DP-PINN: A Dual-Phase Training Scheme for Improving the Performance of Physics-Informed Neural Networks

Da Yan and Ligang He*

Department of Computer Science, University of Warwick, UK
Da.Yan@warwick.ac.uk, Ligang.He@warwick.ac.uk

Abstract. Physics-Informed Neural Networks (PINNs) are a promising application of deep neural networks for the numerical solution of nonlinear partial differential equations (PDEs). However, it has been observed that standard PINNs may not be able to accurately fit all types of PDEs, leading to poor predictions for specific regions in the domain. A common solution is to partition the domain by time and train each time interval separately. However, this approach leads to the prediction errors being accumulated over time, which is especially the case when solving "stiff" PDEs. To address these issues, we propose a new PINN training scheme, called DP-PINN (Dual-Phase PINN). DP-PINN divides the training into two phases based on a carefully chosen time point t_s. The phase-1 training aims to generate the accurate solution at t_s, which will serve as the additional intermediate condition for the phase-2 training. New sampling strategies are also proposed to enhance the training process. These design considerations improve the prediction accuracy significantly. We have conducted the experiments to evaluate DP-PINN with both "stiff" and non-stiff PDEs. The results show that the solutions predicted by DP-PINN exhibit significantly higher accuracy compared to those obtained by the state-of-the-art PINNs in literature.

Keywords: Physics-Informed Neural Networks · Partial Differential Equations · Model Training · Data Sampling.

1 Introduction

Traditional physics-based numerical methods [1][6][15] have had great success in solving partial differential equations (PDEs) for a variety of scientific and engineering problems. While these methods are accurate, they are computationally intensive for complex problems such as nonlinear partial differential equations and often require problem-specific techniques. Over the past decade, data-driven methods have gained significant attention in various areas of science and engineering. These methods can identify highly nonlinear mappings between inputs and outputs, potentially replacing or augmenting expensive physical simulations.

* Corresponding Author

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

However, typical data-driven deep learning methods tend to ignore a physical understanding of the problem domain [5].

In order to incorporate physical priors into the model training, a new deep learning technique, Physics-Informed Neural Networks (PINNs)[10], has been proposed. Propelled by vast advances in computational capabilities and training algorithms, including the availability of automatic differentiation methods, PINNs combine the idea of using neural networks as generalized function approximator for solving PDEs[4] and the idea of using the system of PDEs as physical priors to constrain the output of the neural networks, which makes neural networks a new and effective approach to solving PDEs.

The original PINN algorithm proposed in [10], hereafter referred to as the "standard PINN", is effective in estimating solutions that are reasonably smooth with simple boundary conditions (e.g., the specific boundary values are given), such as the viscous Burger's equation, Poisson's equation, Schrödinger's equation and the wave equation. On the other hand, it has been observed that the standard PINN has the convergence and accuracy problems when solving "stiff" PDEs [3] such as the nonlinear Allen-Cahn equation, where solutions contain sharp space transitions or fast time evolution.

To solve these problems, numerous methods have been proposed recently, including bc-PINN[7], the time adaptive approach[14] and SA-PINN[8]. Among them, bc-PINN is especially noteworthy for its simple and intuitive philosophy. Since the stiff PDE has sharp, fast space/time transitions[13], it is hard to predict a domain as a whole. The key idea of bc-PINN is to retrain the same neural network for solving the PDE over successive time segments while satisfying the already obtained solutions for all previous time segments.

However, our analysis reveals that that the training scheme in bc-PINN may lead to a progressive accumulation of prediction errors across successive time segments. Our explanation for this phenomenon is that after bc-PINN trains the solutions in a time segment, the trained results are used as the ground truth to train the solutions in subsequent segments. Consequently, since the predictions in initial segments inevitably contain inaccuracies, these errors propagate and amplify throughout the training process for later segments.

To address this issue, we propose a new training method called DP-PINN. In DP-PINN, the training is divided into two distinct phases. The division is informed by our observations in the benchmark experiments with existing PINN methods. Our benchmark experiments revealed that the prediction errors became notably more severe after a certain point (denoted as t_s) within the time domain $[t_{start}, t_{end}]$, where t_{start} is usually 0. In DP-PINN, we use t_s as the pivotal time point in the dual-phase training scheme.

In phase 1, DP-PINN focuses on training the network to predict solutions from t_{start} (i.e., 0) to t_s. In phase 2, we diverge from the bc-PINN's practice of using the entire solutions predicted within $[0, t_s]$ as the ground truth for subsequent training during $(t_s, t_{end}]$, Rather, we found that what is more important is the accuracy of the solutions predicted at t_s. In phase 1 of DP-PINN, we will obtain the predicted solutions on t_s.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

In phase 2, we extend the training across the entire time domain $[0, t_{end}]$ using the same neural network architecture. This phase not only trains the network for $(t_s, t_{end}]$, but also continue to refine the solutions in $[0, t_s]$. Crucially, the solutions predicted at t_s serve as "intermediate" conditions (augmenting the original boundary and initial conditions of the PDE) to guide the training in phase 2.

