Check for updates

DOI: 10.1111/mice.12685

RESEARCH ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

Efficient training of physics-informed neural networks via importance sampling

Mohammad Amin Nabian[1] | **Rini Jasmine Gladstone**[2] | **Hadi Meidani**[2]

[1] NVIDIA, Santa Clara, California, USA

[2] Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA

Correspondence
Hadi Meidani, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA.
Email: meidani@illinois.edu

Abstract

Physics-informed neural networks (PINNs) are a class of deep neural networks that are trained, using automatic differentiation, to compute the response of systems governed by partial differential equations (PDEs). The training of PINNs is simulation free, and does not require any training data set to be obtained from numerical PDE solvers. Instead, it only requires the physical problem description, including the governing laws of physics, domain geometry, initial/boundary conditions, and the material properties. This training usually involves solving a nonconvex optimization problem using variants of the stochastic gradient descent method, with the gradient of the loss function approximated on a batch of collocation points, selected randomly in each iteration according to a uniform distribution. Despite the success of PINNs in accurately solving a wide variety of PDEs, the method still requires improvements in terms of computational efficiency. To this end, in this paper, we study the performance of an importance sampling approach for efficient training of PINNs. Using numerical examples together with theoretical evidences, we show that in each training iteration, sampling the collocation points according to a distribution proportional to the loss function will improve the convergence behavior of the PINNs training. Additionally, we show that providing a piecewise constant approximation to the loss function for faster importance sampling can further improve the training efficiency. This importance sampling approach is straightforward and easy to implement in the existing PINN codes, and also does not introduce any new hyperparameter to calibrate. The numerical examples include elasticity, diffusion, and plane stress problems, through which we numerically verify the accuracy and efficiency of the importance sampling approach compared to the predominant uniform sampling approach.

1 | INTRODUCTION

Physics-informed neural networks (PINNs) leverage recent advances in deep neural networks to calculate the response of systems governed by partial differential equations (PDEs). Specifically, PINNs are trained to satisfy the governing laws of physics described in the form of PDEs, as well as initial/boundary conditions and

© 2021 *Computer-Aided Civil and Infrastructure Engineering*

measurement data (Lagaris et al., 1998; Raissi, Perdikaris, et al., 2019). In this approach, the solution to a PDE is considered to be in the form of a deep neural network (with second or higher order differentiable nonlinearities) whose parameters are estimated by minimizing the squared residuals over specified collocation points using automatic differentiation (AD) (Baydin et al., 2018) and variants of the stochastic gradient descent (SGD) algorithm (Bottou, 2012).

The idea of using physics-informed training for neural network solutions of differential equations was first introduced in Dissanayake and Phan-Thien (1994), Lagaris et al. (1998), and Psichogios and Ungar (1992), where neural network solutions for initial/boundary value problems were developed. The method, however, did not gain much attention due to limitations in computational resources and optimization algorithms, until recently when researchers revisited this idea in (1) solving challenging dynamic problems described by time-dependent nonlinear PDEs (Raissi et al., 2017; Raissi, Perdikaris, et al., 2019), (2) solving variants of nonlinear PDEs (Berg & Nyström, 2018; Goswami et al., 2019; Guo et al., 2019 Jagtap & Karniadakis, 2019; Sirignano & Spiliopoulos, 2017; Weinan & Yu, 2018;), (3) data-driven discovery of PDEs (e.g., Long et al., 2017; Qin et al., 2018; Raissi, 2018a; Raissi, Perdikaris, et al., 2019), (4) uncertainty quantification (e.g., Meng & Karniadakis, 2019; Kissas et al., 2019; Nabian & Meidani, 2019; Raissi et al., 2018; Raissi, Wang, et al., 2019; Xu & Darve, 2019; Y. Yang & Perdikarisa, 2019; Zhu et al., 2019), (5) solving stochastic PDEs (e.g., Beck et al., 2018; Raissi, 2018b; Weinan et al., 2017; L. Yang et al., 2018), and (6) physics-driven regularization of neural network surrogates (e.g., Nabian & Meidani, 2020).

Training of PINNs usually involves solving a nonconvex optimization problem using an iterative method, with the gradient of loss function approximated on a batch of collocation points, selected randomly in each iteration according to a uniform distribution. Although this iterative update is shown to result in an unbiased estimation of the gradient with bounded variance (Bottou, 2010), such batch selection may seem to be naïve in terms of efficiency. In a given iteration, such batch selection may result in computing the gradient at a number of collocation points at which the approximate solution already satisfies the differential operator to a satisfactory extent relative to other points. As a result, little or no gradient information will be obtained, which can delay the convergence. Alternatively, by following an importance sampling (Press et al., 2007) scheme, in each iteration we can select a batch of collocation points that can offer more gradient information for accelerated convergence.

The performance of implementing an importance sampling–based training has recently been evaluated on classification tasks using convolutional neural networks and recurrent neural networks (Alain et al., 2015; Katharopoulos & Fleuret, 2018, 2017), where the authors provided theoretical and numerical evidences showing that the training convergence speed can be maximized if, at each training iteration, samples from the training images or texts are drawn according to a proposal distribution that is proportional to the two-norm of loss gradient with respect to model parameters. Further, it has been illustrated that computing such proposal distributions can be computationally expensive, and the authors used an approximate proposal distribution proportional to the loss function itself to improve the computational efficiency. Finally, the performance of such importance sampling approach is evaluated for image classification and language modeling tasks.

Our contribution in this paper is twofold. First, we borrow the theoretical findings in Katharopoulos and Fleuret (2018) and Katharopoulos and Fleuret (2017) to propose an efficient approach for accelerated training of PINNs based on importance sampling. To the authors' knowledge, this is the first time that an importance sampling scheme is used for training of PINNs. This can be an important step toward improving the computational efficiency of PINNs compared to their traditional numerical counterparts, that is, finite difference, finite element, and finite volume methods. Second, we show how a piecewise constant (PWC) approximation to the loss function, using nearest neighbor search (Marsland, 2014) or Voronoi tessellation (Aurenhammer, 1991), can be used to approximate the proposal distribution to further improve the convergence behavior of PINNs training. The proposed importance sampling approach is straightforward and can be easily applied to the existing PINN codes by modifying only a few lines of the code. Furthermore, no new hyperparameters are introduced in the proposed approach.

The remainder of this paper is organized as follows. A theoretical background on PINNs is presented in Section 2. Our proposed importance sampling approach for training of PINNs is then introduced in Section 3. Section 4 includes three numerical examples, on which the performance of the proposed importance sampling approach is evaluated. Finally, Section 5 concludes the paper.

2 | DEEP LEARNING OF DIFFERENTIAL EQUATIONS

2.1 | Feed-forward fully connected deep neural networks

A basic architecture for deep neural networks is the feed-forward fully connected deep neural network, which will

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

be explained here (a more detailed introduction can be found in LeCun et al., 2015 and Goodfellow et al., 2016). Given the d-dimensional row vector $\boldsymbol{x} \in \mathbb{R}^d$ as model input, the k-dimensional output of a standard single hidden layer neural network is in the form of

$$\boldsymbol{y} = \sigma(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{1}$$

in which $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are weight matrices of size $d \times q$ and $q \times k$, and $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are bias vectors of size $1 \times q$ and $1 \times k$, respectively. The function $\sigma(\cdot)$ is an element-wise nonlinear model, known as the activation function. In deep neural networks, for each additional hidden layer, a new set of weight matrix and biases is added to Equation (1). Popular choices of activation functions include Sigmoid, hyperbolic tangent (Tanh), rectified linear unit (ReLU), and sine functions.

The model parameters are estimated according to

$$(\boldsymbol{W}_1^*, \boldsymbol{W}_2^*, \dots, \boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \dots) = \underset{(\boldsymbol{W}_1, \dots, \boldsymbol{b}_1 \cdots)}{\operatorname{argmin}} J(\theta; \boldsymbol{X}, \boldsymbol{Y}) \tag{2}$$

where $\theta = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \dots, \boldsymbol{b}_1, \boldsymbol{b}_2, \dots\}$ is the set of model parameters (i.e., weights and biases). This optimization is performed iteratively using SGD and its variants (Bottou, 2012; Duchi et al., 2011; Kingma & Ba, 2014; Sutskever et al., 2013; Zeiler, 2012). Specifically, at the ith iteration, the model parameters are updated according to

$$\theta^{(i+1)} = \theta^{(i)} - \eta^{(i)} \nabla_\theta J(\theta^{(i)}; \boldsymbol{X}, \boldsymbol{Y}) \tag{3}$$

where $\eta^{(i)}$ is the step size in the ith iteration. The gradient of loss function with respect to model parameters $\nabla_\theta J$ is usually computed using *backpropagation* (LeCun et al., 2015), which is a special case of the more general technique called reverse-mode AD (Baydin et al., 2018). In simplest terms, in backpropagation, the required gradient information is obtained by the backward propagation of the sensitivity of objective value at the output, utilizing the chain rule successively to compute partial derivatives of the objective with respect to each weight (Baydin et al., 2018). In other words, the gradient of the last layer is calculated first and the gradient of the first layer is calculated last. Partial gradient computations for one layer are reused in the gradient computations for the foregoing layers. This backward flow of information facilitates efficient computation of the gradient at each layer of the deep neural network (LeCun et al., 2015). It is important to note that automatic (or reverse automatic) differentiation is different from symbolic or numerical differentiation, which are two common alternatives for computing derivatives (Baydin et al., 2018). Detailed discussions about the backpropagation algorithm can be found in Goodfellow et al. (2016), LeCun et al. (2015), and Baydin et al. (2018).

Deep neural networks have been successfully applied in a variety of civil engineering applications. Examples include, but are not limited to, structural damage detection and image recognition (e.g., Azimi & Pekcan, 2019; Cha et al., 2017; Koziarski & Cyganek, 2017; Liang, 2019; Lin et al., 2017; Rafiei & Adeli, 2018, 2017; Zeinali & Story, 2017; X. Zhang et al., 2019), estimation of concrete compressive strength (e.g., Rafiei et al., 2017), traffic forecasting and management (e.g., Hashemi & Abdelghany, 2018; Liu et al., 2018; Y. Zhang et al., 2019), reliability analysis of transportation networks (e.g., Nabian & Meidani, 2018), and uncertainty quantification (e.g., Luo & Kareem, 2019).

