for the test-cases with sharp sources.

4.2 2D diffusion-advection equation

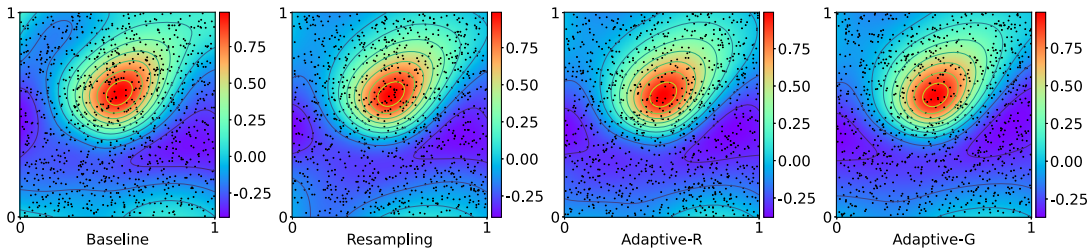
We next look at a steady-state 2D diffusion-advection equation with source function $f(\boldsymbol{x})$, diffusion tensor $\boldsymbol{K}$, and velocity vector $\boldsymbol{v}$:

$$-\boldsymbol{v} \cdot \nabla u + \text{div}\, \boldsymbol{K} \nabla u = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{8}$$

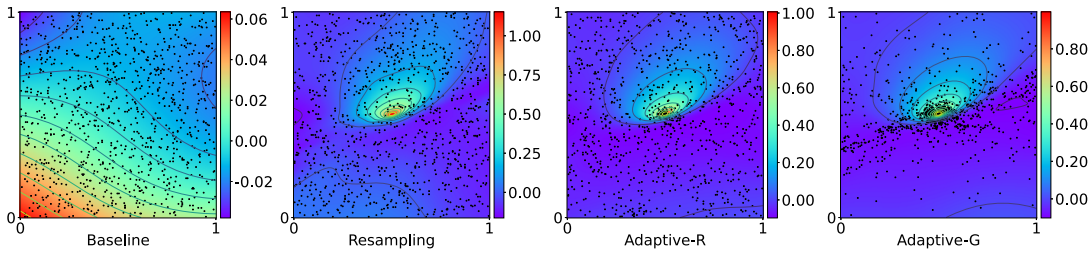
For homogeneous diffusion tensors and velocity vectors, the solution of the advection-diffusion equation with doubly-periodic boundary conditions can also be computed using the fast Fourier transform on a discrete grid as:

$$\begin{aligned} \hat{u} &= F^{-1}\big(\frac{-1}{g_1 - g_2} F(f(\boldsymbol{x}))\big), \\ g_1 &= -(k_x^2 k_{11} + k_y^2 k_{22} + 2k_x k_y k_{12}), \\ g_2 &= i k_x v_1 + i k_y v_2, \end{aligned} \tag{9}$$

where F is the Fourier transform, $i = \sqrt{-1}$, k_x, k_y are the frequencies in the Fourier domain, k_{11}, k_{22}, k_{12} are the diagonal and off-diagonal coefficients of the diffusion tensor $\boldsymbol{K}$, and v_1, v_2 are the velocity components. As before, we enforce the boundary conditions as Dirichlet boundary conditions using the true solution, and we consider a Gaussian function as the source with standard deviation σ_f. We use a diffusion tensor $k_{11} = 1, k_{22} = 8, k_{12} = 4$ and velocity vector $v_1 = 40, v_2 = 10$ to simulate sufficient advection. We consider two test-cases as before:



(a) $TC3$



(b) $TC4$

Figure 2: *We visualize the predicted solution at all testing points for both test-cases TC3 (top) and TC4 (bottom). We observe that resampling can help significantly for this system, but the adaptive schemes still show the least errors for both test-cases.*

- TC3: smooth source function with $\sigma_f = 0.1$
- TC4: sharp source function with $\sigma_f = 0.01$.

We show the source functions and the target solutions in the appendix.

Observations. We observe a very similar behaviour for the reconstruction errors as in the previous experiments. As before, we start with $n_c = 1,000$ collocation points, and we report the results in Tab. 3 and the visualizations in Fig. 2. Here, the baseline achieves slightly better performance for TC3 (14%) but completely fails (100% error) for the TC4 test case, which includes a sharper forcing function. However, the RESAMPLING achieves better results for both cases. Furthermore, the best performance is achieved by the adaptive methods. Finally, in Tab. 4 we report the relative errors for the baseline and adaptive schemes with various numbers of collocation points n_c. Similar to the Poisson system, larger values of n_c show good performance, but the baselines underperform in the low data regime for sharp sources. The resampling achieves better errors, while the adaptive methods once again consistently achieve the best performance for both data regimes.