To improve the accuracy of predictions at t_s and overall model accuracy, we propose the new sampling strategies in DP-PINN. With the use of intermediate conditions and the new sampling strategies, DP-PINN is able to achieve much higher accuracy than the state-of-the-art methods.

In summary, DP-PINN incorporates three optimization strategies to enhance its prediction accuracy, particularly in solving complex PDEs such as Allen-Cahn equation. These strategies include: i) strategically dividing the network training into two phases around a specifically identified point t_s, ii) leveraging the predictions at t_s obtained in phase 1 as the extra "intermediate" conditions to improve accuracy in subsequent predictions, and iii) incorporating the new sampling strategies to improve the accuracy of solutions at t_s obtained in phase 1. Together, these strategies enpower DP-PINN to effectively solve a variety of PDEs, including the stiff PDE - the Allen-Cahn PDE, with significantly higher accuracy than other state-of-the-art PINN algorithms. We have conducted the experiments to validate the effectiveness of DP-PINN.

2 Related Work

The standard PINN algorithm can be unstable during training and produce inaccurate approximations around sharp space and time transitions or fail to converge entirely in the solution of "stiff" PDEs, such as the Allen-Cahn equation. Much of the recent studies on PINNs has been devoted to mitigating these issues by introducing modifications to the standard PINN algorithm that can increase training stability and accuracy of the approximation, mostly via splitting the solution domain evenly into several smaller time segments, or by using a weighted loss function during training. We discuss the main approaches of those below.

Non-Adaptive Weighting. The work in [14] points out that the neural network should be forced to satisfy the initial condition closely. Accordingly, a loss function with the form, $\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + C\mathcal{L}_I(\theta) + \mathcal{L}_r(\theta)$, was proposed, where C ($C \gg 1$) is a hyper-parameter.

Adaptive weighting. In [8], PINNs are trained adaptively, using the fully-trainable weights that force the neural network to focus on the difficult regions of the solution, which is an approach reminiscent of soft multiplicative attention masks used in Computer vision[9][12]. The key idea is to increase the weights as the corresponding losses increase, which is accomplished by training the network to minimize the losses while maximizing the weights.

Backward compatible PINN. In [7], the proposed method, termed backward compatible PINN (bc-PINN), addresses the limitation of retraining a sin-

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

gle neural network over successive time segments by ensuring that the network satisfies the solutions obtained in all previous time segments (i.e., treating the solutions in previous time segments as the ground truth for the training in subsequent time segments) during the progressive solution of a PDE system. The bc-PINN divides the time axis into several even time intervals, ensuring a comprehensive and backward-compatible solution across the entire temporal range.

Time-Adaptive Approaches. In [8], the time axis is divided into several time intervals, then PINNs are trained on them, either separately or sequentially. The initial condition for each time interval relies on the predictions in the preceding time step. This approach is time consuming due to the dependency between time steps and and the need for training multiple PINNs.

3 Method

In this section, we first give a brief overview of PINN. Next, we present DP-PINN. Finally, we describe the sampling method used in DP-PINN.

3.1 Overview of Standard PINN and Motivation of DP-PINN

Overview of Standard PINN The general form of the PDE solved by PINN can be defined as follows:

$$u_t + \mathcal{N}[u] = 0, \quad \boldsymbol{x} \in \Omega, t \in [0, t_{end}], \tag{1}$$

$$u(x,t) = g(x,t), \quad \boldsymbol{x} \in \partial\Omega, t \in [0, t_{end}], \tag{2}$$

$$u(x,0) = h(x), \quad \boldsymbol{x} \in \Omega \tag{3}$$

where x is a spatial vector variable, which includes a vector of spatial points on which we need to find solutions of u, t is time, u_t is the partial derivative of u over t, Ω is a subset of R^d (d is the dimension of the space), $\partial\Omega$ denotes the set of all boundary spatial vector variables where the boundary conditions of the PDE are enforced, and $\mathcal{N}[\cdot]$ is non-linear differential operator. Note that Equations (2) and (3) represent the boundary conditions and initial conditions of the PDE, respectively.

the solution $u(\boldsymbol{x}, t)$ is approximated by the output $u_\theta(\boldsymbol{x}, t)$ of the deep neural network (θ denotes the network parameters) with inputs $\boldsymbol{x}$ and t. The residual, $r_\theta(x, t)$, is defined as:

$$r_\theta(\boldsymbol{x}, t) = \frac{\partial}{\partial t} u_\theta(\boldsymbol{x}, t) + \mathcal{N}[u_\theta(\boldsymbol{x}, t)], \tag{4}$$

where all partial derivatives can be computed by automatic differentiation methods [2]. With the use of back-propagation [11] during training, the parameters θ can be obtained by minimizing the following loss function:

