Bias-Variance Trade-off in Physics-Informed Neural Networks with Randomized Smoothing for High-Dimensional PDEs

Zheyuan Hu[*] [†] Zhouhao Yang[*] [†] Yezhen Wang[*] [†] George Em Karniadakis[‡] [§]
Kenji Kawaguchi[†]

Abstract

Physics-Informed Neural Networks (PINNs) have triggered a paradigm shift in scientific computing, leveraging mesh-free properties and robust approximation capabilities. While proving effective for low-dimensional partial differential equations (PDEs), the computational cost of PINNs remains a hurdle in high-dimensional scenarios. This is particularly pronounced when computing high-order and high-dimensional derivatives in the physics-informed loss. Randomized Smoothing PINN (RS-PINN) introduces Gaussian noise for stochastic smoothing of the original neural net model, enabling the use of Monte Carlo methods for derivative approximation, which eliminates the need for costly automatic differentiation. Despite its computational efficiency, especially in the approximation of high-dimensional derivatives, RS-PINN introduces biases in both loss and gradients, negatively impacting convergence, especially when coupled with stochastic gradient descent (SGD) algorithms. We present a comprehensive analysis of biases in RS-PINN, attributing them to the nonlinearity of the Mean Squared Error (MSE) loss as well as the intrinsic nonlinearity of the PDE itself. We propose tailored bias correction techniques, delineating their application based on the order of PDE nonlinearity. The derivation of an unbiased RS-PINN allows for a detailed examination of its advantages and disadvantages compared to the biased version. Specifically, the biased version has a lower variance and runs faster than the unbiased version, but it is less accurate due to the bias. To optimize the bias-variance trade-off, we combine the two approaches in a hybrid method that balances the rapid convergence of the biased version with the high accuracy of the unbiased version. In addition to methodological contributions, we present an enhanced implementation of RS-PINN. Extensive experiments on diverse high-dimensional PDEs, including Fokker-Planck, Hamilton-Jacobi-Bellman (HJB), viscous Burgers', Allen-Cahn, and Sine-Gordon equations, illustrate the bias-variance trade-off and highlight the effectiveness of the hybrid RS-PINN. Empirical guidelines are provided for selecting biased, unbiased, or hybrid versions, depending on the dimensionality and nonlinearity of the specific PDE problem.

1 Introduction

Physics-Informed Neural Networks (PINNs) [34] have revolutionized the scientific computing field thanks to their mesh-free properties, robust approximation, rapid convergence, and strong generalization capabilities [21, 24, 26]. Although PINNs have proven effective in solving many low-dimensional PDEs, the computational cost remains significant in high-dimensional scenarios. This is particularly evident when calculating high-order, high-dimensional derivatives of the neural network model concerning its inputs, especially in the context of computing the physics-informed loss. The intrinsic value of unlocking the potential of PINN lies in their mesh-free training, which allows them to overcome the curse-of-dimensionality. The capability to address high-dimensional PDE problems holds immense significance, offering substantial value in addressing a myriad of practical applications, e.g., the Hamilton-Jacobi-Bellman (HJB) equation in control theory, the Black-Scholes equation in mathematical finance, and the Schrödinger equation in quantum physics.

Among the variants of PINNs, the randomized smoothing PINN is a promising approach, see [14]. Specifically, Randomized Smoothing PINN (RS-PINN) [14] introduces Gaussian noise for stochastic smoothing of a neural network model, allowing its derivatives with respect to inputs to be expressed as expectations. This enables the model and its derivatives to be approximated using Monte Carlo methods, circumventing the challenges associated with high-order, high-dimensional derivatives, where computation by automatic differentiation can be prohibitively expensive. While RS-PINN presents an efficient backpropagation-free method for PINN parameterization and training, its reliance on Monte Carlo to approximate expectations introduces biases in both its loss and gradients. Given that RS-PINNs

[*] Equal Contribution

[†] Department of Computer Science, National University of Singapore, Singapore, 119077 (e0792494@u.nus.edu,kenji@nus.edu.sg)

[‡] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA (george_karniadakis@brown.edu)

[§] Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory, Richland, WA, United States

are commonly coupled with stochastic gradient descent (SGD) algorithms, such as Adam [27], the unbiasedness of stochastic gradients is crucial for the convergence of RS-PINNs. In fact, the unbiasedness of stochastic gradients constitutes the fundamental assumption for the convergence of SGD. This phenomenon hinders the model from converging to the optimal point, making RS-PINNs perform much worse than vanilla PINNs based on automatic differentiation.

In this paper, we conduct an in-depth analysis of the sources of bias in RS-PINNs, stemming from the nonlinearity of the commonly used Mean Squared Error (MSE) loss in PINN as well as the inherent nonlinearity of the PDE itself. We demonstrate how to correct these biases separately and extend our formulation to the specific order of PDE nonlinearity. Specifically, for nonlinear PDEs with various orders, we illustrate how their biases can be corrected differently. Overall, the essence of bias correction lies in the re-sampling using distinct Gaussian random samples. After derivation of the unbiased version of RS-PINN, we analyze the advantages and disadvantages of biased and unbiased versions. To this end, the biased version exhibits faster running speed, while under the same sample size, the unbiased version tends to have a larger variance, leading to the exploration of the bias-variance trade-off. We discuss various scenarios where the unbiased/biased version might perform better and propose a combination of both to achieve convergence speeds comparable to the biased version and the accuracy of the unbiased version. Concretely, in the optimization's initial stages, we use the biased version to rapidly converge the model to a reasonably good point. Once the loss of the biased version ceases to decrease, we transition to the unbiased version for fine-tuning. In addition to our methodological contributions, we also present an improved implementation of RS-PINN.

Finally, through extensive experiments on several high-dimensional PDEs, including the linear Fokker-Planck PDE, the nonlinear HJB equation, the viscous Burgers' equation, the Allen-Cahn equation, and the Sine-Gordon equation in different high-dimensional scenarios, we illustrate the bias-variance trade-off and how the hybrid version adeptly assimilates the strengths and weaknesses of both versions to achieve optimal results. Through experiments, we also empirically provide guidelines for the usage of biased, unbiased, and hybrid versions, which are dependent on the dimensionality as well as the nonlinearity of the PDE problem.

The rest of this paper is arranged as follows. We discuss related work in Section 2. We provide an introduction to RS-PINNs in Section 3. Then, we introduce the bias correction techniques and our main algorithms in Section 4. Computational experiments are conducted in Section 5, and we conclude the paper in Section 6.

2 Related Work

Randomized Smoothing. Randomized smoothing was initially proposed to tackle the adversarial robustness problem via certified robustness in neural networks, especially in image classification [10, 28]. Later, it was extended to train PINNs without stacked backpropagation [14], which avoids the huge computational cost, especially in high-dimensional PDEs. The generalization property of the RS-PINN can be understood via the information bottleneck theory [25]. It is expected to improve the generalization property by reducing the mutual information between the input and the hidden layer via the additionally injected noise. More recently, randomized smoothing has also been applied to backpropagation-free federated learning [11].

Backpropagation-Free PINNs. In [8], the authors proposed a coupled-automatic-numerical PINN (CAN-PINN), which combines automatic differentiation (AD) and numerical differentiation (ND) to become both accurate like AD and efficient like ND. The authors of [30] proposed a hybrid finite difference PINN (HFD-PINN), which adopts AD for smooth scales and a weighted essentially non-oscillatory (WENO) scheme to capture discontinuity. Fractional PINN (fPINN) [32] was proposed to solve fractional advection-diffusion equations, where the ND scheme is adopted for fractional differentiation. The Deep Galerkin method (DGM) [36] proposed a Monte-Carlo-based algorithm for fast second-order derivatives calculation. Taylor mode AD [6] was proposed to mitigate the exponential computational burden with increasing order of derivatives, which is currently available in the Jax framework.

High-Dimensional PDE Solver. In the broader field of high-dimensional PDE solvers, numerous attempts have been made. The authors of [37] proved the importance of L^∞ loss in solving high-dimensional Hamilton-Jacobi-Bellman equations. Separable PINN [9] adopts a separable structure, enabling the residual point to be a tensor product of per-dimension points, thereby expanding the batch size. However, for problems exceeding ten dimensions, memory usage becomes a significant concern. DeepBSDE [12, 13] and its extensions [2, 7, 15, 18, 22] are based on the classical BSDE interpretation of certain high-dimensional parabolic PDEs, and deep learning models are employed to approximate the unknowns in the formulation. The deep splitting method [1] integrates the classical splitting method with deep learning. FBSNN [33] connects high-dimensional parabolic PDEs with forward-backward stochastic differential equations and adopts deep learning for approximating the unknown solution. The multilevel Picard methods [3, 4, 5, 19, 20] are a nonlinear extension of Monte Carlo that can provably solve parabolic PDEs under certain constraints. The authors of [38, 39] proposed tensor neural networks, which adopt a separable structure for cheap numerical integration in solving high-dimensional Schrödinger equations. More recently, SDGD [17] was proposed to sample the dimension in PDEs for scaling up and speeding up high-dimensional PINNs.

Physics-Informed Machine Learning. The algorithmic concepts in this paper are based on Physics-Informed Machine Learning [23], especially PINNs [34], which utilize neural networks as surrogate models for PDE solution approximation and optimize the boundary and residual losses, which are theoretically grounded to help the neural network model discover the correct solution [16, 31, 35].

3 Preliminary

3.1 Physics-Informed Neural Networks (PINNs)

This paper focuses on solving the following partial differential equations (PDEs) defined on a domain $\Omega \subset \mathbb{R}^d$:

$$\mathcal{B}u(\boldsymbol{x}) = B(\boldsymbol{x}) \text{ on } \Gamma, \qquad \mathcal{L}u(\boldsymbol{x}) = g(\boldsymbol{x}) \text{ in } \Omega, \tag{1}$$

where $\mathcal{L}$ and $\mathcal{B}$ are the differential operators for the residual condition in Ω and for the boundary/initial condition on Γ. PINNs [34] is a neural network-based PDE solver via minimizing the following boundary and residual loss functions. Concretely, given the boundary points $\{\boldsymbol{x}_{b,i}\}_{i=1}^{n_b} \subset \Gamma$ and the residual points $\{\boldsymbol{x}_{r,i}\}_{i=1}^{n_r} \subset \Omega$, the PINN loss is composed of the mean square error in the residual and on the boundary:

$$\begin{aligned}\mathcal{L}(\theta) &= \lambda_b \mathcal{L}_b(\theta) + \lambda_r \mathcal{L}_r(\theta) \\ &= \frac{\lambda_b}{n_b} \sum_{i=1}^{n_b} \left|\mathcal{B}u_\theta(\boldsymbol{x}_{b,i}) - B(\boldsymbol{x}_{b,i})\right|^2 + \frac{\lambda_r}{n_r} \sum_{i=1}^{n_r} \left|\mathcal{L}u_\theta(\boldsymbol{x}_{r,i}) - g(\boldsymbol{x}_{r,i})\right|^2,\end{aligned} \tag{2}$$

where λ_b is the weight for the boundary loss while λ_r is that for the residual loss.

