Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations

Kejun Tang[a], Xiaoliang Wan[b], Chao Yang[c,d,*]

[a] Peng Cheng Laboratory, Shenzhen 518052, China
[b] Department of Mathematics and Center for Computation and Technology, Louisiana State University, Baton Rouge 70803, USA
[c] School of Mathematical Sciences, Peking University, Beijing 100871, China
[d] Institute for Computing and Digital Economy, Peking University, Changsha 410205, China

A R T I C L E   I N F O

Article history:
Received 13 July 2022
Received in revised form 3 December 2022
Accepted 16 December 2022
Available online 28 December 2022

Keywords:
Deep learning
Numerical approximation of PDEs
Adaptive sampling
Deep generative models

A B S T R A C T

In this work we propose a deep adaptive sampling (DAS-PINNs) method for solving partial differential equations (PDEs), where deep neural networks are utilized to approximate the solutions of PDEs and deep generative models are employed to generate new collocation points to refine the training set. The overall procedure of DAS consists of two components: solving the PDEs by minimizing the residual loss on the collocation points in the training set and generating a new training set to further improve the accuracy of the current approximate solution. In particular, we treat the residual as a probability density function and approximate it with a deep generative model, called KRnet. The new samples from KRnet are consistent with the distribution induced by the residual, i.e., more samples are located in the region of large residual and less samples are located in the region of small residual. Analogous to classical adaptive methods such as the adaptive finite element, KRnet acts as an error indicator that guides the refinement of the training set. Compared to the neural network approximation obtained with uniformly distributed collocation points, the developed algorithms can significantly improve the accuracy, especially for low regularity and high-dimensional problems. We demonstrate the effectiveness of the proposed DAS-PINNs method with numerical experiments.

1. Introduction

In recent years, solving partial differential equations (PDEs) with deep learning methods has been receiving increasing attention [1–3]. Two major types of deep learning methods have been proposed for solving PDEs, including the variational form subject to deep learning techniques [4–7] and the physics-informed neural networks (PINNs) [8–10,3], both of which reformulate a PDE problem as an optimization problem and train a deep neural network (DNN) to approximate the solution of PDE through minimizing the corresponding loss functional. The variational form is based on the weak formulation of PDEs, while PINN is based on the residual loss of PDEs. Similar ideas of solving PDEs via minimizing the residual loss can be traced back to the works [11,12] in the 1990's, where a shallow neural network is optimized on a priori fixed mesh as an approximation of the solution. Some efforts have been made to incorporate traditional computational techniques to enhance the performance of solving PDEs with deep neural networks. In [13–16,7], deep neural networks based on domain

* Corresponding author at: School of Mathematical Sciences, Peking University, Beijing 100871, China.
E-mail addresses: tangkj@pcl.ac.cn (K. Tang), xlwan@math.lsu.edu (X. Wan), chao_yang@pku.edu.cn (C. Yang).

https://doi.org/10.1016/j.jcp.2022.111868
0021-9991/© 2022 Elsevier Inc. All rights reserved.

decomposition are proposed to improve the efficiency. A penalty free neural network method [17] and Phygeonet [16] are developed to deal with complex geometries and irregular domains. A weak formulation with primal and adversarial networks is proposed in [18], where the PDE problem is converted to an operator norm minimization problem induced by the weak formulation.

One critical step for all these methods is to approximate the loss functional, where the integral is usually approximated by the Monte Carlo method with collocation points randomly generated by a uniform distribution on the computational domain. Since the minimization of the discrete loss functional yields the approximate solution, the accuracy of the approximate solution is closely related to the accuracy of the discrete loss functional. In contrast to classical computational methods, where the main concern is the approximation error, one needs to balance the approximation error and the generalization error for the neural network approximation, where the approximation error mainly originates from the modeling capability of the neural network and the generalization error is mainly related to the data points in the training set, i.e., the random samples for the discretization of the loss functional. However, for many PDE models, the uniform random sampling strategy is not efficient especially when the PDE solution has a low regularity, in other words, an integrand of low regularity may have a large variance in terms of a uniform distribution such that the Monte Carlo approximation of the loss functional has a large prefactor before the convergence rate $O(N^{-1/2})$. This issue becomes worse for high-dimensional problems due to the curse of dimensionality. In high-dimensional spaces, most of the volume of the computational domain concentrates around its surface [19–21], which means that uniform samples may become less effective for training deep neural networks to approximate high-dimensional PDEs. For example, the collocation points from the uniform distribution are not suitable for solving high-dimensional Fokker-Planck equations, while an adaptive strategy through sampling the current approximate solution is effective [22]. In [23], a selection network is introduced to serve as a weight function to assign higher weights for samples with large point-wise residuals, which yields a more accurate approximate solution if the selection network is properly chosen. However, to obtain a valid selection network, one needs to impose additional constraints on the selection network, which is often a non-trivial task. For low-dimensional problems, it is well known that one can employ adaptive numerical schemes to deal with PDEs with low-regularity solutions [24–26], which also suggests that the uniform samples are not the best choice. Therefore, adaptive sampling strategies are crucial for developing more efficient and reliable deep learning techniques for the approximation of PDEs.

In this work, we develop a deep adaptive sampling method (DAS-PINNs) for the neural network approximation of PDEs based on residual minimization, where a deep generative model, called KRnet [27–29], is used to guide the sample generation for the training set. To this end, we need to construct two deep neural network models: one for approximating the solution and the other one for refining the training set. The neural network approximation is achieved by the standard procedure of residual minimization. KRnet defines a transport map [30] from the data distribution to a prior distribution (e.g. the standard Gaussian). KRnet retains two traits of flow-based generative models [31,32]: exact invertibility of the transport map and efficient computation of the Jacobian determinant, based on which one can obtain an explicit density model using the change of variables and an effective approach for generating samples through the invertible mapping. The key point in our proposed framework is that the residual is viewed as a probability density function (PDF) up to a constant and approximating this PDF can be achieved by minimizing the Kullback-Leibler (KL) divergence between the KRnet-induced density model and the residual-induced distribution. We use the trained KRnet to generate new collocation points to replace or refine the training set, where more points are put in the region of large residual and less points are put in the region of small residual. The updated training set is then used to further improve the accuracy of the current approximate solution. Simply speaking, KRnet acts as an error indicator for the improvement of the training set, which shares similarities with the classical adaptive finite element method subject to a residual-based posteriori error estimator. In summary, the main contributions of this work are as follows.

- We utilize a deep generative model as a generic means to reflect the correspondence between the residual and the error of approximation through efficient PDF approximation and sample generation.
- We propose a deep adaptive sampling (DAS-PINNs) framework, including efficient sampling procedures and training algorithms, for the adaptive improvement of neural network approximation of PDEs.

The remainder of the paper is organized as follows. In the next section, we briefly describe the deep learning method used in this work for the approximation of PDEs. After that, the statistical error of the machine learning technique is illustrated from the perspective of function approximation. Our DAS approach is presented in section 4. We provide the theoretical analysis of DAS in section 5. In section 6, we demonstrate the efficiency of our adaptive sampling approach with numerical experiments. The paper is concluded in section 7.

2. Deep learning for PDEs

Let $\Omega \subset \mathbb{R}^d$ be a spatial domain, which is bounded, connected and with a polygonal boundary $\partial\Omega$, and $\boldsymbol{x} \in \mathbb{R}^d$ denote a spatial variable. The PDE problem is stated as: find $u(\boldsymbol{x}) \in F : \mathbb{R}^d \mapsto \mathbb{R}$ where F is a proper function space defined on Ω, such that

$$\begin{aligned}\mathcal{L}u(\boldsymbol{x}) &= s(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Omega, \\ \mathfrak{b}u(\boldsymbol{x}) &= g(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \partial\Omega,\end{aligned} \tag{1}$$

where $\mathcal{L}$ is the partial differential operator, $\mathfrak{b}$ is the boundary operator, $s(\boldsymbol{x})$ is the source function, and $g(\boldsymbol{x})$ represents the boundary conditions.

Let $u(\boldsymbol{x};\Theta)$ be a neural network with parameters Θ. In the framework of PINNs, the goal is to use $u(\boldsymbol{x};\Theta)$ to approximate the solution $u(\boldsymbol{x})$ through optimizing a loss functional defined as [8,9]

$$J\left(u(\cdot;\Theta)\right) = \|r(\boldsymbol{x};\Theta)\|_{2,\Omega}^2 + \gamma\,\|b(\boldsymbol{x};\Theta)\|_{2,\partial\Omega}^2 = J_r(u(\boldsymbol{x};\Theta)) + \gamma\, J_b(u(\boldsymbol{x};\Theta)), \tag{2}$$

where $r(\boldsymbol{x};\Theta) = \mathcal{L}u(\boldsymbol{x};\Theta) - s(\boldsymbol{x})$, and $b(\boldsymbol{x};\Theta) = \mathfrak{b}u(\boldsymbol{x};\Theta) - g(\boldsymbol{x})$ measure how well $u(\boldsymbol{x};\Theta)$ satisfies the partial differential equations and the boundary conditions, respectively, and $\gamma > 0$ is a penalty parameter. Here, $\|u\|_{2,\Omega}^2 = \int_\Omega |u(\boldsymbol{x})|^2 d\boldsymbol{x}$ and $\|u\|_{2,\partial\Omega}^2 = \int_{\partial\Omega} |u(\boldsymbol{x})|^2 d\boldsymbol{x}$. The loss functional (2) is usually discretized numerically before the optimization with respect to Θ is addressed. In practice, one often chooses two sets of uniformly distributed collocation points $\mathsf{S}_\Omega = \{\boldsymbol{x}_\Omega^{(i)}\}_{i=1}^{N_r}$ and $\mathsf{S}_{\partial\Omega} = \{\boldsymbol{x}_{\partial\Omega}^{(i)}\}_{i=1}^{N_b}$ respectively for the discretization of the two terms in the objective functional (2), leading to the following empirical loss

$$J_N\left(u(\cdot;\Theta)\right) = \|r(\boldsymbol{x};\Theta)\|_{N_r,\mathsf{S}_\Omega}^2 + \hat{\gamma}\,\|b(\boldsymbol{x};\Theta)\|_{N_b,\mathsf{S}_{\partial\Omega}}^2, \tag{3}$$

where $\hat{\gamma} > 0$, and

$$\|u\|_{N_r,\mathsf{S}_\Omega} = \left(\frac{1}{N_r}\sum_{i=1}^{N_r} u^2(\boldsymbol{x}_\Omega^{(i)})\right)^{\frac{1}{2}}, \quad \|u\|_{N_b,\mathsf{S}_{\partial\Omega}} = \left(\frac{1}{N_b}\sum_{i=1}^{N_b} u^2(\boldsymbol{x}_{\partial\Omega}^{(i)})\right)^{\frac{1}{2}}.$$

Note that in the definition of J_N we do not take into account the constants $|\Omega| = \int_\Omega d\boldsymbol{x}$ and $|\partial\Omega| = \int_{\partial\Omega} d\boldsymbol{x}$ and the ratio induced by these two constants can be dealt with by choosing $\hat{\gamma} = \frac{\gamma|\partial\Omega|}{|\Omega|}$ such that $J_N(u)$ is a Monte Carlo approximation of $J(u)$ up to a constant scaling factor $|\Omega|$. We then seek an approximate solution by minimizing the empirical loss (3), i.e.,

$$\min_\Theta J_N(u(\cdot;\Theta)), \tag{4}$$

which can be solved by stochastic gradient-based methods [33,34].

Recently, some prior error estimates of neural-network-based methods for solving PDEs are established. Combining the analysis techniques of the least square finite element method [35] with the universal approximation property of neural networks [36–39], Shin et al. propose an abstract framework for the error estimation of PINNs [40]. Lu et al. derive a prior estimate of the generalization error for the deep Ritz method with two-layer neural networks [41]. Suppose that $u(\cdot,\Theta_N^*)$ is the minimizer of the empirical loss $J_N(u(\cdot;\Theta))$ and $u(\cdot;\Theta^*)$ is the minimizer of $J(u(\cdot;\Theta))$, i.e.,

$$\begin{aligned} u(\cdot;\Theta^*) &= \arg\min_\Theta J(u(\cdot;\Theta)), \\ u(\cdot;\Theta_N^*) &= \arg\min_\Theta J_N(u(\cdot;\Theta)). \end{aligned}$$

We have

$$u(\boldsymbol{x};\Theta_N^*) - u(\boldsymbol{x}) = u(\boldsymbol{x},\Theta_N^*) - u(\boldsymbol{x};\Theta^*) + u(\boldsymbol{x};\Theta^*) - u(\boldsymbol{x}), \tag{5}$$

i.e.,

$$\mathbb{E}\left(\left\|u(\cdot;\Theta_N^*) - u\right\|_\Omega\right) \le \mathbb{E}\left(\left\|u(\cdot,\Theta_N^*) - u(\cdot;\Theta^*)\right\|_\Omega\right) + \left\|u(\cdot;\Theta^*) - u\right\|_\Omega, \tag{6}$$

where $\mathbb{E}$ indicates the expectation and the norm $\|\cdot\|_\Omega$ corresponds to the function space F for u. The first term describes the statistical error from discretizing the loss functional with the Monte Carlo approximation, and the second term is the approximation error of minimizing the loss functional over the hypothesis space. The approximation error depends on the capability of neural networks, while the statistical error depends on the definition of S_Ω and $\mathsf{S}_{\partial\Omega}$. In this work, we focus on how to reduce the statistical error for problem (4) and our algorithm can also be generalized to other formulations for the neural network approximation of PDEs. For simplicity, we focus on the integration of the residual $r(\boldsymbol{x};\Theta)$ and assume that the integral on the boundary is well approximated by a prescribed $\mathsf{S}_{\partial\Omega}$.

3. Illustration of the statistical error

We first use function approximation as an example to illustrate the statistical error in machine learning. Let $\boldsymbol{X} \in \mathbb{R}^d$ and $Y \in \mathbb{R}$ subject to a joint distribution $\rho_{\boldsymbol{X},Y}$. Let $\hat{Y} = \boldsymbol{m}(\boldsymbol{X})$ be a model and $h : x \mapsto y$ be a function to be approximated. We know in the L_2 sense the optimal model is

$$\boldsymbol{m}^* = \arg\min_{\boldsymbol{m}} \left[L(\boldsymbol{m}) = \int (y - \boldsymbol{m}(\boldsymbol{x}))^2 \rho_{\boldsymbol{X},Y}(\boldsymbol{x}, y) d\boldsymbol{x} dy \right]. \tag{7}$$

In reality, we usually do not know $\rho_{\boldsymbol{X},Y}$ and only have a set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^N$ of data which can be regarded as samples of $\rho_{\boldsymbol{X},Y}$. For a certain hypothesis space W, we obtain a regression problem

$$\boldsymbol{m}_{\boldsymbol{w}^*} = \arg \min_{\boldsymbol{m}_{\boldsymbol{w}} \in W} \left[L_N(\boldsymbol{m}_{\boldsymbol{w}}) = \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \boldsymbol{m}_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}))^2 \right], \tag{8}$$

where L_N can be regarded as a Monte Carlo approximation of L and the subscript $\boldsymbol{w}$ indicates the model parameters specified by W. If we let $\rho_{\boldsymbol{X},Y}(\boldsymbol{x}, y) = \delta(y - h(\boldsymbol{x}))\rho(\boldsymbol{x})$, and assume $m \in V$ with V being a linear space, we then obtain the continuous least-squares method for function approximation

$$\boldsymbol{m}_V^* = \arg\min_{\boldsymbol{m} \in V} \left[L_V(\boldsymbol{m}) = \int (\boldsymbol{m}(\boldsymbol{x}) - h(\boldsymbol{x}))^2 \rho(\boldsymbol{x}) d\boldsymbol{x} \right], \tag{9}$$

where $\boldsymbol{m}_V^*$ is the best approximation of h located in V subject to a weighted L_2 norm in terms of a probability density function ρ. To approximate h, we consider

$$\boldsymbol{m}_{\hat{\boldsymbol{v}}^*} = \arg \min_{\boldsymbol{m}_{\hat{\boldsymbol{v}}} \in V} \left[L_{V,N}(\boldsymbol{m}_{\hat{\boldsymbol{v}}}) = \frac{1}{N} \sum_{i=1}^N (\boldsymbol{m}_{\hat{\boldsymbol{v}}}(\boldsymbol{x}^{(i)}) - h(\boldsymbol{x}^{(i)}))^2 \right], \tag{10}$$

where $\{\boldsymbol{x}^{(i)}\}_{i=1}^N$ are samples of ρ, and $L_{V,N}$ is the Monte Carlo approximation of L_V. Here, we use the linear space spanned by polynomials as an example to show how the statistical error naturally arises from machine learning. Although the algorithm in this paper does not consider the linear space, it can be used to explain the existence of statistical errors and clarify the roles of samples. We derive the error estimate for $m_{\hat{\boldsymbol{v}}^*}$ as follows.

