Simulation-Free Reliability Analysis with Active Learning and Physics-Informed Neural Network

Chi Zhang[a], Abdollah Shafieezadeh[a*]

[a]Risk Assessment and Management of Structural and Infrastructure Systems (RAMSIS) Lab, Department of Civil, Environmental, and Geodetic Engineering, The Ohio State University, Columbus, OH, 43210, United States

Abstract: Physical phenomena are often described by partial differential equations (PDEs), which have been traditionally solved using computationally demanding finite element, difference, or volume methods to produce labeled data. Due to its multi-query nature, characterization of event probabilities requires many such simulations, which can become prohibitive given the high costs of acquiring labeled data. As opposed to conventional PDE solution methods, Physics-Informed Neural Network (PINN) is directly trained using the physics knowledge encoded in PDEs, and therefore is simulation free. Building on this capability, we propose a simulation-free uncertainty quantification method called adaptively trained PINN for reliability analysis (AT-PINN-RA). We introduce an active learning approach with the dual objective of training PINN for solving PDEs and characterizing the limit state. The approach actively learns from the responses of the PINN model to identify the limit state and subsequently, adaptively shifts the focus of the training of the PINN model to regions of high importance for failure probability characterization to boost the accuracy and efficiency of reliability estimation. The performance of AT-PINN-RA is investigated using four benchmark problems with varying complexities. In all examples, AT-PINN-RA provides accurate estimates of event probabilities with high efficiency.

Key words: Physics-informed neural network, Deep neural network, Reliability Analysis, Adaptive method

1. Introduction

The primary goal of reliability analysis is to estimate the probability of an event of interest, often a failure event, considering the uncertainties that affect the occurrence of that event. The failure probability is an essential measure for various natural and engineered systems and is an integral part of risk assessment and management frameworks for design, monitoring, maintenance, and upgrade decisions. The failure probability, P_f, can be determined as follows:

$$P_f = P(g(\boldsymbol{\xi}) \leq 0) = \int_{g(\boldsymbol{\xi})\leq 0} f(\boldsymbol{\xi}) d\boldsymbol{x} \tag{1}$$

where $\boldsymbol{\xi}$ is the vector of random variables, $g(\boldsymbol{\xi})$ is the limit state function and $f(\boldsymbol{\xi})$ is the joint probability density function of the random variables. Despite this relatively simple formulation, the integral is often high-dimensional and may involve a complex limit state function. Thus, in most cases, the failure probability can be quite difficult to obtain due to the absence of an analytical solution for the irregular integral.

The most straight forward ways of estimating the failure probability are simulation approaches, such as Monte Carlo Simulation (MCS). They often entail drawing many samples from the probability distribution of the random variables. The evaluation of the limit state function is required for each sample, which results in a high computational demand. Using MCS, P_f can be estimated as follows:

$$P_f = \int \quad f(\boldsymbol{\xi}) I(\boldsymbol{\xi}) d\boldsymbol{x} \approx \frac{1}{N_{MCS}} \sum_{i=1}^{N_{MCS}} I(\boldsymbol{\xi}_i) \tag{2}$$

where $I(\boldsymbol{\xi})$ is the indicator function yielding zero when the system is safe and one when the system fails, respectively, N_{MCS} is the population size of the MCS samples and $\boldsymbol{\xi}_i$ is one of the MSC samples. There are other simulation methods developed to reduce the computational cost, such as importance sampling and subset simulation. The limit state function in many problems in science and engineering is at core driven by physical laws and are expressed in the form of linear or nonlinear ODEs/PDEs. For spatiotemporal phenomena, these equations can be expressed as follows:

$$u(\boldsymbol{x},t)_t + F(u(\boldsymbol{x},t),\lambda) = 0, x \in \Omega, t \in [0,T] \tag{3}$$

where $\boldsymbol{x}$ is the spatial variable with D dimensions, t is the temporal variable, $u(\boldsymbol{x},t)_t$ is the derivative of the hidden solution $u(\boldsymbol{x},t)$ with respect to time, $F(\cdot)$ is a nonlinear operator parametrized by λ, and Ω is a subset of $\mathbb{R}^D$. This type of problem is normally solved using methods such as finite element (FE) and finite difference (FD) techniques. Using these methods, it can be time-consuming to solve complex problems, hence resulting in prohibitive computational costs when considering multi-query analyses such as reliability analysis. To reduce the number of required simulations, one can use approximation methods, such as first and second order reliability method (FORM and SORM)) [1], [2], or alternatively adopt surrogate model-based methods. Candidate surrogate models include Polynomial Response Surface [3]–[5], Polynomial Chaos Expansion (PCE) [6], Support Vector Regression (SVR) [7]–[9], and Kriging [10]–[14]. However, these methods still require labeled data, and for each data point, we still require one FE or FD simulation.

In the past decade, deep neural networks (DNNs) have been gaining a lot of attention due its promising performance in many scientific and engineering fields. The applications of DNN in reliability are steadily growing [15]. The DNN-based method can outperform other methods [16]. For example, Xiang et al. [17] proposed an active learning reliability method combining deep neural network and weighted sampling. Bao et al. [18] developed an adaptive subset searching-based DNN for reliability analysis. However, these methods based on DNN still require labeled data. Recently, a type of DNN that is free of labeled data has been developed. Different than the traditional DNNs that require a large number of labeled data to train, this novel DNN relies on the knowledge of the physics embedded in the ODE/PDE instead of the labeled data. The idea of using the information of ODE/PDE was discussed in some works as early as the 90s [19]–[21]. Recently, it was revisited and revised by Raissi et al. in their proposed approach called Physics-Informed Neural Network (PINN) [22]. PINN leverages the capability of DNN as a universal approximator [23] and utilizes the automatic differentiation (AD) [24] to differentiate neural networks. The output, which is regarded as the estimation of the solution to the ODE/PDE, and its derivatives, which are obtained with AD, are plugged into PDEs that are in the form of Eq. (3). If the squared value of the left-hand side of Eq. (3) is calculated, it can be regarded as the violation of physics. Thus, it can be regarded as a part of the loss function in DNN to be minimized, which can be referred to as physics-informed loss. The other components of the loss functions are comprised of the error estimated by the labeled data coming from the initial and boundary conditions, which are often very easy to obtain. During the training of the PINN, no evaluation of the solution of the ODE/PDE is needed, thus it can be regarded as simulation free. The PINN provides new opportunities for analyzing reliability of stochastic problems that are expressed in the form of ODE/PDEs.

The application of PINN for reliability analysis has been first investigated by Chakraborty [25]. For the sake of convenience, his work will be referred to as PINN-RA in the remainder of the paper. In PINN-RA, a modified PINN is used as the surrogate model for the function evaluation in reliability analysis. The random variables are considered as a part of the input along with the spatial and temporal variables for the modified PINN. After training the modified PINN, the neural network is able to make predictions given different realizations of the random variables. However, the generation of the random variables in the collocation points in that method is purely based on Latin hypercube sampling (LHS), which is not particularly most efficient for reliability analysis. In reliability analysis, accurate understanding and representation of true responses is critical only near the limit state. Therefore, the accuracy of the PINN model in regions other than the vicinity of the limit state is not essential. Due to the even distribution of the training focus for random variables in PINN-RA, the unimportant regions are treated equally as the

important regions, which can lead to an ineffective allocation of computational resources for achieving the desired accuracy. Moreover, as the dimension of the random vector increases, maintaining high accuracy over the entire domain of random variables becomes more challenging for the PINN model, thus potentially degrading the accuracy of reliability analysis using PINN. To overcome this limitation, we propose an adaptively trained PINN for reliability analysis (AT-PINN-RA). An active learning approach called adaptive training (AT) sampling is developed with the dual objective of training PINN for solving PDEs and characterizing the limit state. During the training process, the approach actively learns from the responses of the PINN model as it becomes more accurate in terms of characterizing the limit state. This information is subsequently utilized to guide the training focus to regions of high importance which then enhances the fidelity of limit state characterization, thus forming a virtuous circle. A set of anchor points are generated to facilitate adaptive sampling.

The rest of the paper is organized as follows: Section 2 presents a review of PINN and PINN-RA. Section 3 introduces the proposed method AT-PINN-RA. The performance of the proposed method is then demonstrated via four numerical examples in Section 4. In the end, conclusions are provided in Section 5.

2. PINN for Reliability Analysis

In this section, PINN is briefly reviewed first. Next, the existing approach PINN-RA proposed by Chakraborty [25] is presented.

2.1 Overview of PINN

For the architecture of the network, the original PINN uses fully connected deep neural networks (FC-DNNs), which are the simplest and most common ones in the domain of deep learning. A FC-DNN can be seen as nested nonlinear transformations of simple affine transformations [26]. These models can be expressed as follows:

$$\begin{aligned} \boldsymbol{Y} &= \varphi_{L+1}(\boldsymbol{W}_{L+1}\boldsymbol{z}_L + \boldsymbol{b}_{L+1}) \\ \boldsymbol{z}_L &= \varphi_L(\boldsymbol{W}_L\boldsymbol{z}_{L-1} + \boldsymbol{b}_L) \\ \boldsymbol{z}_{L-1} &= \varphi_{L-1}(\boldsymbol{W}_{L-1}\boldsymbol{z}_{L-2} + \boldsymbol{b}_{L-1}) \\ &\cdots \\ \boldsymbol{z}_1 &= \varphi_1(\boldsymbol{W}_1\boldsymbol{z}_0 + \boldsymbol{b}_1) \end{aligned} \tag{4}$$

where $\boldsymbol{Y}$ is the final output of the FC-DNN, φ_i is a nonlinear activation function for the i_{th} layer, of which some typical choices are ReLUs, leaky ReLUs, Sigmoids and hyperbolic tangents [27], $\boldsymbol{W}_i$ is the matrix that stores the weights in the i_{th} layer, $\boldsymbol{z}_i$ is the output of the i_{th} layer and the input of the $(i+1)_{th}$ layer, and $\boldsymbol{b}_i$ is the vector that stores the bias parameters in the i_{th} layer. From this point, $\boldsymbol{\theta}$ will be used to denote the collection of all the $\boldsymbol{W}_i{}'$s and $\boldsymbol{b}_i$'s in the network for the sake of convenience. The network presented in Eq. (4) has $L+2$ layers in total, with 0_{th} layer being the input layer and $(L+1)_{th}$ layer being the output layer. The layers between the input layer and output layer are often referred to as hidden layers. In this work, the hyperbolic tangent functions are chosen as the activation functions for the hidden layers and the output layer uses a linear activation function. For the sake of convenience, Eq. (4) is written in a more compact form as follows:

$$\boldsymbol{Y} = \mathbb{N}(\boldsymbol{z_0}) \tag{5}$$

where $\mathbb{N}(\cdot)$ represents the neural network.

In PINN, a FC-DNN is used to estimate the hidden solution of the ODE/PDE $u(\boldsymbol{x}, t)$. The input of the neural network is the spatial variable $\boldsymbol{x}$ and temporal variable t. The output of the variable, which is denoted as $\hat{u}(\boldsymbol{x}, t) = \mathbb{N}(\boldsymbol{x}, t)$, is the solution estimated by PINN. Training a deep neural network often requires a significant amount of training data. However, for forward ODE/PDE problems, PINN is data-free. It only requires N collocation points $\left\{\left(\boldsymbol{x}^i, t^i\right)\right\}_{i=1}^{N_{total}}$ that can easily be generated using a preferred design of

experiment (DOE). The automatic differentiation [24] is used to calculate the derivatives of $\hat{u}(\boldsymbol{x}, t)$ with respect to $\boldsymbol{x}$ and t for these collocation points. Then, the values of $\hat{u}(\boldsymbol{x}, t)$ and its derivatives for the collocation points are used to calculate the losses that represent the violations of the PDE, initial and boundary condition states, respectively. When the weighted sum of the losses is being minimized, the violations of physics are also being optimized. Thus, $\hat{u}(\boldsymbol{x}, t)$ converges toward $u(\boldsymbol{x}, t)$ during the process of training the network. Before the training starts, a dataset of training points, with a population size of N_{total}, is generated. A graphical instruction of PINN is shown in Fig 1(a).