2.2 | PINNs

PINNs are a class of deep neural networks that are used to calculate the approximate solution $u(t, \boldsymbol{x}; \theta)$ for the following generic differential equation:

$$\begin{aligned} \mathcal{N}(t, \boldsymbol{x}; u(t, \boldsymbol{x}; \theta)) &= 0, \quad t \in [0, T], \boldsymbol{x} \in \mathcal{D}, \\ \mathcal{I}(\boldsymbol{x}; u(0, \boldsymbol{x}; \theta)) &= 0, \quad \boldsymbol{x} \in \mathcal{D}, \\ \mathcal{B}(t, \boldsymbol{x}; u(t, \boldsymbol{x}; \theta)) &= 0, \quad t \in [0, T], \boldsymbol{x} \in \partial\mathcal{D} \end{aligned} \tag{4}$$

where θ include the parameters of the function form of the solution; $\mathcal{N}(\cdot)$ is a general differential operator that may consist of time derivatives, spatial derivatives, and linear and nonlinear terms; and $\boldsymbol{x}$ is a position vector defined on a bounded continuous spatial domain $\mathcal{D} \subseteq \mathbb{R}^D, D \in \{1, 2, 3\}$ with boundary $\partial\mathcal{D}$. Also, $\mathcal{I}(\cdot)$ and $\mathcal{B}(\cdot)$ denote, respectively, the initial and boundary conditions and may consist of differential, linear, or nonlinear operators.

In order to calculate the solution, that is, calculate the parameters θ, let us consider the following nonnegative residuals, defined over the entire spatial and temporal domains

$$\begin{aligned} r_{\mathcal{N}}(\theta) &= \int_{[0,T]\times D} (\mathcal{N}(t, \boldsymbol{x}; \theta))^2 dt\, d\boldsymbol{x}, \\ r_{\mathcal{I}}(\theta) &= \int_{D} (\mathcal{I}(\boldsymbol{x}, \boldsymbol{p}; \theta))^2 d\boldsymbol{x}, \\ r_{\mathcal{B}}(\theta) &= \int_{[0,T]\times \partial D} (\mathcal{B}(t, \boldsymbol{x}; \theta))^2 dt\, d\boldsymbol{x} \end{aligned} \tag{5}$$

The optimal parameters θ^* can then be calculated according to

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmin}}\, r_{\mathcal{N}}(\theta), \\ \text{s.t.} \quad & r_{\mathcal{I}}(\theta) = 0,\ r_{\mathcal{B}}(\theta) = 0 \end{aligned} \tag{6}$$

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Therefore, the solution to the differential equation defined in Equation (4) is reduced to an optimization problem, where initial and boundary conditions can be viewed as constraints. This constrained optimization can be reformulated as an unconstrained optimization with a modified loss function that also accommodates the constraints. The predominant approach to do so is the soft assignment of constraints, where the constraints are translated into additive penalty terms in the loss function (see, e.g., Sirignano & Spiliopoulos, 2017). Other approaches also exist—for example, hard assignment of constraints (Lagaris et al., 1998) and the unified approach (Berg & Nyström, 2018)—but are not discussed here for brevity.

Let us denote the solution obtained by a PINN by $\tilde{u}(t, \boldsymbol{x}; \theta)$. The inputs to this deep neural network are realizations from t and $\boldsymbol{x}$. With soft assignment of constraints, we solve the following unconstrained optimization problem:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \underbrace{r_{\mathcal{N}}(\theta) + \lambda_1 r_{\mathcal{I}}(\theta) + \lambda_2 r_{\mathcal{B}}(\theta)}_{J(\theta)} \tag{7}$$

in which λ_1 and λ_2 are weight parameters, analogous to collocation finite element method in which weights are used to adjust the relative importance of each residual term (Bochev & Gunzburger, 2006).

To solve this unconstrained optimization problem, mini-batch SGD optimization algorithms (Ruder, 2016) are used. In each iteration of a mini-batch SGD algorithm, the gradient of loss function is approximated through backpropagation using a batch of points of size m in the input space, based on which the neural network parameters are updated. This iterative update is shown to result in an unbiased estimation of the gradient, with bounded variance (Bottou, 2010). Specifically, in the ith iteration, we select a subset of collocation points uniformly drawn in $[0, T], \mathcal{D}, \partial\mathcal{D}$. and compute the loss function as

$$J(\theta) \approx \frac{1}{m} \sum_{j \in M^{(i)}} J(\theta; \boldsymbol{x}_j) = \frac{1}{m} \sum_{j \in M^{(i)}} \Big[\big[\mathcal{N}\left(t_j, \boldsymbol{x}_j; \tilde{u}(t_j, \boldsymbol{x}_j; \theta)\right)\big]^2 + \lambda_1 \big[\mathcal{I}\left(\boldsymbol{x}_j; \tilde{u}(0, \boldsymbol{x}_j; \theta)\right)\big]^2 + \lambda_2 \Big[\mathcal{B}\Big(t_j, \underline{\boldsymbol{x}}_j; \tilde{u}(t_j, \underline{\boldsymbol{x}}_j; \theta)\Big)\Big]^2 \Big] \tag{8}$$

where $M^{(i)}$ is the set of indices of selected collocation points at iteration i with $|M^{(i)}| = m$, $J(\theta; \boldsymbol{x}_j)$ is the per-sample loss evaluated at the jth collocation point, and $\{\underline{\boldsymbol{x}}_j\}$ denotes the boundary collocation points. The model parameters are updated according to

$$\theta^{(i+1)} = \theta^{(i)} - \eta^{(i)} \nabla_\theta J(\theta^{(i)}) \tag{9}$$

ALGORITHM 1 Training of the PINNs

1:	Generate N collocation points $\{t_j, \boldsymbol{x}_j\}_{j=1}^N$ sampled from $[0, T] \times \mathcal{D}$, and N boundary points $\{\underline{\boldsymbol{x}}_j\}_{j=1}^N$ sampled from $\partial\mathcal{D}$.
2:	Set the model architecture (number of layers, dimensionality of each layer, and nonlinearities). Also specify optimizer hyper-parameters, λ_1, λ_2, batch size m, and error tolerance ϵ.
3:	Initialize model parameters $\theta^{(0)}$.
4:	**while** $J(\theta^{(i)}) > \epsilon$ **do**
5:	Randomly select a batch of m points out of the N collocation points according to a uniform distribution.
6:	Take a descent step $\theta^{(i+1)} = \theta^{(i)} - \eta^{(i)} \nabla_\theta J(\theta^{(i)})$.
7:	**end while**

where $\nabla_\theta J$ is calculated using backpropagation (Baydin et al., 2018). Algorithm 1 summarizes the steps for training of a PINN.

Collocation points may be generated according to a uniform distribution, or alternatively according to a low-discrepancy sequence generator algorithm. Low-discrepancy sequences provide a means to generate quasi-random numbers with a high level of uniformity. Among the popular low-discrepancy sequences are the generalized Halton sequences (Faure & Lemieux, 2009; Halton, 1960;), the Sobol sequences (Joe & Kuo, 2008; Sobol', 1967), and the Hammersley sets (Hammersley, 1960, 2013).

3 | IMPORTANCE SAMPLING FOR TRAINING OF PINNs

Consider the following parameter estimation problem:

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_f[J(\theta)] \\ &= \underset{\theta}{\operatorname{argmin}}\, \frac{1}{N} \sum_{j=1}^{N} J(\theta; \boldsymbol{x}_j), \quad \boldsymbol{x}_j \sim f(\boldsymbol{x}) \end{aligned} \tag{10}$$

where $f(\boldsymbol{x})$ is the sampling distribution for the training points in the physical domain $\boldsymbol{x} \in \mathcal{D}$. The typical choice for this sampling distribution is the uniform distribution defined over the physical domain, that is, $\mathcal{U}(\mathcal{D})$. In an importance sampling approach, we seek to draw training samples from an alternative sampling distribution, denoted by $q(\boldsymbol{x})$, and instead estimate the neural network

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

parameters according to

$$\theta^* \approx \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{j=1}^{N} \frac{f(\boldsymbol{x}_j)}{q(\boldsymbol{x}_j)} J(\theta; \boldsymbol{x}_j), \quad \boldsymbol{x}_j \sim q(\boldsymbol{x}) \tag{11}$$

In this work, we effectively implement a discrete sampling scheme, by turning the continuous domain $\mathcal{D}$ into a discrete set of N sample locations uniformly selected in $\mathcal{D}$, with $N >> 1$. Thus, instead of the sampling density functions $f(\boldsymbol{x}_j)$ and $q(\boldsymbol{x}_j)$, we will work with discrete distributions $\{f_j\}_{j=1}^N$ and $\{q_j\}_{j=1}^N$, respectively, with $f_j = \frac{1}{N}$ at any candidate point j.

In order to build the corresponding SGD, for the sake of brevity, let us first consider no mini batch, that is, $m = 1$, that is,

$$\theta^{(i+1)} = \theta^{(i)} - \eta^{(i)} \underbrace{\nabla_\theta \left(\frac{f_i}{q_i} J(\theta^{(i)}; \boldsymbol{x}_i) \right)}_{\boldsymbol{G}^{(i)}} \tag{12}$$

Our objective in this work is to design a training scheme with a sampling distribution q that can accelerate the convergence of Equation (12). Authors in Katharopoulos and Fleuret (2018) considered the following definition for convergence speed:

$$S^{(i)} = -\mathbb{E}_f \left[||\theta^{(i+1)} - \theta^*||_2^2 - ||\theta^{(i)} - \theta^*||_2^2 \right] \tag{13}$$

and showed that

$$\begin{aligned} S^{(i)} = {} & 2\eta \left(\theta^{(i)} - \theta^*\right) \mathbb{E}_f \left[\boldsymbol{G}^{(i)}\right] - \eta^2 \mathbb{E}_f \left[\boldsymbol{G}^{(i)}\right]^T \mathbb{E}_f \left[\boldsymbol{G}^{(i)}\right] \\ & - \eta^2 \mathrm{Tr}\left(\mathbb{V}_f \left[\boldsymbol{G}^{(i)}\right]\right) \end{aligned} \tag{14}$$

It was then concluded that the convergence can be accelerated by sampling the input variables from a distribution that minimizes $\mathrm{Tr}(\mathbb{V}_P[\boldsymbol{G}^{(i)}])$ (Katharopoulos & Fleuret, 2018).