Lemma 1. *Let $h \in C(D)$ be a continuous function defined on a compact domain $D \subset \mathbb{R}^d$ and $\rho(\boldsymbol{x}) > 0$ be a PDF on D. Let $V = \operatorname{span}\{q_i : i = 1, \dots, n\}$ with q_i being orthonormal polynomials in terms of ρ. For any $\delta > 0$ and with probability at least $1 - 2\delta$, we have for a sufficiently large N*

$$\|\boldsymbol{m}_{\hat{\boldsymbol{v}}^*} - h\|_\rho \le C\sqrt{\frac{\ln \delta^{-1}}{N}} + \|\boldsymbol{m}_V^* - h\|_\rho,$$

where C is a constant, and $\|\cdot\|_\rho$ is the weighted L_2 norm in terms of ρ.

The first term on the right-hand is the statistical error due to the random samples for the approximation of L_V (see the proof in Appendix A for more details) and its existence does not depend on the choice of V. When N goes to infinity, the statistical error goes to zero and only the approximation error is left. In other words, when applying neural networks to function approximation, we need to pay attention to both the hypothesis space W and the choice of random samples $\{\boldsymbol{x}^{(i)}\}_{i=1}^N$, i.e., the training set, to obtain a trade-off between the statistical error and the approximation error. For low-dimensional problems, classical methods such as finite element methods avoid the statistical error by using Gauss quadrature rules, which implies that deep learning methods are in general less efficient than classical methods due to the existence of statistical error. On the other hand, for high-dimensional problems, classical methods may not be able to obtain a relatively small approximation error due to the curse of dimensionality and deep learning methods may perform better by using a capable hypothesis space such as deep neural networks and an affordable sample size for a relatively small statistical error. A more general estimate about the statistical error needs the Rademacher complexity of the hypothesis space. In this work, we are interested in the reduction of the statistical error instead of its bound for a certain set of random samples.

4. Deep adaptive sampling method

We now focus on the reduction of the statistical error when neural networks are used to approximate PDEs. Our deep adaptive sampling method, i.e., the DAS method, will be established from the viewpoint of variance reduction since the statistical error is induced by the Monte Carlo approximation of the loss. Consider the term $J_r(u)$ in equation (2). It is easy to see that if $r^2(\boldsymbol{x})$ is a smooth function with a good regularity, the most effective way to reduce the error of $J_N(u)$ is to increase the sample size N. However, if there exists low regularity, the situation might be different. For example, if the residual is strongly localized, the scenario can be regarded as a rare event. Assume that the residual $r^2(\boldsymbol{x})$ in equation (2) has a similar behavior to an indicator function $1_I(\boldsymbol{x})$ where $I \subset \Omega$ and is much smaller than Ω, i.e.,

$$\zeta = \int_\Omega 1_I(\boldsymbol{x}) d\boldsymbol{x} \approx \int_\Omega r^2(\boldsymbol{x}) d\boldsymbol{x} \ll 1.$$

For simplicity, we assume that $|\Omega| = 1$ when presenting the algorithm. To improve the approximation in I, we need to compute ζ accurately. Consider a Monte Carlo estimator of ζ in terms of uniform samples

$$\hat{P}_{\mathsf{MC}} = \frac{1}{N} \sum_{i=1}^{N} 1_I(\boldsymbol{x}^{(i)}).$$

The relative error of $\hat{P}_{\mathsf{MC}}$ is

$$\frac{\mathrm{Var}^{1/2}(\hat{P}_{\mathsf{MC}})}{\zeta} = N^{-1/2}((1-\zeta)/\zeta)^{1/2} \approx (\zeta N)^{-1/2}.$$

We then need a sample size of $O(1/\zeta)$ to obtain a relative error of $O(1)$. This implies that a set of uniform samples with size N quickly becomes less effective if the residual is strongly localized and such a problem can be worsened in the approximation of high-dimensional PDEs. Instead, adaptive sampling can help accelerate the training process and reduce the relative error (see the example of solving high-dimensional Fokker-Planck equations in [22]). So, we need to choose more effective random samples instead of uniform samples, which can incorporate the problem properties.

4.1. Some ideas on variance reduction

We outline our basic ideas on variance reduction in this section and more details about the algorithm will be presented later. We first consider the importance sampling technique. For simplicity, we only consider $J_r(u(\boldsymbol{x};\Theta))$ in equation (2), which is the loss induced by the residual. We have

$$J_r(u(\boldsymbol{x};\Theta)) = \mathbb{E}[r^2] = \int_\Omega r^2(\boldsymbol{x};\Theta) d\boldsymbol{x} = \int_\Omega \frac{r^2(\boldsymbol{x};\Theta)}{p(\boldsymbol{x})} p(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(\boldsymbol{x}_\Omega^{(i)};\Theta)}{p(\boldsymbol{x}_\Omega^{(i)})}, \tag{11}$$

where the set $\{\boldsymbol{x}_\Omega^{(i)}\}_{i=1}^{N_r}$ is generated with respect to the probability density function (PDF) $p(\boldsymbol{x})$ instead of a uniform distribution as in equation (3). Note that r^2 is defined in Ω, so a valid p needs to satisfy $p(x) = 0$ for $\boldsymbol{x} \in \mathbb{R}^d \backslash \Omega$. The details of designing such p are presented in the next subsection. If the variance of $r^2(\boldsymbol{X})p^{-1}(\boldsymbol{X})$ in terms of $p(\boldsymbol{x})$ is smaller than the variance of $r^2(\boldsymbol{X})$ in terms of the uniform distribution, the accuracy of the Monte Carlo approximation will be improved for a fixed sample size N_r. The optimal choice for $p(\boldsymbol{x})$ is

$$p^*(\boldsymbol{x}) = \frac{r^2(\boldsymbol{x};\Theta)}{\mu}, \tag{12}$$

where $\mu = \int_\Omega r^2(\boldsymbol{x};\Theta) d\boldsymbol{x}$. The optimal choice is useless in practice since μ is the quantity to be computed. One question that needs to be clarified is whether variance reduction can be achieved if p is sufficiently close to p^* in a certain sense. In practice, one commonly used quantity to measure the difference between two probability distributions is the Kullback-Leibler (KL) divergence defined as

$$D_{\mathsf{KL}}(p\|p^*) = \mathbb{E}_p\left[\log \frac{p(\boldsymbol{x})}{p^*(\boldsymbol{x})}\right],$$

where $\mathbb{E}_p$ indicates the expectation with respect to p. Assuming that p approaches p^* in terms of the KL divergence, we have the following lemma.

Lemma 2. *Assume that $|\Omega| = 1$ and $p(\boldsymbol{x})$ is a PDF satisfying*

$$D_{\mathsf{KL}}(p\|p^*) \le \varepsilon < \infty,$$

where D_{KL} indicates the Kullback-Leibler divergence. For any $0 < a < \infty$, we have

$$\mathbb{E}\left|Q_p[r^2] - \mathbb{E}[r^2]\right| \le aN_r^{-1/2} + 2\|r^2/p\|_p \sqrt{\mathbb{P}(|r^2/p - \mu| > a; p)}, \tag{13}$$

where

$$Q_p(r^2) = \frac{1}{N_r}\sum_{i=1}^{N_r}\frac{r^2(\boldsymbol{X}^{(i)})}{p(\boldsymbol{X}^{(i)})},$$

and $\boldsymbol{X}^{(i)} \sim p(\boldsymbol{x})$ are i.i.d. random variables. $\|\cdot\|_p$ is the weighted L_2 norm in terms of p. The tail probability can be bounded as

$$\mathbb{P}(|r^2/p - \mu| > a; p) \le \frac{\mu(2\varepsilon)^{1/2}}{a}.$$

When $p(\boldsymbol{x})$ approaches $p^*(\boldsymbol{x})$, the ratio $r^2(\boldsymbol{x})/p(\boldsymbol{x})$ approaches μ. The second term on the right-hand side of inequality (13) is related to the tail probability $\mathbb{P}(|r^2/p - \mu| > a; p)$ in terms of $p(\boldsymbol{x})$, which goes to zero for any a as $\varepsilon \to 0$. If we let $a = \varepsilon^\beta$ with $0 < \beta < \frac{1}{2}$, the error can be arbitrarily small for any N_r when ε is sufficiently small. In other words, if we rewrite the error bound as $c(a = \varepsilon^\beta, \varepsilon)N_r^{-1/2}$, we have $c(a, \varepsilon) \to 0$ for a fixed N_r as $\varepsilon \to 0$. We will obtain variance reduction when $c(a, \varepsilon)$ is small enough, together with an extra assumption that $Q_p[r^2] \le M < \infty$ since

$$\mathbb{E}\left[\left(Q_p[r^2] - \mathbb{E}[r^2]\right)^2\right] \le \max\{M, \mu\}\mathbb{E}\left|Q_p[r^2] - \mathbb{E}[r^2]\right|.$$

However, we are not able to obtain variance reduction directly because the KL divergence is weaker than the L_2 norm.

Another option for variance reduction is to relax the definition of $J_r(u)$ as:

$$J_{r,p}(u(\boldsymbol{x};\Theta)) = \int_\Omega r^2(\boldsymbol{x};\Theta)p(\boldsymbol{x})d\boldsymbol{x} \approx \frac{1}{N_r}\sum_{i=1}^{N_r} r^2(\boldsymbol{x}_\Omega^{(i)};\Theta), \tag{14}$$

where the set $\{\boldsymbol{x}_\Omega^{(i)}\}_{i=1}^{N_r}$ is sampled from the PDF $p(\boldsymbol{x})$ and $p(\boldsymbol{x}) > 0$ on Ω. If the minimal value of $J_{r,p}(u(\boldsymbol{x};\Theta))$ is zero, it is easy to see that the minimizer of $J_{r,p}(u(\boldsymbol{x};\Theta))$ is also the solution of problem (1) when the boundary conditions are satisfied. To reduce the error induced by the Monte Carlo approximation, we may adjust $p(\boldsymbol{x})$ such that the residual $r^2(\boldsymbol{x};\Theta)$ does not vary dramatically. This is similar to the classical adaptive finite element method, where the mesh refinement/coarsening is supposed to make the approximation error nearly uniform. For our case, we may sample from a distribution that is close to the residual-induced distribution and add more samples from the region of large residual into the training set.

4.2. PDF approximation and sample generation

Two options for variance reduction are considered in the previous section. To make both ideas practical, the key issue is to generate samples efficiently from a PDF $p(\boldsymbol{x}) \approx \mu^{-1}r^2(\boldsymbol{x};\Theta)$ for a fixed Θ. The first option is based on importance sampling, meaning that an explicit PDF model $p(\boldsymbol{x})$ is needed. The second option is based on refinement of the training set, which only needs samples from the region of large residual and does not need the likelihood of the samples. The second option has been employed in the recent literature to improve the neural network approximation, which is either ad hoc [42] (only for low-dimensional problems) or based on traditional sampling strategies such as MCMC [43]. To the best of our knowledge, there does not exist a generic algorithm that can be effectively adapted to both options. We intend to fill this gap in this work. It is seen from Lemma 2 that the tail probability $\mathbb{P}(|r^2/p - \mu| > a; p)$ should be small enough for the effectiveness of p, which means that p must be close enough to the distribution induced by r^2. However, the approximation of PDF is a challenging task especially in high-dimensional spaces. Classical explicit PDF models such as the exponential family of distributions and Gaussian mixture models are in general not sufficient as an approximator for the PDF induced by r^2. To alleviate this difficulty, we resort to deep generative modes. In particular, we employ a recently developed deep generative model called KRnet for both probability approximation and sample generation [22,27]. KRnet is one type of normalizing flows [44], which provides an invertible transport map between a prior distribution and the target distribution. Unlike other types of deep generative models such as GAN [45–47] and VAE [48], normalizing flows provide an explicit likelihood. KRnet can be used to address both options for variance reduction while other deep generative models may also be employed if only the second option is considered. However, one computational issue faced by all deep generative models is that the distribution induced by $r^2(\boldsymbol{x})$ is usually defined on a compact domain while deep generative models are in general defined on the whole space. We subsequently address this issue without going into details about the structure of KRnet.

Let $\boldsymbol{X} \in \mathbb{R}^d$ be a random vector associated with a given data set, and its PDF is denoted by $p_{\boldsymbol{X}}(\boldsymbol{x})$. The target is to estimate $p_{\boldsymbol{X}}(\boldsymbol{x})$ from data or to generate samples that are consistent with a given $p_{\boldsymbol{X}}(\boldsymbol{x})$. Let $\boldsymbol{Z} \in \mathbb{R}^d$ be a random vector associated with a PDF $p_{\boldsymbol{Z}}(\boldsymbol{z})$, where $p_{\boldsymbol{Z}}(\boldsymbol{z})$ is a prior distribution (e.g., Gaussian distributions). The flow-based generative modeling is to seek an invertible mapping $\boldsymbol{z} = f(\boldsymbol{x})$ [31]. By the change of variables, we have the PDF of $\boldsymbol{X} = f^{-1}(\boldsymbol{Z})$ as

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = p_{\boldsymbol{Z}}(f(\boldsymbol{x}))\left|\det \nabla_{\boldsymbol{x}} f\right|. \tag{15}$$

Once the prior distribution $p_{\boldsymbol{Z}}(\boldsymbol{z})$ is specified, equation (15) provides an explicit density model for $\boldsymbol{X}$. The inverse of $f(\cdot)$ provides a convenient way to sample $\boldsymbol{X}$ as $\boldsymbol{X} = f^{-1}(\boldsymbol{Z})$. The basic idea of KRnet is to define the structure of $f(\boldsymbol{x})$ in terms of the Knothe-Rosenblatt rearrangement. Let $\mu_{\boldsymbol{Z}}$ and $\mu_{\boldsymbol{X}}$ be the probability measures of two random variables $\boldsymbol{Z}, \boldsymbol{X} \in \mathbb{R}^d$ respectively. A mapping $\mathcal{T}$: $\boldsymbol{Z} \mapsto \boldsymbol{X}$ is called a transport map such that $\mathcal{T}_{\#}\mu_{\boldsymbol{Z}} = \mu_{\boldsymbol{X}}$, where $\mathcal{T}_{\#}\mu_{\boldsymbol{Z}}$ is the push-forward of $\mu_{\boldsymbol{Z}}$ such that $\mu_{\boldsymbol{X}}(B) = \mu_{\boldsymbol{Z}}(\mathcal{T}^{-1}(B))$ for every Borel set B [49]. The transport map $\mathcal{T}$ given by the Knothe-Rosenblatt (K-R) rearrangement [49,30] has a lower-triangular structure

$$\boldsymbol{z} = \mathcal{T}^{-1}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{T}_1(x_1) \\ \mathcal{T}_2(x_1, x_2) \\ \vdots \\ \mathcal{T}_d(x_1, \dots, x_d) \end{bmatrix}. \tag{16}$$

Simply speaking, KRnet integrates the triangular structure of the K-R rearrangement into the definition of the invertible mapping $\boldsymbol{z} = f(\boldsymbol{x})$, which can be regarded as a transport map. More details can be found in [27,22,29].

Let $f_{\mathrm{KRnet}}(\cdot; \Theta_f)$ indicate the invertible transport map induced by KRnet, where Θ_f includes the model parameters. An explicit PDF model $p_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$ can be obtained by letting $f = f_{\mathrm{KRnet}}$ in equation (15), i.e.,

$$p_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f) = p_{\boldsymbol{Z}}(f_{\mathrm{KRnet}}(\boldsymbol{x})) \left|\det \nabla_{\boldsymbol{x}} f_{\mathrm{KRnet}}\right|. \tag{17}$$

The samples of $p_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$ are given as $\boldsymbol{X} = f_{\mathrm{KRnet}}^{-1}(\boldsymbol{Z})$ by sampling $\boldsymbol{Z}$. A common choice for the distribution of $\boldsymbol{Z}$ is the standard Gaussian distribution. Depending on the prior knowledge of the problem, a more general model such as Gaussian mixture model can also be used as the prior distribution. The KRnet f_{KRnet} does not have any constraint on the range of the mapped data, meaning that both $\boldsymbol{X}$ and $\boldsymbol{Z}$ are defined on $\mathbb{R}^d$. Let $\boldsymbol{Z}$ be Gaussian. Due to the invertibility, $|\det \nabla_{\boldsymbol{x}} f_{\mathrm{KRnet}}| > 0$ for any $\boldsymbol{x} \in \mathbb{R}^d$, which implies that $p_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f) > 0$ for any $\boldsymbol{x} \in \mathbb{R}^d$. So $p_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$ is not consistent with the distribution induced by $r^2(\boldsymbol{x})$, which is equal to zero on $\mathbb{R}^d \backslash \Omega$. To deal with this issue, we propose the following strategy.