3.2 Randomized Smoothing PINNs

He et al. [14] proposed the randomly smoothed neural network structure for backpropagation-free PINN computation:

$$u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} f(\boldsymbol{x} + \delta; \theta), \tag{3}$$

where $f(\boldsymbol{x}; \theta)$ is a vanilla neural network parameterized by θ; $u(\boldsymbol{x}; \theta)$ is the corresponding smoothed version of the network $f(\boldsymbol{x}; \theta)$ serving as the surrogate model in PINNs.

The derivative of $u(\boldsymbol{x}; \theta)$ can be analytically computed without backpropagation. For instance, its gradient, Laplacian, and Hessian with respect to the input $\boldsymbol{x}$ can be written as follows (see He et al. [14]):

$$\nabla_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\delta}{\sigma^2} f(\boldsymbol{x} + \delta; \theta)\right]. \tag{4}$$

$$\Delta_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\|\delta\|^2 - \sigma^2 d}{\sigma^4} f(\boldsymbol{x} + \delta; \theta)\right]. \tag{5}$$

$$\text{Hess}_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\delta\delta^{\mathrm{T}} - \sigma^2 d}{\sigma^4} f(\boldsymbol{x} + \delta; \theta)\right]. \tag{6}$$

Here $\nabla_{\boldsymbol{x}}, \Delta_{\boldsymbol{x}}, \text{Hessian}_{\boldsymbol{x}}$ are the gradient, Laplacian, and Hessian operators with respect to the input $\boldsymbol{x}$, respectively. All of them can be simulated by Monte Carlo sampling for the expectation estimator to calculate the derivatives without the expensive automatic differentiation, and then the PINN loss is used to solve the PDE.

He et al. [14] also introduced a corresponding variance reduction form, as the entire derivative estimation involves Monte Carlo estimation of expectations, which contains certain variance. Specifically, the variance reduction is related to control variate and antithetic variable method, whose ultimate forms are similar to the numerical differentiation:

$$\nabla_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\delta}{2\sigma^2} \left(f(\boldsymbol{x} + \delta; \theta) - f(\boldsymbol{x} - \delta; \theta)\right)\right]. \tag{7}$$

$$\Delta_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\|\delta\|^2 - \sigma^2 d}{2\sigma^4} \left(f(\boldsymbol{x} + \delta; \theta) + f(\boldsymbol{x} - \delta; \theta) - 2f(\boldsymbol{x}; \theta)\right)\right]. \tag{8}$$

$$\text{Hess}_{\boldsymbol{x}} u(\boldsymbol{x}; \theta) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\frac{\delta\delta^{\mathrm{T}} - \sigma^2 d}{2\sigma^4} \left(f(\boldsymbol{x} + \delta; \theta) + f(\boldsymbol{x} - \delta; \theta) - 2f(\boldsymbol{x}; \theta)\right)\right]. \tag{9}$$

The essence of the randomized smoothing PINN lies in transforming the derivatives and model inference into an expectation. Especially in high-dimensional scenarios, this approach is highly cost-effective since computing the Hessian and other high-order derivatives of a complicated neural network in high dimensions is prohibitively expensive. The Monte Carlo sampling method for estimating expectations serves as a powerful tool to combat the curse-of-dimensionality, especially when combined with the mesh-free PINN approach. This synergy positions it as a formidable tool for addressing high-dimensional PDEs.

Specifically, given a sample size of $K \in \mathbb{Z}^+$ in Monte Carlo, we can approximate the expectations above as follows.

$$\widehat{u}(\boldsymbol{x};\theta;\delta) := \frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta) \approx \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} f(\boldsymbol{x}+\delta;\theta) = u(\boldsymbol{x};\theta), \tag{10}$$

where $\delta = \{\delta_i\}_{i=1}^{K}$ are K $i.i.d.$ Gaussian samples and $\widehat{u}(\boldsymbol{x};\theta;\delta)$ is the Monte-Carlo-based estimation of the exact $u(\boldsymbol{x};\delta)$ on the point $\boldsymbol{x}$. Similarly, for the gradient, Laplacian, and Hessian, we have the following Monte-Carlo-based estimators, which are then substituted into the PINN loss for optimization:

$$\widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta) := \frac{1}{K}\sum_{i=1}^{K}\left[\frac{\delta}{2\sigma^2}\left(f(\boldsymbol{x}+\delta_i;\theta) - f(\boldsymbol{x}-\delta_i;\theta)\right)\right]. \tag{11}$$

$$\widehat{\Delta_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta) = \frac{1}{K}\sum_{i=1}^{K}\left[\frac{\|\delta_i\|^2 - \sigma^2 d}{2\sigma^4}\left(f(\boldsymbol{x}+\delta_i;\theta) + f(\boldsymbol{x}-\delta_i;\theta) - 2f(\boldsymbol{x};\theta)\right)\right]. \tag{12}$$

$$\widehat{\text{Hess}_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta) = \frac{1}{K}\sum_{i=1}^{K}\left[\frac{\delta_i\delta_i^{\mathrm{T}} - \sigma^2 d}{2\sigma^4}\left(f(\boldsymbol{x}+\delta_i;\theta) + f(\boldsymbol{x}-\delta_i;\theta) - 2f(\boldsymbol{x};\theta)\right)\right]. \tag{13}$$

4 Proposed Method

In this section, we show that the original formulation of randomized smoothing PINN leads to a biased gradient. We further show that bias comes from two contributions: the nonlinear mean square error loss function in the PINN loss and the nonlinearity of PDE itself. These nonlinearities disrupt the linearity of the mathematical expectation in the RS-PINN for model inference and the model's derivatives in the PINN loss, thereby introducing bias. Then, we correct these biases for better performance, demonstrating how the two biases affect PINN's performance differently and showing that the biased (unbiased) version has a lower (higher) gradient variance and that the biased version runs faster than the unbiased one per epoch. Hence, we finally combine them to propose a hybrid version, which runs as fast as the biased version and as accurate as the unbiased one.

4.1 Bias from the Mean Square Error Loss Function

In this subsection, we illustrate the bias of RS-PINN's loss function and its gradient with respect to model parameters induced by the nonlinear mean square error loss function in the PINN loss, since its nonlinearity violates the linearity of expectation that guarantees unbiasedness.

Without loss of generality, since the inference and the derivatives of the surrogate model $u(\boldsymbol{x};\theta)$ can all be written as an expectations, let us use the boundary loss to demonstrate the first bias from the nonlinearity of the mean square error loss function, which includes the following expectation related to model inference:

$$u(\boldsymbol{x};\theta) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} f(\boldsymbol{x}+\delta;\theta). \tag{14}$$

The boundary loss on a boundary point $\boldsymbol{x}$ is:

$$L_b(\theta) = \left(u(\boldsymbol{x};\theta) - g(\boldsymbol{x})\right)^2 = \left(\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} f(\boldsymbol{x}+\delta;\theta) - g(\boldsymbol{x})\right)^2, \tag{15}$$

where $g(\boldsymbol{x})$ is the given boundary condition. He et al. [14] approximate the boundary loss by Monte Carlo:

$$L_b^{(0)}(\theta) = \left(\widehat{u}(\boldsymbol{x};\theta;\delta) - g(\boldsymbol{x})\right)^2 = \left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta) - g(\boldsymbol{x})\right)^2. \tag{16}$$

Although $\widehat{u}(\boldsymbol{x};\theta;\delta)$ is an unbiased estimator of $u(\boldsymbol{x};\theta)$, due to the nonlinear quadratic form of the mean square error loss function, the expectation of the loss function $L_b^{(0)}(\theta)$ is not the true loss $L_b(\theta)$:

$$\mathbb{E}_\delta\left[L_b^{(0)}(\theta)\right] = \mathbb{E}_\delta\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta) - g(\boldsymbol{x})\right)^2\right] \neq L_b(\theta) = \left(\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})} f(\boldsymbol{x}+\delta;\theta) - g(\boldsymbol{x})\right)^2, \tag{17}$$

i.e., the formulation of the loss function by He et al. [14] is biased since the nonlinear mean square error loss function violates the linearity of mathematical expectation.

To correct the bias, we just need to sample independently two groups of Gaussian variables δ_i and δ_i' and compute the following debiased loss function:

$$L_b^{(1)}(\theta) = [\widehat{u}(\boldsymbol{x};\theta;\delta') - g(\boldsymbol{x})]\,[\widehat{u}(\boldsymbol{x};\theta;\delta) - g(\boldsymbol{x})] = \left[\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i';\theta) - g(\boldsymbol{x})\right]\left[\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta) - g(\boldsymbol{x})\right], \tag{18}$$

where $\delta_i \sim \mathcal{N}(0,\sigma^2\boldsymbol{I})$ and $\delta_i' \sim \mathcal{N}(0,\sigma^2\boldsymbol{I})$ are independent. Then, the derived loss function and its gradient with respect to θ for optimization are all unbiased, since we eliminate the bias introduced by nonlinearity through re-sampling, which breaks down the nonlinearity. This is summarized in the following theorem.

Theorem 4.1. *The loss $L_b^{(1)}(\theta)$ and its gradient with respect to θ are unbiased estimators for the exact loss $L_b(\theta)$ and its gradient with respect to θ, respectively, i.e.,*

$$\mathbb{E}_{\delta,\delta'}\left[L_b^{(1)}(\theta)\right] = L_b(\theta), \quad \mathbb{E}_{\delta,\delta'}\left[\frac{\partial L_b^{(1)}(\theta)}{\partial\theta}\right] = \frac{\partial L_b(\theta)}{\partial\theta}. \tag{19}$$

Proof. The proof is presented in Appendix A □

Our previous discussion on boundary loss can be extended to the case of residual loss in linear PDEs. Due to the linearity of the mathematical expectation, the residual part of a linear PDE preserves the unbiased nature of Monte Carlo sampling. Therefore, the sole source of bias stems from the nonlinearity of the mean square error loss function.