It was shown in Katharopoulos and Fleuret (2018) and Alain et al. (2015) that this term is minimized if training samples are selected according to $q^* \propto \|\nabla_\theta J(\theta^{(i)})\|_2$. In the case of mini-batch SGD with batches of size m, this was effectively done by calculating the sampling distributions according to

$$q_j^{(i)} = \frac{\left\| \nabla_\theta J(\theta^{(i)}; \boldsymbol{x}_j) \right\|_2}{\sum_{j=1}^{N} \left\| \nabla_\theta J(\theta^{(i)}; \boldsymbol{x}_j) \right\|_2}, \quad \forall j \in \{1, \dots, N\} \tag{15}$$

and selecting the mini-batch sample set $M^{(i)}$, with $|M^{(i)}| = m$, by sampling m indices from a multinomial with probabilities $\boldsymbol{p}^{(i)} = \{q_1^{(i)}, \dots, q_N^{(i)}\}$. In order to get an unbiased estimate of the gradient $\nabla_\theta J(\theta)$, following Equations (8) and (11), the mini-batch gradient descent update rule will then be given by (Alain et al., 2015; Katharopoulos & Fleuret, 2018)

$$\theta^{(i+1)} = \theta^{(i)} - \frac{\eta^{(i)}}{m} \sum_{j \in M} \frac{1}{N q_j^{(i)}} \nabla_\theta J(\theta^{(i)}; \boldsymbol{x}_j) \tag{16}$$

This derivation provides a theoretical evidence that training of PINNs can be accelerated using an importance sampling approach where training samples are obtained from a distribution proportional to the 2-norm of the gradient of loss function with respect to model parameters. However, computing the 2-norm of this gradient for all of the collocation point at each iteration requires extra backpropagations through the computational graph, which can be computationally expensive. To alleviate this issue, it was shown theoretically and numerically in Katharopoulos and Fleuret (2017) that a linear transformation of the loss value at a training example is always greater than the 2-norm of loss gradient at that example, and that the ordering of collocation points according to their gradient norm is consistent with their ordering according to their loss value. Thus, one can use the loss value, instead of the gradient value, as the importance metric, according to

$$q_j^{(i)} \approx \frac{J(\theta^{(i)}; \boldsymbol{x}_j)}{\sum_{j=1}^{N} J(\theta^{(i)}; \boldsymbol{x}_j)}, \quad \forall j \in \{1, \dots, N\} \tag{17}$$

Specifically, using this proposal distribution, one can select m mini-batch samples as explained earlier together with the gradient descent rule of Equation (16) to update the model parameters.

Although evaluation of the loss function is computationally less expensive compared to that of the gradient, such evaluation for the entire set of collocation points in each iteration can still be very expensive. To alleviate this, we propose a PWC approximation to the loss function. That is, instead of evaluating the loss function at every collocation point, we evaluate the loss only at a subset of points, hereinafter referred to as "seeds," denoted by $\{\boldsymbol{x}_s\}_{s=1}^S$, with $S < N$. Next, using a nearest neighbor search algorithm (Marsland, 2014), for each collocation point j, we identified the nearest seed $s = \rho(j)$, and set the loss value at that collocation point equal to the loss at the nearest seed, that is, $J(\cdot; \boldsymbol{x}_j) := J(\cdot; \boldsymbol{x}_{\rho(j)})$. This is equivalent to generating a Voronoi tesselation (Aurenhammer, 1991) using the seeds, and using a constant approximation for loss within each Voronoi cell, as shown in Figure 1. It will be shown in the numerical examples that such PWC approximation provides improved computational efficiency compared to

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

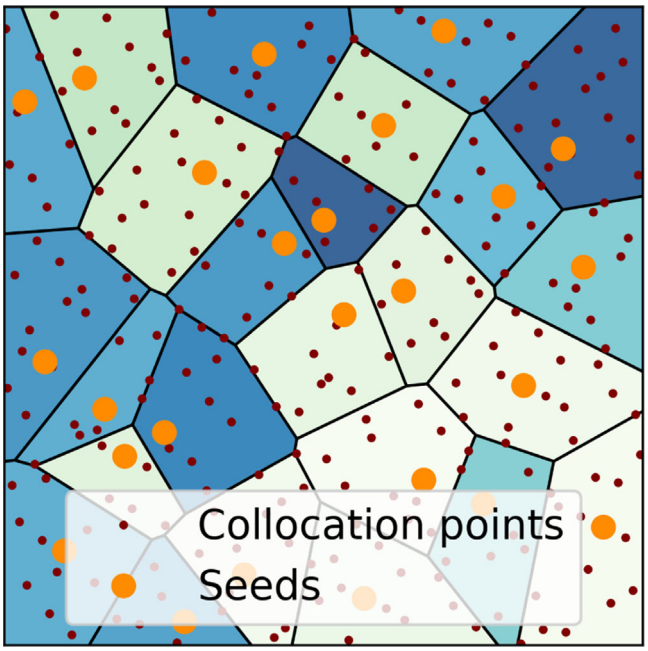

FIGURE 1 Piecewise constant approximation of loss on a sample 2D domain. Within each cell, the loss is evaluated only at the course-level point and is assigned to the neighboring collocation points

the case where the loss function is evaluated for the entire collocation points.

Algorithm 2 summarizes the steps for the proposed importance sampling method for efficient training of a PINN.

4 | NUMERICAL EXAMPLES

In this section, we numerically demonstrate the performance of the proposed importance sampling approach, with PWC loss approximation, in solving sample PDEs. In the first example, the proposed approach is applied to solve an elasticity problem on an irregular plate. Next, a plane stress problem is solved on a 2D plate with three holes using the proposed approach. In the final example, a steady diffusion example is considered. Training is performed using TensorFlow (Abadi et al., 2016) on a NVIDIA Tesla P100-PCIE-16GB GPU. The Adam optimization algorithm (Kingma & Ba, 2014) is used to find the optimal neural network parameters, with β_1 (the exponential decay rate for the first moment estimates) and β_2 (the exponential decay rate for the second moment estimates) set to 0.9 and 0.999, respectively.

ALGORITHM 2 Efficient training of PINNs via importance sampling

1:	Generate N collocation points $\{t_j, \boldsymbol{x}_j\}_{j=1}^{N}$ sampled from $[0,T]\times\mathcal{D}$, n boundary points $\{\underline{\boldsymbol{x}}_j\}_{j=1}^{N}$ sampled from $\partial\mathcal{D}$, and S seeds $\{t_s, \boldsymbol{x}_s\}_{s=1}^{S}$ sampled from $[0,T]\times\mathcal{D}$.
2:	For each collocation point, find the nearest seed.
3:	Set the model architecture (number of layers, dimensionality of each layer, and nonlinearities). Also specify optimizer hyper-parameters, λ_1, λ_2, batch size m, and error tolerance ϵ.
4:	Initialize model parameters $\theta^{(0)}$.
5:	**while** $J(\theta) > \epsilon$ **do**
6:	Compute the loss value at each seed $\{J(\theta^{(i)}; \boldsymbol{x}_s)\}_{s=1}^{S}$.
7:	Compute $\tilde{q}_j^{(i)} = J(\theta^{(i)}; \boldsymbol{x}_{\rho(j)}) / \sum_{j=1}^{N} J(\theta^{(i)}; \boldsymbol{x}_{\rho(j)})$ for $\forall j \in \{1, \dots, N\}$.
8:	Select a batch of collocation points according to a multinomial with $\boldsymbol{p}^{(i)} = \{\tilde{q}_1^{(i)}, \dots, \tilde{q}_N^{(i)}\}$.
9:	Take a step $\theta^{(i+1)} = \theta^{(i)} - \frac{\eta^{(i)}}{mN} \sum_{j\in M^{(i)}} \frac{1}{\tilde{q}_j^{(i)}} \nabla_\theta J(\theta^{(i)}; \boldsymbol{x}_j)$.
10:	**end while**

4.1 | A 2D isotropic elasticity problem

In the first example, we consider the governing equations of the displacement of a 2D isotropic elastic structure (Chikazawa et al., 2001) as follows:

$$\mathcal{N}_1(x,y,u,v) = (\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f_x,$$
$$\mathcal{N}_2(x,y,u,v) = (\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+f_y \tag{18}$$

where u and v denote the displacement along the x- and y-axes, and f_x and f_y the external force terms along the x- and y-axes, respectively. The constants λ and μ are defined as

$$\lambda = \frac{\nu E}{(1+\nu)(1-\nu)}, \tag{19}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{20}$$

where ν and E are the Poisson's ratio and the Young's modulus of the structure.

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

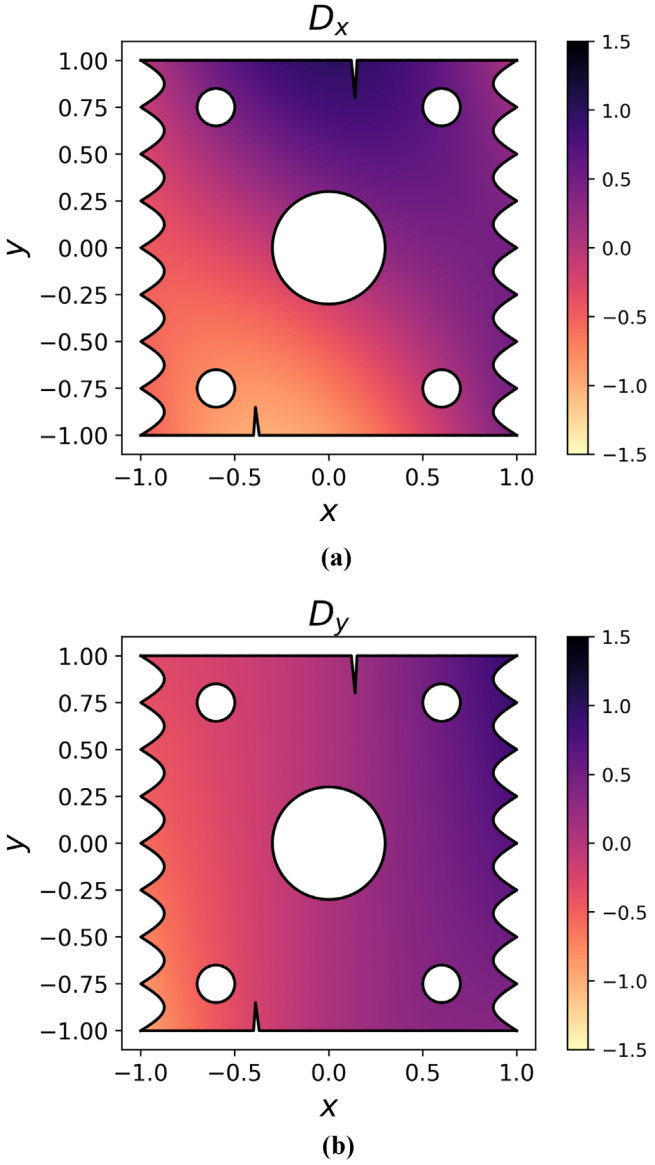

FIGURE 2 Plate displacement: (a) x-axis displacement (D_x); (b) y-axis displacement (D_y)

As a representative test case, we synthesize a governing equation that would generate a given solution. In particular, we prescribe the plate displacement to take the following analytic form:

$$\begin{aligned} u(x,y) = {} & 0.8\sin\left(\frac{\pi}{2}(x+0.78)\right)\cos(y-1) \\ & - 0.8\sin\left(\frac{\pi}{2}(x+1.50)\right)\cos(y+1), \\ v(x,y) = {} & 0.72 - 0.65\left[\exp\left(\frac{-x^2 y}{2}\right) + x\right] \end{aligned} \tag{21}$$

We further consider an irregular plate geometry that is shown in Figure 2 upon which the two displacement fields of Equations (21) are defined. With E and ν set to 0.25 and 0.2, respectively, we can then analytically back-calculate the terms f_x and f_y in Equation (18) that would correspond to these prescribed geometry and displacement fields. Also, we establish a "numerical" boundary condition $\mathcal{B}$, where for any given collocation point on the boundary, we set the displacement to be equal to the prescribed displacement fields $u_{\mathcal{B}}$ at those points.