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_I(\theta) + \mathcal{L}_r(\theta) \tag{5}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

where $\mathcal{L}_b$ is the loss corresponding to the boundary condition, $\mathcal{L}_I$ is the loss due to the initial condition, and $\mathcal{L}_r$ is the loss corresponding to the residual. $\mathcal{L}_b, \mathcal{L}_I, \mathcal{L}_r$ are essentially penalties for outputs that do not satisfy (2), (3) and (4) respectively. $\mathcal{L}_b$, $\mathcal{L}_I$ and $\mathcal{L}_r$ can be defined by Equations (6)-(8), respectively.

$$\mathcal{L}_b(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_b} |[u_\theta(x_b^i, t_b^i) - g(x_b^i, t_b^i)]|^2, \tag{6}$$

$$\mathcal{L}_I(\theta) = \frac{1}{N_I}\sum_{i=1}^{N_I} |u_\theta(x_I^i, 0) - h(x_I^i)|^2, \tag{7}$$

$$\mathcal{L}_r(\theta) = \frac{1}{N_r}\sum_{i=1}^{N_r} |r_\theta(x_r^i, t_r^i)|^2 \tag{8}$$

where $\{x_I^i\}_{i=1}^{N_I}$ and $\{h(x_I^i)\}_{i=1}^{N_I}$ denote the initial points of the PDE and their corresponding values; $\{x_b^i, t_b^i\}_{i=1}^{N_b}$ and $\{g(x_b^i, t_b^i)\}_{i=1}^{N_b}$ are the points on the boundary of the PDE and their values; $\{x_r^i, t_r^i\}_{i=1}^{N_r}$ is the set of collocation points randomly sampled from the domain Ω; N_b, N_I and N_r denote the number of boundary points, initial points and the collocation points, respectively. To tune the parameters θ of the neural network, the training is done for 10k iterations of Adam, followed by 10k iterations of L-BFGS, consistent with the related work for a fair comparison in the experiments. L-BFGS uses Hessian matrix (second derivative) to identify the direction of steepest descent.

Motivation of DP-PINN We conducted the experiment to use bc-PINN to train the model for solving Allen-Cahn (AC) equation, known for its stiff nature. Figure 1 shows the distribution of its prediction errors (L2-error) across the domain. It can be observed that prediction errors escalate as the training progresses over time. This outcome can be attributed to bc-PINN's unique training approach. bc-PINN divided the entire time domain into four discrete segments, and train the model sequentially across these segments. This scheme allows bc-PINN to focus on smaller, more manageable portions of the time domain at any given moment, which effectively circumvents the challenges faced by the standard PINN that train across the full domain simultaneously. Initially, prediction errors within the early segments may appear harmless. Nonetheless, as the training adopts the outcomes of preceding segments as the groundtruth for training in subsequent ones, prediction errors will accumulate as more segments are processed.

The benchmark experiments with bc-PINN provided insights that led us to propose the ideas in our DP-PINN. A key observation is that bc-PINN tends to generate more accurate predictions in the initial segments of the time domain. This can be attributed to two main factors. Firstly, by focusing on a smaller time segment, as opposed to tackling the entire time domain in a single sweep like standard PINN, the model is better positioned to learn the features

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

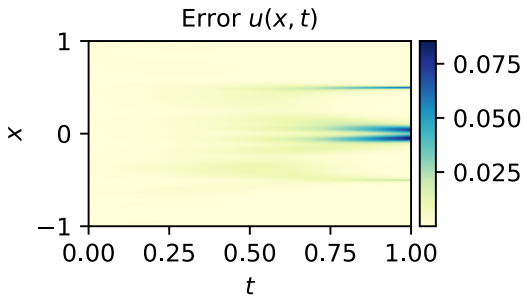

Fig. 1. bc-PINN's absolute prediction error on Allen-Cahn equation.

and relationships among points. This reduced time span simplifies the learning process, enabling more precise predictions. Secondly, the proximity of the early segment to the initial conditions, which are the true ground truth, allows the model to more accurately capture the genuine relationships between points. As the training progresses to points further from the initial conditions, the reliance on previous predictions introduces a compounding effect of inaccuracies. This observation underpins the rationale of the dual-phase training adopted in DP-PINN, aiming to mitigate the propagation of errors.

The core principle behind our DP-PINN involves identifying a critical time point, t_s, beyond which prediction errors notably intensify. We divide the model training into two phases. In phase 1, DP-PINN aims to generate the predictions at t_s as accurate as possible. Subsequently, in the phase-2 training, the predictions at t_s act as the initial condition for training the model over the subsequent time segment $(t_s, t_{end}]$. We anticipate an enhancement in prediction accuracy for $(t_s, t_{end}]$ according to the understanding gleaned from bc-PINN, where predictions are more accurate in segments closer to the initial conditions. We will present the dual-phase training scheme in subsection 3.2.