Without loss of generality, we can assume that $\Omega = [-1/2, 1/2]^d$, since any square domain in $\mathbb{R}^d$ can be mapped to Ω. Let $B = (-(1/2 + \delta/2), 1/2 + \delta/2)^d$ with $0 < \delta < \infty$ such that $\Omega \subset B$. For each dimension of $\boldsymbol{x}$, we define the following logarithmic mapping

$$y = \ell(x) = \frac{s}{2} \log \frac{2x + (1+\delta)}{(1+\delta) - 2x}, \quad x = \ell^{-1}(y) = \frac{1+\delta}{2} \frac{e^{2y/s} - 1}{e^{2y/s} + 1},$$

with $s > 0$ being a scale parameter, which defines a one-to-one correspondence between $x \in (-(1/2 + \delta/2), 1/2 + \delta/2)$ and $y \in (-\infty, +\infty)$. Let $\boldsymbol{\ell}(\boldsymbol{x}) : B \mapsto \mathbb{R}^d$ be a d-dimensional mapping such that

$$\ell_i(x_i) = \ell(x_i), \quad i = 1, \dots, d.$$

Then the following invertible mapping

$$\boldsymbol{z} = f_{\mathrm{KRnet}} \circ \boldsymbol{\ell}(\boldsymbol{x}) \tag{18}$$

defines a PDF

$$\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f) = p_{\mathrm{KRnet}}(\boldsymbol{\ell}(\boldsymbol{x}); \Theta_f) |\nabla_{\boldsymbol{x}} \boldsymbol{\ell}(\boldsymbol{x})|, \tag{19}$$

where the support of $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$ is B.

We now consider a modification of $r^2(\boldsymbol{x}; \Theta)$. Define a cutoff function as

$$h(\boldsymbol{x}) = \begin{cases} 1, \ \boldsymbol{x} \in \Omega, \\ \prod_{i=1}^d h_\delta(x_i), \ \boldsymbol{x} \in B \backslash \Omega, \end{cases}$$

where $h_\delta(x)$ is a piecewise linear function

$$h_\delta(x) = \begin{cases} 1, \ x \in [-1/2, 1/2], \\ \delta^{-1}(2x + 1 + \delta), \ x \in (-1/2 - \delta/2, -1/2), \\ \delta^{-1}(1 + \delta - 2x), \ x \in (1/2, 1/2 + \delta/2), \\ 0, \ x \in (-\infty, -1/2 - \delta/2] \cup [1/2 + \delta/2, \infty). \end{cases}$$

We consider a modified PDF for any Θ

$$\hat{r}_{\boldsymbol{X}}(\boldsymbol{x}) \propto r^2(\boldsymbol{x}; \Theta) h(\boldsymbol{x}). \tag{20}$$

Note that both $\hat{r}_{\boldsymbol{X}}(\boldsymbol{x})$ and $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$ have the support B. We then solve the following optimization problem

$$\Theta_f^* = \arg\min_{\Theta_f} D_{\mathrm{KL}}(\hat{r}_{\boldsymbol{X}}(\boldsymbol{x}) \| \hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)), \tag{21}$$

where $D_{\mathrm{KL}}(\cdot\|\cdot)$ indicates the Kullback-Leibler (KL) divergence between two distributions. We finally use

$$p_{\boldsymbol{X}}(\boldsymbol{x}) \propto \hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f^*) 1_\Omega(\boldsymbol{x}) \tag{22}$$

as an approximation of the PDF induced by $r^2(\boldsymbol{x}; \Theta)$. If $\delta \ll 1$, most the samples $\boldsymbol{\ell}^{-1} \circ f_{\mathrm{KRnet}}^{-1}(\boldsymbol{z}^{(i)}; \Theta_f^*)$ will be located in Ω. Since $\hat{r}_{\boldsymbol{X}}(\boldsymbol{x}) \propto r^2(\boldsymbol{x}; \Theta)$ on Ω, $p_{\boldsymbol{X}}(\boldsymbol{x})$ approximates the $r^2(\boldsymbol{x}; \Theta)$-induced PDF well when δ is small. In our numerical experiments, we set $\delta = 0.01$ and $s = 2$.

We now look at the approximation of Θ_f^*. The KL divergence in the optimization problem (21) can be written as

$$D_{\mathrm{KL}}(\hat{r}_{\boldsymbol{X}}(\boldsymbol{x}) \| \hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)) = \int_B \hat{r}_{\boldsymbol{X}} \log \hat{r}_{\boldsymbol{X}} d\boldsymbol{x} - \int_B \hat{r}_{\boldsymbol{X}} \log \hat{p}_{\mathrm{KRnet}} d\boldsymbol{x}. \tag{23}$$

The first term on the right-hand side corresponds to the differential entropy of $\hat{r}_{\boldsymbol{X}}$, which does not affect the optimization with respect to Θ_f. So minimizing the KL divergence is equivalent to minimizing the cross entropy between $\hat{r}_{\boldsymbol{X}}$ and $\hat{p}_{\mathrm{KRnet}}$ [50,51]:

$$H(\hat{r}_{\boldsymbol{X}}, \hat{p}_{\mathrm{KRnet}}) = -\int_B \hat{r}_{\boldsymbol{X}} \log \hat{p}_{\mathrm{KRnet}} d\boldsymbol{x}. \tag{24}$$

Since the samples from $\hat{r}_{\boldsymbol{X}}$ are not available, we approximate the cross entropy using the importance sampling technique:

$$H(\hat{r}_{\boldsymbol{X}}, \hat{p}_{\mathrm{KRnet}}) \approx -\frac{1}{N_r} \sum_{i=1}^{N_r} \frac{\hat{r}_{\boldsymbol{X}}(\boldsymbol{x}_B^{(i)})}{\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}_B^{(i)}; \hat{\Theta}_f)} \log \hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}_B^{(i)}; \Theta_f), \tag{25}$$

where $\hat{p}(\boldsymbol{x}; \hat{\Theta}_f)$ is a PDF model with known parameter $\hat{\Theta}_f$ and its samples $\{\boldsymbol{x}_B^{(i)}\}_{i=1}^{N_r}$ can be generated efficiently as

$$\boldsymbol{x}_B^{(i)} = \boldsymbol{\ell}^{-1} \circ f_{\mathrm{KRnet}}^{-1}(\boldsymbol{z}^{(i)}; \hat{\Theta}_f) \tag{26}$$

with $\boldsymbol{z}^{(i)}$ being sampled from the prior distribution. We then minimize the discretized cross entropy (25) to obtain an approximation of Θ_f^*. The choice for $\hat{\Theta}_f$ will be specified in section 4.3 when the adaptive sampling method is defined.

Remark 1. An alternative approach for the approximation of $\hat{r}_{\boldsymbol{X}}(\boldsymbol{x})$ is to minimize the following KL divergence:

$$D_{\mathrm{KL}}(\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f) \| \hat{r}_{\boldsymbol{X}}(\boldsymbol{x})) = \int_B \hat{p}_{\mathrm{KRnet}} \log \hat{p}_{\mathrm{KRnet}} d\boldsymbol{x} - \int_B \hat{p}_{\mathrm{KRnet}} \log \hat{r}_{\boldsymbol{X}} d\boldsymbol{x}, \tag{27}$$

which can be approximated by samples from $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f)$. Note that the KL divergence is asymmetric. Minimizing the KL divergence (23) is not equivalent to the minimization of the KL divergence (27), although both minimizers will be achieved at $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f) = \hat{r}_{\boldsymbol{X}}(\boldsymbol{x})$ if $\hat{r}_{\boldsymbol{X}}$ can be reached exactly by a certain parameter Θ_f.

Remark 2. To apply the DAS method to a complex domain, we may use a square or a cube to cover the complex geometry together with rejection sampling. Moreover, we can also couple DAS with the penalty-free method [17] to handle boundary conditions more effectively. In this work, we only consider Ω to be a hypercube in $\mathbb{R}^d$.

4.3. Adaptive sampling procedure

We are now ready to present our algorithms. In this work, we mainly focus on the adaptivity of S_Ω for simplicity. The key step of our adaptivity strategy is to improve the effectiveness of the random samples in the training set S_Ω, and we provide two algorithms corresponding to the two options discussed in section 4.1.

I. We replace all the collocation points in the current training set using the new samples for importance sampling. This corresponds to equation (11).
II. We gradually add more collocation points to the current training set. This corresponds to equation (14), where the new samples are mainly from the region of large residual.

We first present strategy I. Let $S_{\Omega,0} = \{\boldsymbol{x}_{\Omega,0}^{(i)}\}_{i=1}^{N_r}$ and $S_{\partial\Omega,0}$ be two sets of collocation points that are uniformly sampled from Ω and $\partial\Omega$ respectively. Using $S_{\Omega,0}$ and $S_{\partial\Omega,0}$, we minimize the empirical loss (3) to obtain $u(\boldsymbol{x}; \Theta_N^{*,(1)})$. With $u(\boldsymbol{x}; \Theta_N^{*,(1)})$, we minimize the cross entropy (25) to get $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}; \Theta_f^{*,(1)})$, where we simply use uniform samples for importance sampling. To refine the training set, a new set $S_{\Omega,1} = \{\boldsymbol{x}_{\Omega,1}^{(i)}\}_{i=1}^{N_r}$ is generated by $\boldsymbol{\ell}^{-1} \circ f_{\mathrm{KRnet}}^{-1}(\boldsymbol{z}^{(i)}; \Theta_f^{*,(1)})$ (see equation

(18)). Then we continue to update the approximate solution $u(\boldsymbol{x};\Theta_N^{*,(1)})$ using $\mathsf{S}_{\Omega,1}$ as the training set. In general, we use $\mathsf{S}_{\Omega,k}=\{\boldsymbol{x}_{\Omega,k}^{(i)}\}_{i=1}^{N_r}$ to obtain $u(\boldsymbol{x};\Theta_N^{*,(k+1)})$ as

$$\Theta_N^{*,(k+1)}=\arg\min_{\Theta} J_N^{\mathsf{IS}}(u(\boldsymbol{x};\Theta)),$$

where $u(\boldsymbol{x};\Theta)$ is initialized as $u(\boldsymbol{x};\Theta_N^{*,(k)})$ and J_N^{IS} is defined as

$$J_N^{\mathsf{IS}}(u(\boldsymbol{x};\Theta))=\frac{1}{N_r}\sum_{i=1}^{N_r}\frac{r^2(\boldsymbol{x}_{\Omega,k}^{(i)};\Theta)}{\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x}_{\Omega,k}^{(i)};\Theta_f^{*,(k)})}+\frac{1}{N_b}\sum_{i=1}^{N_b}b^2(\boldsymbol{x}_{\partial\Omega,k}^{(i)};\Theta). \tag{28}$$

Starting with $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$, the density model $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f)$ is updated as

$$\Theta_f^{*,(k+1)}=\arg\min_{\Theta_f}-\frac{1}{N_r}\sum_{i=1}^{N_r}\frac{r^2(\boldsymbol{x}_{B,k}^{(i)};\Theta_N^{*,(k+1)})h(\boldsymbol{x}_{B,k}^{(i)})}{\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x}_{B,k}^{(i)};\Theta_f^{*,(k)})}\log\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x}_{B,k}^{(i)};\Theta_f), \tag{29}$$

where we let $\hat{\Theta}_f=\Theta_f^{*,(k)}$ in equation (25), i.e., the previous PDF model $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$ is used for importance sampling when computing the cross entropy. A new set $\mathsf{S}_{\Omega,k+1}=\{\boldsymbol{x}_{\Omega,k+1}^{(i)}\}_{i=1}^{N_r}$ of collocation points is then generated. As detailed in section 4.2, the support of data points generated by KRnet is $B=(-(1/2+\delta/2),1/2+\delta/2)^d$, while the computation domain is $\Omega=[-1/2,1/2]^d$. So we need to deal with the collocation points located in $B\backslash\Omega$. Instead of neglecting these points, we project them onto $\partial\Omega$. We define an entry-wise projection operator $\mathcal{P}(\boldsymbol{x}):B\mapsto\Omega$ as

$$\mathcal{P}(x_i)=\begin{cases}-1/2, & \text{if } x_i<-1/2,\\ x_i, & \text{if } -1/2\le x_i\le 1/2, \quad i=1,\dots,d,\\ 1/2, & \text{if } x_i>1/2.\end{cases} \tag{30}$$

For a sequence of i.i.d. samples $\boldsymbol{z}^{(j)}$ generated from the standard Gaussian with $j=1,2\dots$, we compute $\boldsymbol{x}_B^{(j)}=\boldsymbol{\ell}^{-1}\circ f_{\mathsf{KRnet}}^{-1}(\boldsymbol{z}^{(j)})$. if $\boldsymbol{x}_B^{(j)}=\mathcal{P}(\boldsymbol{x}_B^{(j)})$, we assign $\boldsymbol{x}_B^{(j)}$ to $\mathsf{S}_{\Omega,k+1}$; otherwise, we add $\mathcal{P}(\boldsymbol{x}_B^{(j)})$ to $\mathsf{S}_{\partial\Omega,k}$. The updated training set $\mathsf{S}_{\Omega,k+1}$ and $\mathsf{S}_{\partial\Omega,k+1}$ will be used for the next training stage. This procedure is repeated until the stopping criterion is satisfied (see Algorithm 1). Since the collocation points in $\mathsf{S}_{\Omega,k}$ will be completely replaced at the next stage, we call this type of deep adaptive sampling strategy DAS-R for short. The alternative approach given in Remark 1 can also be used to obtain $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$.

We now look at strategy II. Unlike DAS-R, the number of collocation points in the training set S_{Ω} increases gradually. So we denote this type of deep adaptive sampling strategy by DAS-G for short. Starting with an initial set of collocation points $\mathsf{S}_{\Omega,0}=\{\boldsymbol{x}_{\Omega}^{(i)}\}_{i=1}^{n_r}$ (as well as $\mathsf{S}_{\partial\Omega,0}$) drawn from a uniform distribution defined on Ω, we minimize the empirical loss (3) on the training set $\mathsf{S}_{\Omega,0}$ (as well as $\mathsf{S}_{\partial\Omega,0}$) to obtain $u(\boldsymbol{x};\Theta_N^{*,(1)})$. Once we have $u(\boldsymbol{x};\Theta_N^{*,(1)})$ in hand, we can seek $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(1)})$ using the residual $r^2(\boldsymbol{x};\Theta_N^{*,(1)})$. We here use uniform samples to approximate and minimize the cross entropy (25). Similar to DAS-R, a new set of collocation points $\mathsf{S}_{\Omega,1}^g=\{\boldsymbol{x}_{\Omega,1}^{g,(i)}\}_{i=1}^{n_r}$ is generated by $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(1)})$ while the main difference is that we update the training set as $\mathsf{S}_{\Omega,1}=\mathsf{S}_{\Omega,0}\cup\mathsf{S}_{\Omega,1}^g$, in other words, $\mathsf{S}_{\Omega,0}$ is augmented rather than replaced by $\mathsf{S}_{\Omega,1}^g$. We continue to update $u(\boldsymbol{x};\Theta)$ using $\Theta_N^{*,(1)}$ as the initial parameters and $\mathsf{S}_{\Omega,1}$ as the training set, which yields a refined model $u(\boldsymbol{x};\Theta_N^{*,(2)})$. Staring from $k=2$, we seek $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$ using the approach given in Remark 1. We repeat the procedure to obtain an adaptive algorithm (see Algorithm 2).

Our two adaptive training methods are summarized in Algorithm 1 (DAS-R) and Algorithm 2 (DAS-G), where N_{adaptive} is a given number of maximum adaptivity iterations, m is the batch size for stochastic gradient, and N_e is the number of epochs for training $u(\boldsymbol{x};\Theta)$ and $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f)$. The algorithms consist of three steps: solving PDE, training KRnet and refining the training set.