The physics-informed loss function for the collocation points can be expressed as follows:

$$L_{PDE}(\boldsymbol{\theta}) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left| \hat{u}(\boldsymbol{x}^i, t^i)_t + F\left(\hat{u}(\boldsymbol{x}^i, t^i), \lambda\right) \right|^2 \tag{6}$$

where N_{PDE} is the number of collocation points generated for the PDE residual in a batch, and $\hat{u}(\boldsymbol{x}^i, t^i)_t + F\left(\hat{u}(\boldsymbol{x}^i, t^i), \lambda\right)$ is the residual of the PDE calculated at $(\boldsymbol{x}^i, t^i)$. For the initial condition, the data-driven loss function can be formulated as follows:

$$L_{IC}(\boldsymbol{\theta}) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left| u_{IC,i} - \hat{u}(\boldsymbol{x}^i, t^i) \right|^2 \tag{7}$$

where N_{IC} is the number of collocation points generated for the initial condition in a batch, and $u_{IC,i}$ is the initial condition of the system evaluated at $(\boldsymbol{x}^i, t^i)$. And similarly, for the boundary condition, the data-driven loss function can be formulated as follows:

$$L_{BC}(\boldsymbol{\theta}) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| u_{BC,i} - \hat{u}(\boldsymbol{x}^i, t^i) \right|^2 \tag{8}$$

where N_{BC} is the number of collocation points generated for the boundary condition in a batch, and $u_{IC,i}$ is the boundary condition of the system evaluated at $(\boldsymbol{x}^i, t^i)$. Thus, the loss function in PINN can be formulated as follows:

$$L(\boldsymbol{\theta}) = PL(\boldsymbol{\theta}) + \gamma_b BL(\boldsymbol{\theta}) + \gamma_i IL(\boldsymbol{\theta}), \tag{9}$$

where γ_b and γ_i are the weight parameters for the boundary and initial condition state losses, respectively. The weights γ_b and γ_i can be customized or tuned during the training process [28], [29], to improve the trainability of PINN. In this paper, γ_b and γ_i are both set as one as in the original PINN. The graphical illustration of PINN is shown in Fig 1(a).

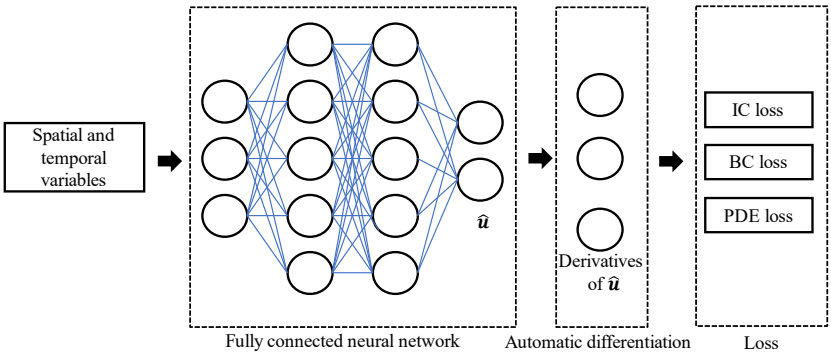

Fig 1(a). The graphic illustration of PINN

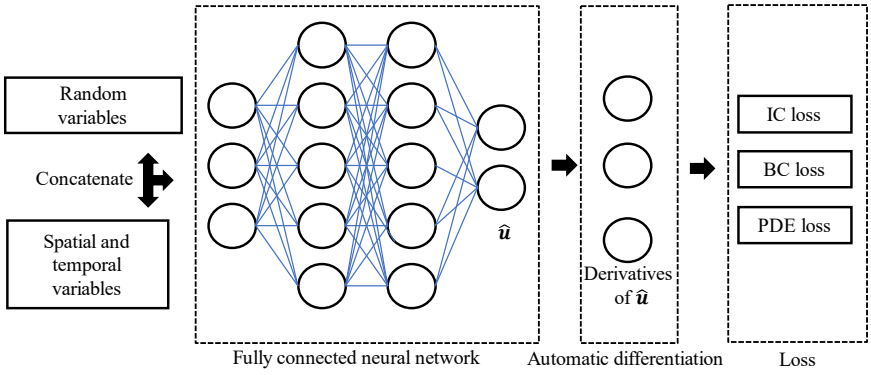

Fig 1(b). The graphic illustration of PINN-RA

The FC-DNN is trained by performing an optimization for $\boldsymbol{\theta}$ to minimize the loss function in Eq. (9). The optimization problem has the following form:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L(\boldsymbol{\theta}) \tag{10}$$

A network is trained in a series of epochs. One epoch is one round of training on the entire dataset. The batch-size determines the number of training points from a dataset used to evaluate one gradient update. Training with a batch-size of the number of the entire training points N_{total} is known as full-batch learning, and training with a batch-size of a smaller number $N < N_{total}$ is known as mini-batch learning. In this paper, mini-batch learning is adopted for all the methods discussed. Once the network is trained, it can be used to make predictions of the solution to the PDE at any unknown point $(\boldsymbol{x}^*, t^*)$.

2.2 PINN-RA

Chakraborty [25] proposed to use PINN for reliability analysis. In his work, a modified version of PINN, which is herein referred to as PINN-RA, was introduced. Let's represent the value of the limit state function as:

$$g(\boldsymbol{\xi}) = u(\boldsymbol{\xi}) - u_c \tag{11}$$

where u_c is the critical value for the solution u, and $u(\boldsymbol{\xi})$ is the system response, which is the solution to the ODE/PDE as follows:

$$u(\boldsymbol{x}, t)_t + F(u(\boldsymbol{x}, t), \lambda, \boldsymbol{\xi}) = 0, x \in \Omega, t \in [0, T] \tag{12}$$

Eq. (12) is very similar to Eq. (3) with the only difference that $\boldsymbol{\xi}$ is introduced in the nonlinear operator. This means that some parameters in the original λ are now stochastic and are represented by $\boldsymbol{\xi}$. With this

change in the equation, the current objective is to train a FC-DNN that can act as a surrogate for the response u with respect to $\boldsymbol{\xi}$. Thus, in PINN-RA, $\boldsymbol{\xi}$ is concatenated with the spatial variable $\boldsymbol{x}$ and temporal variable t to form the new collocation point $\{(\boldsymbol{x}^i, t^i, \boldsymbol{\xi}^i)\}_{i=1}^{N_{total}}$. And the collocation points generated using a preferred DOE are fed into the modified PINN $\hat{u}_N(\boldsymbol{x}, \mathrm{t}, \boldsymbol{\xi}) = \mathbb{N}(\boldsymbol{x}, \mathrm{t}, \boldsymbol{\xi})$. The graphical illustration of PINN-RA is shown in Fig 1. (b). The three loss terms are expressed as follows, respectively:

$$
\begin{aligned}
L_{PDE}(\boldsymbol{\theta}) &= \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left| \hat{u}(\boldsymbol{x}^i, t^i, \boldsymbol{\xi}^i)_t + F(\hat{u}(\boldsymbol{x}^i, t^i, \boldsymbol{\xi}^i), \lambda) \right|^2, \\
L_{IC}(\boldsymbol{\theta}) &= \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left| u_{IC,i}(\boldsymbol{\xi}^i) - \hat{u}(\boldsymbol{x}^i, t^i, \boldsymbol{\xi}^i) \right|^2, \\
L_{BC}(\boldsymbol{\theta}) &= \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| u_{BC,i}(\boldsymbol{\xi}^i) - \hat{u}(\boldsymbol{x}^i, t^i, \boldsymbol{\xi}^i) \right|^2
\end{aligned}
\tag{13}
$$

The loss terms for initial and boundary conditions can sometimes be eliminated by incorporating the initial and boundary conditions into the formulation of the neural network as follows:

$$
\hat{u}(\boldsymbol{x}, \mathrm{t}, \boldsymbol{\xi}) = u_{IC,BC}(\boldsymbol{x_b}, \mathrm{t}_i, \boldsymbol{\xi}) + B \cdot \hat{u}_N(\boldsymbol{x}, \mathrm{t}, \boldsymbol{\xi}) \tag{14}
$$

where B is a function that satisfies $B = 0$ at points associated with the initial and boundary conditions, and $u_{IC,BC}(\boldsymbol{x_b}, \mathrm{t}_i, \boldsymbol{\xi})$ is a function that satisfies all the initial and boundary conditions. Thus, $\hat{u}(\boldsymbol{x}, \mathrm{t}, \boldsymbol{\xi})$ will automatically satisfy the initial and boundary conditions without the need of $L_{IC}(\boldsymbol{\theta})$ and $L_{BC}(\boldsymbol{\theta})$ being involved in the loss function. More details of this scheme will be provided in the numerical examples section. Similar to the original PINN, the loss function for PINN-RA needs to be minimized to achieve a PINN surrogate model that is able to make predictions of the solution given different realizations of the random variables.

The advantage of PINN-RA is obvious: it is free of simulation. The incurred computational cost merely comes from training the neural network, and the trained network can be directly used for many-query analyses such as reliability analysis. However, the collocation points in PINN-RA only depend on the initial DOE, and the training points are not tailored to reliability analysis. This may lead to the waste of computational resources, as most of the training points are in the regions that are not of interest. Moreover, the accuracy of the results may not be guaranteed as there can be cases where there are no sufficient collocation points in the vicinity of the limit state. To overcome this challenge, a novel method has been proposed, which is introduced in the next section.

3. AT-PINN-RA

3.1 Adaptive training focus

As noted earlier, the sampling of the collocation points in PINN-RA may not be in favor of reliability analysis. In surrogate model assisted reliability analysis methods, an active learning function is evaluated for all the candidate training points in the candidate pool to identify the "best" training points. For example, Bichon et al. [30] developed the expected feasibility function (EFF) that provides an indication of how much the true value of the response at a point can be expected to be less than the current best solution. Echard et al. [31] proposed the *U* learning function that represents a reliability index for the risk of a surrogate model making a mistake about the sign of limit state function for a candidate training point. Lv et al. [32] developed the H learning function based on the information entropy. Sun et al. [33] proposed the least improvement function LIF that values how much the accuracy of estimated failure probability will be improved with adding a specific training point. Wang et al. proposed a learning function considering the probability density and distributed uniformity [34]. These learning functions generally aim at finding the

potential training points that are in the vicinity of the limit state and have large uncertainty. Training these surrogate models is quite different from a PINN, albeit they have the same objective that is to construct a model to perform reliability analysis. The construction of the surrogate model requires training points that are obtained with simulations. The training points, or collocation points, in PINN are randomly generated. The training process is to make the neural network satisfy the physics described by the ODE/PDEs. However, the advantages are obvious if the collocation points can be adaptively tailored, similar to adaptive surrogate-based reliability methods, to boost the convergence of the model and improve the accuracy in terms of calculating the failure probability. In this context, we propose AT-PINN-RA that adaptively samples more collocations points in the vicinity of the limit state according to the current model. For the purpose of illustration, let us consider a reliability problem entailing the ODE studied in [25], [35]. The governing differential equation is presented as follows:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -Zu, t \in [0,1] \tag{15}$$

where Z is the decay rate coefficient and is considered to be a random variable. The random variable follows a normal distribution $N(\mu, \sigma^2)$, where $\mu = -2$ and $\sigma = 1$. The system is subjected to the following initial condition:

$$u(t = 0) = 1.0 \tag{16}$$

There exists an analytical solution as follows:

$$u(t, Z) = u_0 \exp(-Zt) \tag{17}$$

The limit state function is defined as:

$$g(u(t, Z)) = u(t, Z) - u_c \tag{18}$$

where u_c is the critical value that is set as 0.5 here. For this problem, the reliability is evaluated at $t = 1$. In PINN-RA, the collocation points are generated using Latin Hypercube sampling (LHS) at the beginning of the training process and are kept the same thought the training process. For the temporal variable t, realizations are generated using a uniform distribution $U(0,1.0)$. For the random variable Z, although it follows a normal distribution, a uniform distribution $U(\mu - 5\sigma, \mu + 5\sigma)$ is used for the coverage of collocation points in favor of the reliability analysis. Thus, the collocation points $\{(t^i, Z^i)\}_{i=1}^N$ are evenly scattered in the domain as shown in Fig 2.

