Error Bounds for Physics-Informed Neural Networks in Fokker-Planck PDEs

Chun-Wei Kong[1] **Luca Laurenti**[2] **Jay McMahon**[1] **Morteza Lahijanian**[1]

[1]Dept. of Aerospace Eng. Sciences, University of Colorado Boulder
[2]Center for Systems and Control , Delft University of Technology

Abstract

Stochastic differential equations are commonly used to describe the evolution of stochastic processes. The state uncertainty of such processes is best represented by the probability density function (PDF), whose evolution is governed by the Fokker-Planck partial differential equation (FP-PDE). However, it is generally infeasible to solve the FP-PDE in closed form. In this work, we show that physics-informed neural networks (PINNs) can be trained to approximate the solution PDF. Our main contribution is the analysis of PINN approximation error: we develop a theoretical framework to construct tight error bounds using PINNs. In addition, we derive a practical error bound that can be efficiently constructed with standard training methods. We discuss that this error-bound framework generalizes to approximate solutions of other linear PDEs. Empirical results on nonlinear, high-dimensional, and chaotic systems validate the correctness of our error bounds while demonstrating the scalability of PINNs and their significant computational speedup in obtaining accurate PDF solutions compared to the Monte Carlo approach.

1 INTRODUCTION

Stochastic differential equations (SDEs) are widely used to model the evolution of stochastic processes across various fields, including sciences, engineering, and finance. In many of these applications, particularly in *safety-critical* domains, a key concern is understanding how the state uncertainty in SDEs propagates over space and time. This state uncertainty can be represented by probability density function (PDF), which is governed by the Fokker-Planck partial differential equation (FP-PDE). However, analytical solutions for general FP-PDEs are unavailable, and numerical methods—e.g., finite elements or finite difference [Spencer and Bergman, 1993, Drozdov and Morillo, 1996, Masud and Bergman, 2005, Pichler et al., 2013, Qian et al., 2019, Ureña et al., 2020]—are typically employed, but these methods scale poorly as the dimensionality grows beyond three [Tabandeh et al., 2022]. Recent advancements in deep-learning suggest physics-informed learning frameworks, called *physics-informed neural networks* (PINNs), can effectively learn PDE solutions, showing notable success in handling high-dimensional systems (up to 200 dimensions) and complex geometries [Sirignano and Spiliopoulos, 2018, Lu et al., 2021]. Despite their effectiveness, PINNs are still subject to approximation errors, a crucial concern in safety-critical systems. In this work, we tackle this challenge by developing a novel framework to approximate FP-PDE solutions using PINNs and rigorously bounding the approximation error.

Recent works on using PINNs to approximate solutions to PDEs typically analyze approximation errors in terms of *total* error, capturing cumulative approximation error across space and time [De Ryck and Mishra, 2022b,a, Mishra and Molinaro, 2023, De Ryck et al., 2024]. While useful in some applications, this approach is less informative for SDEs and their PDF propagation. Moreover, total error bounds are often overly loose, sometimes exceeding the actual errors by several orders of magnitude. Crucially, these bounds do not provide insight into the worst-case approximation error at specific time instances or within particular subsets of space, which is essential in many stochastic systems. For example, in autonomous driving scenarios involving pedestrian crossings, accurate prediction and bounding the probability of collision requires precise reasoning over specific time instances and spatial regions. Loose over-approximations can lead to undesirable behaviors, such as sudden braking.

In this work, we show how PINNs can be used to approximate solutions to FP-PDE (i.e., PDF of an SDE's state) and, more importantly, introduce a framework for tightly bounding the worst-case approximation error over the subset of interest in state space as a function of time. Our key insight

is that the approximation error is related to the residual of the FP-PDE and is governed by another PDE. Hence, a second PINN can be used to learn the error, with its own error also following a PDE. This results in a recursive formulation of error functions, each of which can be approximated using a PINN. We establish sufficient training conditions under which this series converges with a finite number of terms. Specifically, we prove that two PINNs are enough to obtain arbitrarily tight error bounds. Additionally, we derive a more practical bound requiring only one error PINN at the cost of losing arbitrary tightness, and provide a method to verify its sufficient condition. Furthermore, we propose a training scheme with regularization and discuss extensions to other linear PDEs. Finally, we illustrate and validate these error bounds through experiments on several SDEs, supporting our theoretical claims.