Remark 3. In both strategies, we use uniform samples to approximate and minimize the cross entropy to obtain $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(1)})$, after which either equation (25) or equation (27) can be employed. The main reason of doing this is to use uniform samples to capture the modes if the residual-induced distribution is multimodal. The obtained PDF $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(1)})$ provides a good initialization when equation (27) is employed to seek $\hat{p}_{\mathsf{KRnet}}(\boldsymbol{x};\Theta_f^{*,(i)})$ with $i>1$. This choice is usually not necessary if the modes are not strongly localized or the location of the modes can be encoded into the prior distribution of KRnet through a Gaussian mixture model.

Remark 4. DAS-R maintains the L_2 norm of the squared residual by generating a completely new training set while DAS-G modifies the weight function in equation (14) by adding new random samples to the current training set. According to

Lemma 2, we should achieve variance reduction for a fixed sample size when the residual-induced distribution is well approximated and the ratio $r^2(\cdot)/\hat{p}_{\text{KRnet}}(\cdot;\Theta_f^{*,(k)})$ is bounded. However, the boundness of $r^2/\hat{p}_{\text{KRnet}}$ is not guaranteed in the current version of DAS-R. This implies that DAS-G may be more robust than DAS-R.

Algorithm 1 DAS-R for PDEs.

```
Input: Initial $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f^{(0)})$ , $u(\boldsymbol{x};\Theta^{(0)})$, maximum epoch number $N_e$, batch size $m$, initial training set $\mathsf{S}_{\Omega,0}=\{\boldsymbol{x}_{\Omega,0}^{(i)}\}_{i=1}^{N_r}$ and $\mathsf{S}_{\partial\Omega,0}=\{\boldsymbol{x}_{\partial\Omega,0}^{(i)}\}_{i=1}^{N_b}$.
 1: for $k=0:N_{\text{adaptive}}-1$ do
 2:   // Solve PDE
 3:   for $i=1:N_e$ do
 4:     for $j$ steps do
 5:       Sample $m$ samples from $\mathsf{S}_{\Omega,k}$.
 6:       Sample $m$ samples from $\mathsf{S}_{\partial\Omega,k}$.
 7:       Update $u(\boldsymbol{x};\Theta)$ by descending the stochastic gradient of $J_N^{\text{IS}}(u(\boldsymbol{x};\Theta))$ (see equation (28)).
 8:     end for
 9:   end for
10:   // Train KRnet
11:   for $i=1:N_e$ do
12:     for $j$ steps do
13:       Sample $m$ samples from $\mathsf{S}_{\Omega,k}$.
14:       Update $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f)$ by descending the stochastic gradient of $H(\hat{r}_{\boldsymbol{X}},\hat{p}_{\text{KRnet}})$ (see equation (25)).
15:     end for
16:   end for
17:   // Refine training set
18:   Generate $\mathsf{S}_{\Omega,k+1}\subset\Omega$ through $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k+1)})$.
19: end for
Output: $u(\boldsymbol{x};\Theta_N^*)$
```

Algorithm 2 DAS-G for PDEs.

```
Input: Initial $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f^{(0)})$ , $u(\boldsymbol{x};\Theta^{(0)})$, maximum epoch number $N_e$, batch size $m$, initial training set $\mathsf{S}_{\Omega,0}=\{\boldsymbol{x}_{\Omega,0}^{(i)}\}_{i=1}^{n_r}$ and $\mathsf{S}_{\partial\Omega,0}=\{\boldsymbol{x}_{\partial\Omega,0}^{(i)}\}_{i=1}^{N_b}$.
 1: for $k=0:N_{\text{adaptive}}-1$ do
 2:   // Solve PDE
 3:   for $i=1:N_e$ do
 4:     for $j$ steps do
 5:       Sample $m$ samples from $\mathsf{S}_{\Omega,k}$.
 6:       Sample $m$ samples from $\mathsf{S}_{\partial\Omega,k}$.
 7:       Update $u(\boldsymbol{x};\Theta)$ by descending the stochastic gradient of $J_N(u(\boldsymbol{x};\Theta))$ (see equation (3)).
 8:     end for
 9:   end for
10:   // Train KRnet
11:   for $i=1:N_e$ do
12:     for $j$ steps do
13:       Sample $m$ samples from $\mathsf{S}_{\Omega,k}$.
14:       Update $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f)$ by descending the stochastic gradient of $H(\hat{r}_{\boldsymbol{X}},\hat{p}_{\text{KRnet}})$ (see equation (25)).
15:     end for
16:   end for
17:   // Refine training set
18:   Generate $\mathsf{S}_{\Omega,k+1}^{g}\subset\Omega$ with size $n_r$ through $\hat{p}_{\text{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k+1)})$.
19:   $\mathsf{S}_{\Omega,k+1}=\mathsf{S}_{\Omega,k}\cup\mathsf{S}_{\Omega,k+1}^{g}$.
20: end for
Output: $u(\boldsymbol{x};\Theta_N^*)$
```

5. Analysis of DAS

As discussed in section 4.3, the key point of our DAS-PINNs method is to achieve variance reduction for the discretization of the residual loss, based on which we expect to improve the accuracy of the approximate solution. Under certain conditions, we show that the expectation of error bound becomes smaller when the adaptive sampling strategy is employed.

Assumption 1. [35] In problem (1), we let $F=\mathscr{H}$ be a Hilbert space and $\mathcal{L}$ a linear operator. Assume that the differential operator $\mathcal{L}$ and the boundary operator $\mathfrak{b}$ satisfy

$$C_1\|v\|_{2,\Omega}\le\|\mathcal{L}v\|_{2,\Omega}+\|\mathfrak{b}v\|_{2,\partial\Omega}\le C_2\|v\|_{2,\Omega}\quad\forall v\in\mathscr{H}\tag{31}$$

where $\mathscr{H}$ is a Hilbert space defined on Ω and the positive constants C_1 and C_2 are independent of v.

The above condition is called the stability bound [35], which is essential to the existence and uniqueness of problem (1). Except for this assumption, the following two assumptions for the relationship between $r(\boldsymbol{x};\Theta_N^{*,(k)})$ and $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$ are given.

Assumption 2. Assume that $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)})$ is the optimal candidate for the change of measure in equation (11)

$$\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k)}) = c_k r^2(\boldsymbol{x};\Theta_N^{*,(k)}) \tag{32}$$

where $c_k = 1/\int_\Omega r^2(\boldsymbol{x};\Theta_N^{*,(k)})d\boldsymbol{x}$ is the normalization constant.

Assumption 3. Let

$$R_k = \frac{1}{N_r}\sum_{i=1}^{N_r}\frac{r^2(\boldsymbol{x}_\Omega^{(i)};\Theta_N^{*,(k)})}{\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}_\Omega^{(i)};\Theta_f^{*,(k-1)})}$$

be the discrete residual loss at the k-th stage, where each $\boldsymbol{x}_\Omega^{(i)}$ is drawn from $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k-1)})$. Assume that $r^2(\boldsymbol{x}_\Omega^{(i)};\Theta_N^{*,(k)})/\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x}_\Omega^{(i)};\Theta_f^{*,(k-1)}) \in [\tau_1,\tau_2]$ almost surely for each $i=1,2,\ldots,N_r$.

At each adaptivity stage, the error of the approximate solution is estimated as follows.

Theorem 1. *Let $u(\boldsymbol{x};\Theta_N^{*,(k)}) \in F$ be a solution of (3) where the collocation points are independently drawn from $\hat{p}_{\mathrm{KRnet}}(\boldsymbol{x};\Theta_f^{*,(k-1)})$. Suppose that Assumption 1 and Assumption 3 are satisfied. Given $0<\varepsilon<1$, the following error estimate holds*

$$\left\|u(\boldsymbol{x};\Theta_N^{*,(k)}) - u(\boldsymbol{x})\right\|_{2,\Omega} \le \sqrt{2}C_1^{-1}\left(R_k + \varepsilon + \left\|b(\boldsymbol{x};\Theta_N^{*,(k)})\right\|_{2,\partial\Omega}^2\right)^{\frac{1}{2}}.$$

with probability at least $1-\exp(-2N_r\varepsilon^2/(\tau_2-\tau_1)^2)$.

The approximate solutions at two adjacent adaptivity stages satisfy:

Corollary 1. *Under the same conditions of Theorem 1, suppose that Assumption 2 is satisfied and the boundary loss $J_b(u)$ is zero, then the following inequality holds*

$$\mathbb{E}(R_{k+1}) \le \mathbb{E}(R_k).$$

Theorem 1 provides an error estimate for the approximate solution similar to the results in [40]. Corollary 1 outlines the error behavior of a sequence of approximate solutions induced by adaptivity. However, it is not quite straightforward to quantify the decay of the error due to adaptive refinement. For example, for DAS-R the reduction in variance is up to a tail probability as shown in Lemma 2 and for DAS-G the loss is changed at each adaptivity stage due to the modification of the training set. These issues are left for future study.

6. Numerical experiments

In this section, we conduct some numerical experiments to demonstrate the effectiveness of the proposed DAS-PINNs method, including two low-dimensional and low regularity test problems, one high-dimensional linear test problem, and one high-dimensional nonlinear test problem. Due to the curse of dimensionality, data are sparse in high-dimensional spaces [20,19,21], which implies that effective samples should be able to deal with localized information. We mainly use low-dimensional and low regularity test problems to demonstrate that the sampling strategy affects significantly the performance of neural network approximation if the residual is strongly localized. For comparison, we also test the performance of the residual-based adaptive refinement (RAR) method [52,42] (see section 6.2 and section 6.3) for high-dimensional problems, where RAR searches uniform samples to find those with large residuals and add them to the current training set. RAR is similar to DAS-G, where the main difference is that the selection of new samples in DAS-G is completely guided by an optimization problem while RAR relies partially on the intuition of users. All deep neural network models are trained by the ADAM method [34]. The penalty parameter in equation (3) is set to $\hat{\gamma}=1$. The activation function of $u(\boldsymbol{x};\Theta)$ is set to the hyperbolic tangent function. The activation function of KRnet is the rectified linear unit (ReLU) function since we only use the KRnet for density approximation.

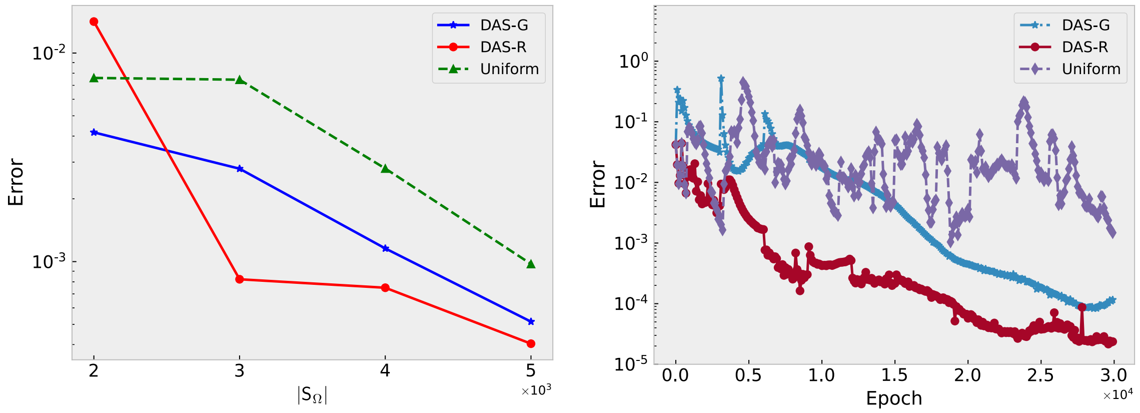


Fig. 1. Approximation errors for the two-dimensional peak test problem. Left: The error w.r.t. sample size $|S_\Omega|$; Right: The error w.r.t. epoch for $|S_\Omega| = 5 \times 10^3$.

6.1. *Low-dimensional and low regularity test problems*

In this part, two-dimensional low regularity problems are considered, where the solution of the first one has a peak and the solution of the second one has two peaks.

6.1.1. *Two-dimensional peak problem*

The following elliptic equation is considered

$$\begin{aligned} -\Delta u(x_1, x_2) &= s(x_1, x_2) \quad \text{in } \Omega, \\ u(x_1, x_2) &= g(x_1, x_2) \quad \text{on } \partial\Omega, \end{aligned} \tag{33}$$

where the computation domain is $\Omega = [-1, 1]^2$. In order to quantify the error, we use the following reference solution

$$u(x_1, x_2) = \exp\left(-1000[(x_1 - r_c)^2 + (x_2 - r_c)^2]\right),$$

which has a peak at the point (r_c, r_c) and decreases rapidly away from (r_c, r_c). This test problem is often used to test the performance of adaptive finite element methods [53,24].

We choose a six-layer fully connected neural network $u(\boldsymbol{x}; \Theta)$ with 32 neurons to approximate the solution. For KRnet, we take $L = 6$ affine coupling layers, and two fully connected layers with 24 neurons for each affine coupling layer. The number of epochs for training both $u(\boldsymbol{x}; \Theta)$ and $p(\boldsymbol{x}; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 500$. Here, we set $(r_c, r_c) = (0.5, 0.5)$. To assess the effectiveness of our DAS methods, we generate a uniform meshgrid with size 256×256 in $[-1, 1]^2$ and compute the mean square error on these grid points.

In Fig. 1, we plot the approximation errors given by different sampling strategies with respect to the sample size in the left plot and with respect to the number of epochs in the right plot. For each $|S_\Omega|$, we take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For DAS strategies, the numbers of adaptivity iterations are set to $N_{\text{adaptive}} = 4, 6, 8, 10$ for $|S_\Omega| = 2 \times 10^3, 3 \times 10^3, 4 \times 10^3, 5 \times 10^3$ respectively, and $n_r = 500$ is set for the DAS-G strategy (see section 4.3). For the uniform sampling strategy, the number of epochs is set to be the same as the total number of epochs of each DAS method. It is clear that for this test problem the DAS methods (DAS-G and DAS-R) have a better performance than the uniform sampling strategy and DAS-R performs better than DAS-G. For the same sample size, both DAS-R and DAS-G yield a smaller error than the uniform sampling method. In terms of the number of epochs, the errors of DAS-R and DAS-G decay in a more consistent way than the uniform sampling method.

In Fig. 2 we compare the exact solution, the DAS solutions given by 5×10^3 nonuniform samples and the approximate solution given by 5×10^3 uniform samples. It is seen that DAS methods are much more effective than the uniform sampling method to capture the information in the region of low regularity. Fig. 3 shows the error evolution of DAS-R at different adaptivity iteration steps. It is seen that the approximation error drops as the adaptivity iteration step k increases, which is consistent with Corollary 1, and the relaxation time for the optimization iterations reduces as well. Fig. 4 shows the evolution of the training sets ($|S_\Omega| = 5 \times 10^3$) of DAS-R method with respect to adaptivity iterations $k = 1, 4, 7, 9$, where the initial training set $S_{\Omega,0}$ consists of uniform collocation points on Ω (see section 4.3). It is seen that the largest density of $S_{\Omega,1}$ for DAS-R is around $[0.5, 0.5]$ since $S_{\Omega,k}$ is consistent with the residual-induced distribution. However, we also expect that the tail of the residual-induced distribution becomes heavier as k increases since the adaptivity tries to make the residual-induced distribution more uniform, which is illustrated by $S_{\Omega,4}$, $S_{\Omega,7}$ and $S_{\Omega,9}$.