To solve the PDE in Equation (18) using a PINN with the proposed PWC importance sampling approach, a neural network is constructed as the trial solution, with four hidden layers, each with 32 neurons. The input layer consists of two neurons, which take realizations from the physical domain (i.e., x, y). The output layer also consists of two neurons, which represent the horizontal and vertical plate displacement for the given realizations in the input layer. A sine function is adopted for the activations in each hidden layer. The parameters of the trial solution are randomly initialized, and are trained following Algorithm 2, with the following loss function:

$$\begin{aligned} & J = J_1 + \lambda_2 J_2, \\ & J_1 = \left[(\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y}\right) + \mu\left(\frac{\partial^2 \tilde{u}}{\partial x^2} + \frac{\partial^2 \tilde{u}}{\partial y^2}\right) + f_x \right. \\ & \qquad \left. + (\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial \tilde{u}}{\partial x} + \frac{\partial \tilde{v}}{\partial y}\right) + \mu\left(\frac{\partial^2 \tilde{v}}{\partial x^2} + \frac{\partial^2 \tilde{v}}{\partial y^2}\right) + f_y \right]^2, \\ & (x,y) \in \mathcal{D}, J_2 = (\tilde{u} - u_{\mathcal{B}})^2, \ (x,y) \in \partial\mathcal{D} \end{aligned} \tag{22}$$

The gradient of this loss function with respect to model parameters is computed through backpropagation (Baydin et al., 2018), as explained in Section 2.1. Parameter λ_2 set to 1 (note that parameter λ_1 is set to zero as the problem is time independent). Batch size is set to 10,000, and the learning rate α is set to 0.002. A generalized Halton sequence generator algorithm is used to generate 100,000 collocation points and 10,000 seeds within the computation domain. Another 100,000 uniformly distributed boundary collocation points are also generated. The model is trained for 500 iterations.

Figure 3 presents a comparison between the PINN solution trained using the proposed approach and the exact solution in Equation (21). It is evident that there is a close agreement between the results, verifying the accuracy of the proposed importance sampling approach. A visualization of the progressive change in the loss value as well as in the sampled points when the model is trained using the proposed approach is also presented in Figure 4. Additionally, Figure 5 evaluates how well the gradient norm $\|\nabla_\theta J(\boldsymbol{\theta}^{(i)}; \boldsymbol{x})\|_2$ used in Equation (15) is approximated by $J(\boldsymbol{\theta}^{(i)}; \boldsymbol{x})$ used in Equation (17). This comparison is shown

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

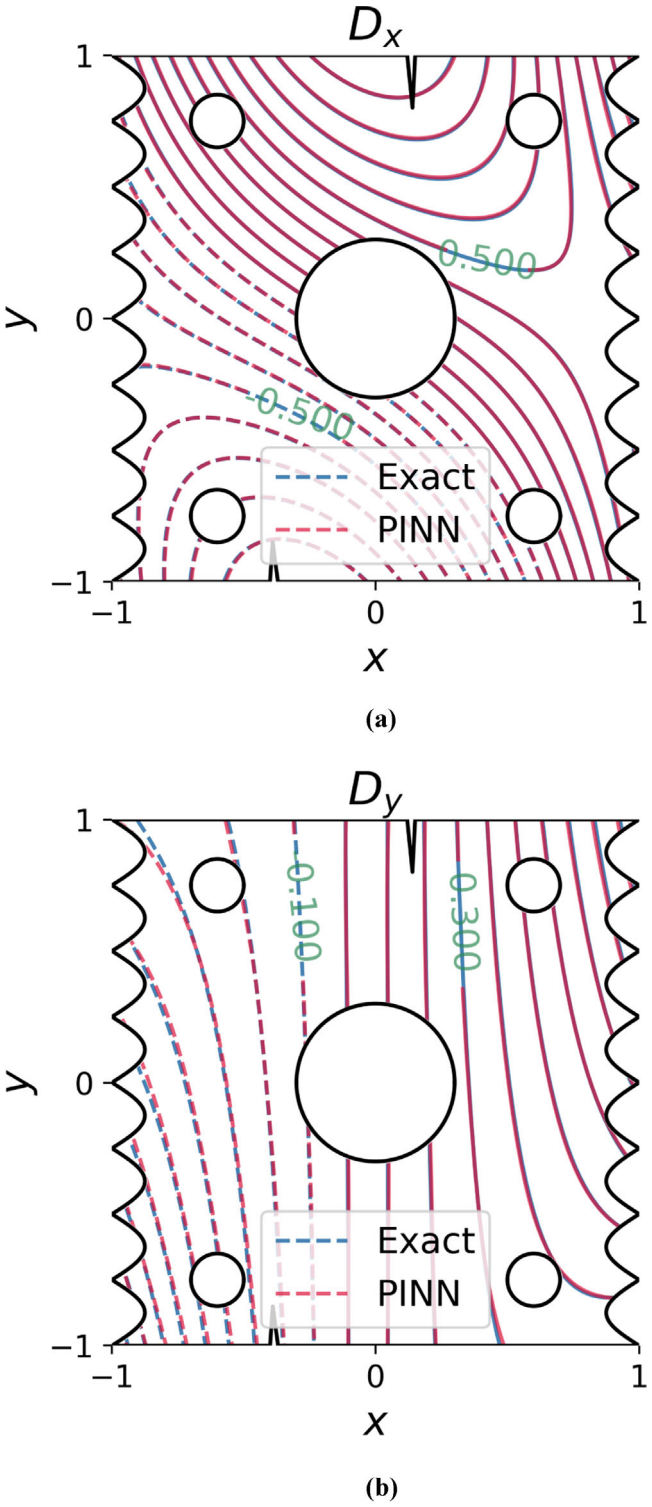

FIGURE 3 A comparison between the exact and the PINN solutions to Equation (18). The PINN solution is trained using the proposed importance sampling approach (Algorithm 2) with piecewise constant approximation to loss. D_x and D_y are, respectively, the displacement in x and y directions. The positive and negative values are shown by solid and dashed lines, respectively

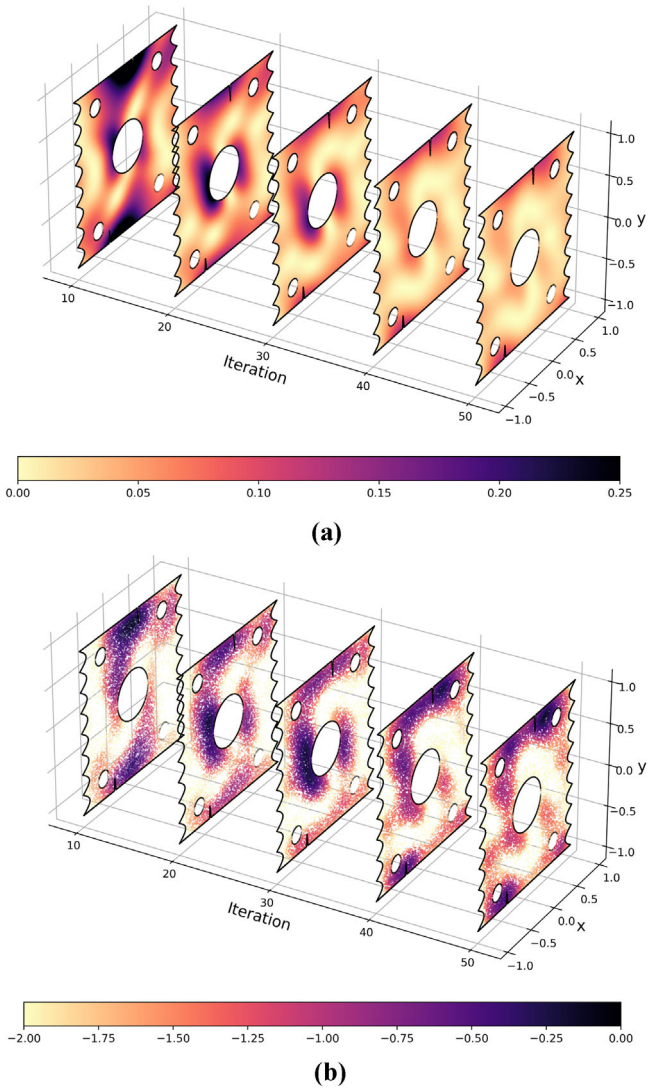

FIGURE 4 A visualization of the progressive change in the (a) loss value and (b) the sampled points, when the PINN solution to Equation (18) is trained using the proposed importance sampling approach. The color map in the two figures show, respectively, the loss value and the log probabilities showing the concentration of sampled points

for the loss fields at three different training steps of the PWC importance sampling process.

To compare the accuracy and computational efficiency of the proposed PWC importance sampling approach with the uniform sampling approach, Figure 6 shows the loss value at different iterations (left) and different times (right) for each approach, and justifies the effectiveness of the PWC importance sampling method in accelerating the convergence of PINNs training. In comparing numerical performances, we also considered a third importance sampling approach, which uses the exact loss values without any PWC approximation. It is evident from Figure 6(a) that the PWC approximation to loss is in fact a good

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

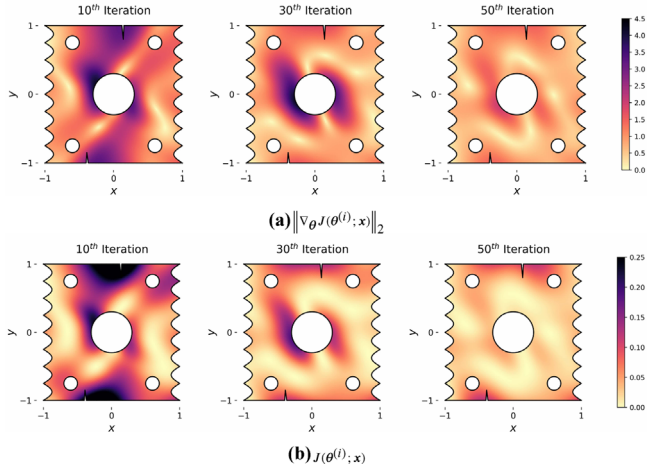

FIGURE 5 Comparison between the 2-norm of the loss gradient w.r.t. model parameters (top) and the loss values (bottom) at three snapshots taken at iterations $i = 10$, $i = 30$, and $i = 50$ of the proposed PWC importance sampling training