In light of the core principle underpinning DP-PINN, an important objective is to achieve utmost accuracy in predictions specifically at t_s. The accuracy of these predictions at t_s first depends critically on the selection of t_s within the time domain. Identifying a optimal t_s presents a challenge: it must be neither too close to the initial condition at time 0 nor excessively distant. The rationale behind this balance is that because the solutions predicted at t_s will be used as the initial condition for the training over $(t_s, t_{end}]$, smaller t_s means that some points in $(t_s, t_{end}]$ are further away from their initial conditions and may consequently yield more inaccurate predictions.

In order to obtain an appropriate t_s and generate accurate predictions at t_s, we first conducted benchmark experiments designed to map out the trend of prediction errors across the time domain, thereby establishing an empirical foundation for selecting t_s. Second, in the phase-2 training of DP-PINN, we incorporate a trainable parameter η in the model to fine-tune the prediction val-

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

ues at t_s. The approach of empirical determination of t_s followed by fine-tuning with η is instrumental in optimizing the accuracy of DP-PINN's predictions and enhancing the model's overall effectiveness.

	Number of Sampling Points		
	5k	10k	20k
Relative L2 error	3.391e-3	**1.375e-3**	2.297e-3

Table 1. Training accuracy of the standard PINN on the 1D viscous Burger's equation with different numbers of sampling collocation points. The numbers of initial points and boundary points were kept unchanged, which are $N_i = 100$ and $N_b = 50$ respectively, in the three sets of sampling points.

Another strategy of generating accurate predictions at t_s stems from another observation we made in the benchmark experiments. We applied the standard PINN to train the model with different numbers of sampling points, as shown in Table 1. These experiments revealed a relationship between the number of sampling points and model accuracy: increasing the sampling density generally led to more accurate training results. However, too many sampling points resulted in a decline in model performance, likely due to overfitting. The reason for this trend may be because while a sparse distribution of sampling points challenges the model's ability to learn underlying relationships, excessively dense sampling may cause the model to memorize the training data too closely, losing its generalization capability.

Building on this understanding, we tailor the sampling strategy for DP-PINN to circumvent these issues. Unlike bc-PINN, which samples the same number of points in each time segment, DP-PINN adopts a differentiated approach. Specifically, we allocate a higher density of sampling points to the phase-1 training. This targeted increase in sampling density for phase-1 training is strategic, aimed at obtaining highly accurate predictions at t_s. We will present our sampling strategies in subsection 3.3.

3.2 DP-PINN

Based on the discussions in subsection 3.1, we propose a dual-phase training scheme for PINNs. In phase 1, the model is trained on the sampled points in the time duration $[0, t_s]$, and predicts the solutions at t_s. In phase 2, the same model undergoes training across the entire domain, integrating the predictions at t_s as the intermediate condition, which also acts as the initial condition for the training in $(t_s, t_{end}]$).

The phase-1 training in DP-PINN takes as input the initial conditions of the PDE, the boundary conditions of the points that fall in $[0, t_s]$, and trains the model on the collocation points in $[0, t_s]$. The loss function used by the phase-1 training is defined in Equation 9, where $\mathcal{L}_I(\theta)$ is the one defined in Equation 7, representing the initial conditions; $\mathcal{L}_{b1}(\theta)$ represents the compliance to the

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

boundary conditions of the points in the phase-1 domain, which is defined in Equation 10; $\mathcal{L}_{r1}(\theta)$ represents the residual of the predicted solution on the collocation points in the phase-1 domain, which is defined in Equation 11; C is a hyper-parameter described in [14], which acts as a weight of the $\mathcal{L}_I(\theta)$ term in the overall loss function.

$$\mathcal{L}_1(\theta) = \mathcal{L}_{b1}(\theta) + C\mathcal{L}_I(\theta) + \mathcal{L}_{r1}(\theta) \tag{9}$$

$$\mathcal{L}_{b1}(\theta) = \frac{1}{N_{b1}} \sum_{i=1}^{N_{b1}} |[u_\theta(x_{b1}^i, t_{b1}^i) - g(x_{b1}^i, t_{b1}^i)|^2, \tag{10}$$

$$\mathcal{L}_{r1}(\theta) = \frac{1}{N_{r1}} \sum_{i=1}^{N_{r1}} |r_\theta(x_{r1}^i, t_{r1}^i)|^2, \tag{11}$$

In the phase-2 training, the model is trained across the entire domain $[0, t_{end}]$. Therefore, the original boundary conditions, initial conditions of the PDE are used as the input of the phase-2 training. this phase extends the evaluation of the model's residual to include all sampled collocation points within the entire domain. Moreover, the solutions at time t_s predicted in the phase-1 training are used as an additional condition in the phase-2 training. Based on these, the loss function for the phase-2 training is defined in Equation 12, where $\mathcal{L}_b(\theta)$ is the one defined in Equation 6, representing the original boundary conditions of the PDE; $\mathcal{L}_{t_s}(\theta, \boldsymbol{\eta})$ represents the additional intermediate condition established based on the solutions predicted in the phase-1 training at time t_s.