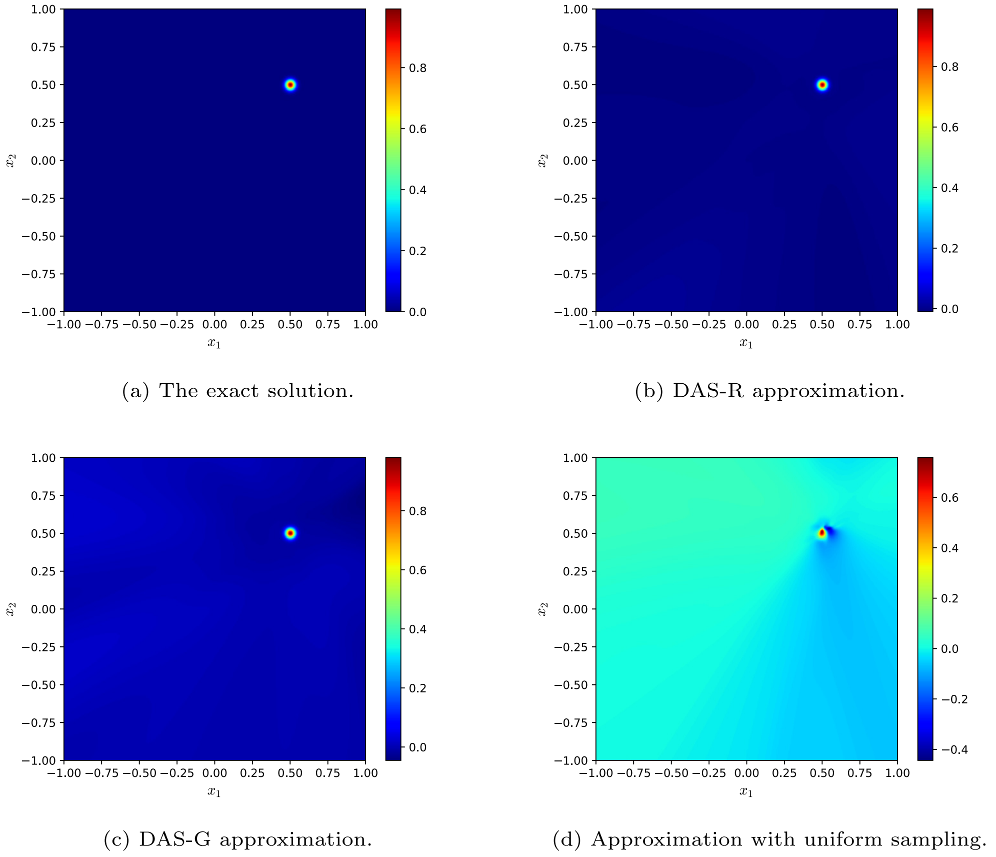


Fig. 2. Solutions, two-dimensional peak test problem. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

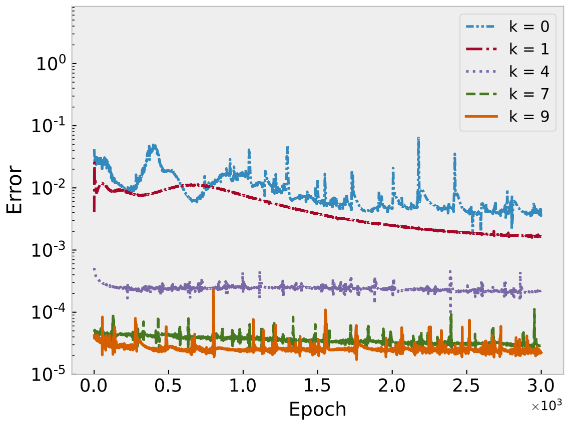


Fig. 3. The errors of DAS-R at certain adaptivity iteration steps for the two-dimensional peak test problem. $|S_\Omega| = 5 \times 10^3$.

6.1.2. Two-dimensional test problem with two peaks

In this test problem, we consider the following equation

$$\begin{aligned} -\nabla \cdot \left[u(x_1, x_2)\nabla(x_1^2 + x_2^2) \right] + \nabla^2 u(x_1, x_2) &= s(x_1, x_2) \quad \text{in } \Omega, \\ u(x_1, x_2) &= g(x_1, x_2) \quad \text{on } \partial\Omega, \end{aligned} \tag{34}$$

where the computation domain is $\Omega = [-1, 1]^2$. The exact solution of (34) is chosen as

$$u(x_1, x_2) = \mathrm{e}^{-1000[(x_1-0.5)^2+(x_2-0.5)^2]} + \mathrm{e}^{-1000[(x_1+0.5)^2+(x_2+0.5)^2]},$$

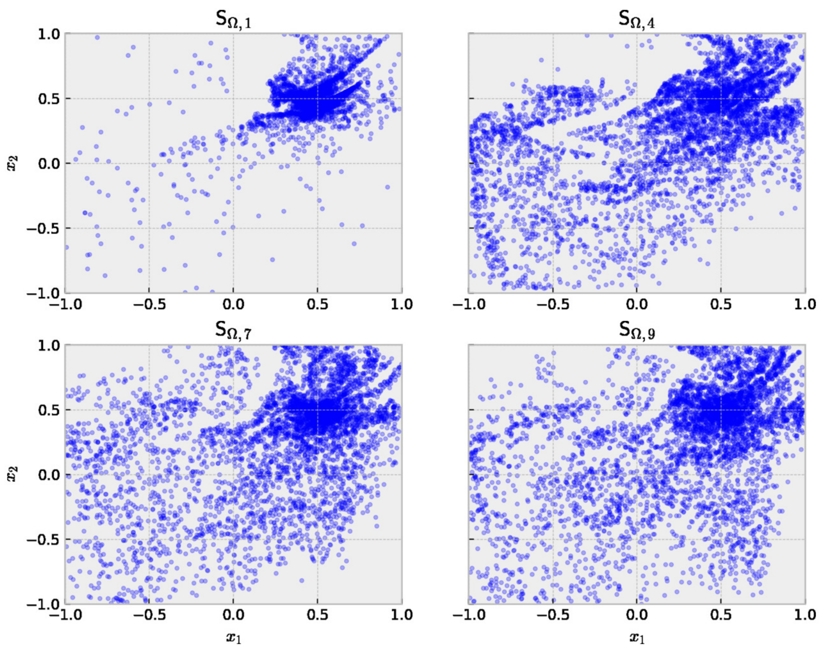


Fig. 4. The evolution of $S_{\Omega,k}$ in DAS-R for the two-dimensional peak test problem.

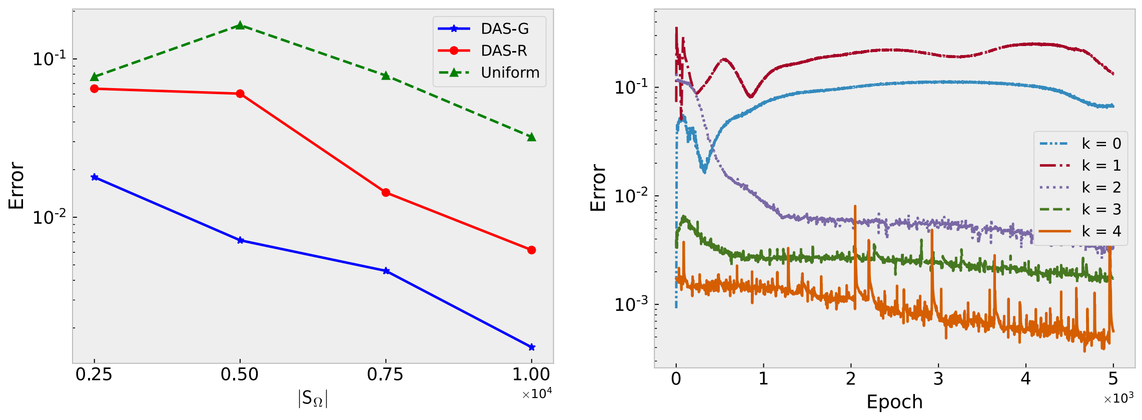


Fig. 5. Approximation errors for the two-dimensional test problem with two peaks. Left: The error w.r.t. sample size $|S_\Omega|$; Right: The errors of DAS-G at each adaptivity iteration steps. $|S_\Omega| = 10^4$.

which has two peaks at the points $(0.5, 0.5)$ and $(-0.5, -0.5)$. Here, the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution.

We choose a six-layer fully connected neural network $u(\boldsymbol{x}; \Theta)$ with 64 neurons to approximate the solution of (34). For KRnet, we take $L = 8$ affine coupling layers, and two fully connected layers with 48 neurons for each affine coupling layer. The number of epochs for training both $u(\boldsymbol{x}; \Theta)$ and $p(\boldsymbol{x}; \Theta_f)$ is set to $N_e = 5000$. The learning rate for ADAM optimizer is set to 0.0001, and the batch size is set to $m = 500$. Again, we generate a uniform meshgrid with size 256×256 in $[-1, 1]^2$ and compute the mean square error on these grid points to assess the effectiveness of our DAS methods.

Fig. 5 shows the approximation errors for this test problem, where the left one displays the errors with respect to the sample size $|S_\Omega|$ for different sampling strategies, and the right one shows the error evolution of DAS-G at different adaptivity iteration steps. For each $|S_\Omega|$, we again take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For the DAS-G strategy, the numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$ (also for DAS-R), and the numbers of collocation points in $S_\Omega^g(k = 1, 2, 3, 4)$ is set to $n_r = 500, 1 \times 10^3, 1.5 \times 10^3, 2 \times 10^3$ for $|S_\Omega| = 2.5 \times 10^3, 5 \times 10^3, 7.5 \times 10^3, 10^4$ respectively. For the uniform sampling strategy, we train the model with 2.5×10^4 epochs to match the total number of epochs of DAS methods. From Fig. 5, it is seen that for this test problem our DAS methods (DAS-G and DAS-R) have a better performance than the uniform sampling strategy and DAS-G performs better than DAS-R. It is also seen that the error decreases as the adaptivity iteration step k increases.

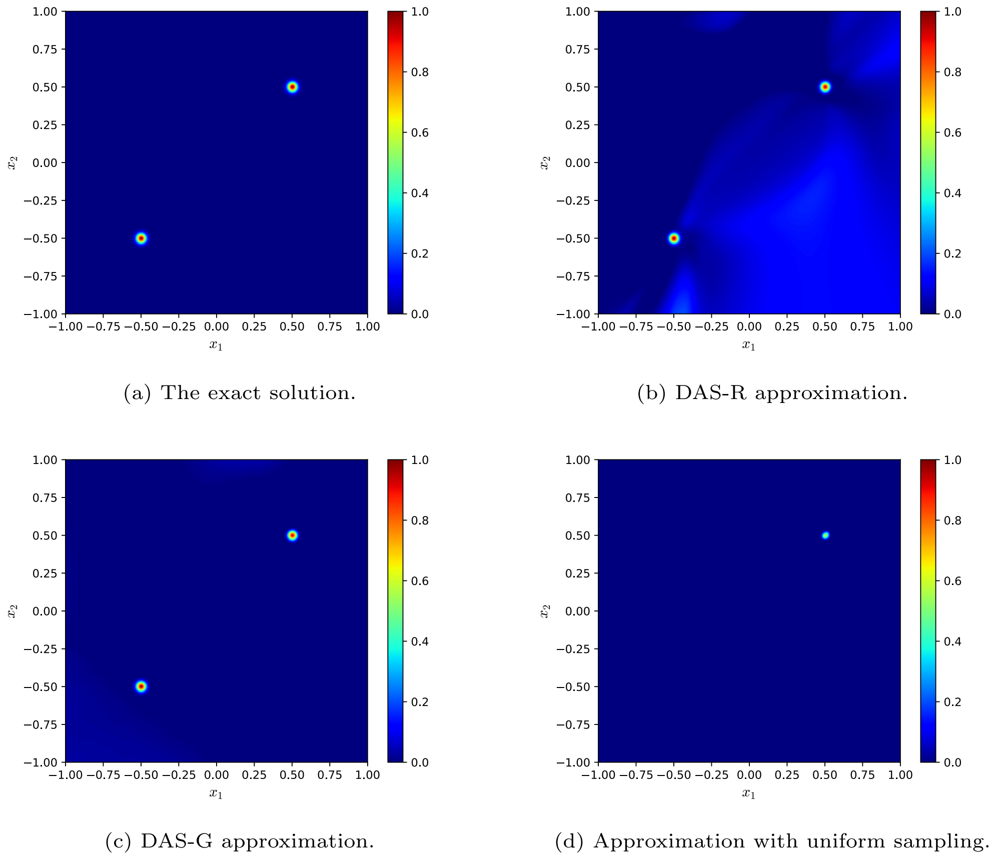

(a) The exact solution. (b) DAS-R approximation. (c) DAS-G approximation. (d) Approximation with uniform sampling.

Fig. 6. Solutions, two-dimensional test problem with two peaks.

In Fig. 6 we compare the exact solution, the DAS solutions given by 10^4 nonuniform samples and the approximate solution given by 10^4 uniform samples. It is seen that DAS methods are much more effective than the uniform sampling method to capture the information around the two peaks. Fig. 7 shows the evolution of $\mathrm{S}_{\Omega,k}^g$ of DAS-G method with respect to adaptivity iterations $k = 1, 2, 3, 4$ ($|\mathrm{S}_{\Omega,k}^g| = 2 \times 10^3$), where the initial training set $\mathrm{S}_{\Omega,0}$ consists of uniform collocation points on Ω (see section 4.3). $\mathrm{S}_{\Omega,1}^g$ shows that the error profile has two peaks. After the training set is augmented with $\mathrm{S}_{\Omega,1}$, the error profile becomes more flat as shown by the distribution of $\mathrm{S}_{\Omega,2}^g$. After the training set is augmented with $\mathrm{S}_{\Omega,2}$, the largest error is found again around the two peaks, and then the subsequent augmentation of the training set yields a more flat error profile. Such a pattern is repeated until no improvement can be reached.

6.2. High-dimensional linear test problems

Next we consider the d-dimensional elliptic equation

$$-\Delta u(\boldsymbol{x}) = s(\boldsymbol{x}), \quad \boldsymbol{x} \text{ in } \Omega = [-1, 1]^d, \tag{35}$$

with an exact solution

$$u(\boldsymbol{x}) = \mathrm{e}^{-10\|\boldsymbol{x}\|_2^2},$$

where the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. We are interested in cases with a large $d > 3$. Note that the geometric properties of high-dimensional spaces are significantly different from our intuitions on low-dimensional ones, e.g., most of the volume of a high-dimensional cube is located around its corners [20,19,21]. If we use uniform samples to generate S_Ω, most of the collocation points in S_Ω are near the surface of the hypercube. Since the information of the exact solution is mainly from the neighborhood of the origin, most of the samples in S_Ω may not contribute to training the neural network when d is large enough.

We choose a six-layer fully connected neural network $u(\boldsymbol{x}; \Theta)$ with 64 neurons to approximate the solution. For KRnet, we set $K = 3$ and take $L = 6$ affine coupling layers, and two fully connected layers with 64 neurons for each affine coupling layer. The number of epochs for training both $u(\boldsymbol{x}; \Theta)$ and $p(\boldsymbol{x}; \Theta_f)$ is set to $N_e = 3000$. The learning rate for ADAM

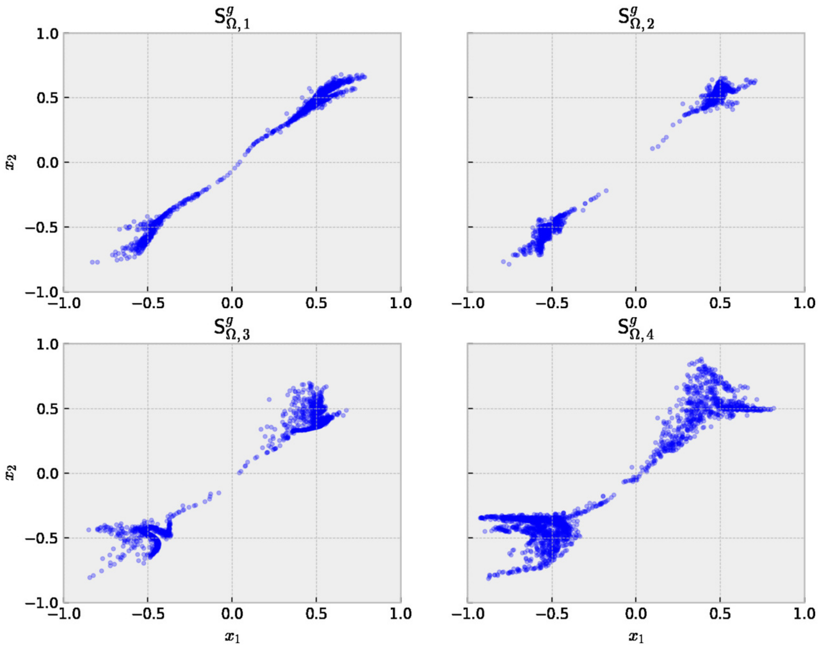


Fig. 7. The evolution of $S^g_{\Omega,k}$ in DAS-G for the two-dimensional test problem with two peaks.

optimizer is set to 0.0001, and the batch size is set to $m = 5000$. The numbers of adaptivity iterations is set to $N_{\text{adaptive}} = 5$. To measure the quality of approximation, we generate a tensor grid with n_t^d points around the origin (in $[-0.1, 0.1]^d$) where n_t is the number of nodes for each dimension. We define the relative error

$$\text{Relative error} = \frac{\|\boldsymbol{u}_{\text{NN}} - \boldsymbol{u}\|_2}{\|\boldsymbol{u}\|_2},$$

where $\boldsymbol{u}_{\text{NN}}$ and $\boldsymbol{u}$ denote two vectors whose elements are the function values of $u(\boldsymbol{x};\Theta)$ and $u(\boldsymbol{x})$ at the tensor grid respectively.