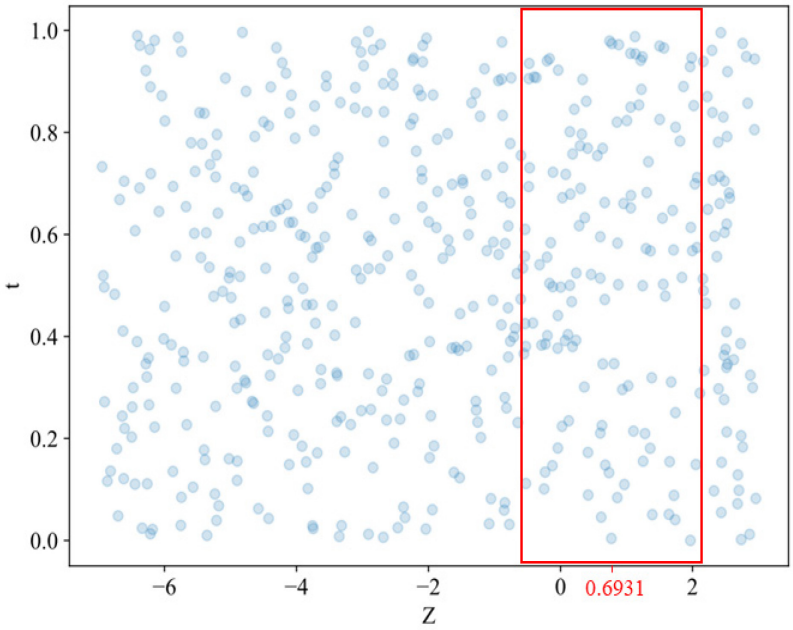

Fig 2. The collocation points generated using LHS

Because of the existence of the analytical solution, we can find that the critical value for Z is 0.6931, that is, the limit state occurs at $Z = 0.6931$. Therefore, if we are able to use PINN to make accurate predictions in the vicinity of $Z = 0.6931$, which is presented in the red box in Fig 2, we will be able to achieve an accurate estimation of the reliability. The collocation points in other regions are not as essential as the ones in the critical region. Focusing the training of the network on this critical region, can reduce the computational cost of training and improve the accuracy of network predictions as they concern estimating the failure probability. However, in most cases, the critical regions are unknown. The proximity information provided by PINN can be utilized for this purpose. Here, we propose the adaptive training (AT) sampling to gradually shift the training focus to the critical region during the training of the network, which will be introduced below. Note that in this work, only the sampling of the random variables in the collocation points is adaptive; the spatial and temporal variables in the collocation points are still generated using LHS. The reason is that only the reliability of the system at a specified time and location is studied in this paper.

The main idea of AT is to use the proximity information of the current network to turn the focus of training to critical regions. This goal is achieved by sampling more random variables in the collocation points in those regions. Thus, we propose a weight function to evaluate the significance of the random variable points as follows:

$$WF\left(g\left(\hat{u}\left(\boldsymbol{\xi}^i\right)\right)\right) = \exp\left(-A\left|g\left(\hat{u}\left(\boldsymbol{\xi}^i\right)\right)\right|\right) \tag{19}$$

As the value of the limit state function $g\left(\hat{u}\left(\boldsymbol{\xi}^i\right)\right)$ approaches the limit state according to the current model, a larger weight is assigned to the realization of the random variable $\boldsymbol{\xi}^i$. A is a factor that can define the shape of the function. The evaluation of the weight function is not significantly computationally expensive; however, it can still get costly when it comes to the entire set of collocation points in each iteration. To overcome this challenge, we propose to generate a smaller set of random variable points, which are referred to as "anchors" $\boldsymbol{\xi}_a^i, i = 1,2,\dots,N_a$, where N_a is the number of anchors. Then, the entire N_{total} random variable samples of the collocation points are categorized into N_a groups based on their proximity to those anchor points. At the beginning of each iteration, only the anchor points, $\boldsymbol{\xi}_a^i, k = 1,2,\dots,N_a$, are evaluated

using the weight function, and then the values of the normalized weight function are calculated for those points using the following formula:

$$NWF\left(g\left(\hat{u}\left(\boldsymbol{\xi}_a^i\right)\right)\right)=\frac{\exp\left(-A\left|g\left(\hat{u}\left(\boldsymbol{\xi}_a^i\right)\right)\right|\right)}{\sum_{k=1}^{N_a}\exp\left(-A\left|g\left(\hat{u}\left(\boldsymbol{\xi}_a^k\right)\right)\right|\right)} \tag{20}$$

Note that this NWF function has a very similar format to the softmax function, which is often used in multinominal logistic regression and the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes. Herein, it can be regarded as a pseudo probability of the evaluated random variable point $\boldsymbol{\xi}_a^i$ being the closest one to the limit state, which means the higher this probability, the more likely $\boldsymbol{\xi}_a^i$ is closer to the limit state. A probability distribution of a discrete random variable can be constructed using the N_a NWF values as the probability mass function (PMF) values. Each value in the PMF indicates the probability of drawing the random points from the corresponding group. In this manner, in each batch, N random points $\boldsymbol{\xi}^i$ are drawn using the PMF values generated based on Eq. (20), and then they are concatenated with the N pair of spatial variables $\boldsymbol{x}$ and temporal variables t from the LHS pool generated at the beginning to form the collocation points. These collocation points in each batch are focused on the critical region based on the belief of the current network. Gradually, the training focus is shifted to the real limit state as the training process progresses. Note that factor A determines how aggressively we shift our training focus, that is, when A is set to be a large value, very high priority will be given to the points that are, based on the current network, are believed to be close to the limit state, and vice versa. In this study, we set A as 1 for all the numerical examples.

3.2 The procedures of AT-PINN-RA

With the AT sampling, we propose the adaptively trained Physics-Informed Neural Network for reliability analysis (AT-PINN-RA). Note that the FC-DNN structure of the proposed method is the same as PINN-RA; however, the AT sampling approach makes a significant difference in terms of the performance of the method. The implementation procedure of AT-PINN-RA is described in Algorithm 1.

Algorithm 1 AT-PINN-RA

Generate $\boldsymbol{N_{total}}$ pairs of spatial and temporal variables $\left\{\left(\boldsymbol{x^i},\boldsymbol{t^i}\right)\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N_{total}}}$ and $\boldsymbol{N_{total}}$ candidate training samples for random variables $\left\{\boldsymbol{\xi_c^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N_{total}}}$ using LHS, and generate $\boldsymbol{N_a}$ anchor points for the random variables $\left\{\boldsymbol{\xi_a^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N_a}}$. Set the batch-size $\boldsymbol{N}$ and total number of epochs $\boldsymbol{N_{epoch}}$

Categorize $\left\{\boldsymbol{\xi_c^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N_{total}}}$ into $\boldsymbol{N_a}$ groups according to their proximity to the anchor points $\left\{\boldsymbol{\xi_a^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N_a}}$

Initialize the PINN for $\hat{\boldsymbol{u}}$

While True:

    For $\boldsymbol{k=1}$: $(\boldsymbol{N_{total}/N})$:

        Use Eq. (20) to determine the NWF values for all the anchor points and construct a PMF

        Use the constructed PMF to sample N random variable points $\left\{\boldsymbol{\xi^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ from random variable candidates $\left\{\boldsymbol{\xi_c^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ from the $\boldsymbol{N_a}$ groups

        Draw $\boldsymbol{N}$ pairs of spatial and temporal variables $\left\{\left(\boldsymbol{x^i},\boldsymbol{t^i}\right)\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ from the $\boldsymbol{N_{total}}$ pairs

        Concatenate $\left\{\left(\boldsymbol{x^i},\boldsymbol{t^i}\right)\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and $\left\{\boldsymbol{\xi^i}\right\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and feed the concatenated variable to FC-DNN

10:	Record the loss
11:	Perform one gradient update for $\widehat{\boldsymbol{u}}$
12:	End (the end of one epoch)
13:	Calculate the average loss for the past epoch
14:	If the average loss has not improved for over 100 epochs:
15:	Break
16:	End (the end of the training process)
17:	Use the trained neural network to perform reliability analysis

In order to cover the domain that contains the limit state, we use LHS to generate anchor points with uniform distribution, and the range for uniform sampling is $[\mu - 5\sigma, \mu + 5\sigma]$, where μ and σ are the mean and standard deviation of the random variable, respectively. This range should be sufficient to solve problems with failure probabilities larger than 3×10^{-7}. One can change the factor 5 to other values that is suitable for the specific problem. It is suggested that the number of anchor points is determined as 10^n, where n is dimension of the random variables. When n is larger than 3, a number of anchor points of 1,000 is found to be enough. In this study, a stopping criterion based on the max stagnation of loss is used. When the average loss over the epoch has not increased for a certain number of epochs (in this study, we find that 100 is sufficient for all the numerical examples), we assume the convergence is achieved and the training is stopped. The performance of the proposed AT-PINN-RA will be demonstrated using four benchmark problems in the next section.

4. Numerical examples

In this work, four numerical examples are tested to demonstrate the performance of the proposed AT-PINN-RA method. In all numerical examples, both PINN-RA and AT-PINN-RA are used to perform the reliability analyses, and the results are directly compared. Both methods are implemented and trained via PyTorch 1.70. Training is performed on a NVIDIA RTX 3090. The loss functions in all the networks are minimized using Adam optimizer [36], which is also used in multiple works related to PINN [22], [37], [38]. In all the numerical examples, the initial learning rate is set as 10^{-3}. If the loss has not improved for 1,000 gradient updates, the learning rate will be multiplied by a factor of 0.7. For all the numerical examples, the total number of collocation points $\boldsymbol{N_{total}}$ and the batch-size are set as 30,000 and 1,000, respectively. In each epoch, the gradient update will be performed 30 times. A maximum epoch number of 5,000 is set for all the numerical examples. The training will be terminated when the stopping criterion is met, or the maximum epoch number is reached. For all the numerical examples in this study, the MCS population used to calculate the failure probability has a size of 10^5. The results of all the examples are averaged over 20 repeated trials. The result of MCS with traditional numerical methods using MALAB [39] and FENICS [40] is used as the reference in all numerical examples.

4.1 ODE example

The first example investigated is the ODE example illustrated in Section 3. For solving this problem, a FC-DNN is constructed. The network has 4 hidden layers, and each hidden layer has 50 neurons. The network has 2 inputs: the temporal variable t and random variable Z. To automatically satisfy the initial condition, the approach in [25] is adopted here. The PINN output is modified as follows:

$$\hat{u}(t, Z) = t \cdot \mathbb{N}(t, Z) + 1.0 \tag{21}$$

where $\mathbb{N}(t, Z)$ is the direct output of the FC-DNN, and $\hat{u}(t, Z)$ is the solution estimated by the PINN. Thus, only one loss term is needed for this problem, and it is presented as follows:

$$L(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{u}\left(t^i, Z^i\right)_t + Z\hat{u}\left(t^i, Z^i\right)\right|^2 \tag{22}$$

To train the network using PINN-RA, the collocation points $\left\{\left(t^i, Z^i\right)\right\}_{i=1}^{30,000}$ are generated using LHS. For AT-PINN-RA, the realizations of the temporal variable $\left\{t^i\right\}_{i=1}^{30,000}$ and candidate training samples for random variable $\left\{Z_C^i\right\}_{i=1}^{30,000}$ as well as 10 anchor points $\left\{Z_a^i\right\}_{i=1}^{10}$ are generated. The Adam optimizer is run until the stopping criterion is met or it reaches 5,000 epochs for both methods.

From the experiments, it is found that the proposed AT sampling in AT-PINN-RA is quite effective. The evolution of the collocation points in a randomly selected trial of AT-PINN-RA can be found in Fig 3. The figure shows the collocation points in the first batch of the corresponding epoch. A clear pattern can be observed where the training focus has been gradually shifted to the critical value of Z. Near the end of the training (Epoch 400), almost all the sampled collocation points are located in the vicinity of the critical value. On the other hand, for PINN-RA, the collocation points are evenly scattered in the domain throughout the entire training process, without the training focus set on the important regions. Thus, the model in PINN-RA cannot be trained as effectively as the one in AT-PINN-RA.

The results by both methods are presented in Table 1. The average error of AT-PINN-RA is only 1.00%, while that for PINN-RA is 3.75%. Moreover, AT-PINN-RA takes 450 epochs to converge on average. on the other hand, PINN-RA, without the adaptive training focus, takes 3,544 epochs to converge. The boxplot of the results is presented in Fig 4. Fig 5 presents the comparison of the error evolution of the two methods with the same pool for initial collocation points. It can be observed that AT-PINN-RA converges significantly faster to the actual failure probability compared to PINN-RA.

A parametric study has been performed for the number of anchor points for this example. Fig 6 shows the results for different number of anchor points over 20 repeated trials. It can be observed that the performance will not be significantly different after 10 anchor points.