The term $\mathcal{L}_{t_s}(\theta, \boldsymbol{\eta})$ is defined in Equation 13, where $u'(x_{t_s}, t_{t_s})$ are the values predicted in phase 1 for the collocation points at t_s, and $\boldsymbol{\eta} = (\eta^1, ..., \eta^{N_{t_s}})$ is a set of trainable parameters aimed at fine-tuning the predicted values for the N_{t_s} collocation points (i.e., $x_{t_s}^i$) at t_s. $\mathcal{L}_{t_s}(\theta, \boldsymbol{\eta})$ essentially calculates the residual of the predictions at t_s in phase 2 by treating the phase-1 predictions at t_s as "prior". The parameter $\boldsymbol{\eta}$ is updated using Equation 14 with k indicating the learning step and μ_k the step-specific learning rate.

The base value of t_s is set to be 30% of the entire time domain. This is an empirical value gathered through our benchmark experiments. For example, in the experiments shown in fig. 1, prediction errors were not significantly pronounced at $t = 0.3$ (given a total time span of 1) according to our experimental records.

$$\mathcal{L}_2(\theta, \boldsymbol{\eta}) = \mathcal{L}_b(\theta) + C\mathcal{L}_I(\theta) + \mathcal{L}_r(\theta) + \mathcal{L}_{t_s}(\theta, \boldsymbol{\eta}) \tag{12}$$

$$\mathcal{L}_{t_s}(\theta, \boldsymbol{\eta}) = \frac{1}{N_{t_s}} \sum_{i=1}^{N_{t_s}} |u_\theta(x_{t_s}^i, t_s) - \eta^i * u'(x_{t_s}^i, t_s)|^2 \tag{13}$$

$$\boldsymbol{\eta}(k+1) = \boldsymbol{\eta}(k) - \mu_k \nabla_{\boldsymbol{\eta}} \mathcal{L}_2(\theta, \boldsymbol{\eta}) \tag{14}$$

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

3.3 Sampling Methods

We propose two sampling strategies to provide the training points for DP-PINN. N_r and N_b denote the set of collocation and boundary points sampled in the entire domain, respectively. N_{r1} and N_{b1} denote the set of collocation and boundary points for the phase-1 training, respectively. $N_{r1} \subset N_r$ and $N_{b1} \subset N_b$. N_I denotes the set of initial points. N_{t_s} denotes the set of collocation points sampled at time t_s. In both sampling strategies proposed in this work, we comply with the constraint that the total numbers of sampled collocation points and boundary points that are input for training are no more than $|N_r|$ and $|N_b|$, respectively.

Fixed sampling strategy Unlike [7] and [14], where the time axis is evenly split into multiple time intervals (e.g., bc-PINN uses 4 intervals), our approach contains two time intervals $[0, t_s]$ and $(t_s, t_{end}]$. In the phase-1 training, i.e., the training in $[0, t_s]$, the number of sampled collocation points is determined by Equation 15, where $\frac{t_s}{t_{end}} \times |N_r|$ is the number of collocation points that is proportional to the ratio of t_s to t_{end}, and therefore α can be regarded as an amplification factor that increases the sampling density in phase-1. α is a hyperparameter in training.

$$|N_{r1}| = \alpha \times \frac{t_s}{t_{end}} \times |N_r| \tag{15}$$

Similarly, the number of sampled boundary points is determined by Equation 16.

$$|N_{b1}| = \alpha \times \frac{t_s}{t_{end}} \times |N_b| \tag{16}$$

In the phase-2 training, the model is trained using the points sampled from both the initial intervals $[0, t_s]$ and the subsequent interval $(t_s, t_{end}]$. In the fixed sampling strategy, the set of sampled points in $[0, t_s]$ in the phase-2 training remains the same as that in the phase-1 training (i.e., N_{r1} and N_{b1}). To adhere to the constraint that the total number of collocation points and boundary points used for training does not exceed $|N_r|$ and $|N_b|$, respectively, we sample $|N_r| - |N_{r1}|$ collocation points and $|N_b| - |N_{b1}|$ boundary points within $(t_s, t_{end}]$ for phase-2 training. Compared to proportional sampling strategy to be presented next, this sampling strategy is deliberately designed to prioritize the accuracy of predictions at the initial stages of the timeline. This focus is particularly important for effectively addressing "stiff" PDEs ,such as Allen Cahn equation, where inaccuracies in early predictions can quickly magnify as the model extends to later time points.