4.3 Sensitivity to random seeds

We test the sensitivity of our methods to the randomization processes involved by repeating all experiments with 10 different random seeds (from 1 to 10) to quantify the variance in our metrics. We report the testing error and the variance in Fig. 3. We observe that the adaptive schemes consistently show the best (or comparable) performance and are superior to the RESAMPLING scheme in the low collocation points regime as well as for TC2 and TC4 (systems with sharp features). Further, the variance in the error for the adaptive schemes is small across all numbers of collocation points and is much smaller than the RESAMPLING for these two sharp source test-cases in the small collocation point regime. With larger number of collocation points the RESAMPLING scheme's variance also reduces and is comparable to the adaptation. For the smooth systems, both schemes show comparable performance and spread.

5 Conclusions

We studied the impact of the location of the collocation points on PINN models and showed that the vanilla PINN strategy of keeping the collocation points fixed throughout training often results in sub-optimal solutions. This is particularly the case for PDE systems with sharp (or very localized) features. We showed that a simple strategy of resampling collocation points during optimization stalls can significantly improve the reconstruction error, especially for moderately large number of collocation points. We also proposed adaptive collocation schemes to obtain a better allocation of the collocation points. This is done by constructing a probability distribution derived from either the PDE residual (ADAPTIVE-R) or its gradient w.r.t. the input (ADAPTIVE-G). We found that by progressively incorporating the adaptive schemes, we can achieve up to an order of magnitude better solutions, as compared to the baseline, especially for the regime of a small number of collocation points and with problems that exhibit sharp (or very localized) features. Some limitations of this current work include the following: we did not change the NN architecture (it was fixed as a feed forward NN) or tune hyperparameters relating to the architecture (which can be a significant factor in any analysis); and we only focused on 2D spatial systems (it is known that time-dependent or 3D or higher-dimensional systems can show different behaviours and may benefit from different kinds of adaptivity in space and time). We leave these directions to future work. However, we expect that techniques such as those we used here that aim to combine in a more principled way domain-driven scientific methods and data-driven ML methods will help in these cases as well.

Acknowledgements

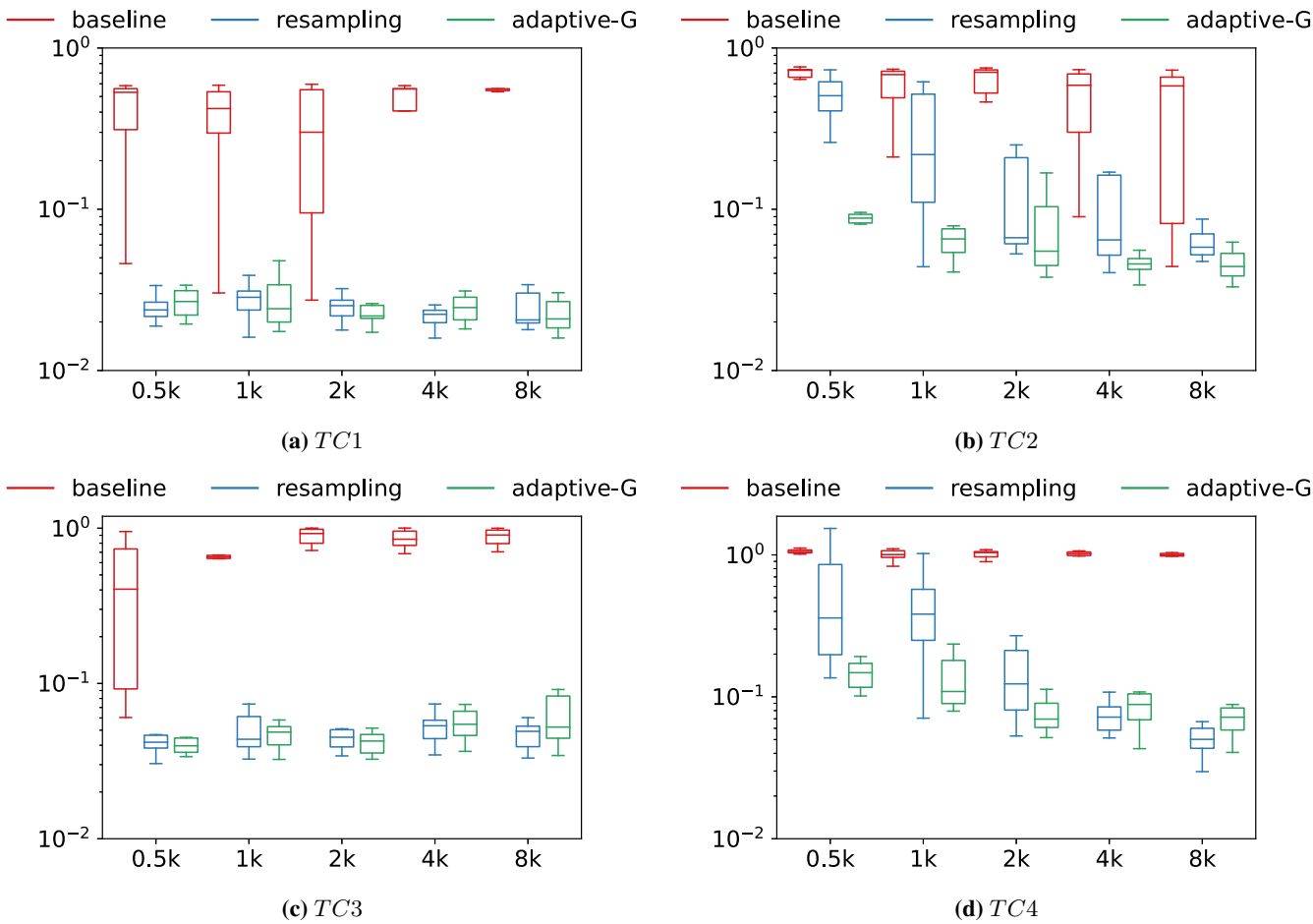
This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231. RMK would like to acknowledge AFOSR MURI FA9550-20-1-0358. MWM would like to acknowledge support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing

Table 3: *We report the relative and absolute errors for the predicted solution for the two test-cases* TC3 *and* TC4 *for the four different schemes. As before, the Baseline fails to converge to a good solution. The* RESAMPLING *method shows significant performance improvements. The two adaptive schemes of* ADAPTIVE-R *and* ADAPTIVE-G *consistently achieve better results than both baseline and* RESAMPLING *for both test-cases.*

Test-case	Errors	Baseline	RESAMPLING	ADAPTIVE-R	ADAPTIVE-G
TC3	μ_1	1.43E-1	4.48E-2	**3.515E-2**	5.5E-2
	μ_2	3.23E-1	1.63E-1	1.47E-1	**1.32E-1**
TC4	μ_1	1.06E+0	4.10E-1	**6.47E-2**	8.70E-2
	μ_2	5.23E+0	1.83E+0	3.28E-1	**3.29E-1**

Table 4: *We report relative errors for the Baseline,* RESAMPLING *and* ADAPTIVE-G *schemes as a function of number of collocation points n_c for five different values in $\{500, 1000, 2000, 4000, 8000\}$. As before, the* RESAMPLING *method performs well at the larger collocation points regime and* ADAPTIVE-G *shows a consistent performance across all n_c values.*

Test-case	Method	$n_c = 500$	$n_c = 1000$	$n_c = 2000$	$n_c = 4000$	$n_c = 8000$
TC3	Baseline	7.23E-1	1.43E-1	7.18E-1	6.99E-1	7.04E-1
	RESAMPLING	4.69E-2	**4.48E-2**	5.10E-2	5.83E-2	**3.82E-2**
	ADAPTIVE-G	**3.54E-2**	5.5E-2	**4E-2**	**4.56E-2**	4.06E-2
TC4	Baseline	1.094E+0	1.07E+0	1.09E+0	1.04E+0	9.91E-1
	RESAMPLING	2.84E-1	4.1E-1	**7.06E-2**	1.05E-1	6.69E-2
	ADAPTIVE-G	**1.13E-1**	**8.70E-2**	7.47E-2	**5.69E-2**	**5.03E-2**



(a) $TC1$ **(b)** $TC2$

(c) $TC3$ **(d)** $TC4$

Figure 3: *Summary of testing errors for baseline,* RESAMPLING *and* ADAPTIVE-G *schemes as a function of different collocation points for all the test-cases to show the variance in error across 10 preset random seeds. We observe that the variance of errors in* ADAPTIVE-G *is the least across all the numbers of collocation points and the variance in* RESAMPLING *reduces steadily and compares to* ADAPTIVE-G *in the large collocation point regime (for the systems with sharp source functions).*

Research, Scientific Discovery through Advanced Computing (SciDAC) program, under Contract Number DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory. AG was supported through funding from Samsung SAIT.

References

[1] Rafael Bischof and Michael Kraus, 'Multi-objective loss balancing for physics-informed deep learning', *arXiv preprint arXiv:2110.09813*, (2021).

[2] Stephen Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[3] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos, 'Machine learning for fluid mechanics', *Annual Review of Fluid Mechanics*, **52**, 477–508, (2020).