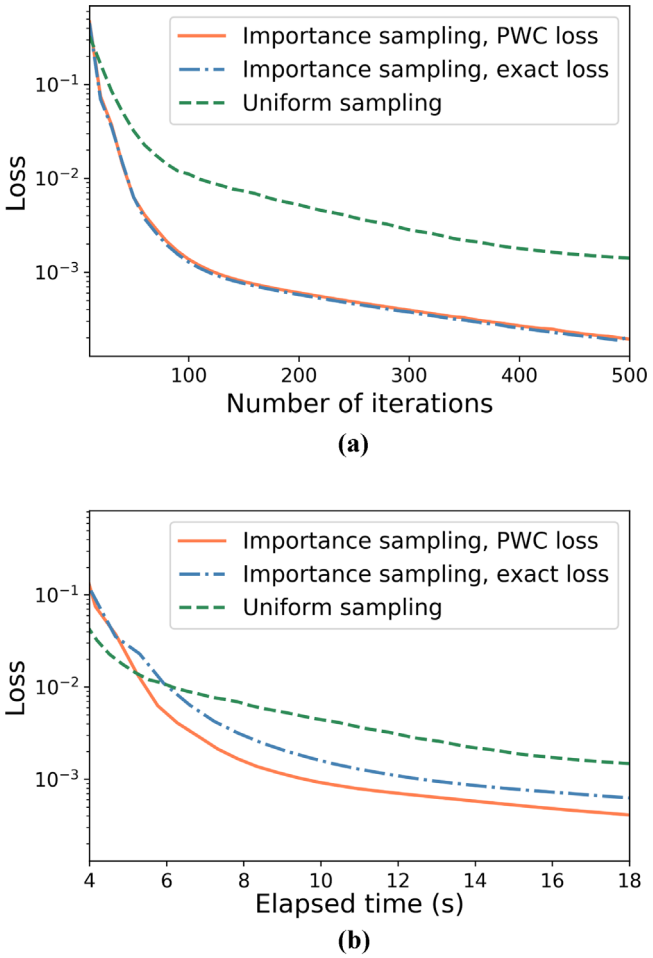

FIGURE 6 A comparison between the performance of the three approaches of uniform sampling, importance sampling with exact loss evaluation, and importance sampling with approximate loss evaluation for training of a PINN solution to the elasticity problem defined in Equation (18)

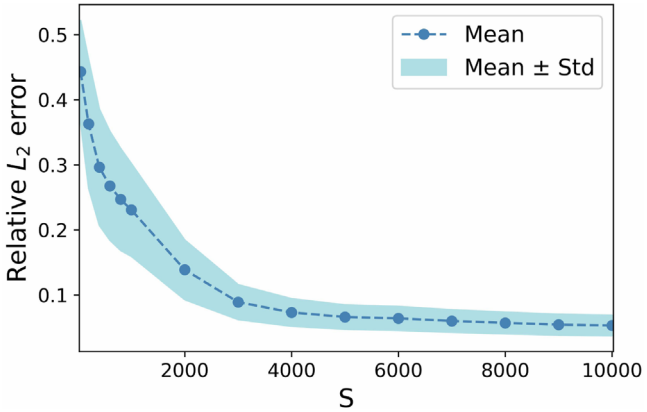

FIGURE 7 Mean and standard deviation, over the training iterations, of the relative L_2 error in piecewise constant approximation of loss values versus the number of seeds

approximation and does not negatively affect the convergence behavior. Figure 6(b) shows that PWC importance sampling approach also outperforms the "exact loss" importance sampling method in terms of computational efficiency. This advantage is apparent in comparing computational times because the PWC importance sampling approach involves significantly less forward model evaluations, compared to the "exact loss" importance sampling method.

Figure 7 shows how the error in piecewise approximation of the loss function varies with respect to the number of seeds. In particular, for each seed size, the loss function approximation error is evaluated at and averaged over the 100,000 collocation points and the variation in this averaged error over the 500 training epochs is depicted by the shaded area in this figure. Moreover, in order to demonstrate the effect of seed size selection on the convergence behavior of model training, Figure 8 shows loss values versus the number of iterations (left) and the elapsed time (right) for different numbers of seeds. It is evident that the number of seeds, if selected reasonably (e.g., $S \geq 500$ in this case), does not significantly affect the convergence behavior and therefore there is no need to consider that number as a new hyperparameter. This is because a PWC approximation with relatively large number of seeds (at least 500 in this case) can potentially provide a good approximation to the loss function, and increasing the number of steps may result in little or no change to the approximation accuracy. One reasonable suggestion is to simply set the seed size equal to the batch size. As a representative case for small seed sizes, we have also considered $S = 50$, and included the corresponding performance in Figure 8. For this seed size, Figure 9 shows the selection probabilities of collocation points, and also the 10,000 sampled collocation points, at the 30th training iteration of the proposed PWC importance sampling approach.

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

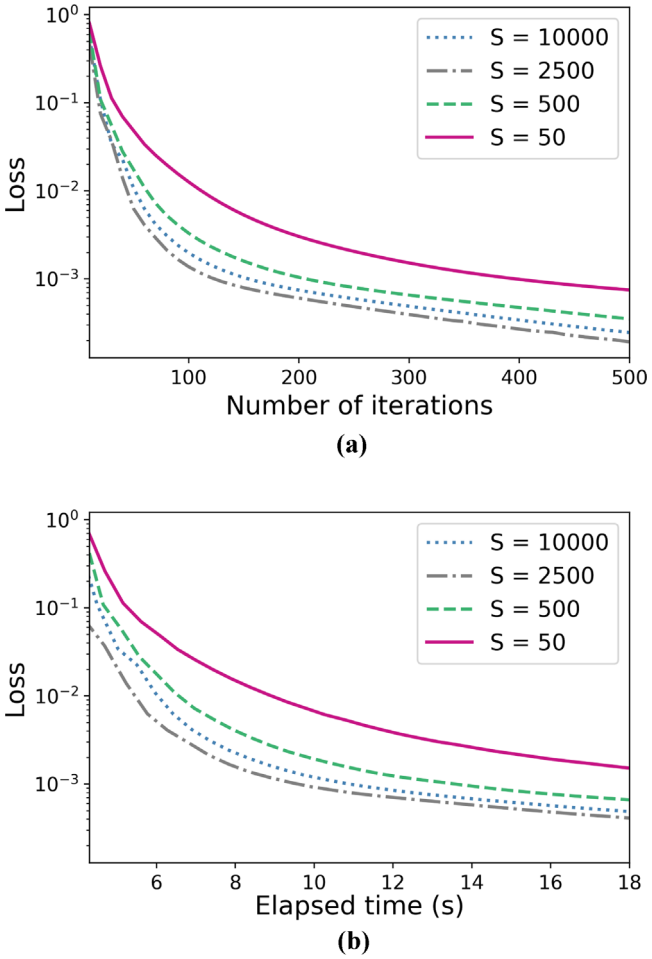

FIGURE 8 A demonstration of the performance of the proposed PWC importance sampling approach with different choices for seed size, S

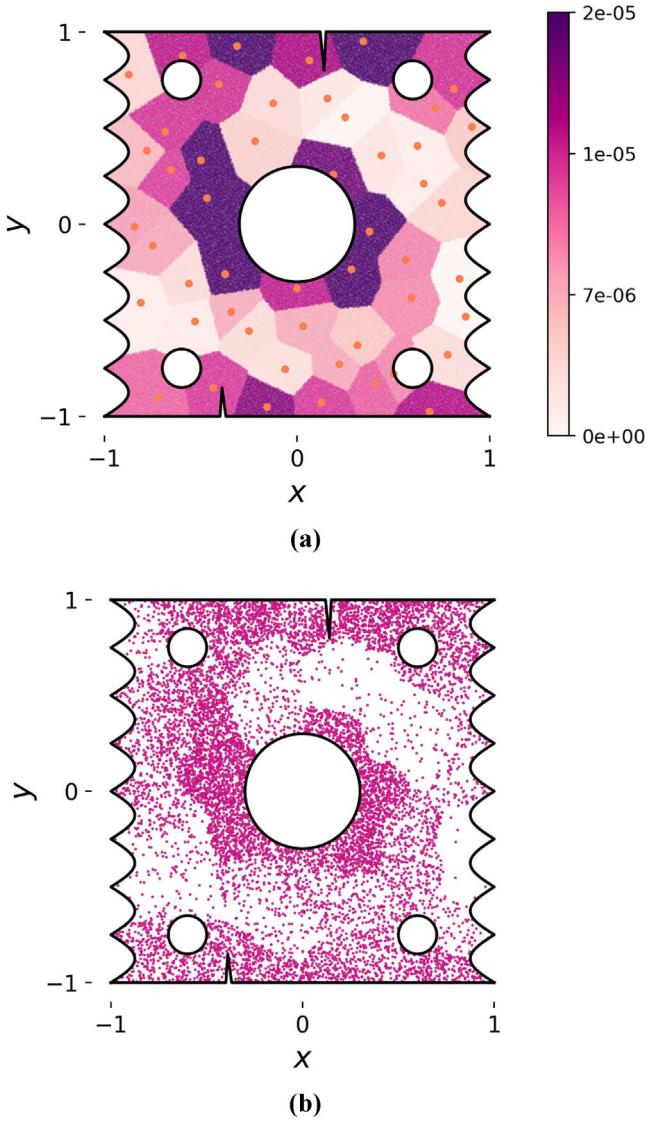

FIGURE 9 (a) A visualization of the seeds and the collocation points color coded with the selection probability at the 30th training iteration of the proposed PWC importance sampling approach with $S = 50$; (b) the 10,000 sampled collocation points

4.2 | A plane stress problem

In the second example, we demonstrate the performance of PINNs for a linear elasticity problem on a 2D plate with multiple holes using plane stress equations. In particular, we focus on the mechanical behavior of cover-plates connections, which informs the design of joint and bolt positions in the plates. This problem entails different behaviors such as the elastic–plastic behavior of the plate, the contact between bolts and holes, and the large displacements that could happen in particular configurations, which collectively determine the failure mode of the plate, based on the complex stress distribution. It differs from the previous example in the following aspects.

- Here, there are stress boundary conditions and hence, Navier equations cannot be directly used.
- The mixed form of the equations used in this example are first order, whereas the Navier equations are second order. This can help increase the accuracy and the speed of the training.
- Since real material properties are used for the calculations, the values of these properties have to be normalized using an appropriate method.