Proportional sampling strategy In the former sampling strategy, we preserve the sampling density within $[0, t_s]$ for phase-2 training but sacrifice the sampling density for the subsequent interval $(t_s, t_{end}]$ due to the constraint on the total number of sampling points. In the proportional sampling strategy, after completing phase-1 training, we randomly select a proportion of the total collocation points, $\frac{t_s}{t_{end}} \times |N_r|$, and boundary points, $\frac{t_s}{t_{end}} \times |N_b|$, from the initial sets

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

N_r1 and N_b1 to use during the $[0, t_s]$ interval of phase-2 training. For the remaining time span $(t_s, t_{end}]$, we then allocate the rest of the points, $(1 - \frac{t_s}{t_{end}}) \times |N_r|$ for collocation and $(1 - \frac{t_s}{t_{end}}) \times |N_b|$ for boundary points.

This strategy adjusts the distribution of sampling points in phase-2 training, reducing the number during $[0, t_s]$ so that the number of points in each interval is proportional to their respective durations. Contrary to fixed sampling, which allocates more sampling points to the early time points, this proportional sampling strategy increases the focus on the later stages (after t_s) in the phase-2 training. This strategy is suited for simpler, non-stiff PDEs.

4 Results

We conducted the experiments to compare our DP-PINN with the state-of-the-art PINN algorithms in literature, SA-PINN[8], BC-PINN[7] and TA (Time-Adaptive) approach[14]. We also used the standard PINN to solve the tested PDEs, whose performance is reported as a baseline. We applied the above PINNs to solve the Allen-Cahn PDE and the Burger's equation.

We carried out the experiments on these two PDEs because the Allen-Cahn equation, as a "stiff" PDE, is regarded as a most challenging benchmark and used in the experiments of all the three state-of-the-art PINN algorithms. The Burger's equation is relatively easier to solve (non-stiff) and is used as the benchmark in the experiments of SA-PINN. The Burger's equation is solved in our experiments in order to demonstrate the generalization of DP-PINN in solving non-stiff PDEs.

Same as in the literature, we use relative L2-error as the metric to measure the performance of the PINN algorithms. Relative L2-error is defined by Equation 17, where N_U is the set of sampled points in the entire domain; $u(x, t)$ and $U(x, t)$ represent the predictions and the ground truth at the point x and time t, respectively.

$$Error_{L2} = \frac{\sqrt{\sum_{i=1}^{N_U} |u(x_i, t_i) - U(x_i, t_i)|^2}}{\sqrt{\sum_{i=1}^{N_U} |U(x_i, t_i)|^2}} \tag{17}$$

4.1 Allen-Cahn Equation

The Allen-Cahn reaction-diffusion equation is commonly used in the phase-field models, which are often used to model solidification and melting processes, providing insights into the behavior of phase boundaries during these transformations. In this experiment, the Allen-Cahn PDE considered is specified as follows:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

$$u_t - 0.0001u_{xx} + 5u^3 - 5u = 0, \quad x \in [-1,1], t \in [0,1], \tag{18}$$

$$u(0,x) = x^2 cos(\pi x), \tag{19}$$

$$u(t,-1) = u(t,1), \tag{20}$$

$$u_x(t,-1) = u_x(t,1) \tag{21}$$

Unlike other PDEs solved by PINNs, Allen-Cahn equation is a nonlinear parabolic PDE that challenges PINNs to approximate solutions with sharp space and time transitions, and also introduces periodic boundary conditions (eq:20-21).

In order to deal with this periodic boundary conditions, the loss function $\mathcal{L}_b(\theta)$ and $\mathcal{L}_{b1}(\theta)$ defined in Equations 6 and 10 are replaced by:

$$\mathcal{L}_b(\theta) = \frac{1}{N_b}\sum_{i=1}^{N_b} |u_\theta(-1,t_b^i) - u_\theta(1,t_b^i)|^2 + |u_{x\theta}(-1,t_b^i) - u_{x\theta}(1,t_b^i)|^2, \tag{22}$$

$$\mathcal{L}_{b1}(\theta) = \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}} |u_\theta(-1,t_{b1}^i) - u_\theta(1,t_{b1}^i)|^2 + |u_{x\theta}(-1,t_{b1}^i) - u_{x\theta}(1,t_{b1}^i)|^2, \tag{23}$$

The neural network architecture is fully connected with 4 hidden layers, each with 128 neurons. The input layer takes two inputs (two neurons) - x and t. The output is the prediction of $u_\theta(x,t)$. This architecture is identical to that used in [14] and [8], which allows a fair comparison. We set the number of collocation, initial and boundary points to $|N_r| = 20000$, $|N_I| = 100$ and $|N_b| = 100$. The amplification factor α as a hyper-parameter is set to 1.6. The number of the points sampled at time t_s for the phase-1 training is set to $|N_{t_s}| = 100$, excluding the points at the boundary. t_s is set to 0.3, and the trainable coefficient η in Equation 13 are initialized following a uniform distribution in the range of [0,1). Mini-batches are used in training, with the batch size of 32. All experiments underwent 10 independent runs with random starts and the performance reported is the average over the 10 runs.

Table 2 shows the performance of different PINNs in solving the Allen-Cahn equation. Comparing to other state-of-the-art algorithms, DP-PINN can achieve a much lower relative L2 error, which is 0.84% ± 0.29%. Note that standard PINN cannot solve the Allen-Cahn equation.