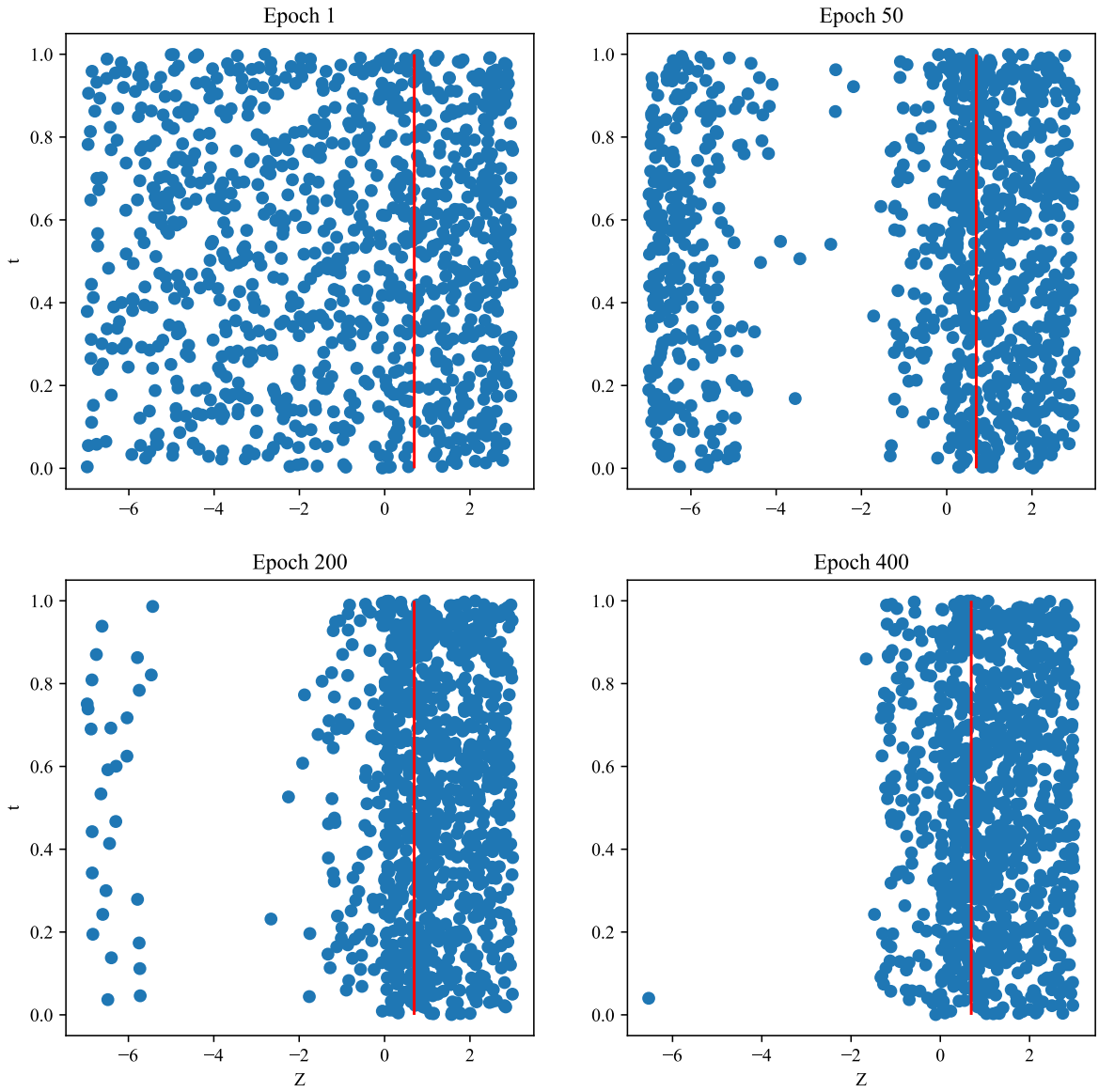

Fig 3. The evolution of the collocation points in AT-PINN-RA (The red line indicates the critical value of Z)

Table 1 Reliability results for Example 1

Method	Epoch number(Standard deviation)	P_f (Standard deviation)	Average error (Standard deviation)
MCS	-	3.54×10^{-3}	-
PINN-RA	3544(1039)	3.57×10^{-3} (3.36×10^{-4})	3.75% (2.23%)
AT-PINN-RA	450(92)	3.65×10^{-3} (1.89×10^{-4})	1.00% (1.20%)

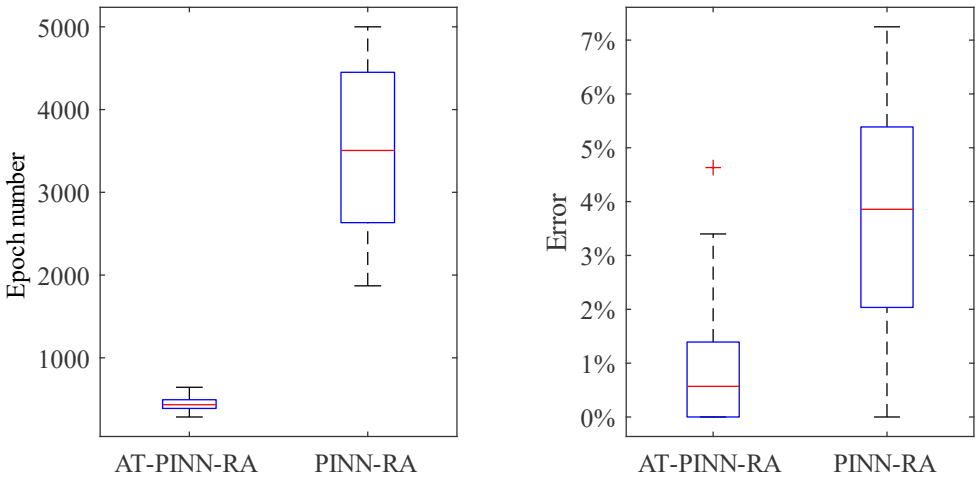

Fig 4. The comparison of the two methods for Example 1

381

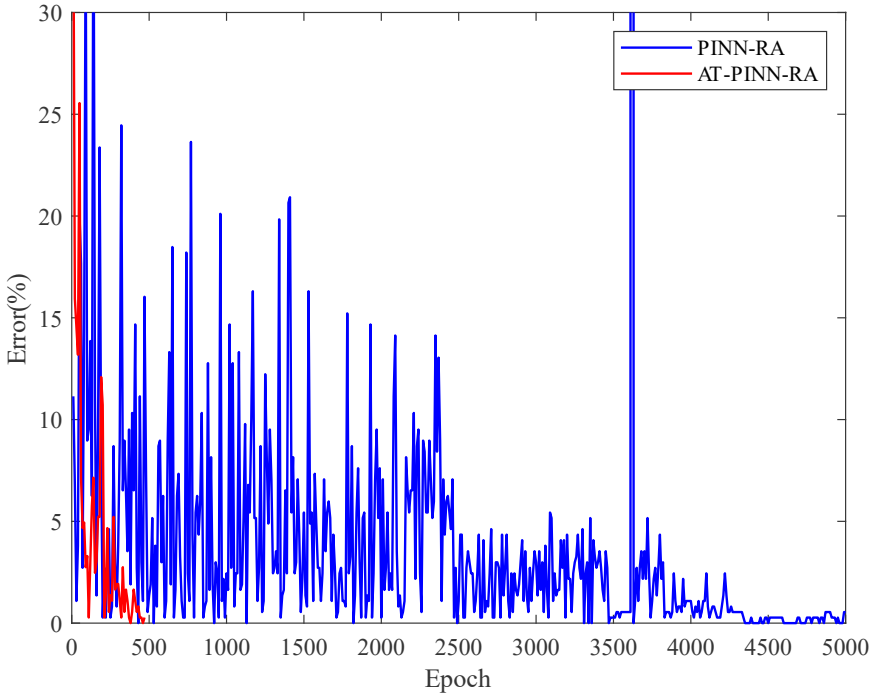

Fig 5. The comparison of error evolution by the two methods for Example 1

382
383

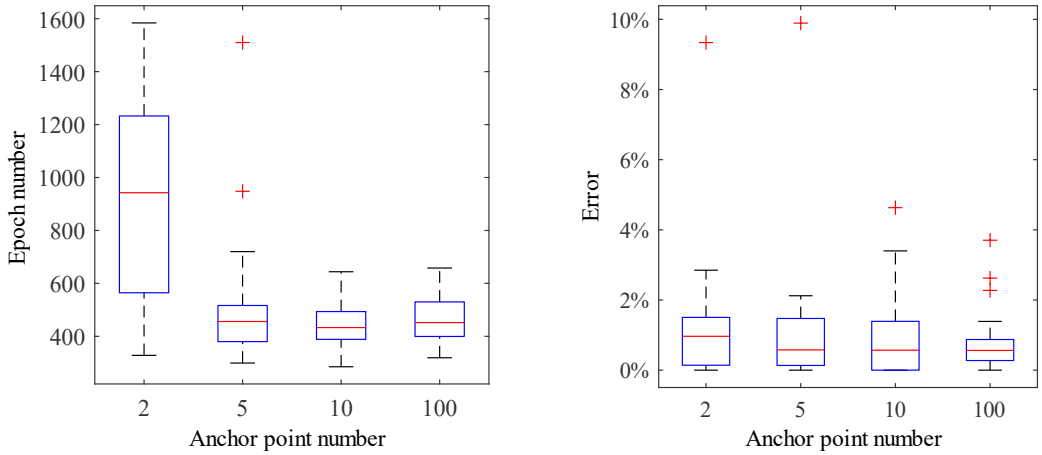

Fig 6. Efficiency and accuracy for different numbers of anchor points

The effect of the max stagnation for the stopping criterion is investigated for this example. Fig 6 shows the results for different stopping criteria over 20 repeated trials. It is observed that when the max stagnation is set to be larger, the average epoch number for the method to converge increases and the average error decreases.

A parametric study on the size of collocation points has also been performed. The results for different sizes of anchor points over 20 repeated trials are presented in Fig 7. Note that when the size of collocation points changes, the number of gradient updates in one epoch will also change accordingly. Thus, the numbers of gradient updates are listed here instead of the numbers of epochs, as they are better representative of the running time. The results show that with a larger size of the collocation points the error tends to be lower. The number of gradient updates for the size of 30,000, which is the one used in this study, is lower than both that of 20,000 and 40,000. Note that these results are affected by the fact that the stopping criterion is changing when the size of collocation points changes. Although the max stagnation of 100 epochs is set for all cases of different collocation point sizes, the numbers of gradient updates for the max stagnation are different. When the size of the collocation points increases, the stopping criterion is more difficult to satisfy.

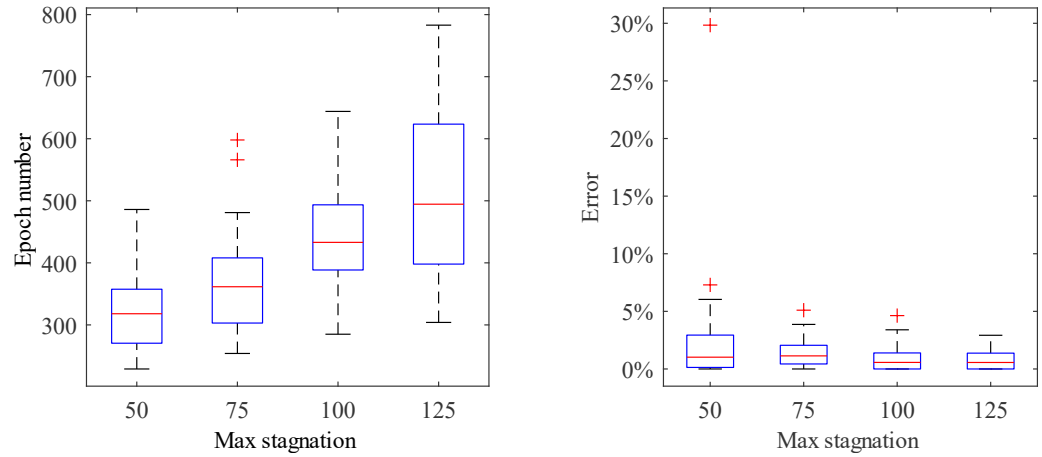

Fig 7. Efficiency and accuracy for different numbers for max stagnation

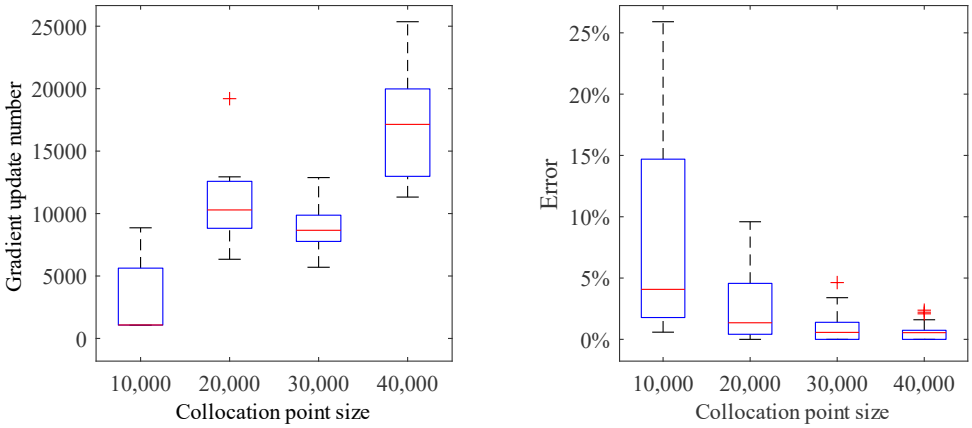

Fig 8. Efficiency and accuracy for different sizes of collocation points

4.2 1D Burgers' equation

A viscous Burgers' equation is considered as the second example to show the performance of the proposed method. The example was also studied in [25], [35]. The governing PDE of the viscous Burgers' equation is presented as follows:

$$u_t + uu_x = \nu u_{xx}, x \in [-1,1], t \in [0,10] \tag{23}$$

where u_x and u_{xx} are the first and second order partial derivatives of u with respect to x, respectively. ν is the viscosity of the system. The system is subjected to the boundary conditions as follows:

$$u(x=-1) = 1 + \delta, u(x=1) = -1 \tag{24}$$

where δ is a random variable that represents a small perturbation that is applied to the boundary at $x = -1$, and it follows a uniform distribution $U(0,0.1)$. The initial condition, which is a linear interpolation of the boundary conditions, can be accordingly expressed as follows:

$$u(x, t=0) = -1 + (1-x)(1 + \frac{\delta}{2}) \tag{25}$$

The solution to this PDE has a transition layer at distance z such that $u(z) = 0$. This transition layer is highly sensitive to the perturbation δ [41]. The limit state function for this problem is defined as:

$$g(Z,t) = -z(\delta) + z_0 \tag{26}$$

where $z(\delta)$ is the distance for the transition layer expressed as a function of the perturbation. For this problem, a FC-DNN with 4 hidden layers and 50 neurons in each layer is constructed. The network has three inputs: the spatial variable x, temporal variable t and random variable δ. The direct output of FC-DNN $\hat{u}(x,t,\delta)$ is used as the estimation of the solution by PINN. The loss function for the PINN can be formulated as follows:

$$L(\boldsymbol{\theta}) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left| \hat{u}\left(x^i, t^i, \delta^i\right)_t + \hat{u}\left(x^i, t^i, \delta^i\right)\hat{u}\left(x^i, t^i, \delta^i\right)_x - \nu \hat{u}\left(x^i, t^i, \delta^i\right)_{xx} \right|^2 + \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} \left| \hat{u}\left(x_{IC}^i, t_{IC}^i, \delta^i\right) - u_{IC}(x_{IC}^i, t_{IC}^i, \delta^i) \right|^2 + \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left| \hat{u}\left(x_{BC}^i, t_{BC}^i, \delta^i\right) - u_{BC}(x_{BC}^i, t_{BC}^i, \delta^i) \right|^2 \quad (27)$$

where $u_{IC}(x_{IC}^i, t_{IC}^i, \delta^i)$ are the initial conditions of the system, and $u_{BC}(x_{BC}^i, t_{BC}^i, \delta^i)$ are the boundary conditions of the system.

To train the network using PINN-RA, the collocation points $\left\{\left(x^i, t^i, \delta^i\right)\right\}_{i=1}^{30,000}$ are generated using LHS for the PDE residual, 1,000 collocation points $\left\{\left(x_{IC}^i, t_{IC}^i, \delta_{IC}^i\right)\right\}_{i=1}^{1,000}$ for the initial condition loss term and 1,000 collocation points $\left\{\left(x_{BC}^i, t_{BC}^i, \delta_{BC}^i\right)\right\}_{i=1}^{1,000}$ for the boundary condition loss term. For AT-PINN-RA, the realizations of spatial and temporal variables $\left\{\left(x^i, t^i\right)\right\}_{i=1}^{30,000}$ and candidate training samples for random variables $\left\{\delta_C^i\right\}_{i=1}^{30,000}$ are generated for the PDE residual. Moreover, 1,000 pairs of spatial and temporal variables $\left\{\left(x_{IC}^i, t_{IC}^i\right)\right\}_{i=1}^{1,000}$ for the initial condition loss term and 1,000 pairs of spatial and temporal variables $\left\{\left(x_{BC}^i, t_{BC}^i\right)\right\}_{i=1}^{1,000}$ for the boundary condition loss term, as well as 10 anchor points $\left\{\delta_a^i\right\}_{i=1}^{10}$ are generated for the random variable δ. In each batch, N_{PDE}, N_{IC} and N_{BC} are all set as 1,000. Note that in AT-PINN-RA, the 1,000 random variable samples generated using AT are shared among the PDE residual, initial condition loss term and boundary condition loss term. The Adam optimizer is run until the stopping criterion is met or it reaches 5,000 epochs for both methods.

In this example, the actual critical value of δ is 0.896. The proposed AT sampling in AT-PINN-RA is able to locate the vicinity of the critical value and shift the training focus accordingly. The evolution of the NWF values of the anchor points in a randomly selected trial of AT-PINN-RA can be found in Fig 9. It can be observed in the figure that before the training process (Epoch 0), there are no significant differences among the NWF values for different groups. Over the training process, the NWF value for the anchor point that is closest to the critical value gradually increases and at the end of the training process, the NWF value for that anchor point becomes the largest among all the NWF values. The training focus has been successfully shifted to the actual critical value. On the other hand, for PINN-RA, the training focus is evenly distributed without considering the importance of the region.

The results by both methods are presented in Table 2. The average error by AT-PINN-RA is about 34% of the error by PINN-RA. The boxplot in Fig 10 also indicates that AT-PINN-RA is able to generate a more robust prediction with much fewer epochs compared to PINN-RA. In over half of the 20 repeated trials, the training for PINN-RA stops due to the maximum epoch instead of the stopping criterion.

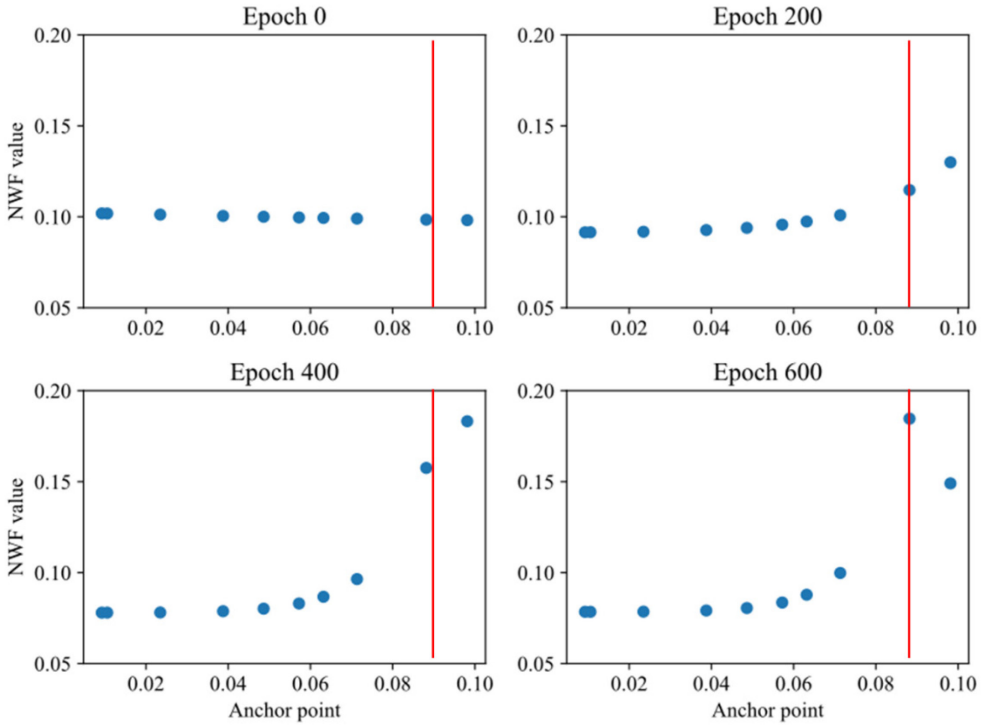

Fig 9. The evolution of the NWF values of the anchor points in AT-PINN-RA (The red line indicates the critical value of δ)

Table 2 Reliability results for Example 2

Method	Epoch number(Standard deviation)	P_f (Standard deviation)	Average error (Standard deviation)
MCS	-	1.037×10^{-1}	-
PINN-RA	4123(1531)	$1.02\times10^{-1}(8.79\times10^{-3})$	5.29% (6.84%)
AT-PINN-RA	1926(235)	1.04×10^{-1} (2.26×10^{-3})	1.80% (1.28%)

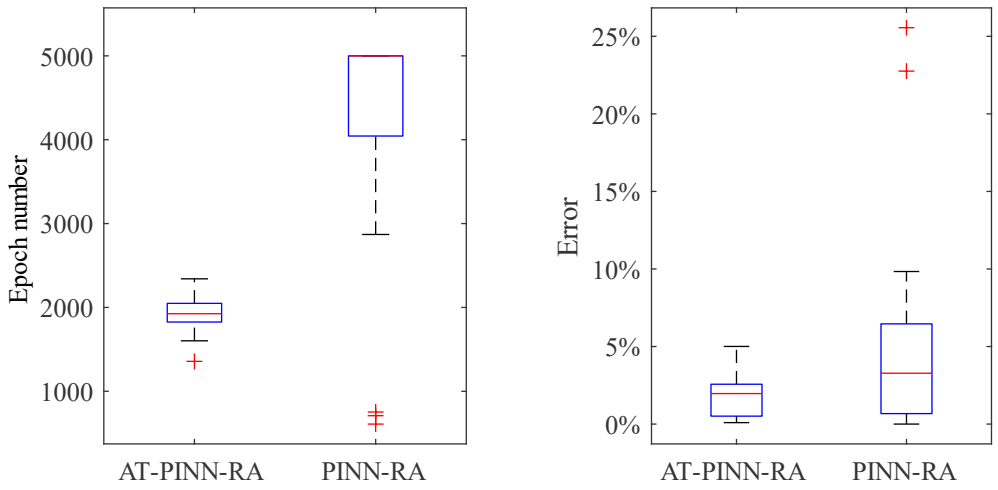

Fig 10. The comparison of the two methods for Example 2

4.3 2D isotropic elasticity problem

The third example considered in this study investigates the displacement of a 2D isotropic elastic structure. The model is described by two coupled PDEs in 2D. Nabian et al. [42] also investigated the application of PINN to the same governing equations and structure. In this problem, a 2D isotropic elastic plate with irregular geometry, which is depicted in Fig 11, is subjected to stochastic external forces. The displacement of a location of interest in this plate should not exceed a predefined critical value. Such problems may emerge in fluid-structure interactions [43]. The objective of this example is to estimate the probability of the displacement of interest exceeding the critical value.

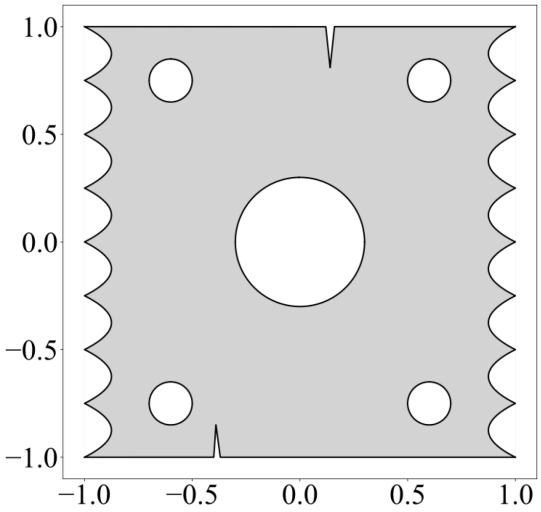

Fig 11. The 2D plate

The governing equation for the structure can be expressed as follows:

$$\begin{aligned}(\lambda+\mu)\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f_x=0\\(\lambda+\mu)\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+f_y=0\end{aligned} \tag{28}$$

where u and v are the displacements along the x and y axes, respectively, μ and λ are constants, and f_x and f_y are the external body forces along the x and y axes, respectively. The constants are defined as follows:

$$\begin{aligned}\lambda&=\frac{\nu E}{(1+\nu)(1-\nu)}\\\mu&=\frac{E}{2(1+\nu)}\end{aligned} \tag{29}$$

where ν and E are the Poisson's ratio and Young's modulus of the structure, respectively. In this study, they are set as 0.2 and 0.25, respectively. Using a similar approach to Nabian et al. [42], we synthesize a governing equation that would generate a prescribed solution. The prescribed structure displacements in our study can be expressed as follows:

$$\begin{aligned}u(x,y,\boldsymbol{Z})&=0\\v(x,y,\boldsymbol{Z})&=sin(\frac{\pi}{2}(x+1))\,[Z_1^3 cos(y-1)-Z_2^3 cos\,(y+1)]\end{aligned} \tag{30}$$

where Z_1 and Z_2 are two independent random variables that both follow a normal distribution $N(1,0.1^2)$. Fig 12 represents four realizations of the solution v. The body forces f_x and f_y can be back-calculated using Eq. (28) and (30) and then applied to synthesize the PDEs used in this problem The displacement along the y axis, v, at point (0,1) (the middle point on the top side) is of interest. The limit state function of this examples is defined as follows:

$$g(\boldsymbol{Z}) = v_c - v(0,1,\boldsymbol{Z}) \tag{31}$$

where v_c is the critical displacement and set as 2.5. The system is determined to fail when the displacement $v(0,1,\boldsymbol{Z})$ exceeds the critical value.