In short, the main contribution is five-fold:

- a method for approximating the PDF of processes modeled by SDEs using PINNs,
- a novel approach to tightly bound the approximation error over time and space through a recursive series of error functions learned by PINNs,
- a proof that this recursive process converges with only two PINNs needed for arbitrarily tight bounds,
- the derivation of a more practical error bound requiring just one PINN, along with a method to verify its sufficiency, and
- validation of the proposed error bounds through experiments on several SDEs.

Related Work Research on using PINNs to approximate PDE solutions often focuses on total error, which represents the cumulative error across all time and space. For instance, Mishra and Molinaro [2023] derive an abstract total error bound. Nevertheless, their numerical experiments reveal that this total error bound is loose, exceeding the actual errors by nearly three orders of magnitude. A similar approach is extended to Navier-Stokes equations [De Ryck et al., 2024], with comparable results. De Ryck and Mishra [2022a] consider FP-PDEs derived from linear SDEs only. They propose an abstract approach to bound the total error, but no numerical experiments are presented. De Ryck and Mishra [2022b] also derive total error bounds for PINNs (and operators) assuming a priori error estimate. In contrast, our work emphasizes bounding the worst-case error at any time of interest for general SDEs, which is particularly valuable in practical applications of stochastic systems (e.g., systems subject to chance constraints [Oguri and McMahon, 2021, Paiola et al., 2024]).

To demonstrate the approximation capabilities of neural networks, error analysis is a key. For example, Hornik [1991] proves that a standard multi-layer feed-forward neural network can approximate a target function arbitrarily well. Yarotsky [2017] considers the worst-case error and shows that deep ReLU neural networks are able to approximate universal functions in the Sobolev space. Recently, deep operator nets (DeepONet) have been suggested to learn PDE operators, with Lanthaler et al. [2022] proving that for every $\epsilon > 0$, there exists DeepONets such that the total error is smaller than ϵ. While these studies show the capabilities of neural networks, they do not address the critical question: what are the quantified errors for a given neural network approximation? This is the central issue tackled by our work.

Error estimates have also been investigated when neural networks are trained as surrogate models for given target functions. For instance, Barron [1994] derives the error between the learned network and target function in terms of training configurations. To learn a latent function with quantified error, Gaussian process regression [Archambeau et al., 2007] is often employed, where observations of an underlying process are required to learn the mean and covariance. Recently, Yang et al. [2022] estimate the worst-case error given target functions and neural network properties. Nevertheless, a fundamental difference between our work and these studies is that we do not assume knowledge of the true solutions (latent functions) or rely on data from the underlying processes.

Solving PDEs is an active research area with various established approaches. For the FP-PDE equation, numerical methods, such as the finite elements, finite differences, or Galerkin projection methods, have been employed [Spencer and Bergman, 1993, Drozdov and Morillo, 1996, Masud and Bergman, 2005, Chakravorty, 2006, Pichler et al., 2013, Qian et al., 2019, Ureña et al., 2020]. For PDF propagation, instead of solving the FP-PDE, some approaches perform a time-discretization of the SDE and use Gaussian mixture models [Terejanu et al., 2008]. Recent works [Khoo et al., 2019, Song et al., 2025, Lin and Ren, 2024] employ numerical methods for approximating transition probability between two regions, which is also governed by the FP-PDE. While these studies show accurate approximations from posterior evaluation, they can be computationally demanding and often lack rigorous error quantification and bounding.

2 PROBLEM FORMULATION

The aim of this work is state uncertainty propagation with quantified error bounds for continuous time and space stochastic processes using deep neural networks. We specifically focus on (possibly nonlinear) Stochastic Differential Equations (SDEs) described by

$$d\boldsymbol{x}(t) = f(\boldsymbol{x}(t), t)dt + g(\boldsymbol{x}(t), t)d\boldsymbol{w}(t), \tag{1}$$

where $t \in T \subseteq \mathbb{R}_{\geq 0}$ is time, $\boldsymbol{x}(t) \in X \subseteq \mathbb{R}^n$ is the system state at time t, and $\boldsymbol{w}(t) \in \mathbb{R}^m$ is a standard Brownian motion. For $\Omega = X \times T$, function $f : \Omega \to \mathbb{R}^n$ represents the deterministic evolution of the system, and function

$g : \Omega \to \mathbb{R}^{n\times m}$ is a term that defines the coupling of the noise. We assume that $f(x,t)$ and $g(x,t)$ satisfy the usual regularity conditions (e.g., see Evans [2022, Ch. 7.1]), and denote the i-th dimension of f and (j,k)-th element of g by f_i and g_{jk}, respectively. The initial state $\boldsymbol{x}(0)$ is a random variable distributed according to a given probability density function (PDF) $p_0 : X \to \mathbb{R}_{\geq 0}$, i.e., $\boldsymbol{x}(0) \sim p_0$. We assume that p_0 is bounded and smooth.