Taking everything together, in this subsection, we have introduced a debiasing method for the loss function of linear PDEs in RS-PINN. The essence lies in eliminating the nonlinearity in the mean square error loss function through resampling, thereby leveraging the linearity of mathematical expectation to ensure unbiasedness. However, we note that this does not hold true for nonlinear PDEs whose residual parts violate the linearity of expectation. Below, we elucidate the additional bias introduced by the nonlinearity of PDEs.

4.2 Bias from the PDE Nonlinearity

In this subsection, we illustrate the bias of RS-PINN induced by the PDE nonlinearity that violates the linearity of expectation. Since nonlinearity differs in various PDEs, we commence with a brief illustration of the HJB equation. Subsequently, considering the diverse orders of nonlinearity inherent in distinct nonlinear equations, we extend our methodology to various and more general scenarios.

We take the Hamilton-Jacobi-Bellman (HJB) equation in He et al. [14] as an example, whose nonlinear PDE part is

$$u_t = \Delta_{\boldsymbol{x}} u - \|\nabla_{\boldsymbol{x}} u(\boldsymbol{x})\|^2. \tag{20}$$

To simplify the discussion and without the loss of generality since the linear PDE case has been tackled in the previous subsection, we ignore the linear term of the HJB equation and only consider the nonlinear term $\|\nabla_{\boldsymbol{x}} u(\boldsymbol{x})\|^2$. The true residual loss on a residual point $\boldsymbol{x}$ is:

$$L_r(\theta) = \left(\|\nabla_{\boldsymbol{x}} u(\boldsymbol{x};\theta)\|^2 - g(\boldsymbol{x})\right)^2 \tag{21}$$

He et al. [14] approximate the boundary loss by Monte Carlo:

$$L_r^{(0)}(\theta) = \left(\left\|\widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta)\right\|^2 - g(\boldsymbol{x})\right)^2, \tag{22}$$

where $\widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta)$ defined in equation (11) is an unbiased estimator of $u(\boldsymbol{x};\theta)$ based on Monte-Carlo sampling. The original formulation of He et al. [14] is biased due to the nonlinearity of the mean square error loss function. However, our previous approach for linear PDEs is still biased due to the quadratic term $\|\cdot\|^2$ term in the loss due to the PDE nonlinearity, which violates the linearity of mathematical expectation. Concretely, the loss correcting the bias from the forward and backward passes is

$$L_r^{(1)}(\theta) = \left(\left\|\widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta)\right\|^2 - g(\boldsymbol{x})\right) \times \left(\left\|\widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta')\right\|^2 - g(\boldsymbol{x})\right). \tag{23}$$

Although $\widehat{u}(\boldsymbol{x};\theta;\delta)$ is an unbiased estimator of $u(\boldsymbol{x};\theta)$, an additional bias comes from the nonlinear term $\|\cdot\|^2$ due to the PDE nonlinearity $\|\nabla u\|^2$.

We can correct the bias via sampling the quadratic terms in $\|f\|^2 = \langle f, f\rangle$ independently, using four groups of Gaussian samples $\delta, \delta', \delta'', \delta'''$:

$$L_r^{(2)}(\theta) = \left(\left\langle \widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta), \widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta')\right\rangle - g(\boldsymbol{x})\right) \times \left(\left\langle \widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta''), \widehat{\nabla_{\boldsymbol{x}} u}(\boldsymbol{x};\theta;\delta''')\right\rangle - g(\boldsymbol{x})\right). \tag{24}$$

Then, the derived loss function and its gradient with respect to θ for optimization are all unbiased.

Theorem 4.2. *The loss $L_r^{(2)}(\theta)$ and its gradient with respect to θ are unbiased estimators for the loss $L_r(\theta)$ and its gradient with respect to θ, respectively, i.e.,*

$$\mathbb{E}_{\delta,\delta',\delta'',\delta'''}\left[L_r^{(2)}(\theta)\right] = L_r(\theta), \quad \mathbb{E}_{\delta,\delta',\delta'',\delta'''}\left[\frac{\partial}{\partial\theta} L_r^{(2)}(\theta)\right] = \frac{\partial}{\partial\theta} L_r(\theta). \tag{25}$$

Proof. The proof is presented in Appendix A. □

So far, we present the bias correction technique for one nonlinear PDE, namely the HJB equation, where we correct the biases from the nonlinear mean square error loss function and the PDE nonlinearity. We will discuss general nonlinear PDEs in the next subsection.

4.3 General Nonlinear PDEs: Order of Nonlinearity

This subsection extends our bias correction technique to general nonlinear PDEs based on the concept of nonlinearity order. Specifically, our idea of bias correction can be easily extended to the general nth order of nonlinearity. In the context of nonlinear PDEs, the "order of nonlinearity" refers to the highest power of the dependent function u or its derivatives in the nonlinear terms of the equation. Nonlinearity in PDEs arises when the equation involves terms that are not proportional and linear to the dependent function or its derivatives. The order of nonlinearity is determined by the highest power of these nonlinear terms. Intuitively, for the nth order nonlinear case, we just need to sample the n terms independently using n different groups of Gaussian variables to break down the nonlinearity and to make the loss function unbiased. Below are some examples of the nonlinearity order and their corresponding biased and unbiased versions of RS-PINN, whose detailed explanation is further provided in Appendix B.

- HJB equation. The previous HJB equation given by $u_t = \Delta_{\boldsymbol{x}} u - \|\nabla_{\boldsymbol{x}} u(\boldsymbol{x})\|^2$, which has a nonlinearity order of two due to the $\|\nabla_{\boldsymbol{x}} u\|^2$ term.
- Allen-Cahn (AC) equation is given by $u_t = \Delta_{\boldsymbol{x}} u + u - u^3$, and its nonlinearity stems from the term u^3, which is a cubic function. Therefore, the nonlinearity order of the AC equation is three. During the model training, we must independently sample the three u terms in $u^3 = u \cdot u \cdot u$ for unbiased gradients.
- Viscous Burgers' equation is given by $u_t + u\sum_{i=1}^d u_{\boldsymbol{x}_i} - \nu\Delta_{\boldsymbol{x}} u(\boldsymbol{x}, t) = 0$, and its nonlinearity stems from the term $u\sum_{i=1}^d u_{\boldsymbol{x}_i}$. Therefore, the nonlinearity order of the viscous Burgers' PDE is two. During the model training, we must independently sample the u and $\nabla_{\boldsymbol{x}} u$ for unbiased gradients.
- Sine-Gordon equation. However, our method cannot deal with nonlinear like $\sin(u)$ in the Sine-Gorden PDE $u_t = \Delta_{\boldsymbol{x}} u + \sin(u)$. Fortunately, we can increase the sample size K in the Monte Carlo to minimize the bias. Furthermore, correcting the bias from the nonlinearity mean square error loss does suffice for the Sine-Gordon equation, i.e., we can still correct the bias from the nonlinear mean square error loss to improve over the original formulation in He et al. [14], which is actually sufficient to obtain a low error in high dimensions.

4.4 Bias-Variance Trade-off and the Hybrid Method

In this subsection, we discuss the bias-variance trade-off in RS-PINN and propose a hybrid version to incorporate the advantages of both methods to achieve the best performance. We also provide guidelines for explaining when the biased/unbiased version can outperform the other, facilitating the choice of the algorithm in practical scenarios.

The unbiased version employs multiple sets of independent Gaussian samples to calculate the loss, making it slower with larger gradient variances due to more sampling and more randomness; however, it provides unbiased gradients. Specifically, while the biased version from He et al. [14] requires only one set of Gaussian variables, correcting the bias from the nonlinear MSE loss functions doubles the number of independent Gaussian variable sets while correcting the additional bias from the PDE nonlinearity further increases the number of samples depending on the nonlinearity order of the PDE. For instance, a totally unbiased version of the HJB equation and the viscous Burgers' equation requires four sets, while that of the Allen-Cahn equation requires six sets.

In contrast, the biased version requires only one set of samples, resulting in faster running speed per iteration and smaller gradient variances, but the gradients are biased. Hence, we propose a hybrid approach that combines the strengths of both methods. In the initial optimization stages, we use the biased version to converge the model rapidly to a reasonably good point. Once the loss of the biased version ceases to decrease, we transition to the unbiased version for fine-tuning.

This theoretical analysis sheds light on the practical choice of algorithms in computational experiments. In higher dimensions, in the unbiased version by sampling more Gaussians will lead to a much larger variance. So, it is expected that the unbiased version will have lower variance in lower dimensions and thus have better performance than the biased one. On the other hand, the biased version will perform better in extremely high dimensions. After the convergence of the biased algorithm, we can further fine-tune it using the unbiased algorithm.

In summary, our guidelines for empirical evaluations based on the theoretical analysis are given as follows. In lower dimensions, where the unbiased version exhibits lower variance, its unbiased nature is crucial, allowing for a direct application of the unbiased version. However, in higher dimensions, utilizing the unbiased version directly introduces significant variance, impeding convergence. Therefore, we employ the biased version initially to converge to a reasonably good position and subsequently fine-tune with the unbiased version.

4.5 Implementation Improvement

Here, we conduct an analysis of He et al.'s [14] approach to implementing randomized smoothing in order to identify its limitations. Subsequently, we propose two more accurate and lower-variance implementations.

Suppose that we would like to implement the second-order derivatives and the network includes both t and $\boldsymbol{x}$ for time-dependent PDEs

$$u(\boldsymbol{x}, t) = \mathbb{E}_{\delta_{\boldsymbol{x}} \sim \mathcal{N}(0, \sigma_{\boldsymbol{x}}^2 I)} \mathbb{E}_{\delta_t \sim \mathcal{N}(0, \sigma_t^2 I)} \left[f(\boldsymbol{x} + \delta_{\boldsymbol{x}}, t + \delta_t) \right], \tag{26}$$

where we randomly smooth $\boldsymbol{x}$ and t using Gaussian with different variance for model flexibility. He et al. [14] implement the randomized smoothing model's derivatives as

$$\boldsymbol{H}_{\boldsymbol{x}} u(\boldsymbol{x}, t) = \mathbb{E}_{\delta_{\boldsymbol{x}} \sim \mathcal{N}(0, \sigma_{\boldsymbol{x}}^2 I)} \mathbb{E}_{\delta_t \sim \mathcal{N}(0, \sigma_t^2 I)} \left[\frac{\delta_{\boldsymbol{x}} \delta_{\boldsymbol{x}}^T - \sigma_{\boldsymbol{x}}^2 I}{2\sigma_{\boldsymbol{x}}^4} \left(f(\boldsymbol{x} + \delta_{\boldsymbol{x}}, t + \delta_t) + f(\boldsymbol{x} - \delta_{\boldsymbol{x}}, t - \delta_t) - 2f(\boldsymbol{x}, t) \right) \right]. \tag{27}$$

However, here we are taking the derivative with respect to $\boldsymbol{x}$, with no relation to t. Nevertheless, the focus has also shifted to t, thereby increasing the variance and impeding convergence.