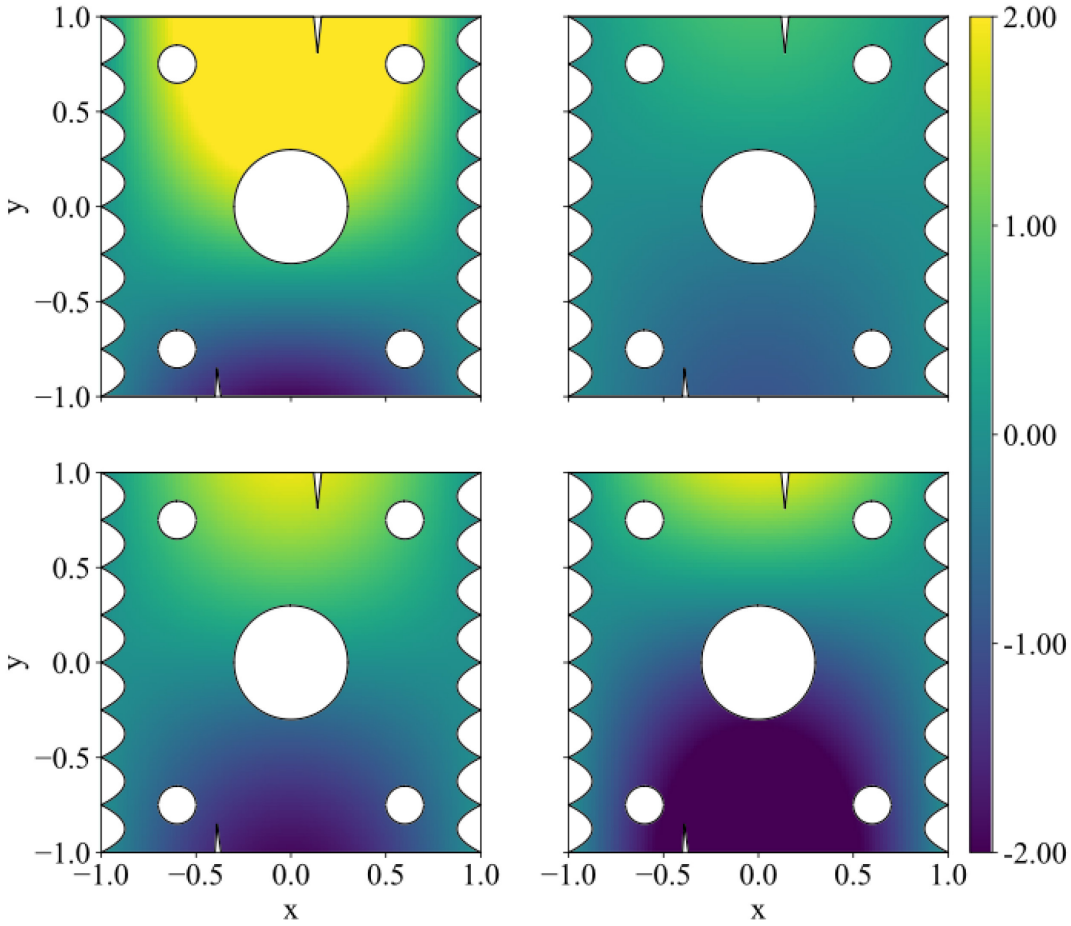

Fig 12. Four realizations of the solution v

For this problem, two FC-DNNs are constructed for u and v, respectively, and each network has 4 hidden layers and 50 neurons in each layer. Both FC-DNNs share the same inputs: the spatial variable x, temporal variable t and random variable Z_1 and Z_2 . The direct outputs of the two FC-DNNs $\hat{u}(x,t,\boldsymbol{Z})$ and $\hat{v}(x,t,\boldsymbol{Z})$ are used as the estimation of the solutions by PINN. The loss function for the PINN can be formulated as follows:

$$
\begin{aligned}
L(\boldsymbol{\theta}) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} & \left| (\lambda+\mu) \frac{\partial}{\partial x}\left(\frac{\partial \hat{u}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial x} + \frac{\partial \hat{v}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial y}\right)\right. \\
& \left. + \mu\left(\frac{\partial^2 \hat{u}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial x^2} + \frac{\partial^2 \hat{u}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial y^2}\right) + f_x\left(x^i, t^i, \boldsymbol{Z}^i\right)\right|^2 \\
& + \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} \left| (\lambda+\mu) \frac{\partial}{\partial y}\left(\frac{\partial \hat{u}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial x} + \frac{\partial \hat{v}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial y}\right)\right. \\
& \left. + \mu\left(\frac{\partial^2 \hat{v}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial x^2} + \frac{\partial^2 \hat{v}\left(x^i, t^i, \boldsymbol{Z}^i\right)}{\partial y^2}\right) + f_y\left(x^i, t^i, \boldsymbol{Z}^i\right)\right|^2 \\
& + \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left|\hat{u}\left(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i\right) - u_{BC}\left(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i\right)\right|^2 \\
& + \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} \left|\hat{v}\left(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i\right) - v_{BC}\left(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i\right)\right|^2
\end{aligned} \tag{32}
$$

where $u_{BC}(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i)$ and $v_{BC}(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{BC}^i)$ are the boundary conditions of the system. These values can be obtained directly using Eq. (30).

To train the network using PINN-RA, the collocation points $\left\{\left(x^i, t^i, \boldsymbol{Z}^i\right)\right\}_{i=1}^{30,000}$ are generated using LHS for the PDE residual, and 1,000 collocation points $\left\{\left(x_{BC}^i, t_{BC}^i, \boldsymbol{Z}_{IC}^i\right)\right\}_{i=1}^{1,000}$ for the boundary condition loss term. For AT-PINN-RA, the realizations of the spatial and temporal variables $\left\{\left(x^i, t^i\right)\right\}_{i=1}^{30,000}$ and candidate training samples for random variable $\left\{\boldsymbol{Z}_C^i\right\}_{i=1}^{30,000}$ are generated for the PDE residual. Moreover, 1,000 pairs of spatial and temporal variables $\left\{\left(x_{BC}^i, t_{BC}^i\right)\right\}_{i=1}^{1,000}$ for the boundary condition loss term as well as 100 anchor points $\left\{\boldsymbol{Z}_a^i\right\}_{i=1}^{100}$ are generated for the random variable vector $\boldsymbol{Z}$. In each batch, N_{PDE} and N_{BC} are both set as 1,000. Note that in AT-PINN-RA, the 1,000 random variable samples generated using AT are shared between the PDE residual and boundary condition loss term. The Adam optimizer is run until the stopping criterion is met or it reaches 5,000 epochs for both methods.

Fig 13 shows two realizations of the solutions for v by AT-PINN-RA, the real solutions and the L_1 errors of the AT-PINN-RA solutions. It can be observed that the PINN model can predict the solution with high accuracy, especially in the region near the point of interest.

In this example, the proposed AT sampling in AT-PINN-RA is able to identify the limit state and shift the focus of training to the region around the limit state. Fig 14 shows the anchor points and the sampled random variable points in the last epoch of the training for AT-PINN-RA. It can be observed that the density of the points around the limit state is higher than other regions. In addition, it can be seen that the estimated limit state by AT-PINN-RA is quite accurate.

The comparisons of PINN-RA and AT-PINN-RA are presented in Table 3. AT-PINN-RA achieves an average error that is about 56% of the one of PINN-RA. Moreover, it takes 1235 epochs on average for AT-PINN-RA to converge, while AT-PINN-RA reaches the predefined maximum epoch number 5,000 in most cases. The boxplot in Fig 15 also indicates that AT-PINN-RA is more accurate and more efficient than PINN-RA.

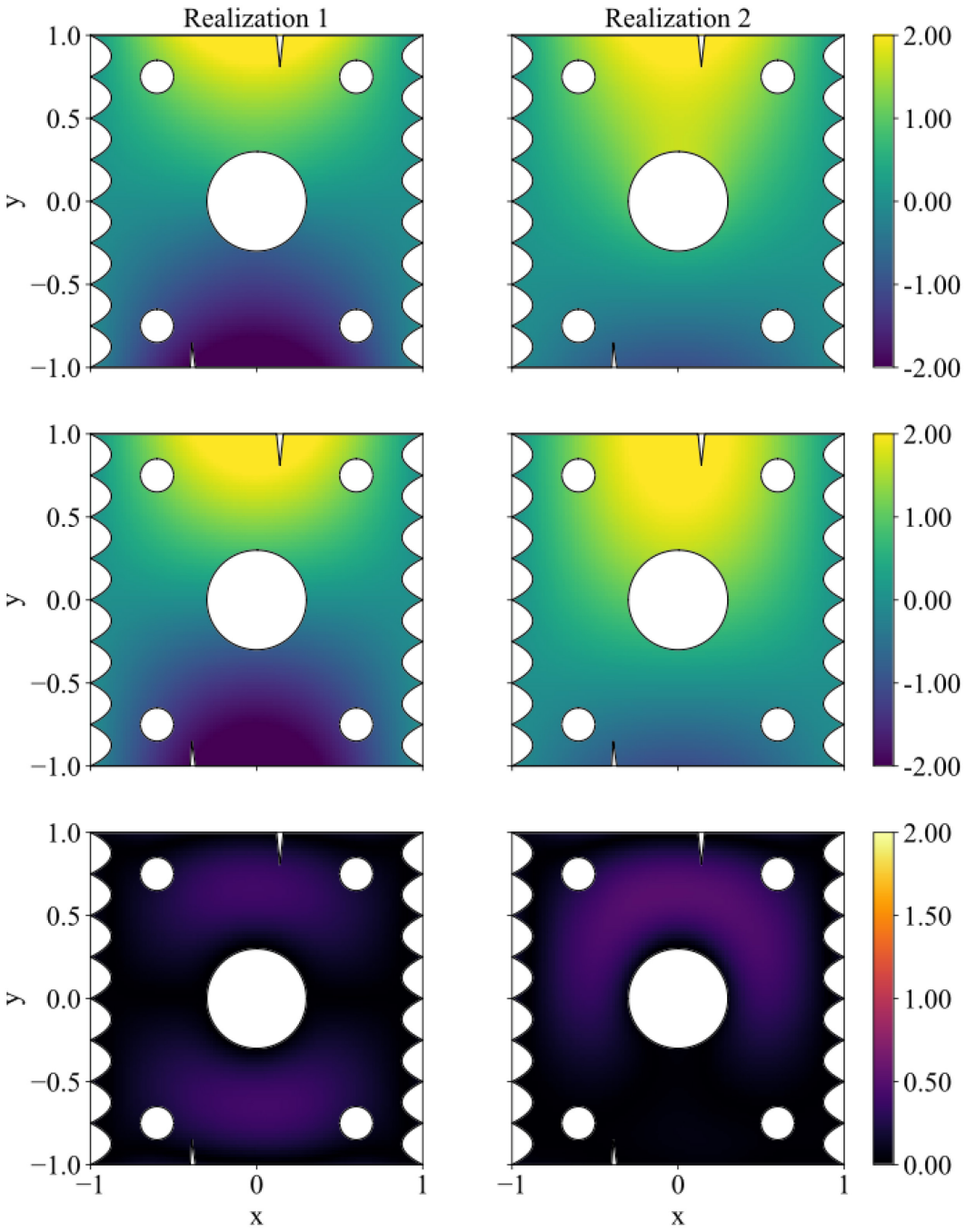

Fig 13. The solutions to v and errors of two randomly selected realizations of the random variables. (First row) The v solutions by AT-PINN-RA. (Second row) The real v solutions. (Third row) L_1 errors of the AT-PINN-RA solutions