For this experiment, following parameters used in the experimental work reported in Toussaint et al. (2017), we consider a 6-mm-thick plate of S235 grade steel with three holes for the bolts, two at the bottom and one at the top, as shown in Figure 10. A uniaxial tension is applied at the bottom of the plate in the form of an imposed displacement of 1.5 mm. The dimensions of plate are 55 mm × 70 mm. The holes have a radius of 7.5 mm. The coordinates of the center

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

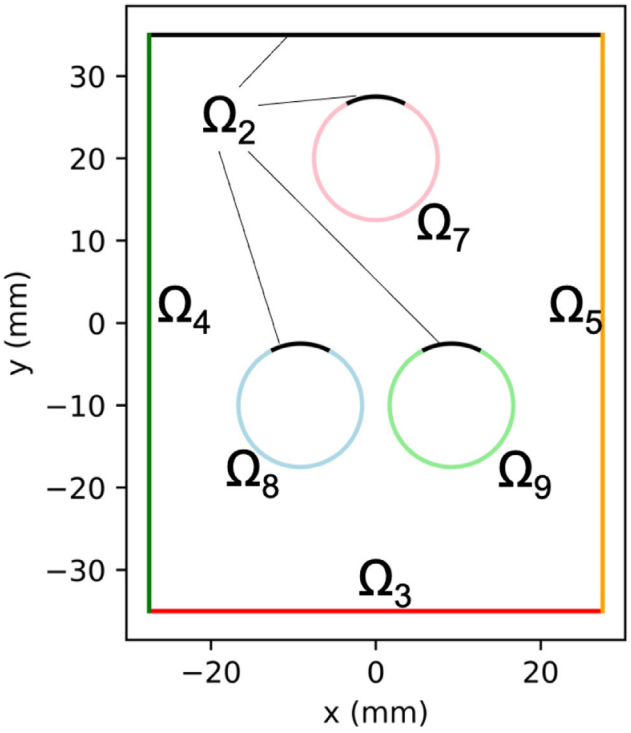

FIGURE 10 The 2D plate with three holes considered for the solving linear elasticity problem using plane stress equations defined in Equation (23). The domains used in Equation (26) are labeled with different colors

of the three holes are (0,20), (−9.67, −10) and (9.67, −10). The boundary conditions for the problem are as follows:

- $u_x = u_y = 0$ for top edge of the plate (at $y = 35.0$ mm) and for the top quarter of the three holes. Here, u_x and u_y are the displacements in the x and y directions, respectively.
- $u_x = 0, u_y = -1.5$ mm (imposed displacement) for the bottom edge of the plate (at $y = -35.0$ mm).
- Zero traction for all the free boundaries.

The governing plane stress equations for the problem are,

$$\begin{aligned}\hat{\sigma}_{xx} &= \hat{\lambda}\left(\frac{\partial \hat{u}}{\partial \hat{x}} + \frac{\partial \hat{v}}{\partial \hat{y}} + \frac{-\hat{\lambda}}{\hat{\lambda}+2\hat{\mu}}\left(\frac{\partial \hat{u}}{\partial \hat{x}} + \frac{\partial \hat{v}}{\partial \hat{y}}\right)\right) + 2\hat{\mu}\frac{\partial \hat{u}}{\partial \hat{x}},\\ \hat{\sigma}_{yy} &= \hat{\lambda}\left(\frac{\partial \hat{u}}{\partial \hat{x}} + \frac{\partial \hat{v}}{\partial \hat{y}} + \frac{-\hat{\lambda}}{\hat{\lambda}+2\hat{\mu}}\left(\frac{\partial \hat{u}}{\partial \hat{x}} + \frac{\partial \hat{v}}{\partial \hat{y}}\right)\right) + 2\hat{\mu}\frac{\partial \hat{v}}{\partial \hat{x}},\end{aligned} \tag{23}$$

where $\hat{u}, \hat{v}$ are the nondimensionalized, normalized displacement vectors, $\hat{x}$ and $\hat{y}$ are the normalized directions. $\hat{\lambda}$ and $\hat{\mu}$ are the corresponding non-dimensionalized, normalized Lamé constants. These values are given by $\hat{x_i} = x_i/L, \hat{u_i} = u_i/U, \hat{\lambda} = \lambda/\mu_c, \hat{\mu} = \mu/\mu_c$, where, L is the characteristic length, U is the characteristic displacement, and μ_c is the nondimensionalizing shear modulus. The nondimensionalized form of the stress-displacement equations are given by

$$\hat{\sigma}_{ij} = \hat{\lambda}\hat{\epsilon}_{kk}\delta_{ij} + 2\hat{\mu}\hat{\epsilon}_{ij} \tag{24}$$

where δ_{ij} is the Kronecker delta function and $\hat{\epsilon}_{ij}$ is the strain tensor that takes the following form:

$$\hat{\epsilon}_{ij} = \frac{1}{2}(\hat{u}_{i,j} + \hat{u}_{j,i}) \tag{25}$$

The nondimensionalized plane stress equations are adopted from SimNet (Hennigh et al., 2020, 2020).

To solve this PDE using a PINN, we constructed a neural network with four hidden layers with 32 units in each hidden layer. The input layer consists of two neurons, which take realizations from the physical domain, $\hat{x}$ and $\hat{y}$. The output layer consists of five neurons representing the solution $(\tilde{u}, \tilde{v}, \tilde{\sigma}_{xx}, \tilde{\sigma}_{xy}, \tilde{\sigma}_{yy})$ for the given realizations in the input layer. We adopt a Swish function for the activation in each hidden layer. The parameters of the neural network are randomly initialized, and are trained following Algorithm 2, with the following loss function (domains used in the equations below are shown in Figure 10):

$$\begin{aligned}J &= \sum_{i=1}^{9}\lambda_i J_i,\\ J_1 &= \left(\frac{\partial \tilde{\sigma}_{xx}}{\partial \hat{x}} + \frac{\partial \tilde{\sigma}_{xy}}{\partial \hat{y}}\right)^2 + \left(\frac{\partial \tilde{\sigma}_{xy}}{\partial \hat{x}} + \frac{\partial \tilde{\sigma}_{yy}}{\partial \hat{y}}\right)^2,\\ J_2 &= (\tilde{u}_x - 0.0)^2 + (\tilde{u}_y - 0.0)^2, \qquad \forall x, y \in \Omega_2,\\ J_3 &= (\tilde{u}_x - 0.0)^2 + (\tilde{u}_y + 1.0)^2, \qquad \forall x, y \in \Omega_3,\\ J_4 &= \tilde{\sigma}_{xx}^2 + \tilde{\sigma}_{xy}^2, \qquad \forall x, y \in \Omega_4,\\ J_5 &= \tilde{\sigma}_{xx}^2 + \tilde{\sigma}_{xy}^2, \qquad \forall x, y \in \Omega_5,\\ J_6 &= (\hat{\sigma}_{xx} - \tilde{\sigma}_{xx})^2 + (\hat{\sigma}_{xy} - \tilde{\sigma}_{xy})^2 + (\hat{\sigma}_{yy} - \tilde{\sigma}_{yy})^2,\\ J_7 &= ((\tilde{\sigma}_{xx}x + \tilde{\sigma}_{xy}(y - 20.0))/7.5)^2, \qquad \forall x, y \in \Omega_7,\\ J_8 &= ((\tilde{\sigma}_{xx}(x + 9.17) + \tilde{\sigma}_{xy}(y + 10.0))/7.5)^2, \qquad \forall x, y \in \Omega_8,\\ J_9 &= ((\tilde{\sigma}_{xx}(x - 9.17) + \tilde{\sigma}_{xy}(y + 10.0))/7.5)^2, \qquad \forall x, y \in \Omega_9\end{aligned} \tag{26}$$

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

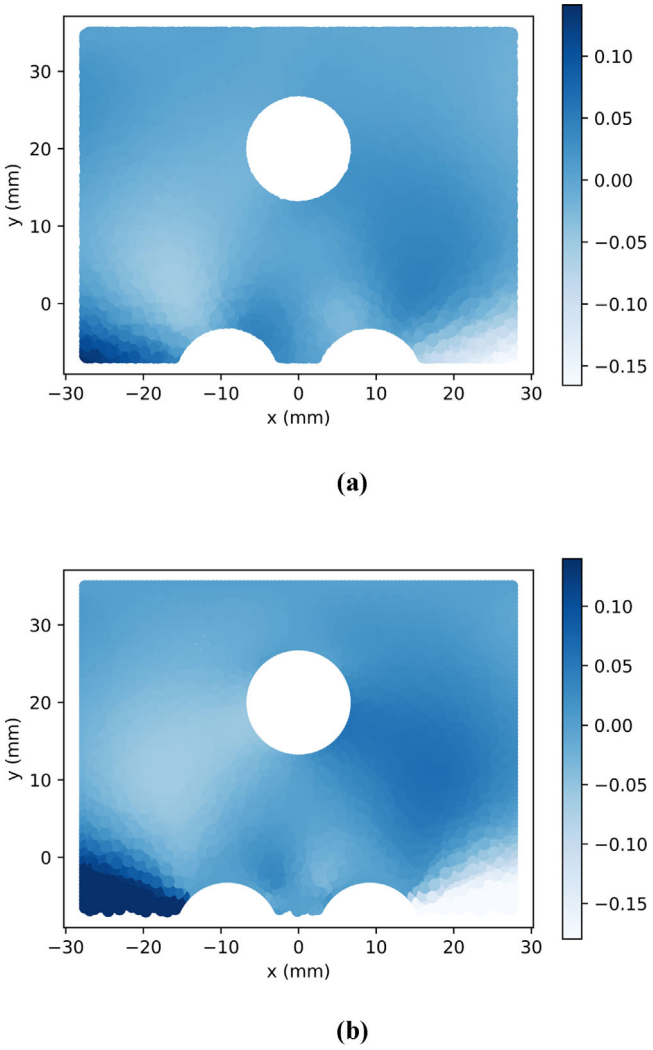

FIGURE 11 A comparison of the displacement in the x-direction, u_x between PINN and finite element (a) u_x computed by PINN; (b) u_x as computed by FEM

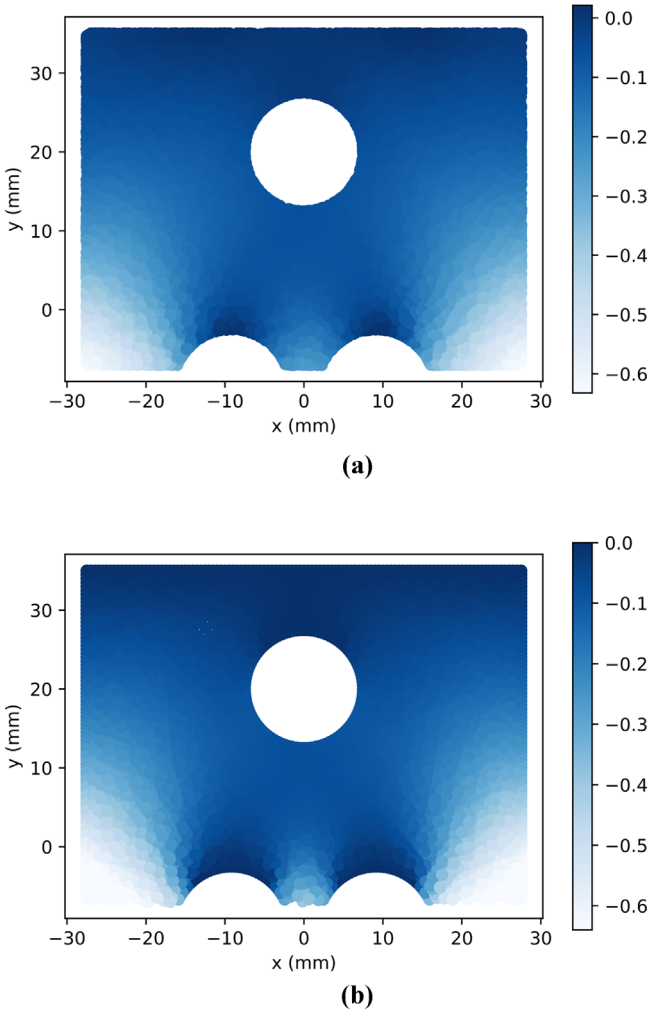

FIGURE 12 A comparison of the displacement in the y-direction, u_y between PINN and finite element (a) u_y computed by PINN; (b) u_y as computed by FEM