The correct approach should treat $\boldsymbol{x}$ and t as independent variables:

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{x}} u(\boldsymbol{x}, t) &= \mathbb{E}_{\delta_{\boldsymbol{x}} \sim \mathcal{N}(0, \sigma_{\boldsymbol{x}}^2 I)} \left[\frac{\delta_{\boldsymbol{x}} \delta_{\boldsymbol{x}}^T - \sigma_{\boldsymbol{x}}^2 I}{2\sigma_{\boldsymbol{x}}^4} \mathbb{E}_{\delta_t \sim \mathcal{N}(0, \sigma_t^2 I)} \left[f(\boldsymbol{x} + \delta_{\boldsymbol{x}}, t + \delta_t) + f(\boldsymbol{x} - \delta_{\boldsymbol{x}}, t + \delta_t) - 2f(\boldsymbol{x}, t + \delta_t) \right] \right] \\ &= \mathbb{E}_{\delta_{\boldsymbol{x}} \sim \mathcal{N}(0, \sigma_{\boldsymbol{x}}^2 I)} \mathbb{E}_{\delta_t \sim \mathcal{N}(0, \sigma_t^2 I)} \left[\frac{\delta_{\boldsymbol{x}} \delta_{\boldsymbol{x}}^T - \sigma_{\boldsymbol{x}}^2 I}{2\sigma_{\boldsymbol{x}}^4} \left(f(\boldsymbol{x} + \delta_{\boldsymbol{x}}, t + \delta_t) + f(\boldsymbol{x} - \delta_{\boldsymbol{x}}, t + \delta_t) - 2f(\boldsymbol{x}, t + \delta_t) \right) \right]. \end{aligned} \tag{28}$$

Another valid implementation approach is to treat $\boldsymbol{x}$ and t as a unified entity and apply the same Gaussian noise smoothing. Then, based on the index, select the model's derivatives concerning both $\boldsymbol{x}$ and t. However, this method compromises the model's flexibility since the PDE exhibits an asymmetry between $\boldsymbol{x}$ and t. Therefore, a more reasonable approach is to model them separately.

5 Computational Experiments

In our computational experiments, for linear equations (Fokker-Planck PDEs in Subsection 5.1), we will use "biased" to denote the biased version and "unbiased" to denote the unbiased version by correcting the bias from the MSE loss. For nonlinear PDEs in the rest of the subsections, we will use "biased" as before, and "unbiased1" to denote the unbiased version by correcting the bias from the MSE loss solely, and "unbiased2" to denote the unbiased version by correcting the two biases from the MSE loss and the PDE nonlinearity. The detailed mathematical formulas for the losses are presented in Appendix B.

5.1 Isotropic and Anisotropic Linear Fokker-Planck PDEs

The isotropic linear Fokker-Planck (heat) PDE is

$$\begin{aligned} u_t &= \frac{1}{2}\Delta_{\boldsymbol{x}} u - \sum_{i=1}^{d} u_{\boldsymbol{x}_i}. \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0,1]. \\ u(\boldsymbol{x}, t=0) &= \|\boldsymbol{x}\|^2. \quad \boldsymbol{x} \in \mathbb{R}^d, \end{aligned} \tag{29}$$

associated with the initial condition at $t = 0$. Its exact solution is

$$u(\boldsymbol{x}, t) = \|\boldsymbol{x} - t\|^2 + dt. \tag{30}$$

Since this PDE corresponds to a Brownian motion with shift, We sample training residual points and test points based on the SDE trajectory:

$$t \sim \text{Unif}(0,1), \boldsymbol{x} \sim \mathcal{N}(t, 2-t). \tag{31}$$

The anisotropic linear Fokker-Planck (heat) PDE is

$$\begin{aligned} u_t &= \frac{1}{2}\Delta_{\boldsymbol{x}} u - \sum_{i=1}^{d} \boldsymbol{\mu}_i u_{\boldsymbol{x}_i}. \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0,1]. \\ u(\boldsymbol{x}, t=0) &= \|\boldsymbol{x}\|^2. \quad \boldsymbol{x} \in \mathbb{R}^d, \end{aligned} \tag{32}$$

associated with an initial condition at $t = 0$. Its solution is

$$u(\boldsymbol{x}, t) = \|\boldsymbol{x} - \boldsymbol{\mu} t\|^2 + dt, \tag{33}$$

where $\boldsymbol{\mu}_i \sim \mathcal{N}(1,1)$ for all dimensions i and $\boldsymbol{\mu} \in \mathbb{R}^d$. This example is designed to show that RS-PINN can deal with anisotropic problems. Since this PDE corresponds to a Brownian motion with shift, We sample training residual points and test points based on the SDE trajectory:

$$t \sim \text{Unif}(0,1), \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu} t, (2-t) \cdot \boldsymbol{I}_{d\times d}). \tag{34}$$

For both isotropic and anisotropic FP PDEs, we randomly sample 100 residual points at each iteration and 20K fixed test points based on the SDE trajectory. The sample size in the RS-PINN is $K = 1024$, and the variance of Gaussian noise is $\sigma = 1e-2$, with a backbone network with 4 layers and 128 hidden units, which is trained by an Adam optimizer [27] with 1e-3 (10, 100, 1K dimension) or 1e-4 (10K dimension) initial learning rate which decays exponentially with coefficient 0.9995 for 10K epochs. We use the boundary augmentation given by the following model output to satisfy the initial condition automatically [29]:

$$u_\theta^{\text{RS}}(\boldsymbol{x}) = u_\theta(\boldsymbol{x}, t)t + \|\boldsymbol{x}\|^2, \tag{35}$$

where $u_\theta(\boldsymbol{x})$ is the randomized smoothing neural network and $u_\theta^{\text{RS}}(\boldsymbol{x})$ is the boundary-augmented model. We repeat our experiment 5 times with 5 independent random seeds. We test RS-PINN with biased, unbiased, and hybrid versions for the 10, 100, 1K, and 10K-dimensional cases. For the hybrid version in the isotropic problem, the transition from the biased version to the unbiased one happens in the 1500th, 1500th, 6000th, and 6000th epochs for the 10, 100, 1K, and 10K dimensional PDEs, respectively. For the hybrid version in the anisotropic problem, the transition from the biased version to the unbiased one happens in the 1000th, 500th, 1000th, 2000th, 6000th, and 8000th epochs for the 10, 100, 250, 500, 1K, and 10K dimensional PDEs, respectively. The transition is chosen by the time when the loss of the biased algorithm ceases to decrease further.

Isotropic FP	10^1D	10^2D	10^3D	10^4D
Biased	3.846E-3	2.367E-2	1.057E-2	8.597E-3
Unbiased	2.244E-3	6.576E-3	1.047E-2	1.197E-2
Hybrid	**2.238E-3**	**6.561E-3**	**1.046E-2**	**7.507E-3**

Table 1: Results for the isotropic FP PDE.

The numerical results for the isotropic FP PDE are shown in Table 1, and Figure 1 shows convergence curves with respect to the epoch (first row) and the running time (second row). Here is the summary of the results:

- In lower dimension (10D, 10^2D), the unbiased version is much better than the biased version in He et al. [14] since the variance of sampling is lower in lower dimensions where the dimensionality of the samples is low, given that the main bottleneck of the unbiased version is the relatively larger variance compared to the biased version of RS-PINN.
- However, as the dimensions goes higher (10^3D, 10^4D), the biased version gets better, i.e., the unbiased version encounters huge variance in higher dimensions, whose disadvantages outweigh its benefit of unbiasedness.
- In $10^1, 10^2, 10^3$D, the hybrid version is as good as the unbiased version.
- In 10^4D, the hybrid version converges well by applying the biased version first to get a relatively good convergence point, then the unbiased version is used for finetuning and gets an even better result.
- In 10^4D, directly applying the unbiased version will lead to huge variances preventing convergence.

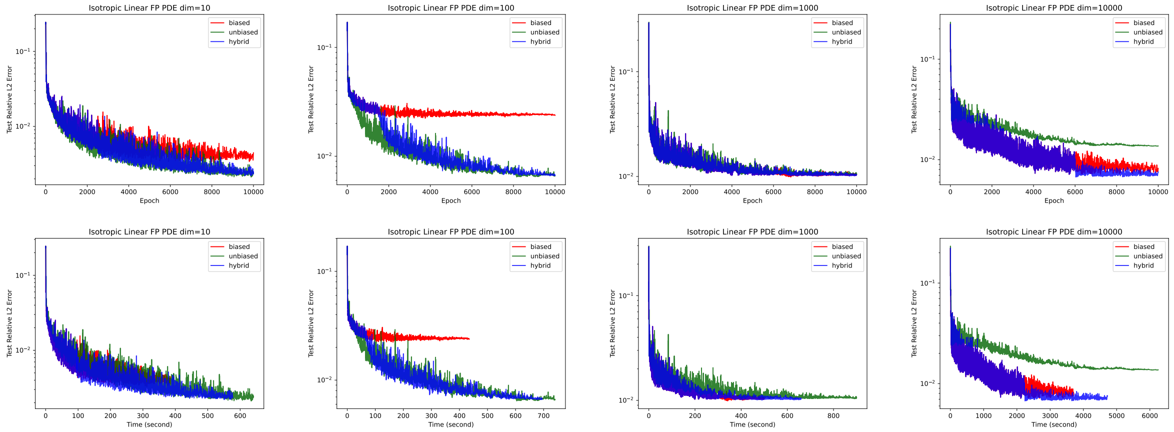

Figure 1: Isotropic FP PDE: 10^1, 10^2, 10^3, and 10^4D convergence curves with respect to the epoch (first row) and the running time (second row). In 10^1, 10^2, and 10^3D, the unbiased version is better than the biased version since the sampling variance is lower in lower dimensions where the dimensionality of the samples is low. Thus, the hybrid method is as good as the unbiased version thanks to the unbiased training at the second training stage and is faster than the unbiased version thanks to the biased pretraining at the early training phase. In 10^4D, the hybrid version converges well by applying the biased version first; then, the unbiased version is used for finetuning and getting an even more stable final convergence result. Solely applying the unbiased version will lead to huge variances preventing convergence.