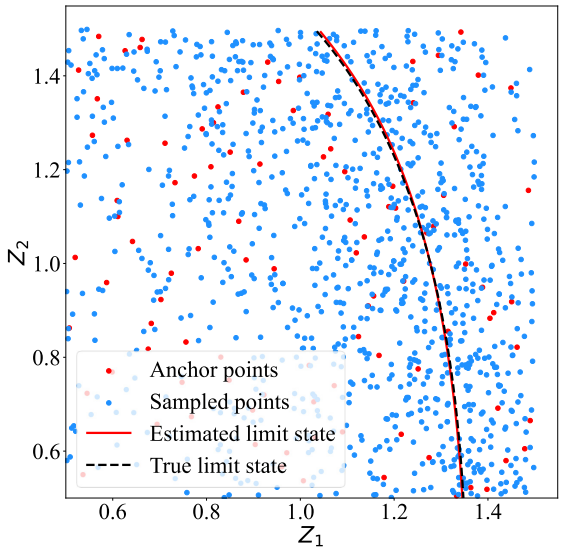

Fig 14. The random variable points in the collocation points in AT-PINN-RA

Table 3 Reliability results for Example 3

Method	Epoch number(Standard deviation)	P_f (Standard deviation)	Average error (Standard deviation)
MCS	-	4.19×10^{-3}	-
PINN-RA	4491(1213)	4.38×10^{-3} (1.63×10^{-4})	4.68% (2.13%)
AT-PINN-RA	1235(229)	4.27×10^{-3} (1.48×10^{-4})	2.64% (1.82%)

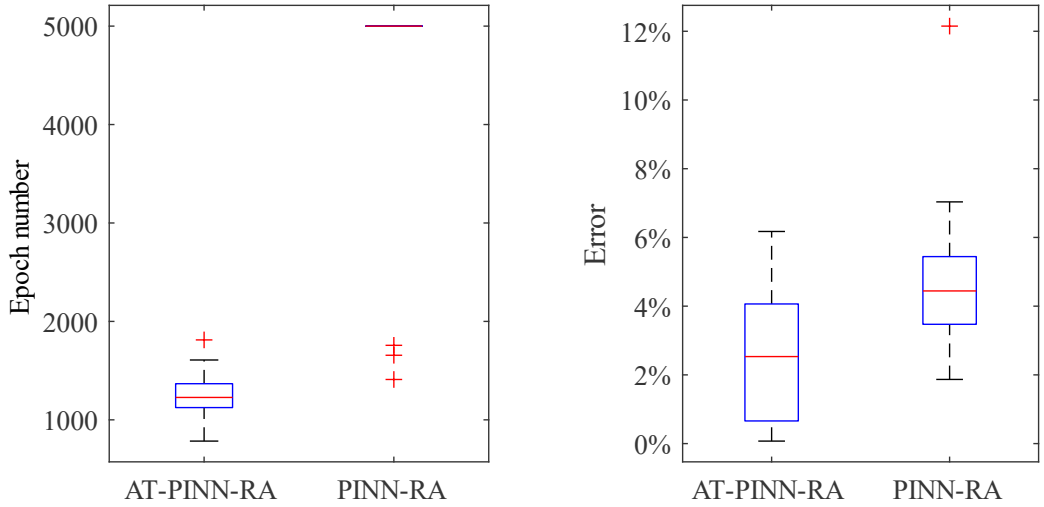

Fig 15. The comparison of the two methods for Example 3

4.4 Cell-signaling cascading

The last example in this study is based on a mathematical model of the autocrine cell signaling cascade. This model was first proposed in [44]. The model describes signal cascading entailing three enzymes in an autocrine loop with positive feedback. The dimensionless concentrations are scaled by the total amount of the enzyme. These concentrations for the active ("phosphorylated") form of the enzymes are investigated. The model was converted into a benchmark reliability analysis problem in [35] and was also studied in [25].

The uncertainties in maximal reaction velocities will influence the dimensionless concentrations. In this problem, the concentration of an enzyme should be above a certain threshold value to speed up the reaction. The governing differential equations for the considered system are presented as follows:

$$\begin{aligned}\frac{\mathrm{d}e_{1p}}{\mathrm{d}t} &= \frac{I(t)}{1+G_4 e_{3p}}\frac{V_{max,1}(1-e_{1p})}{K_{m,1}+(1-e_{1p})} - \frac{V_{max,2}e_{1p}}{K_{m,2}+e_{1p}} \\ \frac{\mathrm{d}e_{2p}}{\mathrm{d}t} &= \frac{V_{max,3}e_{1p}\left(1-e_{2p}\right)}{K_{m,3}+\left(1-e_{2p}\right)} - \frac{V_{max,4}e_{2p}}{K_{m,4}+e_{2p}} \\ \frac{\mathrm{d}e_{3p}}{\mathrm{d}t} &= \frac{V_{max,5}e_{2p}\left(1-e_{3p}\right)}{K_{m,5}+\left(1-e_{3p}\right)} - \frac{V_{max,6}e_{3p}}{K_{m,6}+e_{3p}}\end{aligned} \tag{33}$$

where e_{1p}, e_{2p} and e_{3p} the concentrations of the active form of enzymes, which are the solutions to the equations, $G_4 = 0$, $I(t) = 1$, $K_{m,i} = 0.2, \forall i$, and $V_{max,i}, i = 1,2,\dots,6$ are the random variables considered in the system, which can be expressed as follows:

$$V_{max,i} = \langle V\rangle_{max,i}(1+\sigma Z_i), i = 1,2,\dots,6 \tag{34}$$

where $\langle V\rangle_{max,1} = 0.5$, $\langle V\rangle_{max,2} = 0.15$, $\langle V\rangle_{max,3} = 0.15$, $\langle V\rangle_{max,4} = 0.15$, $\langle V\rangle_{max,5} = 0.25$, $\langle V\rangle_{max,6} = 0.05$, $\sigma = 0.1$ and $Z_i, i = 1,2,\dots,6$ are the independent random variables that all follow uniform distributions $U(-1,1)$.

The system is subjected to the initial condition as follows:

$$e_{1p}(t=0) = 0, e_{2p}(t=0) = 1, e_{3p}(t=0) = 0 \tag{35}$$

The limit state function considered is expressed as follows:

$$g(Z,t) = e_{3p}(\boldsymbol{Z},t) - e_{3p,c} \tag{36}$$

where $e_{3p,c}$ is the threshold value for e_{3p}. When e_{3p} is larger than the critical value, the system is determined to be safe.

For solving this problem, a FC-DNN is constructed. The network has 4 hidden layers, and each hidden layer has 50 neurons. The network has 7 inputs: the temporal variable t and six random variables $V_{max,i}, i = 1,2,\dots,6$. There are 3 outputs for the network, which are denoted as $\mathbb{N}_{e_{1p}}\left(t, V_{max,i}, i = 1,2,\dots,6\right)$, $\mathbb{N}_{e_{2p}}\left(t, V_{max,i}, i = 1,2,\dots,6\right)$ and $\mathbb{N}_{e_{2p}}\left(t, V_{max,i}, i = 1,2,\dots,6\right)$, respectively. To automatically satisfy the initial condition, the approach in [25] is adopted here. The PINN output is modified as follows:

$$\begin{aligned}\hat{e}_{1p} &= t\cdot\mathbb{N}_{e_{1p}} \\ \hat{e}_{2p} &= t\cdot\mathbb{N}_{e_{2p}} + 1.0 \\ \hat{e}_{3p} &= t\cdot\mathbb{N}_{e_{3p}}\end{aligned} \tag{37}$$

where $\hat{e}_{1p}$, $\hat{e}_{2p}$ and $\hat{e}_{3p}$ are the solutions estimated by the PINN. Thus, for each solution, only one loss term is needed. All the loss functions for the three solutions are presented as follows, respectively:

$$
\begin{aligned}
L_{\hat{e}_{1p}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Bigg|&\left(1+G_4\hat{e}_{1p}\right)\left(K_{m,1}+\left(1-\hat{e}_{1p}\right)\right)\left(K_{m,2}+\hat{e}_{1p}\right)\frac{\mathrm{d}\hat{e}_{1p}}{\mathrm{d}t}\\
&-I(t)\left(V_{max,1}(1-\hat{e}_{1p})(K_{m,2}+\hat{e}_{1p})\right)\\
&+\left(V_{max,2}\hat{e}_{1p}\right)(1+G_4\hat{e}_{1p})(K_{m,1}+(1-\hat{e}_{1p}))\Bigg|^2\\
L_{\hat{e}_{2p}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Bigg|&\left(K_{m,3}+\left(1-\hat{e}_{2p}\right)\right)\left(K_{m,4}+\hat{e}_{2p}\right)\frac{\mathrm{d}\hat{e}_{2p}}{\mathrm{d}t}\\
&-\left(V_{max,3}\hat{e}_{1p}(1-\hat{e}_{2p})(K_{m,4}+\hat{e}_{2p})\right)\\
&+\left(V_{max,4}\hat{e}_{2p}\right)(K_{m,3}+(1-\hat{e}_{2p}))\Bigg|^2\\
L_{\hat{e}_{3p}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Bigg|&\left(K_{m,5}+\left(1-\hat{e}_{3p}\right)\right)\left(K_{m,6}+\hat{e}_{3p}\right)\frac{\mathrm{d}\hat{e}_{3p}}{\mathrm{d}t}\\
&-\left(V_{max,5}\hat{e}_{2p}(1-\hat{e}_{3p})(K_{m,6}+\hat{e}_{3p})\right)\\
&+\left(V_{max,6}\hat{e}_{3p}\right)(K_{m,5}+(1-\hat{e}_{3p}))\Bigg|^2
\end{aligned}
\tag{38}
$$

To train the network using PINN-RA, the collocation points $\left\{\left(t^i, V^i_{max,k}, k=1,2,\dots,6\right)\right\}_{i=1}^{30,000}$ are generated using LHS. For AT-PINN-RA, the realizations of temporal variables $\left\{t^i\right\}_{i=1}^{30,000}$ and candidate training samples for random variables $\left\{V^i_{max,k(C)}, k=1,2,\dots,6\right\}_{i=1}^{30,000}$ are generated. Moreover, 1000 anchor points $\left\{\left(V_{max,k}{}^{i}_{(a)}, k=1,2,\dots,6\right)\right\}_{i=1}^{1000}$ are generated for the random variables due to the high dimensions of this problem. It is found that adding more anchor points will not increase the accuracy significantly. The batch-size N is set as 1000 for both PINN-RA and AT-PINN-RA. The Adam optimizer is run until the stopping criterion is met or it reaches 5,000 epochs for both methods.

In this example, AT-PINN-RA outperforms PINN-RA in terms of both efficiency and accuracy. With a fewer epoch number on average, AT-PINN-RA achieves an average error of 1.88%, which is smaller than 3.03% for PINN-RA. The boxplot in Fig 16 also provides a straightforward comparison of the two methods which shows that AT-PINN-RA is more robust than PINN-RA by a great margin.