Figure 2 visualizes the numerical solutions and prediction errors obtained by DP-PINN in Table 2. Comparatively, figure 1 visualizes the prediction errors obtained by bc-PINN in this experiment. It can be seen from the figures that the predictions made by DP-PINN are very accurate.

4.2 1D viscous Burger's Equation

The 1D viscous Burger's equations with Dirichlet boundary conditions are formalized as follows:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

Table 2. Comparing the performance of different PINN algorithms in solving the Allen-Cahn equation. When testing the PINN algorithms in literature, the settings are exactly same as those reported in the literature whenever applicable.

Methods	Relative L2-Error
standard PINN	99.18% ± 0.54%
Time-adaptive approach	7.55% ± 1.03%
SA-PINN	2.06% ± 1.33%
bc-PINN	1.67% ± 0.89%
DP-PINN (fixed sampling)	**0.84% ± 0.29%**

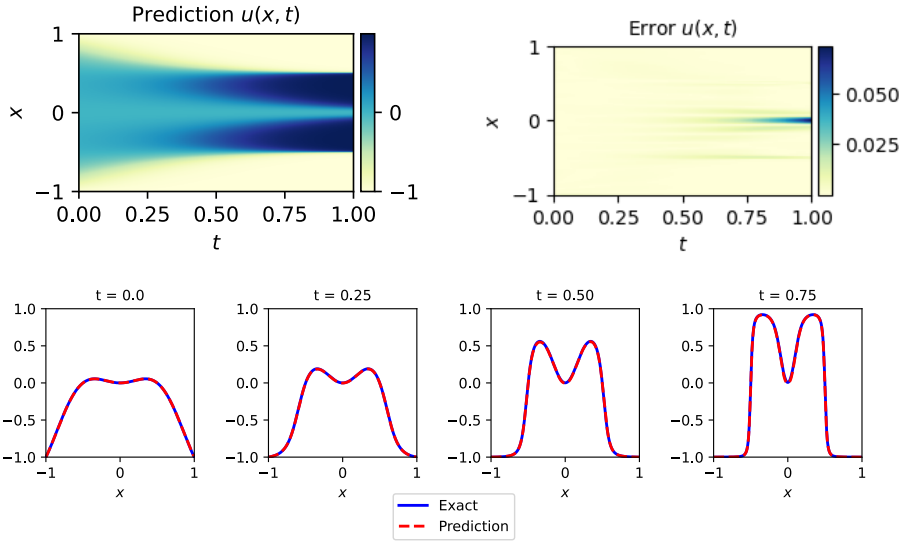

Fig. 2. Visualization of the solutions of Allen Cahn equation and their prediction errors obtained by DP-PINN. The two plots on the top are the predicted solution (left) and the prediction errors (right) across the domain. The four plots at the bottom are the predicted solution at 4 different time point.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], t \in [0, 1], \tag{24}$$
$$u(0, x) = -sin(\pi x), \tag{25}$$
$$u(t, -1) = u(t, 1) = 0. \tag{26}$$

In this set of experiments, the neural network architecture is fully connected with 8 hidden layers, each with 20 neurons. This architecture is identical to [10] and [8]. We set $|N_r| = 10000$, $|N_b| = 50$ and $|N_I| = 100$ for all PINNs, which is the common setting in [10] and [8]. All training is done by 10k Adam iterations, followed by 10k L-BFGS iterations, which is the same as those used in literature. For DP-PINN, $|N_{t_s}| = 256$. η is initialized in the same way as in the previous experiment. In this experiment, we only compared our DP-PINN with SA-PINN since other two state-of-the-art PINN algorithms (bc-PINN and Time Adaptive) are not tested on this simpler PDE.

The performance of different PINN methods are listed in table 3. From this table, we can see that DP-PINN achieves slightly better performance than SA-PINN. The improvement is not as prominent as with Allen Cahn equation because Burger's equation is easy to solve. The state-of-the-art PINN method (SA-PINN) can already generate very accurate solutions.

Table 3. Performance of different PINNs algorithms in solving Burger's equation

Methods	Relative L2 error
Standard-PINN	1.375e-3 ± 1.191e-3
SA-PINN	4.685e-4 ± 1.211e-4
DP-PINN	**4.595e-4 ± 1.381e-4**

Figure 3 visualizes the solutions of the Burger's equation and their prediction errors obtained by DP-PINN in Table 3. Once again, the solutions generated by DP-PINN are very accurate.