The results for the anisotropic FP PDE are shown in Table 2 while the convergence curves and loss records for the highest 10^4D are shown in Figure 2. We can still observe the advantages of the unbiased version in relatively lower dimensions, and as the dimensionality increases, the biased version gradually outperforms the unbiased version. This is primarily due to the increase in variance for the unbiased version, particularly in extremely high dimensions. Notably, the results obtained by the RS-PINN remain quite stable across different dimensions, demonstrating its ability to address the dimensionality curse. Furthermore, RS-PINN exhibits competence in handling anisotropic problems. The convergence curves in the 10,000-dimensional space suggest that, after convergence is achieved with the biased version, fine-tuning with the unbiased version can lead to even better and more stable results, ultimately causing the hybrid method to outperform the rest.

Anisotropic Linear Heat	10D	100D	250D	500D	1,000D	10,000D
Biased	5.611E-02	4.452E-02	2.827E-02	1.935E-02	1.251E-02	1.389E-02
Unbiased	1.043E-02	1.215E-02	1.986E-02	1.846E-02	1.657E-02	4.036E-02
Hybrid	**1.039E-02**	**1.211E-02**	**1.979E-02**	**1.840E-02**	**1.245E-02**	**1.342E-02**

Table 2: Results for the anisotropic FP PDE.

5.2 Hamilton-Jacobi-Bellman PDEs

This section delves into the Hamilton-Jacobi-Bellman (HJB) equation, which is widely used in optimal control problems. The nonlinearity inherent in the HJB equation introduces two biases in RS-PINN. We will demonstrate

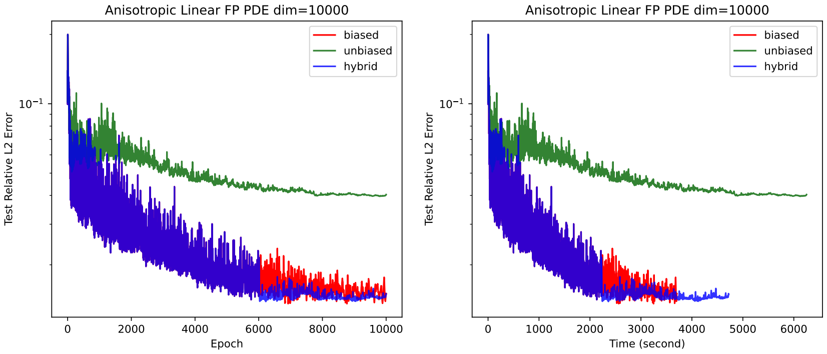

Figure 2: Anisotropic FP PDE: 10^4D convergence curves with respect to the epoch (left) and time (right). The hybrid version converges well by applying the biased version first; then, the unbiased version is used for finetuning and getting an even more stable final convergence result. Solely applying the unbiased version will lead to huge variances preventing convergence.

the remarkable performance of the model after correcting these two biases. Correcting only one bias or adopting a completely biased approach yields suboptimal results. Additionally, we consider two different solutions to showcase the model's versatility.

Specifically, we consider the HJB equation with linear-quadratic-Gaussian (LQG) control:

$$\begin{aligned} &\partial_t u(\boldsymbol{x},t) + \Delta_{\boldsymbol{x}} u(\boldsymbol{x},t) - \|\nabla_{\boldsymbol{x}} u(\boldsymbol{x},t)\|^2 = 0, \quad \boldsymbol{x} \in \mathbb{R}^d, t \in [0,T] \\ &u(\boldsymbol{x},T) = g(\boldsymbol{x}), \end{aligned} \tag{36}$$

where $g(\boldsymbol{x})$ is the terminal condition to be chosen, the PDE has the solution that can be simulated by Monte Carlo for benchmarking over various initial conditions and dimensions:

$$u(\boldsymbol{x},t) = -\log\left(\int_{\mathbb{R}^d} (2\pi)^{-d/2} \exp(-\|\boldsymbol{y}\|^2/2) \exp(-g(\boldsymbol{x} - \sqrt{2(1-t)}\boldsymbol{y}))d\boldsymbol{y}\right). \tag{37}$$

We choose the following cost functions as the terminal conditions:

- Quadratic cost:
$$g(\boldsymbol{x}) = \|\boldsymbol{x}\|^2. \quad u(\boldsymbol{x},t) = \frac{\|\boldsymbol{x}\|^2}{1+4(T-t)} + \frac{d}{2}\log(1+4(T-t)). \tag{38}$$
Here, the solution can be obtained analytically.

- Anisotropic Rosenbrock function:
$$g(\boldsymbol{x}) = \sum_{i=1}^{d/2} \left[c_{1,i}(x_{2i-1} - x_{2i})^2 + c_{2,i}x_{2i}^2\right], \tag{39}$$
where $c_{1,i}, c_{2,i} \sim \text{Unif}[0,1]$ and Monte Carlo is required for simulating the exact solution. We use 10^5 samples for Monte Carlo.

Here is the implementation detail. For all three HJB equations, we randomly sample 1K residual points at each iteration and 20K fixed test points based on the distributions $t \sim \text{Unif}[0,1], \boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_{d\times d})$. The sample sizes in the RS-PINN is $K = 1024$ for training and $K = 128$ for testing, and the variance of Gaussian noise is $\sigma = 1e-2$, with a backbone network with 4 layers and 128 hidden units, which is trained by an Adam optimizer [27] with 1e-3 initial learning rate which decays exponentially with coefficient 0.9995 for 10K epochs. We use the boundary augmentation given by the following model output to satisfy the terminal condition automatically [29]:

$$u_\theta^{\text{RS}}(\boldsymbol{x}) = u_\theta(\boldsymbol{x},t)t + g(\boldsymbol{x}), \tag{40}$$

where $u_\theta(\boldsymbol{x})$ is the randomized smoothing neural network and $u_\theta^{\text{RS}}(\boldsymbol{x})$ is the boundary-augmented model. We repeat our experiment 5 times with 5 independent random seeds.

The computational results for the three HJB equations are shown in Table 3, and the convergence curves for HJB with quadratic cost are shown in Figure 3.

HJB (Quadratic Cost)	10D	20D	30D	40D
Biased	7.423E-2	1.882E-1	2.486E-1	3.628E-1
Unbiased1	3.896E-2	1.281E-1	2.332E-1	3.204E-1
Unbiased2	**1.415E-2**	**2.572E-2**	**2.644E-2**	**4.223E-2**

HJB (Anisotropic Cost)	10D	20D	30D	40D
Biased	9.361E-2	1.820E-1	3.305E-1	3.998E-1
Unbiased1	7.783E-2	2.058E-1	3.652E-1	4.379E-1
Unbiased2	**6.909E-2**	**6.137E-2**	**1.112E-1**	**1.417E-1**

Table 3: Results for the HJB equation: Unbiased2 that corrects all the biases from the mean square error loss function and the PDE nonlinearity performs the best in all dimensions and in different settings with various cost functions.

In most cases of the HJB equation, the model that corrects both biases (unbiased2) performs the best. Following this, the model that corrects one bias in the MSE loss function (unbiased1) shows the next best performance, while the completely biased model performs the worst. This highlights the correctness of our analysis regarding the two biases introduced by analyzing nonlinear equations and underscores the improvement achieved by bias correction. These HJB equations are not particularly high-dimensional, so our theoretical analysis suggests that using the unbiased version in this scenario is better than the biased one. This is because the former's variance won't be substantial in cases that are not highly dimensional. Lastly, this problem is difficult because we do not have a boundary condition but deal with the unbounded domain, and the PDE solution quickly diverges to infinity when $\boldsymbol{x}$ tends to infinity, which makes the model lack information at infinity. Efficient PINN-based algorithms for such HJB equation on unbounded domains are still open questions in the literature. Here, we focus on comparing the performances of the biased and unbiased versions.

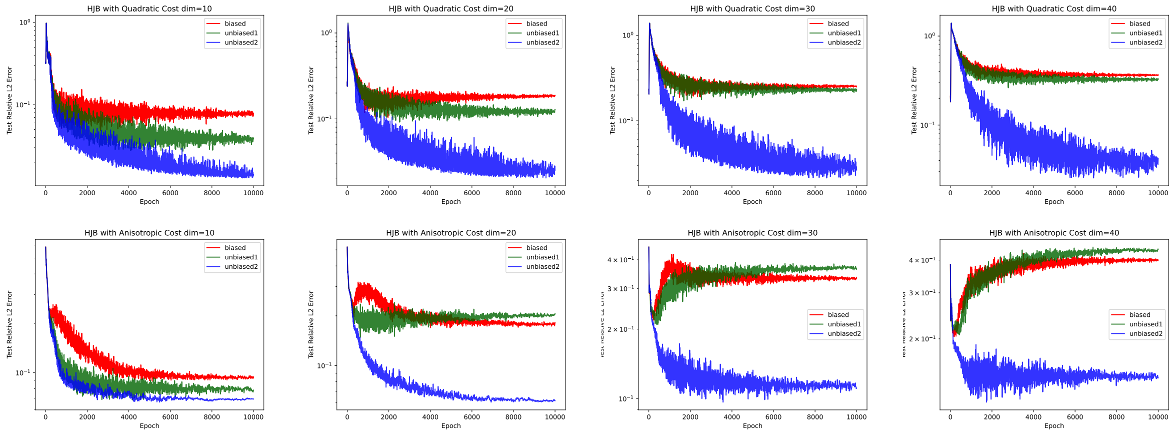

Figure 3: Fisrt row: results for the HJB equation with quadratic cost. Second row: results for the HJB equation with anisotropic Rosenbrock cost. In all dimensions of the HJB equation, the model that corrects both biases (unbiased2) performs the best. Following this, the model that corrects one bias (unbiased1) shows the next best performance, while the completely biased model performs the worst. This highlights the correctness of our analysis regarding the two biases introduced by analyzing nonlinear equations and underscores the improvement achieved by bias correction.

5.3 Viscous Burgers' PDE

In this section, we investigate the nonlinear viscous Burgers' equation. We focus on the influence of different nonlinearities on the biases introduced to RS-PINN.