We first investigate the relation between the error and the dimensionality d when the uniform sampling strategy is employed. Fig. 8(a) shows the relative errors in terms of a varying d for a sample size $|S_\Omega| = 2 \times 10^5$. To roughly match the number of grid points for different d, we set $n_t = 16, 6, 4, 3$ for $d = 4, 6, 8, 10$ respectively. It is seen that the relative error grows quickly to $O(1)$ as d increases. However, as shown in Fig. 8(b), all training losses are finally close to zero for $d = 4, 6, 8, 10$. This is consistent with the fact that in a high-dimensional space most of the uniform samples are located around the boundary, where the solution is close to zero. The optimizer is then in favor of the trivial solution since there are not sufficient samples to resolve the peak at the origin. This phenomenon demonstrates that the uniform sampling method may become less effective as d increases and the convergence of the approximate solution is highly dependent on the choice of S_Ω for a large d.

Fig. 9 shows the relative errors for the uniform sampling strategy, the residual-based adaptive refinement (RAR) method proposed in [52], DAS-R and DAS-G, where different numbers of samples $|S_\Omega|$ are considered. For each $|S_\Omega|$, we again take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error. For the DAS-G strategy, the numbers of collocation points in $S^g_{\Omega,k}$ $(k = 1, 2, 3, 4)$ are set to $n_r = 10^4, 2 \times 10^4, 3 \times 10^4, 4 \times 10^4$ for $|S_\Omega| = 5 \times 10^4, 10^5, 1.5 \times 10^5, 2 \times 10^5$ respectively. For the uniform sampling strategy, we train the model with 1.5×10^4 epochs to match the total number of epochs of DAS methods. For the heuristic method RAR, the numbers of collocation points in $S^g_{\Omega,k}$ $(k = 1, 2, 3, 4)$ are set to $n_r = 5 \times 10^3, 10^4, 1.5 \times 10^4, 2.5 \times 10^4$ for $|S_\Omega| = 5 \times 10^4, 10^5, 1.5 \times 10^5, 2 \times 10^5$ respectively. From Fig. 9, it can be seen that both DAS-G and DAS-R improve the accuracy significantly compared to the uniform sampling strategy and RAR. In addition, the error of DAS-G decreases slightly faster than that of DAS-R for this test problem. Table 1 shows the training time and the error for the uniform sampling strategy, the residual-based adaptive refinement (RAR) method proposed in [52], DAS-R, and DAS-G. It can be seen that the training time of DAS-G is less than that of DAS-R. Moreover, the training time of DAS-G is approximately equal to that of the uniform sampling strategy. However, the errors of the uniform sampling strategy and RAR are much larger than that of DAS since the uniform sampling strategy and RAR are not able to accurately discretize the loss functional for this low-regularity high-dimensional problem. In Fig. 10 we compare the error evolution of different sampling strategies. From the left plot of Fig. 10, as the number of epochs increases, the errors of DAS-G and DAS-R decrease quickly, while the errors of RAR and the uniform sampling strategy do not decrease. This result suggests that for high-dimensional problems DAS methods are able to achieve a good

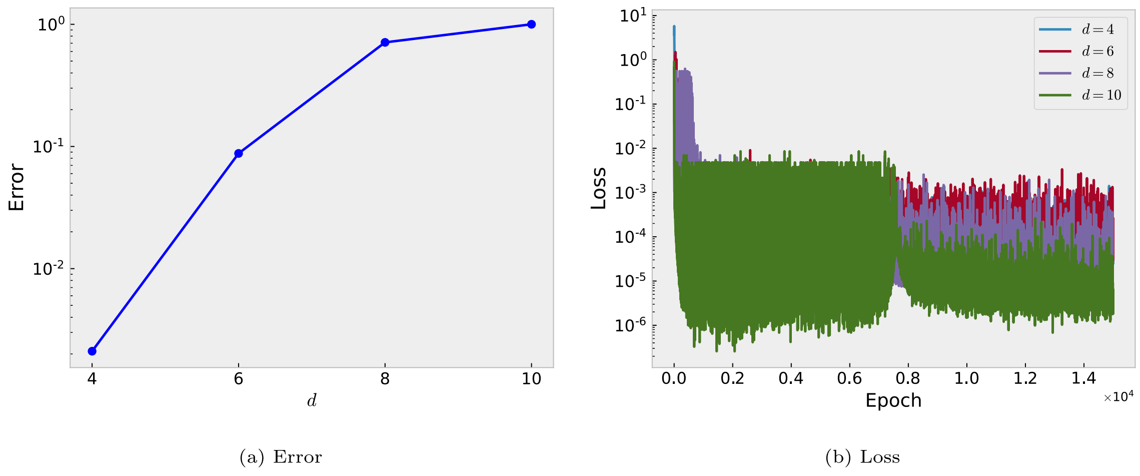


Fig. 8. The convergence behavior of high-dimensional PDEs with uniform sampling method. Loss is close to zero, but the error is still large for the ten-dimensional test problem.

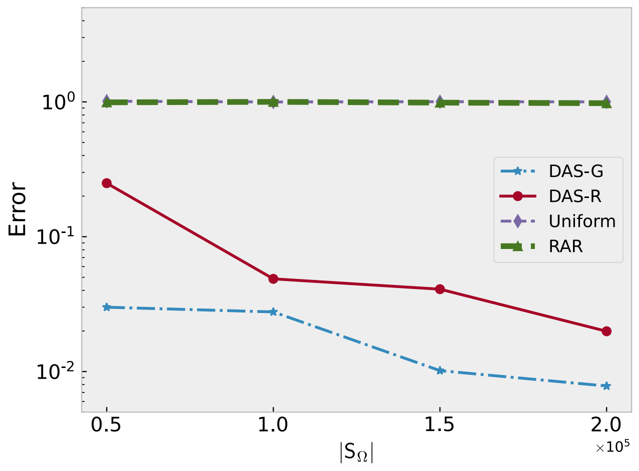


Fig. 9. The error w.r.t. sample size $|S_\Omega|$, ten-dimensional linear test problem.

approximation with a relatively small number of nonuniform samples while much more uniform samples are needed for the same accuracy. The right plot of Fig. 10 shows the error of DAS-G at each adaptivity iteration step k. It is seen that the error drops dramatically after we refine the solution using $S_{\Omega,1}$.

Figs. 11 and 12 show 3000 samples from the training sets ($|S_\Omega| = 2 \times 10^5$) DAS-R and DAS-G for the first four adaptivity iterations, where the components x_6 and x_7 are used for visualization. We have also checked the other components, and no significantly different results were found. For DAS-R, 3000 samples are randomly chosen from $S_{\Omega,k}$ ($k = 1, 2, 3, 4$). For DAS-G, 3000 samples for visualization are randomly selected from $S^g_{\Omega,k}$ ($k = 1, 2, 3, 4$). It is seen that the profile of $S_{\Omega,k}$ is gradually flattened as k increases, meaning the nonuniform samples are able to smooth the error profile which has a peak around the origin. As for DAS-G, the improvement takes a similar path. $S^g_{\Omega,1}$ shows that the error profile has a peak around the origin. After the training set is augmented with $S_{\Omega,1}$, the error profile becomes more flat as shown by the distribution of $S^g_{\Omega,2}$. This is expected since more collocation points are added to the neighborhood of the origin which should reduce the error over there. Such a pattern is similar to what we have observed in Fig. 7. In Fig. 13, we compare the evolution of the variance of the residual for the training with 5×10^4 samples. We estimate the variance of residual using 59049 grid points around the origin (these points are also used to compute the relative errors in the above discussion). It is clear that both DAS-R and DAS-G achieve the variance reduction significantly compared with RAR, which helps reduce the statistical error dramatically for a fixed sample size. Looking more closely, the variance of DAS-R has a transition between two consecutive adaptivity iterations, resulting in the oscillation of errors for DAS-R as observed in the left plot of Fig. 10. From Fig. 13, it can be seen that DAS-G appears more robust than DAS-R for this test problem. We may adjust the communication pattern between the PDE model and the PDF model to reduce the oscillations, which will be left for future study.

Table 1
Training time and error for different $|S_\Omega|$ and sampling strategies, ten-dimensional linear test problem.

sampling strategy / $\lvert S_\Omega \rvert$	DAS-G		DAS-R		Uniform		RAR	
	time	error	time	error	time	error	time	error
5×10^4	1.83 h	0.030	3.38 h	0.250	1.90 h	1.011	1.45 h	0.993
10^5	3.64 h	0.028	6.95 h	0.049	3.92 h	0.999	3.03 h	1.001
1.5×10^5	5.61 h	0.010	10.29 h	0.041	5.85 h	1.003	4.66 h	0.988
2×10^5	7.55 h	0.008	13.49 h	0.020	7.90 h	1.001	5.74 h	0.978

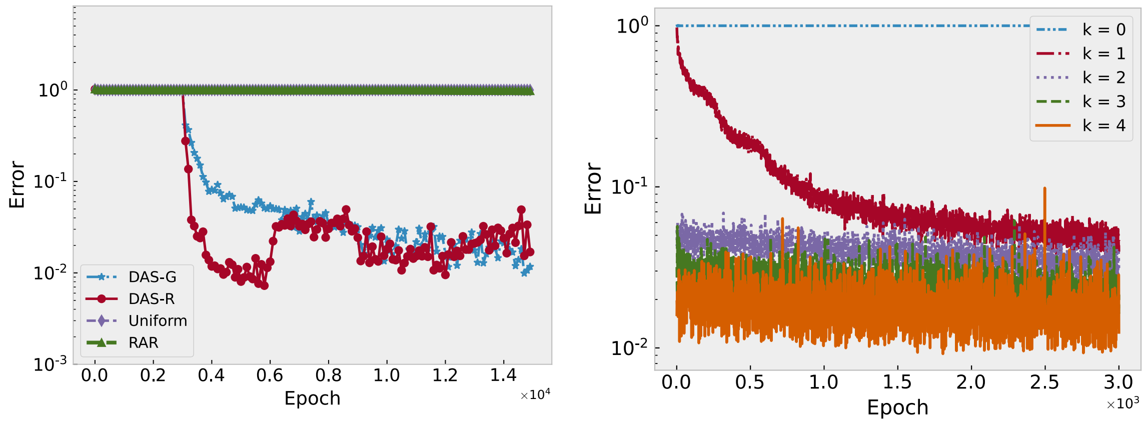


Fig. 10. The error evolution of different sampling strategies with $|S_\Omega| = 2 \times 10^5$ and $d = 10$ (high-dimensional linear test problem). Left: A comparison of DAS-G, DAS-R and the uniform sampling method; Right: The error evolution of DAS-G at different adaptivity iteration steps.

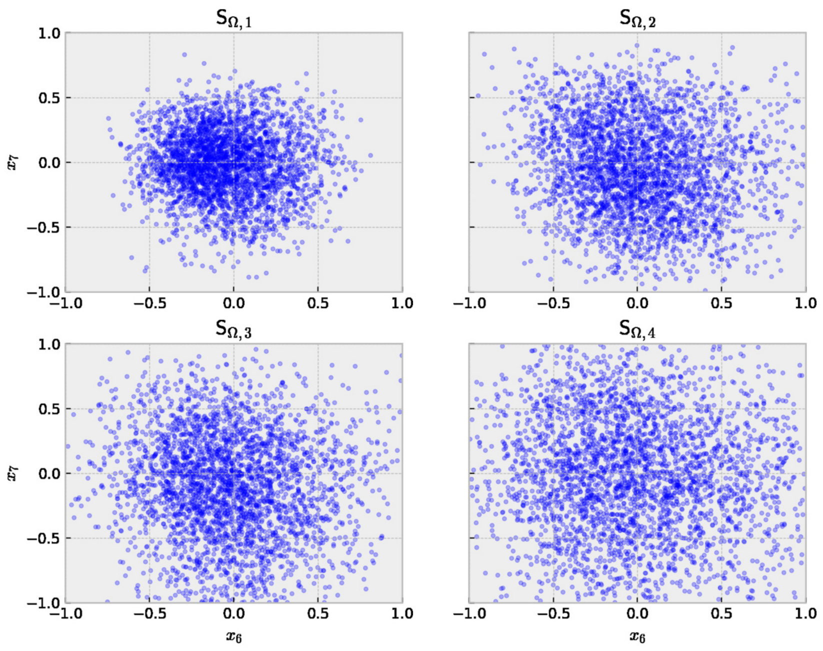


Fig. 11. The evolution of $S_{\Omega,k}$ in DAS-R, ten-dimensional linear test problem.

6.3. High-dimensional nonlinear test problem

In this part, the ten-dimensional nonlinear partial differential equation considered is

$$-\Delta u(\boldsymbol{x}) + u(\boldsymbol{x}) - u^3(\boldsymbol{x}) = s(\boldsymbol{x}), \quad \boldsymbol{x} \text{ in } \Omega = [-1,1]^{10}. \tag{36}$$

The exact solution is set to be the same as (35), and the Dirichlet boundary condition on $\partial\Omega$ is given by the exact solution. The settings of $u(\boldsymbol{x}; \Theta)$ and KRnet are the same as those in section 6.2.

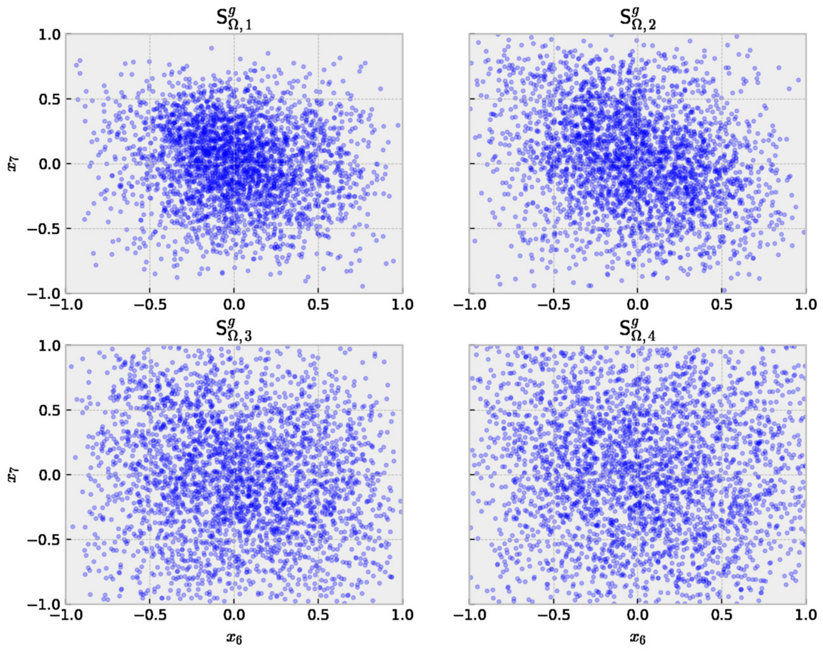


Fig. 12. The evolution of $S^g_{\Omega,k}$ in DAS-G, ten-dimensional linear test problem.

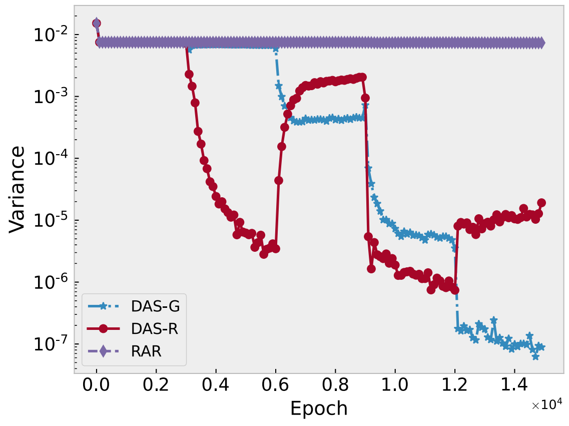


Fig. 13. The evolution for the variance of residual, ten-dimensional linear test problem.