Table 4 Reliability results for Example 3

Method	Epoch number(Standard deviation)	$\boldsymbol{P_f}$ (Standard deviation)	Average error (Standard deviation)
MCS	-	4.59×10^{-2}	-
PINN-RA	2286(925)	4.55×10^{-2} (2.04×10^{-3})	3.03% (3.30%)
AT-PINN-RA	1929(540)	4.56×10^{-2} (1.11×10^{-3})	1.88% (1.57%)

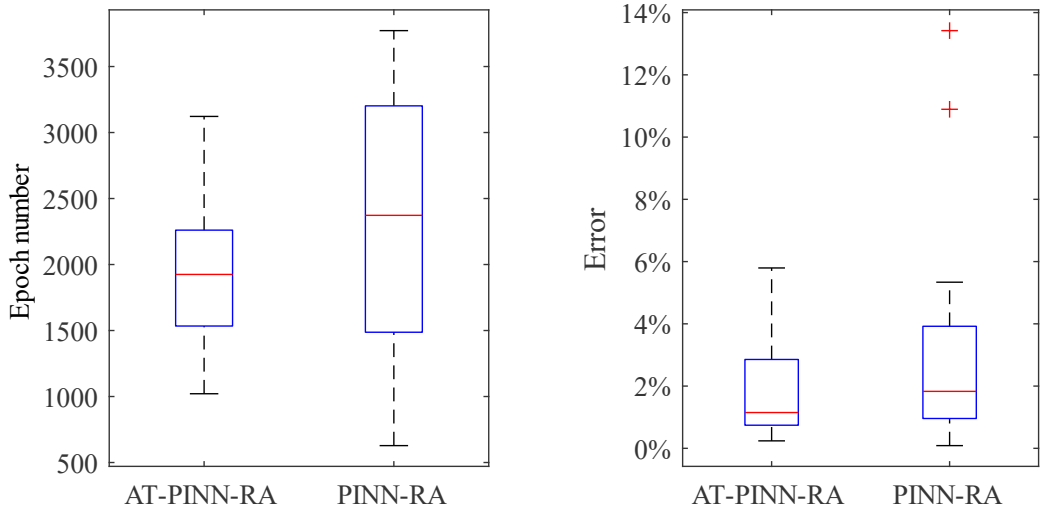

Fig 16. The comparison of two methods for Example 4

5. Conclusions

In this paper, we have proposed a novel method for reliability analysis, which is called adaptively trained Physics-Informed Neural Network for reliability analysis (AT-PINN-RA). The advantages of the proposed method are two-folds: first, the proposed method is simulation free, and the computational costs are only from training the neural network. Second, a novel active learning approach called adaptive training (AT) sampling is proposed to boost the convergence rate and accuracy simultaneously. The proposed AT sampling approach adaptively shifts the training focus to the important regions where the limit state may occur in the entire domain. Anchor points and the corresponding probability mass function are introduced to boost the efficiency of the sampling process. During the training of the surrogate model, more training points are sampled in the vicinity of the limit state, which can help the model converge faster and be more accurate in the regions of interest. The performance of the AT-PINN-RA is demonstrated using four reliability problems. In all the examples, AT-PINN-RA shows significantly better performance than the existing approach, PINN-RA. AT-PINN-RA is able to make accurate predictions in all numerical examples, as the AT sampling can successfully identify the important regions. On the other hand, without the adaptive training focus, PINN-RA cannot always generate accurate results in all examples. The proposed approach facilities quantification of uncertainties and in particular analyzing the probability of events of interest. As the PINN is gaining more attention in solving complex PDEs, there is a growing opportunity to solve reliability analysis problems entailing fundamental governing equations in many engineering problems and physics phenomena using the proposed method. With the increasing complexity of problems, the simulation costs of solving reliability analysis problems can become prohibitive. In this context, the proposed method, as a simulation free method, can have more potential than the traditional methods. In addition, the advancements in the original PINN can be easily incorporated in the proposed method to improve the capability, efficiency and accuracy of the method.

Acknowledgements

This research has been partly funded by the U.S. National Science Foundation (NSF) through awards CMMI-2000156. In addition, this work is supported in part by Lichtenstein endowment at The Ohio State University. These supports are greatly appreciated.

References

[1] O. Ditlevsen and H. O. Madsen, *Structural reliability methods*, vol. 178. Wiley New York, 1996. Accessed: May 12, 2017. [Online]. Available: http://chodor-projekt.net/wp-

content/uploads/BiNSK/Literatura/Dilevsen,Madsen,%20Structural%20Reliability%20Methods%20(2007).pdf
[2] M. Lemaire, *Structural reliability*. John Wiley & Sons, 2013. Accessed: May 12, 2017. [Online]. Available: https://books.google.com/books?hl=zh-CN&lr=&id=oC4MDigl9PsC&oi=fnd&pg=PT8&dq=Lemaire+M.+Structural+reliability.+ISTE+Wiley%3B&ots=gxRZo1u21a&sig=mvKN3vgtKAxvkm_3FF9XpsdedhA
[3] V. J. Romero, L. P. Swiler, and A. A. Giunta, "Construction of response surfaces based on progressive-lattice-sampling experimental designs with application to uncertainty propagation," *Struct. Saf.*, vol. 26, no. 2, pp. 201–219, 2004.
[4] A. A. Giunta, J. M. McFarland, L. P. Swiler, and M. S. Eldred, "The promise and peril of uncertainty quantification using response surface approximations," *Struct. Infrastruct. Eng.*, vol. 2, no. 3–4, pp. 175–189, 2006.
[5] W. Zhao, F. Fan, and W. Wang, "Non-linear partial least squares response surface method for structural reliability analysis," *Reliab. Eng. Syst. Saf.*, vol. 161, pp. 69–77, May 2017, doi: 10.1016/j.ress.2017.01.004.
[6] G. Blatman and B. Sudret, "An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis," *Probabilistic Eng. Mech.*, vol. 25, no. 2, pp. 183–197, 2010.
[7] J.-M. Bourinet, "Rare-event probability estimation with adaptive support vector regression surrogates," *Reliab. Eng. Syst. Saf.*, vol. 150, pp. 210–221, 2016.
[8] H. Dai, H. Zhang, W. Wang, and G. Xue, "Structural reliability assessment by local approximation of limit state functions using adaptive Markov chain simulation and support vector regression," *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 27, no. 9, pp. 676–686, 2012.
[9] A. Roy and S. Chakraborty, "Reliability analysis of structures by a three-stage sequential sampling based adaptive support vector regression model," *Reliab. Eng. Syst. Saf.*, vol. 219, p. 108260, Mar. 2022, doi: 10.1016/j.ress.2021.108260.
[10] B. Echard, N. Gayton, and M. Lemaire, "AK-MCS: an active learning reliability method combining Kriging and Monte Carlo simulation," *Struct. Saf.*, vol. 33, no. 2, pp. 145–154, 2011.
[11] Z. Wang and A. Shafieezadeh, "REAK: Reliability analysis through Error rate-based Adaptive Kriging," *Reliab. Eng. Syst. Saf.*, vol. 182, pp. 33–45, Feb. 2019, doi: 10.1016/j.ress.2018.10.004.
[12] C. Zhang, Z. Wang, and A. Shafieezadeh, "Error Quantification and Control for Adaptive Kriging-Based Reliability Updating with Equality Information," *Reliab. Eng. Syst. Saf.*, vol. 207, p. 107323, Mar. 2021, doi: 10.1016/j.ress.2020.107323.
[13] Z. Wang and A. Shafieezadeh, "Real-time high-fidelity reliability updating with equality information using adaptive Kriging," *Reliab. Eng. Syst. Saf.*, vol. 195, p. 106735, Mar. 2020, doi: 10.1016/j.ress.2019.106735.
[14] Z. Wang and A. Shafieezadeh, "On confidence intervals for failure probability estimates in Kriging-based reliability analysis," *Reliab. Eng. Syst. Saf.*, vol. 196, p. 106758, Apr. 2020, doi: 10.1016/j.ress.2019.106758.
[15] Z. Xu and J. H. Saleh, "Machine learning for reliability engineering and safety applications: Review of current status and future opportunities," *Reliab. Eng. Syst. Saf.*, vol. 211, p. 107530, Jul. 2021, doi: 10.1016/j.ress.2021.107530.
[16] S. Saraygord Afshari, F. Enayatollahi, X. Xu, and X. Liang, "Machine learning-based methods in structural reliability analysis: A review," *Reliab. Eng. Syst. Saf.*, vol. 219, p. 108223, Mar. 2022, doi: 10.1016/j.ress.2021.108223.
[17] Z. Xiang, J. Chen, Y. Bao, and H. Li, "An active learning method combining deep neural network and weighted sampling for structural reliability analysis," *Mech. Syst. Signal Process.*, vol. 140, p. 106684, Jun. 2020, doi: 10.1016/j.ymssp.2020.106684.
[18] Y. Bao, Z. Xiang, and H. Li, "Adaptive subset searching-based deep neural network method for structural reliability analysis," *Reliab. Eng. Syst. Saf.*, vol. 213, p. 107778, Sep. 2021, doi: 10.1016/j.ress.2021.107778.

[19] A. J. Meade and A. A. Fernandez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," *Math. Comput. Model.*, vol. 19, no. 12, pp. 1–25, Jun. 1994, doi: 10.1016/0895-7177(94)90095-7.
[20] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998, doi: 10.1109/72.712178.
[21] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1041–1049, Sep. 2000, doi: 10.1109/72.870037.
[22] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019, doi: 10.1016/j.jcp.2018.10.045.
[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989, doi: 10.1016/0893-6080(89)90020-8.
[24] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *ArXiv150205767 Cs Stat*, Feb. 2018, Accessed: Jul. 29, 2021. [Online]. Available: http://arxiv.org/abs/1502.05767
[25] S. Chakraborty, "Simulation free reliability analysis: A physics-informed deep learning based approach," May 2020, Accessed: Nov. 21, 2021. [Online]. Available: https://arxiv.org/abs/2005.01302v3
[26] C. F. Higham and D. J. Higham, "Deep Learning: An Introduction for Applied Mathematicians," *ArXiv180105894 Cs Math Stat*, Jan. 2018, Accessed: Jul. 30, 2021. [Online]. Available: http://arxiv.org/abs/1801.05894
[27] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *ArXiv171005941 Cs*, Oct. 2017, Accessed: Aug. 01, 2021. [Online]. Available: http://arxiv.org/abs/1710.05941
[28] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," *ArXiv200104536 Cs Math Stat*, Jan. 2020, Accessed: Jul. 30, 2021. [Online]. Available: http://arxiv.org/abs/2001.04536
[29] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective," *ArXiv200714527 Cs Math Stat*, Jul. 2020, Accessed: Jul. 30, 2021. [Online]. Available: http://arxiv.org/abs/2007.14527
[30] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, and J. M. McFarland, "Efficient global reliability analysis for nonlinear implicit performance functions," *AIAA J.*, vol. 46, no. 10, pp. 2459–2468, 2008.
[31] B. Echard, N. Gayton, and M. Lemaire, "AK-MCS: an active learning reliability method combining Kriging and Monte Carlo simulation," *Struct. Saf.*, vol. 33, no. 2, pp. 145–154, 2011.
[32] Z. Lv, Z. Lu, and P. Wang, "A new learning function for Kriging and its applications to solve reliability problems in engineering," *Comput. Math. Appl.*, vol. 70, no. 5, pp. 1182–1197, Sep. 2015, doi: 10.1016/j.camwa.2015.07.004.
[33] Z. Sun, J. Wang, R. Li, and C. Tong, "LIF: A new Kriging based learning function and its application to structural reliability analysis," *Reliab. Eng. Syst. Saf.*, vol. 157, pp. 152–165, 2017.
[34] J. Wang, G. Xu, Y. Li, and A. Kareem, "AKSE: A novel adaptive Kriging method combining sampling region scheme and error-based stopping criterion for structural reliability analysis," *Reliab. Eng. Syst. Saf.*, vol. 219, p. 108214, Mar. 2022, doi: 10.1016/j.ress.2021.108214.
[35] J. Li and D. Xiu, "Evaluation of failure probability via surrogate models," *J. Comput. Phys.*, vol. 229, no. 23, pp. 8966–8980, Nov. 2010, doi: 10.1016/j.jcp.2010.08.022.
[36] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv14126980 Cs*, Jan. 2017, Accessed: Apr. 29, 2020. [Online]. Available: http://arxiv.org/abs/1412.6980

[37] A. Bihlo and R. O. Popovych, “Physics-informed neural networks for the shallow-water equations on the sphere,” *ArXiv210400615 Phys.*, Apr. 2021, Accessed: Jul. 30, 2021. [Online]. Available: http://arxiv.org/abs/2104.00615
[38] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, “A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics,” *Comput. Methods Appl. Mech. Eng.*, vol. 379, p. 113741, Jun. 2021, doi: 10.1016/j.cma.2021.113741.
[39] *MATLAB and Statistics Toolbox Release 2020a*. Natick, Massachusettes, US: The Mathworks Inc., 2020.
[40] M. S. Alnæs *et al.*, “The FEniCS Project Version 1.5,” *Arch. Numer. Softw.*, vol. 3, no. 100, 2015, doi: 10.11588/ans.2015.100.20553.
[41] D. Xiu and G. E. Karniadakis, “Supersensitivity due to uncertain boundary conditions,” *Int. J. Numer. Methods Eng.*, vol. 61, no. 12, pp. 2114–2138, 2004, doi: 10.1002/nme.1152.
[42] M. A. Nabian, R. J. Gladstone, and H. Meidani, “Efficient training of physics-informed neural networks via importance sampling,” *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 36, no. 8, pp. 962–977, 2021, doi: 10.1111/mice.12685.
[43] Y. Chikazawa, S. Koshizuka, and Y. Oka, “A particle method for elastic and visco-plastic structures and fluid-structure interactions,” *Comput. Mech.*, vol. 27, no. 2, pp. 97–106, Feb. 2001, doi: 10.1007/s004660000216.
[44] S. Y. Shvartsman, M. P. Hagan, A. Yacoub, P. Dent, H. S. Wiley, and D. A. Lauffenburger, “Autocrine loops with positive feedback enable context-dependent cell signaling,” *Am. J. Physiol. Cell Physiol.*, vol. 282, no. 3, pp. C545-559, Mar. 2002, doi: 10.1152/ajpcell.00260.2001.