[4] Yuyao Chen, Lu Lu, George Em Karniadakis, and Luca Dal Negro, 'Physics-informed neural networks for inverse problems in nano-optics and metamaterials', *Optics express*, **28**(8), 11618–11633, (2020).

[5] Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne, 'Rethinking the importance of sampling in physics-informed neural networks', *arXiv preprint arXiv:2207.02338*, (2022).

[6] C. Edwards, 'Neural networks learn to speed up simulations', *Communications of the ACM*, **65**(5), 27–29, (2022).

[7] Nicholas Geneva and Nicholas Zabaras, 'Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks', *Journal of Computational Physics*, **403**, 109056, (2020).

[8] John Hanna, Jose V Aguado, Sebastien Comas-Cardona, Ramzi Askri, and Domenico Borzacchiello, 'Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks', *arXiv preprint arXiv:2109.14290*, (2021).

[9] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis, 'Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations', *Journal of Computational Physics*, **426**, 109951, (2021).

[10] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang, 'Physics-informed machine learning', *Nature Reviews Physics*, **3**(6), 422–440, (2021).

[11] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney, 'Characterizing possible failure modes in physics-informed neural networks', *Advances in Neural Information Processing Systems*, **34**, (2021).

[12] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis, 'Artificial neural networks for solving ordinary and partial differential equations', *IEEE transactions on neural networks*, **9**(5), 987–1000, (1998).

[13] Dehao Liu and Yan Wang, 'A dual-dimer method for training physics-constrained neural networks with minimax architecture', *Neural Networks*, **136**, 112–125, (2021).

[14] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis, 'Deepxde: A deep learning library for solving differential equations', *SIAM Review*, **63**(1), 208–228, (2021).

[15] Levi McClenny and Ulisses Braga-Neto, 'Self-adaptive physics-informed neural networks using a soft attention mechanism', *arXiv preprint arXiv:2009.04544*, (2020).

[16] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman, 'Universal differential equations for scientific machine learning', *arXiv preprint arXiv:2001.04385*, (2020).

[17] 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', *Journal of Computational Physics*, **378**, 686–707, (2019).

[18] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis, 'Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations', *Science*, **367**(6481), 1026–1030, (2020).

[19] Amuthan A Ramabathiran and Prabhu Ramachandran, 'Spinn: Sparse, physics-based, and partially interpretable neural networks for pdes', *Journal of Computational Physics*, **445**, 110600, (2021).

[20] Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl, 'Physics-informed neural networks for cardiac activation mapping', *Frontiers in Physics*, **8**, 42, (2020).

[21] Justin Sirignano and Konstantinos Spiliopoulos, 'Dgm: A deep learning algorithm for solving partial differential equations', *Journal of computational physics*, **375**, 1339–1364, (2018).

[22] Shashank Subramanian. https://github.com/ShashankSubramanian/adaptive-selfsupervision-pinns, 2022.

[23] Kejun Tang, Xiaoliang Wan, and Chao Yang, 'Das: A deep adaptive sampling method for solving partial differential equations', *arXiv preprint arXiv:2112.14038*, (2021).

[24] Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, et al., 'Informed machine learning–a taxonomy and survey of integrating knowledge into learning systems', *arXiv preprint arXiv:1903.12394*, (2019).

[25] Sifan Wang, Shyam Sankaran, and Paris Perdikaris, 'Respecting causality is all you need for training physics-informed neural networks', *arXiv preprint arXiv:2203.07404*, (2022).

[26] Sifan Wang, Yujun Teng, and Paris Perdikaris, 'Understanding and mitigating gradient pathologies in physics-informed neural networks', *arXiv preprint arXiv:2001.04536*, (2020).

[27] Sifan Wang, Hanwen Wang, and Paris Perdikaris, 'On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks', *arXiv preprint arXiv:2012.10047*, (2020).

[28] Sifan Wang, Xinling Yu, and Paris Perdikaris, 'When and why pinns fail to train: A neural tangent kernel perspective', *arXiv preprint arXiv:2007.14527*, (2020).

[29] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar, 'Integrating physics-based modeling with machine learning: A survey', *arXiv preprint arXiv:2003.04919*, (2020).

[30] Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu, 'A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks', *arXiv preprint arXiv:2207.10289*, (2022).

[31] Zixue Xiang, Wei Peng, Xiaohu Zheng, Xiaoyu Zhao, and Wen Yao, 'Self-adaptive loss balanced physics-informed neural networks for the incompressible navier-stokes equations', *arXiv preprint arXiv:2104.06217*, (2021).

[32] Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris, 'Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data', *Journal of Computational Physics*, **394**, 56–81, (2019).