The observations are similar to those for the high-dimensional linear problem. Fig. 14 shows the relative errors for the uniform sampling strategy, RAR, DAS-R and DAS-G, where different numbers of samples $|S_\Omega|$ are considered. Again, we take three runs with different random seeds for initialization and compute the mean error of the three runs as the final error for each $|S_\Omega|$. For the DAS-G strategy, the numbers of collocation points in $S^g_{\Omega,k}$ $(k = 1, 2, 3, 4)$ are set to the same as those in section 6.2. For the uniform sampling strategy, we train the model with 1.5×10^4 epochs to match the total number of epochs of DAS methods. For the heuristic method RAR, the numbers of collocation points in $S^g_{\Omega,k}$ $(k = 1, 2, 3, 4)$ are set to $n_r = 5 \times 10^3, 10^4, 1.5 \times 10^4, 2.5 \times 10^4$ for $|S_\Omega| = 5 \times 10^4, 10^5, 1.5 \times 10^5, 2 \times 10^5$ respectively. We compare the training time for different sampling strategies in Table 2. The results are similar to those in section 6.2. It is seen that the training time of DAS-G is less than that of DAS-R. Both DAS-G and DAS-R improve the accuracy significantly compared to the uniform sampling strategy and RAR, and the training time of DAS-G is approximately equal to that of the uniform sampling strategy. In Fig. 15 we compare the error evolution of different sampling strategies. Similar to the high-dimensional linear problem, the errors of DAS-G and DAS-R decrease quickly while the errors of the uniform sampling strategy and RAR do not decrease. The error behavior of DAS-G at each adaptivity iteration step k is shown in the right plot of Fig. 15. It is seen that the approximation is significantly improved when the adaptivity iteration step k increases from 0 to 1. Fig. 16 and 17 show 3000 samples from the training sets ($|S_\Omega| = 2 \times 10^5$) DAS-R and DAS-G for the first four adaptivity iterations, where the components x_6 and x_7 are used for visualization. For DAS-R, 3000 samples are randomly chosen from $S_{\Omega,k}$ $(k = 1, 2, 3, 4)$.

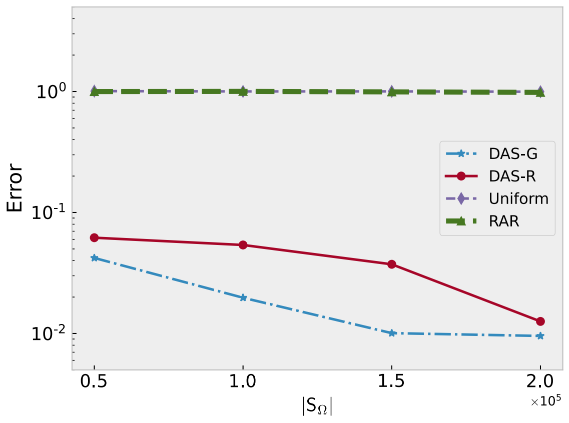
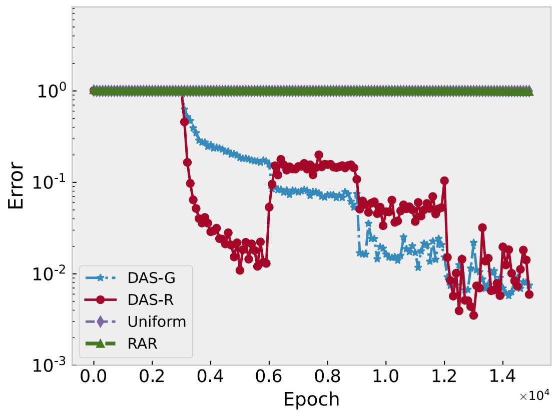


Fig. 14. The error w.r.t. sample size $|S_\Omega|$, ten-dimensional nonlinear test problem.

Table 2
Training time and error for different $|S_\Omega|$ and sampling strategies, ten-dimensional nonlinear test problem.

sampling strategy / $\lvert S_\Omega \rvert$	DAS-G		DAS-R		Uniform		RAR	
	time	error	time	error	time	error	time	error
5×10^4	1.82 h	0.042	3.44 h	0.062	1.84 h	1.008	1.42 h	0.999
10^5	3.65 h	0.020	6.92 h	0.054	3.86 h	1.001	2.97 h	1.002
1.5×10^5	5.81 h	0.010	10.41 h	0.037	5.73 h	1.002	4.63 h	0.993
2×10^5	7.82 h	0.009	13.87 h	0.013	7.80 h	0.996	5.75 h	0.983



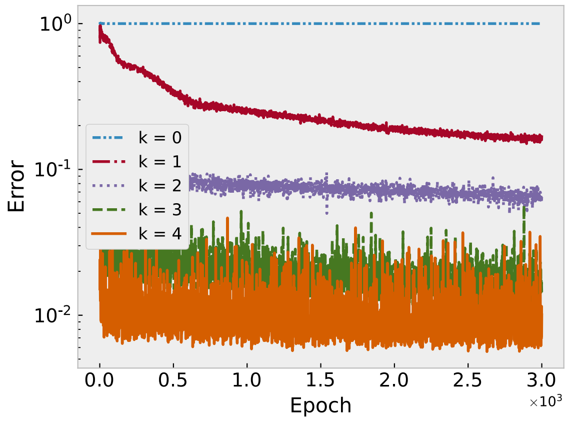


Fig. 15. The error evolution of different sampling strategies with $|S_\Omega| = 2 \times 10^5$ and $d = 10$ (high-dimensional nonlinear test problem). Left: A comparison of DAS-G, DAS-R and the uniform sampling method; Right: The error evolution of DAS-G at different adaptivity iteration steps.

For DAS-G, 3000 samples for visualization are randomly selected from $S^g_{\Omega,k}$ $(k = 1, 2, 3, 4)$. Both DAS-R and DAS-G flatten the error profile through adaptive sampling as we have observed in Figs. 11 and 12. Fig. 18 shows the evolution of the variance of the residual for DAS-R, DAS-G and RAR. The behavior is similar to that in Fig. 13.

7. Conclusion

In this paper we have developed a deep adaptive sampling (DAS) method and coupled it with physics-informed neural networks (PINNs) to improve the neural network approximation of PDEs iteratively. The key idea of DAS is to employ a deep generative model to generate collocation points that are consistent with the distribution induced by an appropriate error indicator function. In this way, the training set is refined according to the regularity of the PDE solution, which follows the similar principle of adaptive mesh refinement of classical numerical methods. Numerical experiments have shown that the DAS method is able to significantly improve the accuracy for the approximation of low regularity problems especially when the dimensionality is relatively large. The proposed DAS method provides a very general and flexible framework for an adaptive learning strategy. There are several possible ways to further improve it. First, DAS consists of two DNN-based models: one model serves as an approximator for the PDE solution and the other one serves as an error indicator for the

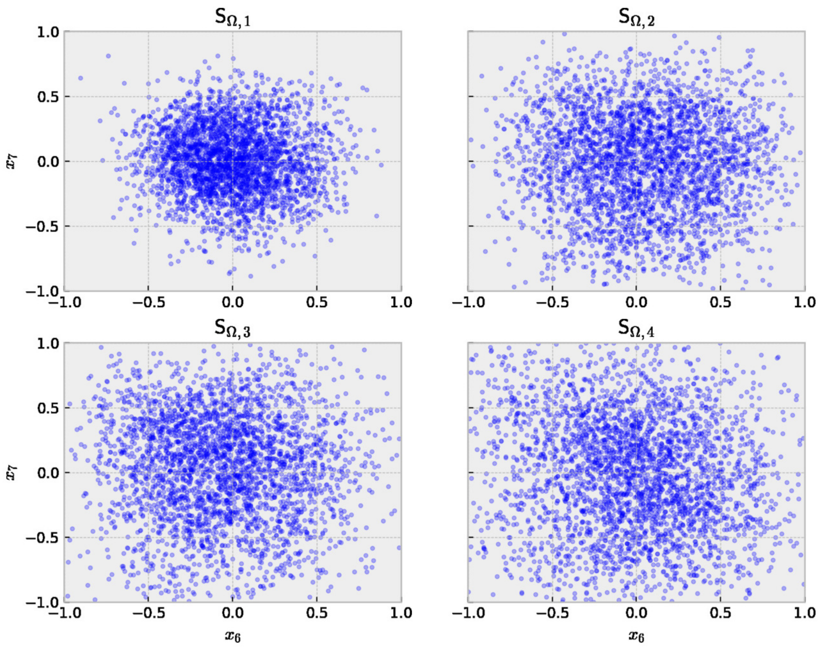


Fig. 16. The evolution of $S_{\Omega,k}$ in DAS-R, ten-dimensional nonlinear test problem.

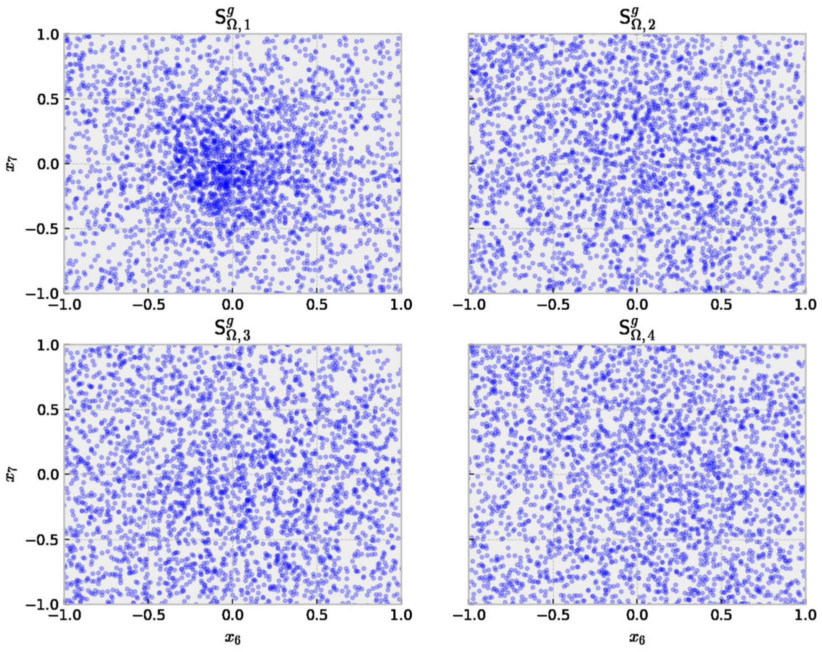


Fig. 17. The evolution of $S^g_{\Omega,k}$ in DAS-G, ten-dimensional nonlinear test problem.

selection of collocation points. Both models can be chosen in terms of a certain criterion. In this work, we use a regular DNN for PDE approximation and KRnet for density approximation and sample generation. Second, the underlying distribution for the training set can be problem dependent. In this work, we choose the residual-induced distribution. We may also use the gradient of the approximation solution to define an indicator distribution. In [22], we employ KRnet to approximate the Fokker-Planck equation, where the collocation points are sampled from the approximate solution. Third, the DAS method is not limited to steady-state PDE problems. We may employ the DAS method on the space-time domain to refine the training set for the approximation of time-dependent problems. Last but not least, the current training process can also be improved. Although the current DAS methods work well enough to demonstrate the effectiveness of the algorithm, many questions

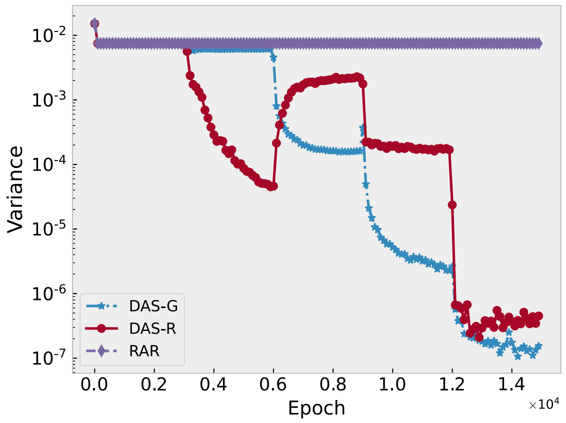


Fig. 18. The evolution for the variance of residual, ten-dimensional nonlinear test problem.

remain open, e.g., what is the optimal way for the two deep models to communicate and what is the optimal sample size for $S^g_{\Omega,k}$. Research on these issues will be reported in forthcoming papers.

CRediT authorship contribution statement

Kejun Tang: Conceptualization, Methodology, Programming, Writing–Original draft preparation. **Xiaoliang Wan**: Conceptualization, Methodology, Writing. **Chao Yang**: Conceptualization, Methodology, Validation, Writing–Reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

K. Tang has been supported by the China Postdoctoral Science Foundation grant 2022M711730. X. Wan has been supported by NSF grant DMS-1913163. C. Yang has been supported by NSFC grant 12131002 and Huawei Technologies Co., Ltd.

Appendix A. Proof of Lemma 1

Proof. We first introduce the following lemma:

Lemma 3. *Consider a perturbed identity matrix* $\mathbf{I}+\delta\mathbf{A}$ *with* $\|\delta\mathbf{A}\|_2 < 1$. *We have*

$$\left\|(\mathbf{I}+\delta\mathbf{A})^{-1}\right\|_2 \le \frac{1}{1-\|\delta\mathbf{A}\|_2}. \tag{A.1}$$

Proof. For any $\boldsymbol{x} \neq 0$, we have

$$\|(\mathbf{I}+\delta\mathbf{A})\boldsymbol{x}\|_2 \ge \|\boldsymbol{x}\|_2 - \|\delta\mathbf{A}\|_2\,\|\boldsymbol{x}\|_2 = (1-\|\delta\mathbf{A}\|_2)\,\|\boldsymbol{x}\|_2 > 0,$$

which implies that $\mathbf{I}+\delta\mathbf{A}$ is nonsingular. We then have

$$\begin{aligned} 1 = \left\|(\mathbf{I}+\delta\mathbf{A})^{-1}(\mathbf{I}+\delta\mathbf{A})\right\|_2 &= \left\|(\mathbf{I}+\delta\mathbf{A})^{-1} + (\mathbf{I}+\delta\mathbf{A})^{-1}\delta\mathbf{A}\right\|_2 \\ &\ge \left\|(\mathbf{I}+\delta\mathbf{A})^{-1}\right\|_2 - \left\|(\mathbf{I}+\delta\mathbf{A})^{-1}\right\|_2 \|\delta\mathbf{A}\|_2, \end{aligned}$$

which yields the conclusion. $\square$

Assume that $\boldsymbol{m}_V^*(\boldsymbol{x}) = (\boldsymbol{v}^*)^{\mathsf{T}}\boldsymbol{q}(\boldsymbol{x})$ and $\boldsymbol{m}_{\hat{\boldsymbol{v}}^*}(\boldsymbol{x}) = (\hat{\boldsymbol{v}}^*)^{\mathsf{T}}\boldsymbol{q}(\boldsymbol{x})$, where $\boldsymbol{q}(\boldsymbol{x}) = [q_1(\boldsymbol{x}), \ldots, q_n(\boldsymbol{x})]^{\mathsf{T}}$ includes the basis functions in $V = \operatorname{span}\{q_i : i = 1, \ldots, n\}$ and the vectors $\boldsymbol{v}^*$ and $\hat{\boldsymbol{v}}^*$ include the coefficients. It is easy to see that $\boldsymbol{v}^*$ and $\hat{\boldsymbol{v}}^*$ satisfy the following linear systems respectively

$$\mathbf{A}\boldsymbol{v}^* = \boldsymbol{b}, \quad \hat{\mathbf{A}}\hat{\boldsymbol{v}}^* = \hat{\boldsymbol{b}}, \tag{A.2}$$

where

$$\hat{a}_{ij} = \frac{1}{N}\sum_{k=1}^{N} q_i(\boldsymbol{x}^{(k)})q_j(\boldsymbol{x}^{(k)}) \approx \langle q_i, q_j\rangle_\rho = a_{ij},$$
$$\hat{b}_i = \frac{1}{N}\sum_{k=1}^{N} q_i(\boldsymbol{x}^{(k)})h(\boldsymbol{x}^{(k)}) \approx \langle q_i, h\rangle_\rho = b_i,$$

and $\langle q_i, q_j\rangle_\rho = \int_D q_i(\boldsymbol{x})q_j(\boldsymbol{x})\rho(\boldsymbol{x})d\boldsymbol{x}$ indicates the inner product of q_i and q_j. We rewrite the linear system for $\hat{\boldsymbol{v}}^*$ as

$$(\mathbf{A} + \delta\mathbf{A})\hat{\boldsymbol{v}}^* = \boldsymbol{b} + \delta\boldsymbol{b}, \tag{A.3}$$

where $\delta\mathbf{A} = \hat{\mathbf{A}} - \mathbf{A}$ and $\delta\boldsymbol{b} = \hat{\boldsymbol{b}} - \boldsymbol{b}$. Let $\mathscr{B} = \{q_i\}_{i=1}^n \cup \{h\}$. Since both $\{q_i\}_{i=1}^n$ and h are continuous on a compact support, we may assume that $|m_1(\boldsymbol{x})m_2(\boldsymbol{x})| \le M$ for any $m_1, m_2 \in \mathscr{B}$ and any $\boldsymbol{x} \in D$, where $0 < M < \infty$ is a constant. Using the Heoffding bound, we have for any $\delta > 0$ with probability at least $1 - 2\delta$,

$$\left|\frac{1}{N}\sum_{k=1}^{N} m_1(\boldsymbol{x}^{(k)})m_2(\boldsymbol{x}^{(k)}) - \langle m_1, m_2\rangle_\rho\right| \le \sqrt{\frac{2M^2\ln\delta^{-1}}{N}}, \tag{A.4}$$

for any $m_1, m_2 \in \mathscr{B}$. This means with probability at least $1 - 2\delta$ we have

$$\|\delta\mathbf{A}\|_2 \le \|\delta\mathbf{A}\|_F \le n\sqrt{\frac{2M^2\ln\delta^{-1}}{N}}, \quad \|\delta\boldsymbol{b}\|_2 \le \sqrt{n}\sqrt{\frac{2M^2\ln\delta^{-1}}{N}}, \tag{A.5}$$

where $\|\cdot\|_F$ indicates the Frobenius norm. Let $\delta\boldsymbol{v}^* = \hat{\boldsymbol{v}}^* - \boldsymbol{v}^*$. It is seen that $\|\delta\mathbf{A}\|_2 \to 0$ as $N \to \infty$. Assume that N is large enough such that $\|\delta\mathbf{A}\|_2 \le (1 - r)$ with $0 < r < 1$, in other words, $(1 - \|\delta\mathbf{A}\|_2)^{-1} \le r^{-1}$. Since q_i are orthonormal, we have $\mathbf{A} = \mathbf{I}$. From equations (A.2) and (A.5), we have

$$\delta\boldsymbol{v}^* = (\mathbf{I} + \delta\mathbf{A})^{-1}(\delta\boldsymbol{b} - \delta\mathbf{A}\boldsymbol{v}^*),$$

to which we apply Lemma 3 and the bounds in equation (A.3) and obtain

$$\left\|\delta\boldsymbol{v}^*\right\|_2 \le r^{-1}(\|\delta\boldsymbol{b}\|_2 + \|\delta\mathbf{A}\|_2\left\|\boldsymbol{v}^*\right\|_2) \le r^{-1}(\sqrt{n} + n\left\|\boldsymbol{v}^*\right\|_2)\sqrt{\frac{2M^2\ln\delta^{-1}}{N}}. \tag{A.6}$$