The gradient of the loss function with respect to model parameters is computed through backpropagation and parameters λ_1 through λ_9, which determine the contributions of each loss value to the overall loss, are finalized through some iterations of training with few epochs. The final values of these parameters are set to be $\lambda_1 = 500$, $\lambda_2 = 1000$, $\lambda_3 = 1000$, $\lambda_4 = 75$, $\lambda_5 = 75$, $\lambda_6 = 200$, $\lambda_7 = 75$, $\lambda_8 = 75$, $\lambda_9 = 75$. Batch size is set to be 5000 and the learning rate is 0.0005. A generalized Halton sequence generator algorithm is used to generate $N = 500{,}000$ collocation points and $S = 5000$ seeds within the computation domain. Another 500,000 uniformly distributed boundary collocation points are also generated. The model is trained for 25,000 iterations. Figures 11 and 12 show the close agreement between displacements obtained by our proposed importance sampling approach for the PINN model and displacements from the finite element solution (using ANSYS). Furthermore, Figure 13 shows that in training the PINN model, the importance sampling approach provides better computational performance compared to the uniform sampling approach, in terms of both the number of iterations and elapsed time.

4.3 | A transient diffusion problem

In the previous two examples, we demonstrated the performance of our proposed PWC importance sampling approach on boundary value problems. However, in addition to boundary value problems, PINNs have also been applied to solve time-dependent PDEs (e.g., Nabian & Meidani, 2020; Raissi, Perdikaris, et al., 2019). Therefore, to numerically verify the accuracy and efficiency of our proposed method in solving time-dependent problems, let us

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

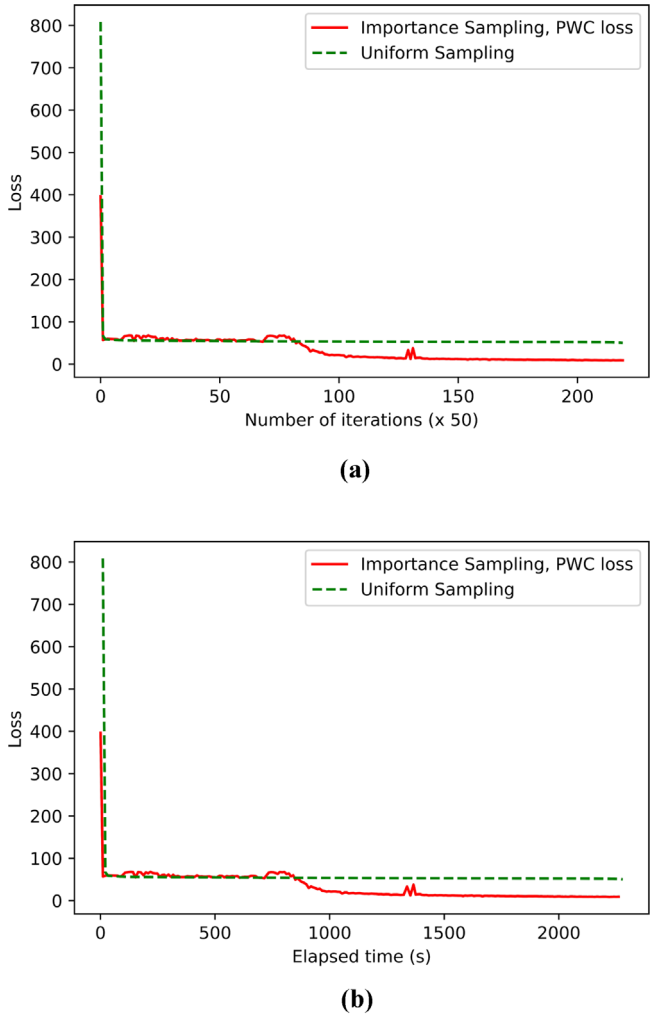

FIGURE 13 Comparisons between the training loss of PINN with importance sampling and PINN with uniform sampling when applied on the plane stress of Section 4.2. Training losses are depicted versus the number of iterations (top figure) and elapsed time (bottom figure)

consider a transient diffusion problem, governed by the following PDE,

$$\begin{aligned} &\mathcal{N}(t,x,u) = \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - 3x, \quad t \in [0,1], x \in [0,1], \\ &\mathcal{I}(x) = u - 10(x - x^2), \quad x \in [0,1], \\ &\mathcal{B}(t,x) = u, \quad t \in [0,1], x \in \{0,1\} \end{aligned} \tag{27}$$

To solve this PDE using a PINN, we construct as the trial solution a neural network with four hidden layers with 32 units in each hidden layer. The input layer consists of two neurons, which take realizations from the physical domain (i.e., t, x). The output layer consists of one neuron, which represents the solution u for the given realizations in the input layer. We adopt a sine function for the activations

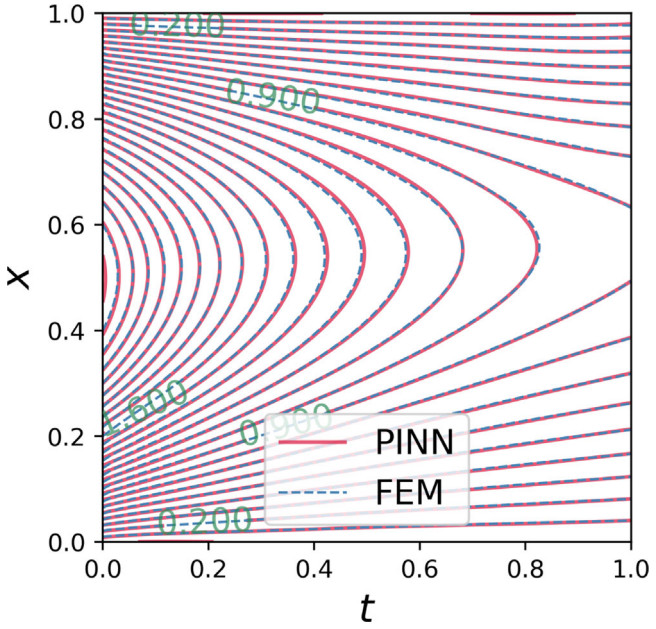

FIGURE 14 A comparison between the finite element and the PINN solutions for u in Equation (27). The PINN solution is trained using the proposed importance sampling approach with piecewise constant approximation to loss

in each hidden layer. The parameters of the trial solution are randomly initialized, and are trained following Algorithm 2, with the following loss function:

$$\begin{aligned} &J = J_1 + \lambda_1 J_2 + \lambda_2 J_3, \\ &J_1 = \left(\frac{\partial \tilde{u}}{\partial t} - \frac{\partial^2 \tilde{u}}{\partial x^2} - 3x \right)^2, \ t \in [0,1], x \in [0,1], \\ &J_2 = \left(\tilde{u} - 10(x - x^2) \right)^2, \ t = 0, x \in [0,1], \\ &J_3 = \tilde{u}^2, \ \ t \in [0,1], x \in \{0,1\} \end{aligned} \tag{28}$$

The gradient of this loss function with respect to model parameters is computed through backpropagation (Baydin et al., 2018), as explained in Section 2.1. Parameters λ_1 and λ_2 are determined based on a few pilot runs, each for a few iterations only, to examine the relative contribution of each term in the loss function to the overall loss. Based on these pilot runs, both of these parameters are set to 500. Batch size is set to 10,000, and the learning rate α to 0.003. A generalized Halton sequence generator algorithm is used to generate $N = 100{,}000$ collocation points and $S = 10{,}000$ seeds within the computation domain. Another 100,000 uniformly distributed boundary collocation points are also generated. The model is trained for 3000 iterations. Figure 14 shows the accuracy of the PINN solution trained using the proposed approach against the finite element solution (using MATLAB Partial Differential Equation Toolbox) in solving a time-dependent

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

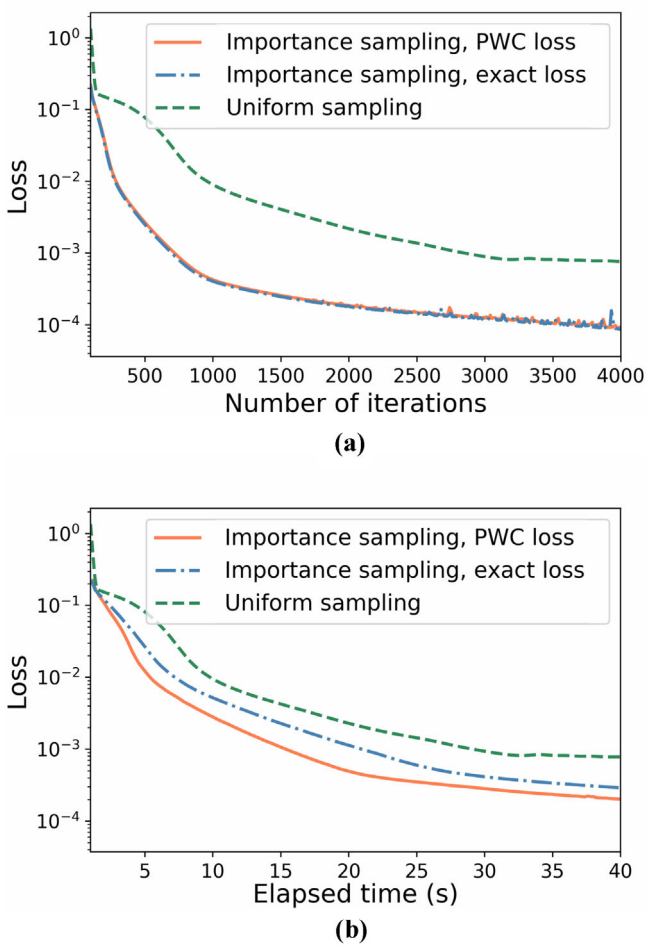

FIGURE 15 A comparison between the performance of the three approaches of uniform sampling, importance sampling with exact loss evaluation, and importance sampling with approximate loss evaluation for training of a PINN solution to transient diffusion defined in Equation (27)

diffusion example. The computational time for the finite element and PINN results presented in this figure are 32 and 40 s, respectively.

Figure 15 compares the computational performance of the importance sampling approach with exact and PWC loss evaluations versus that of the uniform sampling approach, in terms of number of iterations and elapsed time. Once again, it can be seen in Figure 15(a) that the PWC approximation to loss is in fact a good approximation. Also, Figure 15(b) shows that the PWC importance sampling approach provides a better computational efficiency compared to the other two approaches, and numerically demonstrate the effectiveness of this method in accelerating the training of PINNs.