5 Conclusion and Future Works

In this paper, we develop a new training method for PINN, called DP-PINN, to address the limitations of the existing PINNs in solving "stiff" PDEs. By dividing the training into two phases at a carefully chosen time point t_s, DP-PINN significantly reduces prediction errors that accumulate over time in the existing methods like bc-PINN. The first phase of DP-PINN focuses on training the network and predicting accurate solutions at t_s, while the second phase trains the entire time domain, using the solutions at t_s as an intermediate condition. The experiments were conducted to compare DP-PINN with the state-of-the-art PINNs in solving Allen Cahn equation (stiff PDE) and Burger's equation (non-stiff PDE). The results show that DP-PINN achieve much higher performance

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

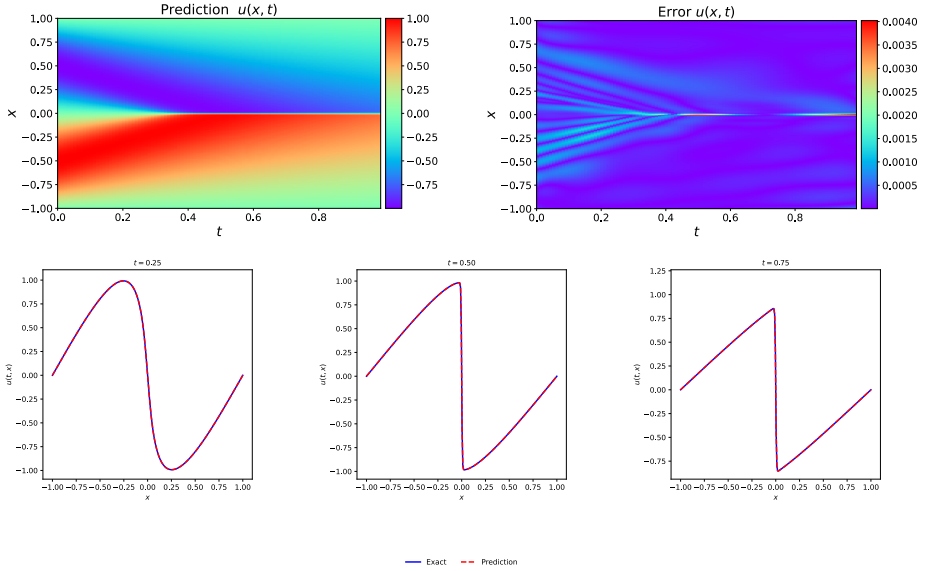

Fig. 3. Visualization of the solution of the 1D viscous Burger's equation and the prediction errors obtained by DP-PINN. The two plots on the top show the predicted solution (left) and prediction error (right) across the entire domain. The plots at the bottom show the predicted solutions at three different time points.

in terms of relative L2 error in solving Allen Cahn equation. DP-PINN only achieves slightly lower related L2 error in solving Burger's equation because it is easier to solve and the existing methods already performs very well. Despite the aforementioned achievements, some further work can be conducted. On the one hand, there have not been theoretical analyses about the impact of network size on the approximation accuracy of PDE solutions, which will be part of our future work. On the other hand, we would like to explore the techniques for solving PDEs from other application domains such as chemistry and biology.

References

1. Ames, W.F.: Numerical methods for partial differential equations. Academic press (2014)
2. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. Journal of Marchine Learning Research **18**, 1–43 (2018)
3. Burden, R.L.: Numerical analysis. Brooks/Cole Cengage Learning (2011)
4. Dissanayake, M., Phan-Thien, N.: Neural-network-based approximations for solving partial differential equations. communications in Numerical Methods in Engineering **10**(3), 195–201 (1994)
5. Huang, S., Feng, W., Tang, C., Lv, J.: Partial differential equations meet deep neural networks: A survey. arXiv preprint arXiv:2211.05567 (2022)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2

6. Lee, J.Y., Ko, S., Hong, Y.: Finite element operator network for solving parametric pdes. arXiv preprint arXiv:2308.04690 (2023)
7. Mattey, R., Ghosh, S.: A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. Computer Methods in Applied Mechanics and Engineering **390**, 114474 (2022)
8. McClenny, L., Braga-Neto, U.: Self-adaptive physics-informed neural networks using a soft attention mechanism. arXiv preprint arXiv:2009.04544 (2020)
9. Pang, Y., Xie, J., Khan, M.H., Anwer, R.M., Khan, F.S., Shao, L.: Mask-guided attention network for occluded pedestrian detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4967–4975 (2019)
10. Raissi, M., Perdikaris, P., Karniadakis, G.E.: 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)
11. Rumelhart, D.E., Durbin, R., Golden, R., Chauvin, Y.: Backpropagation: The basic theory. In: Backpropagation, pp. 1–34. Psychology Press (2013)
12. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3156–3164 (2017)
13. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing **43**(5), A3055–A3081 (2021)
14. Wight, C.L., Zhao, J.: Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. arXiv preprint arXiv:2007.04542 (2020)
15. Zawawi, M.H., Saleha, A., Salwa, A., Hassan, N., Zahari, N.M., Ramli, M.Z., Muda, Z.C.: A review: Fundamentals of computational fluid dynamics (cfd). In: AIP conference proceedings. vol. 2030. AIP Publishing (2018)

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63749-0_2