Using the Pythagorean theorem, we have

$$\begin{aligned}\|\boldsymbol{m}_{\hat{\boldsymbol{v}}^*} - h\|_\rho^2 &= \|\boldsymbol{m}_{\hat{\boldsymbol{v}}^*} - \boldsymbol{m}_V^*\|_\rho^2 + \|h - \boldsymbol{m}_V^*\|_\rho^2\\ &= \|\delta\boldsymbol{v}^*\|_2^2 + \|h - \boldsymbol{m}^*\|_\rho^2\\ &\le (\|\delta\boldsymbol{v}^*\|_2 + \|h - \boldsymbol{m}_V^*\|_\rho)^2,\end{aligned}$$

which yields that

$$\|\boldsymbol{m}_{\hat{\boldsymbol{v}}^*} - h\|_\rho \le \|\delta\boldsymbol{v}^*\|_2 + \|h - \boldsymbol{m}_V^*\|_\rho. \tag{A.7}$$

Combining equations (A.6) and (A.7), we reach the conclusion. □

Appendix B. Proof of Lemma 2

Proof. Let

$$\hat{r}(\boldsymbol{x}) = \begin{cases} r(\boldsymbol{x}), & \text{if } |r^2/p - \mu| \le a;\\ 0, & \text{otherwise,}\end{cases}$$

where $a > 0$. We consider

$$\left|Q_p[r^2]-\mathbb{E}[r^2]\right| \le \left|Q_p(r^2)-Q_p(\hat{r}^2)\right| + \left|Q_p(\hat{r}^2)-\mathbb{E}[\hat{r}^2]\right| + \left|\mathbb{E}[\hat{r}^2]-\mathbb{E}[r^2]\right| = I_1+I_2+I_3. \tag{B.1}$$

The first term I_1 on the right-hand side is bounded as

$$\begin{aligned}\mathbb{E}_p\left|Q_p(r^2)-Q_p(\hat{r}^2)\right| &= \mathbb{E}_p\left|r^2(\boldsymbol{X})p^{-1}(\boldsymbol{X})-\hat{r}^2(\boldsymbol{X})p^{-1}(\boldsymbol{X})\right| \\ &= \int\limits_{|r^2/p-\mu|>a} r^2(\boldsymbol{x})d\boldsymbol{x} \\ &\le \|r^2/p\|_p\sqrt{\mathbb{P}(|r^2/p-\mu|>a;\,p)},\end{aligned} \tag{B.2}$$

where the Cauchy-Schwarz inequality is used in the last step.

The second term I_2 on the right-hand side can be bounded as

$$\begin{aligned}\mathbb{E}_p\left|Q_p(\hat{r}^2)-\mathbb{E}[\hat{r}^2]\right| &\le \sqrt{\mathrm{Var}_p(Q_p(\hat{r}^2))} \\ &\le N^{-1/2}\sqrt{\mathrm{Var}_p(\hat{r}^2(\boldsymbol{X})/p(\boldsymbol{X}))} \\ &\le aN^{-1/2},\end{aligned} \tag{B.3}$$

where in the last step we used the fact that for any variable $\alpha \le Y \le \beta$, $\mathrm{Var}(Y) \le \frac{(\alpha-\beta)^2}{4}$ with probability 1. The third term I_3 on the right-hand side can be bounded the same way as I_1.

We now estimate the tail probability $\mathbb{P}(|r^2/p-\mu|>a;\,p)$. Using the correspondence between L_1 norm and total variation distance for two probability measures as well as the Pinsker's inequality, we have

$$\|p-p^*\|_{L_1} = 2\delta(p,p^*) \le \sqrt{2D_{\mathrm{KL}}(p\|p^*)} \le \sqrt{2\varepsilon}, \tag{B.4}$$

which yields that

$$\mathbb{E}_p\left[\left|r^2/p-\mu\right|\right] \le \mu\sqrt{2\varepsilon}. \tag{B.5}$$

From the Markov inequality, we have

$$\mathbb{P}\left(\left|r^2/p-\mu\right| \ge a;\,p\right) \le \frac{\mu\sqrt{2\varepsilon}}{a}, \tag{B.6}$$

where the probability is with respect to PDF $p(\boldsymbol{x})$. Combining the bounds for I_i, $i=1,2,3$, and equation (B.6), we reach the conclusion. □

Appendix C. Proof of Theorem 1

Proof. By Assumption 1, we have

$$\begin{aligned}&\left\|u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x})\right\|_{2,\Omega} \\ \le & C_1^{-1}\left[\left\|\mathcal{L}(u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x}))\right\|_{2,\Omega}+\left\|\mathfrak{b}(u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x}))\right\|_{2,\partial\Omega}\right] \\ \le & \sqrt{2}C_1^{-1}\left(\left\|\mathcal{L}(u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x}))\right\|_{2,\Omega}^2+\left\|\mathfrak{b}(u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x}))\right\|_{2,\partial\Omega}^2\right)^{\frac{1}{2}}.\end{aligned}$$

Combining $\mathcal{L}u(\boldsymbol{x})=s(\boldsymbol{x})$, $\mathfrak{b}u(\boldsymbol{x})=g(\boldsymbol{x})$, $r(\boldsymbol{x};\Theta_N^{*,(k)})=\mathcal{L}u(\boldsymbol{x};\Theta_N^{*,(k)})-s(\boldsymbol{x})$ and $b(\boldsymbol{x};\Theta_N^{*,(k)})=\mathfrak{b}u(\boldsymbol{x};\Theta_N^{*,(k)})-g(\boldsymbol{x})$ gives

$$\left\|u(\boldsymbol{x};\Theta_N^{*,(k)})-u(\boldsymbol{x})\right\|_{2,\Omega} \le \sqrt{2}C_1^{-1}\left(\left\|r(\boldsymbol{x};\Theta_N^{*,(k)})\right\|_{2,\Omega}^2+\left\|b(\boldsymbol{x};\Theta_N^{*,(k)})\right\|_{2,\partial\Omega}^2\right)^{\frac{1}{2}}. \tag{C.1}$$

Noting that $\mathbb{E}(R_k)=\left\|r(\boldsymbol{x};\Theta_N^{*,(k)})\right\|_{2,\Omega}^2$, and according to the Hoeffding inequality, we have

$$\mathbb{P}\left(R_k-\mathbb{E}(R_k)\ge-\varepsilon\right) \ge 1-\exp\left(\frac{-2N_r\varepsilon^2}{(\tau_2-\tau_1)^2}\right). \tag{C.2}$$

Combining (C.1) and (C.2) gives that

$$\left\| u(\boldsymbol{x}; \Theta_N^{*,(k)}) - u(\boldsymbol{x}) \right\|_{2,\Omega} \leq \sqrt{2} C_1^{-1} \left(R_k + \varepsilon + \left\| b(\boldsymbol{x}; \Theta_N^{*,(k)}) \right\|_{2,\partial\Omega}^2 \right)^{\frac{1}{2}}$$

with probability at least $1 - \exp(-2N_r\varepsilon^2/(\tau_2 - \tau_1)^2)$. □

Appendix D. Proof of Corollary 1

Proof. Noting that

$$\Theta_N^{*,(k+1)} = \arg\min_{\Theta} \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(\boldsymbol{x}_\Omega^{(i)}; \Theta)}{\hat{p}_{\text{KRnet}}(\boldsymbol{x}_\Omega^{(i)}; \Theta_f^{*,(k)})}.$$

Since $\Theta_N^{*,(k+1)}$ is the optimal solution at the $(k+1)$-th stage, we have

$$R_{k+1} = \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(\boldsymbol{x}_\Omega^{(i)}; \Theta_N^{*,(k+1)})}{\hat{p}_{\text{KRnet}}(\boldsymbol{x}_\Omega^{(i)}; \Theta_f^{*,(k)})} \leq \frac{1}{N_r} \sum_{i=1}^{N_r} \frac{r^2(\boldsymbol{x}_\Omega^{(i)}; \Theta_N^{*,(k)})}{\hat{p}_{\text{KRnet}}(\boldsymbol{x}_\Omega^{(i)}; \Theta_f^{*,(k)})}. \tag{D.1}$$

Plugging $\hat{p}_{\text{KRnet}}(\boldsymbol{x}; \Theta_f^{*,(k)}) = c_k r^2(\boldsymbol{x}; \Theta_N^{*,(k)})$ into (D.1) gives that

$$R_{k+1} \leq \frac{1}{c_k}.$$

Noting that R_{k+1} is a random variable and taking its expectation, it follows that

$$\mathbb{E}(R_{k+1}) \leq \frac{1}{c_k} = \int_\Omega r^2(\boldsymbol{x}; \Theta_N^{*,(k)}) d\boldsymbol{x} = \mathbb{E}(R_k),$$

which completes the proof. □

References

[1] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.
[2] E. Weinan, The dawning of a new era in applied mathematics, Not. Am. Math. Soc. 68 (4) (2021) 565–571.
[3] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.
[4] W. E, B. Yu, The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[5] E. Kharazmi, Z. Zhang, G.E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, arXiv preprint, arXiv:1912.00873.
[6] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[7] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547.
[8] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[9] M. Raissi, P. Perdikaris, 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. 378 (2019) 686–707.
[10] G. Pang, L. Lu, G.E. Karniadakis, fPINNs: fractional physics-informed neural networks, SIAM J. Sci. Comput. 41 (4) (2019) A2603–A2626.
[11] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[12] M. Dissanayake, N. Phan-Thien, Neural-network-based approximations for solving partial differential equations, Commun. Numer. Methods Eng. 10 (3) (1994) 195–201.
[13] K. Li, K. Tang, T. Wu, Q. Liao, D3M: a deep domain decomposition method for partial differential equations, IEEE Access 8 (2020) 5283–5294.
[14] W. Li, X. Xiang, Y. Xu, Deep domain decomposition method: elliptic problems, in: J. Lu, R. Ward (Eds.), Proceedings of the First Mathematical and Scientific Machine Learning Conference, in: Proceedings of Machine Learning Research, vol. 107, PMLR, Princeton University, Princeton, NJ, USA, 2020, pp. 269–286.
[15] S. Dong, Z. Li, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Comput. Methods Appl. Mech. Eng. 387 (2021) 114129.
[16] H. Gao, L. Sun, J.-X. Wang, Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. 428 (2021) 110079.
[17] H. Sheng, C. Yang, PFNN: a penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries, J. Comput. Phys. (2020) 110085.
[18] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. 411 (2020) 109409.
[19] A. Blum, J. Hopcroft, R. Kannan, Foundations of Data Science, Cambridge University Press, 2020.
[20] R. Vershynin, High-Dimensional Probability: An Introduction with Applications in Data Science, vol. 47, Cambridge University Press, 2018.
[21] J. Wright, Y. Ma, High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications, Cambridge University Press, 2021.
[22] K. Tang, X. Wan, Q. Liao, Adaptive deep density approximation for Fokker-Planck equations, J. Comput. Phys. 457 (2022) 111080.

[23] Y. Gu, H. Yang, C. Zhou, Selectnet: self-paced learning for high-dimensional partial differential equations, J. Comput. Phys. 441 (2021) 110444.
[24] P. Morin, R.H. Nochetto, K.G. Siebert, Convergence of adaptive finite element methods, SIAM Rev. 44 (4) (2002) 631–658.
[25] K. Mekchay, R.H. Nochetto, Convergence of adaptive finite element methods for general second order linear elliptic pdes, SIAM J. Numer. Anal. 43 (5) (2005) 1803–1827.
[26] H.C. Elman, D.J. Silvester, A.J. Wathen, Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, Oxford University Press, USA, 2014.
[27] K. Tang, X. Wan, Q. Liao, Deep density estimation via invertible block-triangular mapping, Theor. Appl. Mech. Lett. 10 (2020) 143.
[28] X. Wan, S. Wei, VAE-KRnet and its applications to variational Bayes, arXiv preprint, arXiv:2006.16431.
[29] X. Wan, K. Tang, Augmented KRnet for density estimation and approximation, arXiv preprint, arXiv:2105.12866.
[30] F. Santambrogio, Optimal Transport for Applied Mathematicians, Birkäuser, NY, 2015.
[31] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real NVP, arXiv preprint, arXiv:1605.08803.
[32] D.P. Kingma, P. Dhariwa, Glow: generative flow with invertible 1x1 convolutions, in: Advances in Neural Information Processing Systems, 2018, pp. 10215–10224.
[33] L. Bottou, F.E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Rev. 60 (2) (2018) 223–311.
[34] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980.
[35] P. Bochev, M. Gunzburger, Least-squares methods for hyperbolic problems, in: Handbook of Numerical Analysis, vol. 17, Elsevier, 2016, pp. 289–317.
[36] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[37] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[38] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[39] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867.
[40] Y. Shin, Z. Zhang, G.E. Karniadakis, Error estimates of residual minimization using neural networks for linear PDEs, arXiv preprint, arXiv:2010.08019.
[41] J. Lu, Y. Lu, M. Wang, A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic equations, arXiv preprint, arXiv: 2101.01708.
[42] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Eng. 393 (2022) 114823.
[43] W. Gao, C. Wang, Active learning based sampling for high-dimensional nonlinear partial differential equations, arXiv preprint, arXiv:2112.13988.
[44] I. Kobyzev, S.J. Prince, M.A. Brubaker, Normalizing flows: an introduction and review of current methods, IEEE Trans. Pattern Anal. Mach. Intell. 43 (11) (2020) 3964–3979.
[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[46] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.
[47] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein GANs, in: Advances in Neural Information Processing Systems, vol. 30, 2017.
[48] D.P. Kingma, M. Welling, Auto-Encoding Variational Bayes, 2014, stat 1050, 1.
[49] G. Carlier, A. Galichon, F. Santambrogio, From Knothe's transport to Brenier's map and a continuation method for optimal transport, SIAM J. Math. Anal. 41 (6) (2010) 2554–2576.
[50] P.-T. De Boer, D.P. Kroese, S. Mannor, R.Y. Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. 134 (1) (2005) 19–67.
[51] R.Y. Rubinstein, D.P. Kroese, The Cross-Entropy Method: a Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning, Springer Science & Business Media, 2013.
[52] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, Deepxde: a deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[53] W.F. Mitchell, A collection of 2D elliptic problems for testing adaptive grid refinement algorithms, Appl. Math. Comput. 220 (2013) 350–364.