5 | CONCLUSION

PINNs are a recently developed class of machine learning–based methods that can be used to solve PDEs. Although PINNs offer unique advantages over the existing classical numerical methods for PDE, in their current form they are not expected to dominate the best of the classical methods, which have been substantially improved over the past few decades and are optimized in terms of efficiency and robustness. With that in mind, this paper takes a step forward toward improving the computational efficiency of PINNs for solving PDEs. Specifically, in this paper we presented a new approach for training of PINNs based on importance sampling, which selects training points according to a proposal distribution proportional to a PWC approximation of the loss function. We showed that this approach does not introduce any new hyperparameters, and is straightforward to implement into the existing PINN codes. With some theoretical evidences and also three numerical examples of elasticity and transient diffusion, we demonstrated the effectiveness of the proposed importance sampling approach in improvising the efficiency of PINN training.

While the theoretical background and numerical results presented in this paper provide sufficient evidence that use of the proposed importance sampling approach is promising for accelerating the convergence of PINN training, it is still necessary to further investigate the success of our proposed importance sampling approach in solving more challenging problems, such as stochastic PDEs and time-dependent PDEs with highly oscillatory or non-monotonic solutions. Also, the following extensions to this work can be addressed in future studies: (1) generalizing the algorithm to distributed importance sampling for even more computational efficiency, where a number of workers search for the most informative collocation points while a single worker updates the model parameters; (2) modifying the algorithm for execution across the recently developed tensor processing units (TPUs) (Jouppi et al., 2017), which are artificial intelligence (AI) accelerator application–specific integrated circuits developed specifically for efficient training of deep neural networks; (3) investigating the performance of the proposed algorithm in improving the efficiency of PINNs for data-driven PDE discovery.

ACKNOWLEDGMENT

We would like to express our gratitude to Amir Kazemi for his help toward generating the finite element analysis results using ANSYS for the plane stress problem.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P. Wicke, M., …, Google Brain (2016). Tensorflow: A system for large-scale machine learning. In *12th USENIX*

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Symposium on Operating Systems Design and Implementation (OSDI 16), (pp. 265–283). Savannah, GA, USA.

Alain, G., Lamb, A., Sankar, C., Courville, A., & Bengio, Y. (2015). Variance reduction in SGD by distributed importance sampling. *arXiv preprint arXiv:1511.06481*.

Aurenhammer, F. (1991). Voronoi diagrams–a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, *23*(3), 345–405.

Azimi, M., & Pekcan, G. (2019). Structural health monitoring using extremely compressed data through deep learning. *Computer-Aided Civil and Infrastructure Engineering*. *35*, 6, (pp. 597–614).

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, *18*, 1–43.

Beck, C., Becker, S., Grohs, P., Jaafari, N., & Jentzen, A. (2018). Solving stochastic differential equations and Kolmogorov equations by means of deep learning. *arXiv preprint arXiv:1806.00421*.

Berg, J., & Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, *317*, 28–41.

Bochev, P. B., & Gunzburger, M. D. (2006). *Least-squares finite element methods*. Springer.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. G.B. Orr & K.R. Müller *Neural networks: Tricks of the trade* (pp. 421–436). Berlin, Heidelberg: Springer.

Cha, Y.-J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378.

Chikazawa, Y., Koshizuka, S., & Oka, Y. (2001). A particle method for elastic and visco-plastic structures and fluid-structure interactions. *Computational Mechanics*, *27*(2), 97–106.

Dissanayake, M., & Phan-Thien, N. (1994). Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, *10*(3), 195–201.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*(Jul), 2121–2159.

Faure, H., & Lemieux, C. (2009). Generalized Halton sequences in 2008: A comparative study. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *19*(4), 15.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goswami, S., Anitescu, C., Chakraborty, S., & Rabczuk, T. (2019). Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *arXiv preprint arXiv:1907.02531*.

Guo, H., Zhuang, X., & Rabczuk, T. (2019). A deep collocation method for the bending analysis of Kirchhoff plate. *CMC-Computers Materials & Continua*, *59*(2), 433–456.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, *2*(1), 84–90.

Hammersley, J. M. (2013). *Monte Carlo methods*. Springer Science & Business Media.

Hammersley, J. M. (1960). Monte Carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, *86*(3), 844–874.

Hashemi, H., & Abdelghany, K. (2018). End-to-end deep learning methodology for real-time traffic network management. *Computer-Aided Civil and Infrastructure Engineering*, *33*(10), 849–863.

Hennigh, O., Narasimhan, S., Nabian, M. A., Subramaniam, A., Tangsali, K., Rietmann, M., Ferrandis, J. d. A., Byeon, W., Fang, Z., & Choudhry, S. (2020). Nvidia simnet {TM}: An AI-accelerated multi-physics simulation framework. *arXiv preprint arXiv:2012.07938*.

Hennigh, O., Tangsali, K., Subramaniam, A., Narasimhan, S., Nabian, M., Ferrandis, J. d. A., & Choudhry, S. (2020). Simnet: A neural network solver for multi-physics applications. *Bulletin of the American Physical Society*. https://meetings.aps.org/Meeting/DFD20/Session/S01.7.

Jagtap, A. D., & Karniadakis, G. E. (2019). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *arXiv preprint arXiv:1906.01170*.

Joe, S., & Kuo, F. Y. (2008). Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, *30*(5), 2635–2654.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T. V., … Yoon, D. H. (2017). In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, (pp. 1–12). IEEE.

Katharopoulos, A., & Fleuret, F. (2017). Biased importance sampling for deep neural network training. *arXiv preprint arXiv:1706.00043*.

Katharopoulos, A., & Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. *arXiv preprint arXiv:1803.00942*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kissas, G., Yang, Y., Hwuang, E., Witschey, W. R., Detre, J. A., & Perdikaris, P. (2019). Machine learning in cardiovascular flows modeling: Predicting pulse wave propagation from non-invasive clinical measurements using physics-informed deep learning. *arXiv preprint arXiv:1905.04817*.

Koziarski, M., & Cyganek, B. (2017). Image recognition with deep neural networks in presence of noise: Dealing with and taking advantage of distortions. *Integrated Computer-Aided Engineering*, *24*(4), 337–349.

Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, *9*(5), 987–1000.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Liang, X. (2019). Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with Bayesian optimization. *Computer-Aided Civil and Infrastructure Engineering*, *34*(5), 415–430.

Lin, Y.-z., Nie, Z.-h., & Ma, H.-w. (2017). Structural damage detection with automatic feature-extraction through deep learning. *Computer-Aided Civil and Infrastructure Engineering*, *32*(12), 1025–1046.

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Liu, Q., Wang, B., & Zhu, Y. (2018). Short-term traffic speed forecasting based on attention convolutional neural network for arterials. *Computer-Aided Civil and Infrastructure Engineering*, *33*(11), 999–1016.

Long, Z., Lu, Y., Ma, X., & Dong, B. (2017). PDE-net: Learning PDEs from data. *arXiv preprint arXiv:1710.09668*.

Luo, X., & Kareem, A. (2019). Deep convolutional neural networks for uncertainty propagation in random fields. *Computer-Aided Civil and Infrastructure Engineering*. *34*, 12, (pp. 1043–1054).

Marsland, S. (2014). *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC.

Meng, X., & Karniadakis, G. E. (2019). A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems. *arXiv preprint arXiv:1903.00104*.

Nabian, M. A., & Meidani, H. (2018). Deep learning for accelerated seismic reliability analysis of transportation networks. *Computer-Aided Civil and Infrastructure Engineering*, *33*(6), 443–458.

Nabian, M. A., & Meidani, H. (2019). A deep learning solution approach for high-dimensional random differential equations. *Probabilistic Engineering Mechanics*, *57*, 14–25.

Nabian, M. A., & Meidani, H. (2020). Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. *Journal of Computing and Information Science in Engineering*, *20*(1).011006-1–011006-10.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge University Press.

Psichogios, D. C., & Ungar, L. H. (1992). A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, *38*(10), 1499–1511.

Qin, T., Wu, K., & Xiu, D. (2018). Data driven governing equations approximation using deep neural networks. *arXiv preprint arXiv:1811.05537*.

Rafiei, M. H., & Adeli, H. (2017). A novel machine learning-based algorithm to detect damage in high-rise building structures. *The Structural Design of Tall and Special Buildings*, *26*(18), e1400.

Rafiei, M. H., & Adeli, H. (2018). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, *156*, 598–607.

Rafiei, M. H., Khushefati, W. H., Demirboga, R., & Adeli, H. (2017). Supervised deep restricted Boltzmann machine for estimation of concrete. *ACI Materials Journal*, *114*(2).237–244.

Raissi, M. (2018a). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, *19*(1), 932–955.

Raissi, M. (2018b). Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*.

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). 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.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics informed deep learning (part I): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Wang, Z., Triantafyllou, M. S., & Karniadakis, G. E. (2019). Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, *861*, 119–137.

Raissi, M., Yazdani, A., & Karniadakis, G. E. (2018). Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Sirignano, J. & Spiliopoulos, K. (2017). DGM: A deep learning algorithm for solving partial differential equations. *arXiv preprint arXiv:1708.07469*.

Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, *7*(4), 784–802.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139–1147).

Toussaint, E., Durif, S., Bouchaïr, A., & Grédiac, M. (2017). Strain measurements and analyses around the bolt holes of structural steel plate connections using full-field measurements. *Engineering Structures*, *131*, 148–162.

Weinan, E., Han, J., & Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 1–32.

Weinan, E., & Yu, B. (2018). The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, *6*(1), 1–12.

Xu, K., & Darve, E. (2019). The neural network approach to inverse problems in differential equations. *arXiv preprint arXiv:1901.07758*.

Yang, L., Zhang, D., & Karniadakis, G. E. (2018). Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:1811.02033*.

Yang, Y., & Perdikaris, P. (2019). Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, *394*, 136–152.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zeinali, Y., & Story, B. A. (2017). Competitive probabilistic neural network. *Integrated Computer-Aided Engineering*, *24*(2), 105–118.

Zhang, X., Rajan, D., & Story, B. (2019). Concrete crack detection using context-aware deep semantic segmentation network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(11), 951–971.

Zhang, Y., Cheng, T., & Ren, Y. (2019). A graph deep learning method for short-term traffic forecasting on large road networks. *Computer-Aided Civil and Infrastructure Engineering*, *34*(10), 877–896.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, *394*, 56–81.

How to cite this article: Nabian MA, Gladstone RJ, Meidani H. Efficient training of physics-informed neural networks via importance sampling. *Comput Aided Civ Inf*. 2021;*36*:962–977. https://doi.org/10.1111/mice.12685

14678667, 2021, 8, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/mice.12685 by University of Adelaide Alumni, Wiley Online Library on [20/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License