The d-dimensional viscous Burgers' equation with the initial condition at $t = 0$ is given by

$$\begin{aligned} &u_t + u\left(\sum_{i=1}^{d}\frac{\partial u(\boldsymbol{x})}{\partial \boldsymbol{x}_i}\right) - \nu\left(\sum_{i=1}^{d}\frac{\partial^2 u(\boldsymbol{x})}{\partial \boldsymbol{x}_i^2}\right) = 0, \boldsymbol{x} \in \mathbb{R}^d, t \in [0,1]. \\ &u(\boldsymbol{x}, t=0) = \frac{1}{1+\exp\left(\frac{\sum_{i=1}^{d}\boldsymbol{x}_i}{2\nu}\right)}. \end{aligned} \tag{41}$$

Its analytical solution is given by

$$u(\boldsymbol{x}, t) = \frac{1}{1 + \exp\left(\frac{\sum_{i=1}^{d} \boldsymbol{x}_i - dt/2}{2\nu}\right)}. \tag{42}$$

We choose $\nu = 0.5$ and sample points based on its corresponding SDE trajectory as before: $t \sim \text{Unif}(0, 1), \boldsymbol{x} \sim \mathcal{N}(t, 2 - t)$. We use the boundary augmentation given by the following model output to satisfy the initial condition automatically [29]:

$$u_\theta(\boldsymbol{x}, t)t + \frac{1}{1 + \exp\left(\frac{\sum_{i=1}^{d} \boldsymbol{x}_i}{2\nu}\right)}. \tag{43}$$

The solution of this Burgers' equation is highly complex and exhibits stiff regions where $\sum_{i=1}^{d} \boldsymbol{x}_i = dt$. In these regions, the PDE solution experiences abrupt changes. This equation helps us evaluate the performance of RS-PINN on nonlinear equations with complex stiff solutions.

Here are the implementation details. The model is a 4-layer fully connected network with 128 hidden units, which is trained via Adam [27] for 10K epochs, with an initial learning rate 1e-3, which linearly exponentially with exponent 0.9995. We select 100 random residual points at each Adam epoch and 20K fixed testing points from the SDE trajectory. The sample sizes for randomized smoothing in both training and testing are 1024 and 128, respectively, and the variance of Gaussian noise is $\sigma = 1e - 2$.

Viscous Burgers' Equation	5D	10D	20D	25D
Biased	1.085E-02	5.293E-02	7.672E-02	5.698E-02
Unbiased1	1.089E-03	2.412E-03	1.497E-02	2.841E-02
Unbiased2	**1.030E-03**	**2.411E-03**	**1.165E-02**	**2.783E-02**

Table 4: Results for the viscous Burgers' equation.

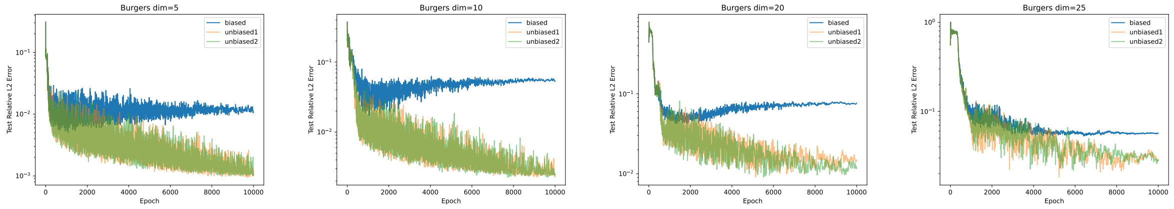

Figure 4: Convergence curves for the Burgers' equation. In this example, correcting the bias from the MSE loss (unbiased1) suffices to achieve optimal performance, while the unbiased2 method can only slightly improve over the unbiased1 method.

The results for the viscous Burgers' equation are shown in Table 4, and convergence curves with respect to epoch are shown in Figure 4. In all-dimensional cases in this example, unbiased1 and unbiased2 consistently outperform biased, underscoring once again the critical significance of unbiasedness for the convergence of RS-PINN. However, in contrast to the previous HJB equation case, unbiased1 alone is sufficient to achieve excellent results here, rendering unbiased2 unnecessary. Hence, the effect of bias-variance trade-off differs in various PDEs.

5.4 Allen-Cahn and Sine-Gordon PDEs with Anisotropic Solution

Here, we aim to consider nonseparable and anisotropic solutions for nonlinear PDEs to form complicated and nontrivial high-dimensional PDEs:

$$u_{\text{exact}}(\boldsymbol{x}) = \left(1 - \|\boldsymbol{x}\|_2^2\right) \left(\sum_{i=1}^{d-1} c_i \sin(\boldsymbol{x}_i + \cos(\boldsymbol{x}_{i+1}) + \boldsymbol{x}_{i+1} \cos(\boldsymbol{x}_i)) \right), \tag{44}$$

where $c_i \sim \mathcal{N}(0, 1)$. We do not want the boundary to leak most information about the exact solution, and thus, the term $1 - \|\boldsymbol{x}\|_2^2$ is added for a zero boundary condition. In addition to the exact solution, the following PDEs defined within the unit ball $\mathbb{B}^d$ associated with zero boundary conditions on the unit sphere are under consideration:

Sine-Gordon	10D	100D	10,00D
biased	5.712E-3	7.835E-3	6.744E-4
unbiased1	1.410E-3	7.223E-3	6.647E-3
unbiased2	N.A.	N.A.	N.A.
hybrid	**1.407E-3**	**7.209E-3**	**4.732E-4**

Allen-Cahn	10D	100D	10,00D
biased	5.062E-3	7.923E-3	5.504E-4
unbiased1	3.233E-3	7.298E-3	1.308E-3
unbiased2	3.217E-3	7.293E-3	1.957E-2
hybrid	**2.768E-3**	**7.285E-3**	**4.856E-4**

Table 5: Results for the Sine-Gordon PDE (first row) and the Allen-Cahn PDE (second row) with anisotropic exact solutions. Note that since the Sine-Gordon contains a sine nonlinearity, its unbiased2 version does not exist.

- Allen-Cahn equation

$$\Delta u(\boldsymbol{x}) + u(\boldsymbol{x}) - u(\boldsymbol{x})^3 = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{B}^d, \tag{45}$$

  where $g(\boldsymbol{x}) = \Delta u_{\text{exact}}(\boldsymbol{x}) + u_{\text{exact}}(\boldsymbol{x}) - u_{\text{exact}}(\boldsymbol{x})^3$.

- Sine-Gordon equation

$$\Delta u(\boldsymbol{x}) + \sin\left(u(\boldsymbol{x})\right) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{B}^d, \tag{46}$$

  where $g(\boldsymbol{x}) = \Delta u_{\text{exact}}(\boldsymbol{x}) + \sin\left(u_{\text{exact}}(\boldsymbol{x})\right)$.

These PDEs exhibit different levels of nonlinearity. Allen-Cahn involves third-order nonlinearity, and Sine-Gordon's nonlinearity stems from the term $\sin(u)$, making it infinite-order nonlinear in theory. It is not feasible to achieve a completely unbiased version for this case. However, we will demonstrate that correcting the bias originating from the mean square error loss in the PINN loss is sufficient.

Here are the implementation details. The model is a 4-layer fully connected network with 128 hidden units, which is trained via Adam [27] for 10K epochs, with an initial learning rate 1e-3, which linearly decays to zero at the end of the optimization. We select 100 random residual points at each Adam epoch and 20K fixed testing points uniformly from the unit ball. The sample size for randomized smoothing in both training and testing is 128, and the variance of Gaussian noise is $\sigma = 1e-2$. We adopt the following model structure to satisfy the zero boundary condition with hard constraint and to avoid the boundary loss [29]:

$$u_\theta^{\text{RS}}(\boldsymbol{x}) = (1 - \|\boldsymbol{x}\|_2^2) u_\theta(\boldsymbol{x}), \tag{47}$$

where $u_\theta(\boldsymbol{x})$ is the randomized smoothing neural network and $u_\theta^{\text{RS}}(\boldsymbol{x})$ is the boundary-augmented model. We repeat our experiment 5 times with 5 independent random seeds.

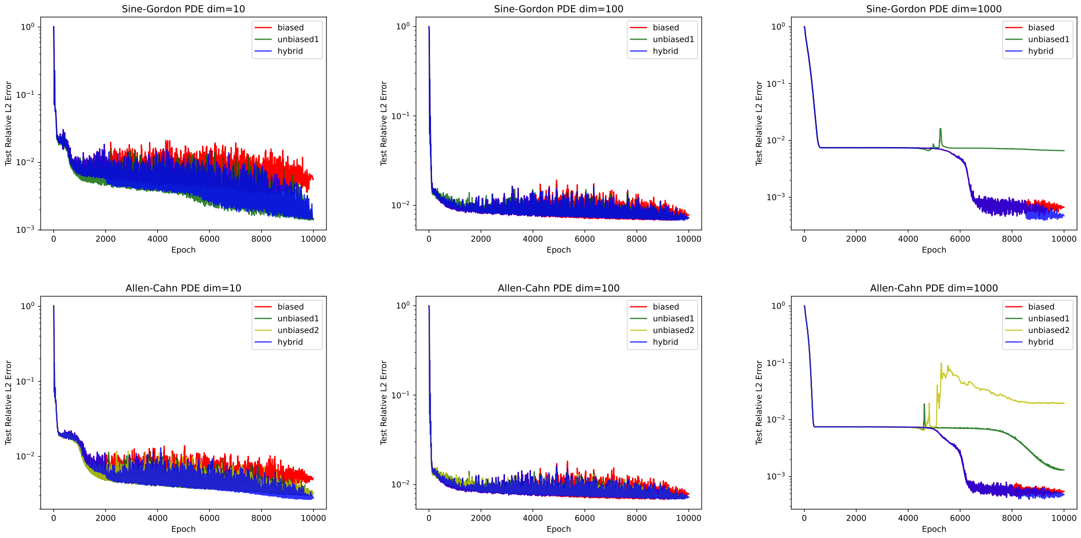

Figure 5: Results for the Sine-Gordon PDE (first row) and the Allen-Cahn PDE (second row) with anisotropic exact solutions.

The final convergence results and the convergence curves are shown in Table 5 and Figure 5, respectively. Overall, RS-PINN is able to deal with highly complicated and anisotropic PDE solutions in high dimensions. In terms of the comparison between the biased, unbiased, and hybrid versions of RS-PINN, we obtain similar observations as the first linear Fokker-Planck equation. Concretely, in the relatively lower 10D scenarios, unbiased versions surpass the biased version, primarily because at lower dimensions, the smaller variance of the unbiased version makes its unbiasedness more crucial for convergence. However, in 100D, the relatively larger variance of the unbiased version balances its negative effects with the positive aspects of its unbiasedness, resulting in similar performances between unbiased and biased versions. Moving to higher dimensions, particularly in the 1000D scenario, the considerable variance of the unbiased version prevents its convergence. Additionally, in the Allen-Cahn equation, unbiased2, with its amplified variance due to increased sampling, performs worse than unbiased1 with relatively less sampling. In this context, the hybrid version, starting with the biased version to reach an acceptable solution and fine-tuning with unbiased2, achieves the optimal outcome. Among all the dimensions, the hybrid version is the most stable, since it incorporates the low variance of the biased version with the optimal convergence results by unbiased version-based fine-tuning.

6 Summary

We have developed an extension of Physics-Informed Neural Networks (PINNs) and their Randomized Smoothing variant (RS-PINN) to address computational challenges in high-dimensional scenarios. The identified biases in RS-PINN, originating from the nonlinearity of the Mean Squared Error (MSE) loss and the inherent nonlinearity of the PDE, have been systematically corrected using tailored techniques. The derivation of an unbiased RS-PINN has allowed us to investigate its attributes, comparing them with the biased version and paving the way for the formulation of a novel combined hybrid approach.

Our proposed bias-variance trade-off strategy, incorporating both biased and unbiased versions, introduces a novel perspective on optimizing simultaneously convergence speed and accuracy. By strategically employing the rapid convergence of the biased version in the initial stages and transitioning to the accuracy of the unbiased version for fine-tuning, we strike a balance that adapts to the dynamic nature of the optimization process.

This work contributes not only to the advancement of RS-PINN methodology but also provides a new paradigm to address biases in high-dimensional scenarios. The presented guidelines offer practical insights for navigating the complexities of biased and unbiased RS-PINN implementations. The extensive experimental validation across various high-dimensional PDEs underscores the efficacy of our bias correction techniques, reinforcing the versatility and applicability of the hybrid RS-PINN approach.

Acknowledgement

This research of ZH, ZY, YW, and KK is partially supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-010-SGIL) and the Singapore Ministry of Education Academic Research Fund Tier 1 (Award No: T1 251RES2207). The work of GEK was supported by the MURI-AFOSR FA9550-20-1-0358 projects and by the DOE SEA-CROGS project (DE-SC0023191). GEK was also supported by the ONR Vannevar Bush Faculty Fellowship (N00014-22-1-2795).

A Proof

A.1 Proof of Theorem 4.1

Proof. Thanks to the independence between the random variables δ_i and δ'_i

$$\begin{aligned}
\mathbb{E}_{\delta,\delta'}\left[L_b^{(1)}(\theta)\right] &= \mathbb{E}_{\delta,\delta'}\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta'_i;\theta)-g(\boldsymbol{x})\right)\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)-g(\boldsymbol{x})\right)\right] \\
&= \mathbb{E}_{\delta'}\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta'_i;\theta)-g(\boldsymbol{x})\right)\right]\cdot\mathbb{E}_{\delta}\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)-g(\boldsymbol{x})\right)\right] \\
&= \left(\mathbb{E}_{\delta'}\left[\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta'_i;\theta)\right]-g(\boldsymbol{x})\right)\cdot\left(\mathbb{E}_{\delta}\left[\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right]-g(\boldsymbol{x})\right) \\
&= \left(\mathbb{E}_{\delta}\left[f(\boldsymbol{x}+\delta;\theta)\right]-g(\boldsymbol{x})\right)\cdot\left(\mathbb{E}_{\delta}\left[f(\boldsymbol{x}+\delta;\theta)\right]-g(\boldsymbol{x})\right) \\
&= L_b(\theta).
\end{aligned} \tag{48}$$

For their gradients with respect to θ, by the chain rule

$$\begin{aligned}
\mathbb{E}_{\delta,\delta'}\left[\frac{\partial L_b^{(1)}(\theta)}{\partial\theta}\right] &= 2\mathbb{E}_{\delta,\delta'}\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta';\theta)-g(\boldsymbol{x})\right)\cdot\frac{\partial}{\partial\theta}\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)-g(\boldsymbol{x})\right)\right] \\
&= 2\mathbb{E}_{\delta'}\left[\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta'_i;\theta)-g(\boldsymbol{x})\right)\right]\cdot\mathbb{E}_{\delta}\left[\frac{\partial}{\partial\theta}\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)-g(\boldsymbol{x})\right)\right] \\
&= 2\mathbb{E}_{\delta}\left[\left(f(\boldsymbol{x}+\delta;\theta)-g(\boldsymbol{x})\right)\right]\cdot\mathbb{E}_{\delta}\left[\frac{\partial}{\partial\theta}\left(f(\boldsymbol{x}+\delta;\theta)-g(\boldsymbol{x})\right)\right] \\
&= \mathbb{E}_{\delta}\left[\frac{\partial}{\partial\theta}\left(f(\boldsymbol{x}+\delta;\theta)-g(\boldsymbol{x})\right)^2\right] \\
&= \frac{\partial L_b(\theta)}{\partial\theta}.
\end{aligned} \tag{49}$$

□

A.2 Proof of Theorem 4.2

Proof.

$$\begin{aligned}
\mathbb{E}_{\delta,\delta',\delta'',\delta'''}\left[L_r^{(2)}(\theta)\right] = &\left(\left\langle\mathbb{E}_{\delta}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right],\mathbb{E}_{\delta'}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{\delta'_i}{\sigma^2}f(\boldsymbol{x}+\delta'_i;\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\times \\
&\left(\left\langle\mathbb{E}_{\delta''}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{\delta''_i}{\sigma^2}f(\boldsymbol{x}+\delta''_i;\theta)\right],\mathbb{E}_{\delta'''}\left[\frac{1}{K}\sum_{i=1}^{K}\frac{\delta'''_i}{\sigma^2}f(\boldsymbol{x}+\delta'''_i;\theta)\right]\right\rangle-g(\boldsymbol{x})\right) \\
= &\left(\left\langle\mathbb{E}_{\delta}\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right],\mathbb{E}_{\delta}\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\times \\
&\left(\left\langle\mathbb{E}_{\delta}\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right],\mathbb{E}_{\delta}\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right]\right\rangle-g(\boldsymbol{x})\right) \\
= &L_r(\theta).
\end{aligned} \tag{50}$$

$$\begin{aligned}\mathbb{E}_{\delta,\delta',\delta'',\delta'''}\left[\frac{\partial}{\partial\theta}L_r^{(2)}(\theta)\right] &= 2\left(\left\langle\mathbb{E}_\delta\left[\frac{1}{K}\sum_{i=1}^K\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right],\mathbb{E}_{\delta'}\left[\frac{1}{K}\sum_{i=1}^K\frac{\delta_i'}{\sigma^2}f(\boldsymbol{x}+\delta_i';\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\times\\ &\quad\frac{\partial}{\partial\theta}\left(\left\langle\mathbb{E}_{\delta''}\left[\frac{1}{K}\sum_{i=1}^K\frac{\delta_i''}{\sigma^2}f(\boldsymbol{x}+\delta_i'';\theta)\right],\mathbb{E}_{\delta'''}\left[\frac{1}{K}\sum_{i=1}^K\frac{\delta_i'''}{\sigma^2}f(\boldsymbol{x}+\delta_i''';\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\\ &= 2\left(\left\langle\mathbb{E}_\delta\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right],\mathbb{E}_\delta\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\times\\ &\quad\frac{\partial}{\partial\theta}\left(\left\langle\mathbb{E}_\delta\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right],\mathbb{E}_\delta\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right]\right\rangle-g(\boldsymbol{x})\right)\\ &= \frac{\partial}{\partial\theta}L_r(\theta).\end{aligned}\tag{51}$$

□

B Detailed Loss Function

B.1 Hamilton-Jacobi-Bellman PDE

The previous HJB equation given by

$$u_t=\Delta_{\boldsymbol{x}}u-\|\nabla_{\boldsymbol{x}}u(\boldsymbol{x})\|^2. \tag{52}$$

which has a nonlinearity order of two due to the $\|\nabla_{\boldsymbol{x}}u\|^2$ term.

As before, to simplify the discussion, we assume the residual condition is $g(\boldsymbol{x})$, and we ignore the linear term of the HJB equation and only consider the nonlinear term: $\|\nabla_{\boldsymbol{x}}u(\boldsymbol{x})\|^2$. Recall that $\nabla_{\boldsymbol{x}}u(\boldsymbol{x};\theta)=\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[\frac{\delta}{\sigma^2}f(\boldsymbol{x}+\delta;\theta)\right]$.

The true residual loss on a residual point $\boldsymbol{x}$ is:

$$L_r(\theta)=\left(\|\nabla_{\boldsymbol{x}}u(\boldsymbol{x};\theta)\|^2-g(\boldsymbol{x})\right)^2 \tag{53}$$

The biased loss from He et al. [14] is

$$L_r^{(0)}(\theta)=\left(\left\|\frac{1}{K}\sum_{i=1}^K\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right\|^2-g(\boldsymbol{x})\right)^2. \tag{54}$$

The unbiased1 loss by correcting the bias from the nonlinear MSE loss solely is

$$L_r^{(1)}(\theta)=\left(\left\|\frac{1}{K}\sum_{i=1}^K\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right\|^2-g(\boldsymbol{x})\right)\times\left(\left\|\frac{1}{K}\sum_{i=1}^K\frac{\delta_i'}{\sigma^2}f(\boldsymbol{x}+\delta_i';\theta)\right\|^2-g(\boldsymbol{x})\right). \tag{55}$$

The unbiased2 loss by correcting the biases from the MSE loss and the PDE nonlinearity is

$$\begin{aligned}L_r^{(2)}(\theta)=&\left(\left\langle\frac{1}{K}\sum_{i=1}^K\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta),\frac{1}{K}\sum_{i=1}^K\frac{\delta_i'}{\sigma^2}f(\boldsymbol{x}+\delta_i';\theta)\right\rangle-g(\boldsymbol{x})\right)\times\\&\left(\left\langle\frac{1}{K}\sum_{i=1}^K\frac{\delta_i''}{\sigma^2}f(\boldsymbol{x}+\delta_i'';\theta),\frac{1}{K}\sum_{i=1}^K\frac{\delta_i'''}{\sigma^2}f(\boldsymbol{x}+\delta_i''';\theta)\right\rangle-g(\boldsymbol{x})\right).\end{aligned}\tag{56}$$

B.2 Allen-Cahn PDE

For the Allen-Cahn (AC) PDE given by

$$u_t=\Delta_{\boldsymbol{x}}u+u-u^3, \tag{57}$$

its nonlinearity stems from the term u^3, which is a cubic function. Therefore, the nonlinearity order of the AC equation is three. During the model training, we are required to sample the three u terms in $u^3=u\cdot u\cdot u$ independently for unbiased gradients.

As before, to simplify the discussion, we assume the residual condition is $g(\boldsymbol{x})$, and we ignore the linear term of the HJB equation and only consider the nonlinear term: u^3,. Recall that $u(\boldsymbol{x};\theta)=\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2\boldsymbol{I})}\left[f(\boldsymbol{x}+\delta;\theta)\right]$.

The true residual loss on a residual point $\boldsymbol{x}$ is:

$$L_r(\theta) = \big(u(\boldsymbol{x};\theta)^3 - g(\boldsymbol{x})\big)^2 \tag{58}$$

The biased loss from He et al. [14] is

$$L_r^{(0)}(\theta) = \left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right)^3 - g(\boldsymbol{x})\right)^2. \tag{59}$$

The unbiased1 loss by correcting the bias from the nonlinear MSE loss solely is

$$L_r^{(1)}(\theta) = \left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right)^3 - g(\boldsymbol{x})\right) \times \left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i';\theta)\right)^3 - g(\boldsymbol{x})\right). \tag{60}$$

The unbiased2 loss by correcting the biases from the MSE loss and the PDE nonlinearity is

$$\begin{aligned} L_r^{(2)}(\theta) = &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(1)};\theta)\right)\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(2)};\theta)\right)\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(3)};\theta)\right) - g(\boldsymbol{x})\right) \times \\ &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(4)};\theta)\right)\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(5)};\theta)\right)\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i^{(6)};\theta)\right) - g(\boldsymbol{x})\right), \end{aligned} \tag{61}$$

where $\delta_i^{(1)}, \delta_i^{(2)}, \delta_i^{(3)}, \delta_i^{(4)}, \delta_i^{(5)}, \delta_i^{(6)}$ are six independent groups of Gaussian random samples.

B.3 Viscous Burgers' PDE

For the viscous Burgers' equation:

$$u_t + u\left(\sum_{i=1}^{d}\frac{\partial u(\boldsymbol{x})}{\partial \boldsymbol{x}_i}\right) - \nu\left(\sum_{i=1}^{d}\frac{\partial^2 u(\boldsymbol{x})}{\partial \boldsymbol{x}_i^2}\right) = 0, \boldsymbol{x}\in\mathbb{R}^d, t\in[0,1], \tag{62}$$

its nonlinearity stems from the term $u\sum_{i=1}^{d} u_{\boldsymbol{x}_i}$. Therefore, the nonlinearity order of the viscous Burgers' PDE is three. During the model training, we are required to sample the u and $\nabla_{\boldsymbol{x}} u$ independently for unbiased gradients.

As before, to simplify the discussion, we assume the residual condition is $g(\boldsymbol{x})$, and we ignore the linear term of the HJB equation and only consider the nonlinear term: $u\sum_{i=1}^{d} u_{\boldsymbol{x}_i}$. Furthermore, we will use the operator sum to denote the element-wise sum of a vector.

The true residual loss on a residual point $\boldsymbol{x}$ is:

$$L_r(\theta) = \left(u(\boldsymbol{x};\theta)\sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}u(\boldsymbol{x};\theta) - g(\boldsymbol{x})\right)^2 \tag{63}$$

The biased loss from He et al. [14] is

$$L_r^{(0)}(\theta) = \left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right)\operatorname{sum}\left(\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right) - g(\boldsymbol{x})\right)^2. \tag{64}$$

The unbiased1 loss by correcting the bias from the nonlinear MSE loss solely is

$$\begin{aligned} L_r^{(1)}(\theta) = &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right)\operatorname{sum}\left(\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i}{\sigma^2}f(\boldsymbol{x}+\delta_i;\theta)\right) - g(\boldsymbol{x})\right) \times \\ &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i';\theta)\right)\operatorname{sum}\left(\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i'}{\sigma^2}f(\boldsymbol{x}+\delta_i';\theta)\right) - g(\boldsymbol{x})\right). \end{aligned} \tag{65}$$

The unbiased2 loss by correcting the biases from the MSE loss and the PDE nonlinearity is

$$\begin{aligned} L_r^{(2)}(\theta) = &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right)\operatorname{sum}\left(\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i'}{\sigma^2}f(\boldsymbol{x}+\delta_i';\theta)\right) - g(\boldsymbol{x})\right) \times \\ &\left(\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i'';\theta)\right)\operatorname{sum}\left(\frac{1}{K}\sum_{i=1}^{K}\frac{\delta_i'''}{\sigma^2}f(\boldsymbol{x}+\delta_i''';\theta)\right) - g(\boldsymbol{x})\right). \end{aligned} \tag{66}$$

B.4 Sine-Gordon PDE

However, our method cannot deal with nonlinear like $\sin(u)$ in the Sine-Gorden PDE:

$$u_t = \Delta_{\boldsymbol{x}} u + \sin(u) \tag{67}$$

However, we can increase the sample size K in the Monte Carlo to minimize the bias. Fortunately, correcting the forward and backward bias does suffice for the Sine-Gordon equation, i.e., we can still correct the bias from the forward and backward passes to improve over the original formulation in He et al. [14], which is actually sufficient to obtain a low error in high dimensions.

As before, to simplify the discussion, we assume the residual condition is $g(\boldsymbol{x})$, and we ignore the linear term of the HJB equation and only consider the nonlinear term: $\sin\left(u(\boldsymbol{x})\right)$.

The true residual loss on a residual point $\boldsymbol{x}$ is:

$$L_r(\theta) = \left(\sin\left(u(\boldsymbol{x};\theta)\right) - g(\boldsymbol{x})\right)^2 \tag{68}$$

The biased loss from He et al. [14] is

$$L_r^{(0)}(\theta) = \left(\sin\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right) - g(\boldsymbol{x})\right)^2 \tag{69}$$

The unbiased1 loss by correcting the bias from the nonlinear MSE loss solely is

$$L_r^{(1)}(\theta) = \left(\sin\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i;\theta)\right) - g(\boldsymbol{x})\right) \times \left(\sin\left(\frac{1}{K}\sum_{i=1}^{K} f(\boldsymbol{x}+\delta_i';\theta)\right) - g(\boldsymbol{x})\right). \tag{70}$$

The unbiased2 loss that corrects the two biases does not exist for this equation due to the sine nonlinearity.

References

[1] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. Deep splitting method for parabolic pdes. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154, 2021.

[2] Christian Beck, Weinan E, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29:1563–1619, 2019.

[3] Christian Beck, Lukas Gonon, and Arnulf Jentzen. Overcoming the curse of dimensionality in the numerical approximation of high-dimensional semilinear elliptic partial differential equations. *arXiv preprint arXiv:2003.00596*, 2020.

[4] Christian Beck, Fabian Hornung, Martin Hutzenthaler, Arnulf Jentzen, and Thomas Kruse. Overcoming the curse of dimensionality in the numerical approximation of allen–cahn partial differential equations via truncated full-history recursive multilevel picard approximations. *Journal of Numerical Mathematics*, 28(4):197–222, 2020.

[5] Sebastian Becker, Ramon Braunwarth, Martin Hutzenthaler, Arnulf Jentzen, and Philippe von Wurstemberger. Numerical simulations for full history recursive multilevel picard approximations for systems of high-dimensional partial differential equations. *arXiv preprint arXiv:2005.10206*, 2020.

[6] Jesse Bettencourt, Matthew J. Johnson, and David Duvenaud. Taylor-mode automatic differentiation for higher-order derivatives in JAX. In *Program Transformations for ML Workshop at NeurIPS 2019*, 2019.

[7] Quentin Chan-Wai-Nam, Joseph Mikael, and Xavier Warin. Machine learning for semi linear pdes. *Journal of scientific computing*, 79(3):1667–1712, 2019.

[8] Pao-Hsiung Chiu, Jian Cheng Wong, Chinchun Ooi, My Ha Dao, and Yew-Soon Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

[9] Junwoo Cho, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, and Eunbyung Park. Separable pinn: Mitigating the curse of dimensionality in physics-informed neural networks. *arXiv preprint arXiv:2211.08761*, 2022.

[10] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019.

[11] Haozhe Feng, Tianyu Pang, Chao Du, Wei Chen, Shuicheng Yan, and Min Lin. Does federated learning really need backpropagation? *arXiv preprint arXiv:2301.12195*, 2023.

[12] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[13] Jiequn Han, Arnulf Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, 5(4):349–380, 2017.

[14] Di He, Shanda Li, Wenlei Shi, Xiaotian Gao, Jia Zhang, Jiang Bian, Liwei Wang, and Tie-Yan Liu. Learning physics-informed neural networks without stacked back-propagation. In *International Conference on Artificial Intelligence and Statistics*, pages 3034–3047. PMLR, 2023.

[15] Pierre Henry-Labordere. Deep primal-dual algorithm for bsdes: Applications of machine learning to cva and im. *Available at SSRN 3071506*, 2017.

[16] Zheyuan Hu, Ameya D. Jagtap, George Em Karniadakis, and Kenji Kawaguchi. When do extended physics-informed neural networks (xpinns) improve generalization? *SIAM Journal on Scientific Computing*, 44(5):A3158–A3182, 2022.

[17] Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *arXiv preprint arXiv:2307.12306*, 2023.

[18] Côme Huré, Huyên Pham, and Xavier Warin. Deep backward schemes for high-dimensional nonlinear pdes. *Mathematics of Computation*, 89(324):1547–1579, 2020.

[19] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, Tuan Anh Nguyen, and Philippe von Wurstemberger. Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. *Proceedings of the Royal Society A*, 476(2244):20190630, 2020.

[20] Martin Hutzenthaler, Arnulf Jentzen, Thomas Kruse, et al. Multilevel picard iterations for solving smooth semilinear parabolic heat equations. *Partial Differential Equations and Applications*, 2(6):1–31, 2021.

[21] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[22] Shaolin Ji, Shige Peng, Ying Peng, and Xichuan Zhang. Three algorithms for solving high-dimensional fully coupled fbsdes through deep learning. *IEEE Intelligent Systems*, 35(3):71–84, 2020.

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

[24] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems (NeurIPS)*, pages 586–594, 2016.

[25] Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. How does information bottleneck help deep learning? In *International Conference on Machine Learning (ICML)*, 2023.

[26] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *Cambridge University Press*, 2022.

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[28] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[29] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[30] Chunyue Lv, Lei Wang, and Chenming Xie. A hybrid physics-informed neural network for nonlinear partial differential equation. *arXiv preprint arXiv:2112.01696*, 2021.

[31] Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes. *arXiv preprint arXiv:2006.16144*, 2020.

[32] Guofei Pang, Lu Lu, and George Em Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[33] Maziar Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

[34] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[35] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *arXiv preprint arXiv:2004.01806*, 2020.

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

[37] Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is \$l\^2\$ physics informed loss always suitable for training physics informed neural network? In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[38] Yifan Wang, Pengzhan Jin, and Hehu Xie. Tensor neural network and its numerical integration. *arXiv preprint arXiv:2207.02754*, 2022.

[39] Yifan Wang, Yangfei Liao, and Hehu Xie. Solving schr\"{o} dinger equation using tensor neural network. *arXiv preprint arXiv:2209.12572*, 2022.