The solution to the SDE in Eq. (1) is a stochastic process $\boldsymbol{x}$ with a corresponding PDF $p : \Omega \to \mathbb{R}_{\geq 0}$ over space and time, i.e., $\boldsymbol{x}(t) \sim p(\cdot, t)$ [Øksendal, 2003]. PDF p is governed by the Fokker-Planck partial differential equation (FP-PDE):

$$\frac{\partial p(x,t)}{\partial t} + \sum_{i=1}^{n} \frac{\partial}{\partial x_i}[f_i p(x,t)] - \frac{1}{2}\sum_{i=1,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}\left[\sum_{k=1}^{m} g_{ik} g_{jk} p(x,t)\right] = 0, \quad (2)$$

and must satisfy the initial condition

$$p(x,0) = p_0(x) \qquad \forall x \in X. \quad (3)$$

To simplify notation, we denote by $\mathcal{D}[\cdot]$ the differential operator associated with the FP-PDE, i.e.,

$$\mathcal{D}[\cdot] := \frac{\partial}{\partial t}[\cdot] + \sum_{i=1}^{n} \frac{\partial}{\partial x_i}[f_i \cdot] - \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}\left[\sum_{k=1}^{m} g_{ik} g_{jk} \cdot\right].$$

Then, Eqs. (2) and (3) can be rewritten in a compact form as

$$\mathcal{D}[p(x,t)] = 0 \quad \text{subject to} \quad p(x,0) = p_0(x). \quad (4)$$

Note that, since f and g are assumed to be regular, the PDE in Eq. (4) is well-posed, i.e., there exists a smooth and unique solution.

Obtaining solution $p(x,t)$ to Eq. (4) in closed-form is generally not possible, and even numerical approaches are limited to simple SDEs [Spencer and Bergman, 1993, Drozdov and Morillo, 1996, Masud and Bergman, 2005, Chakravorty, 2006, Pichler et al., 2013, Qian et al., 2019, Ureña et al., 2020]. In this work, we focus on using PINNs to approximate p, and crucially, we aim to formally bound the resulting approximation error.

Problem 1 *Given FP-PDE in Eq.* (4)*, a bounded subset $X' \subset X$, and a bounded time interval $T' \subset T$, train a neural network $\hat{p}(x,t)$ that approximates the solution $p(x,t)$, and for every $t \in T'$ construct an error bound $B : T' \to \mathbb{R}_{>0}$ such that*

$$\sup_{x \in X'} |p(x,t) - \hat{p}(x,t)| \leq B(t). \quad (5)$$

Note that no data is assumed on p. Instead, our approach leverages the governing Eq. (4) for both training $\hat{p}$ and quantifying its error. Specifically, we first show that existing PINN training methods for PDE solutions can be adapted to approximate p effectively if the training loss is sufficiently small. Then, we show that the resulting approximation error can be written as a series of approximate error functions, each of which satisfying a PDE similar to Eq. (4). This implies that each error function itself can be approximated using a PINN. Then, we derive conditions, under which only a finite number of such PINNs is needed to obtain a guaranteed error bound $B(t)$.

Remark 1 *While we focus on $\hat{p}$ being a neural network, our method of deriving error bound $B(t)$ is not limited to neural networks and generalizes to any smooth function $\hat{p}$ that approximates the true solution p.*

3 PDF APPROXIMATION AND ERROR CHARACTERIZATION VIA PINNS

Here, we first describe a method for training a neural network to approximate PDF $p(x,t)$ for Problem 1, and then derive a recursive error-learning approach to estimate the approximation error.

Learning PDF p We approximate $p(x,t)$ by learning a neural network $\hat{p}(x,t;\theta)$, where θ represents the parameters. During training, spatio-temporal data points $\{(x_j,0)_j\}_{j=1}^{N_0}, \{(x_j,t_j)_j\}_{j=1}^{N_r} \subset \Omega$, for some $N_0, N_r \in \mathbb{N}$, are sampled, and the loss function is derived from the governing physics in Eq. (4) as

$$\mathcal{L} = w_0 \mathcal{L}_0 + w_r \mathcal{L}_r, \quad (6)$$

where $w_0, w_r \in \mathbb{R}^+$ are the weights, and

$$\mathcal{L}_0 = \frac{1}{N_0}\sum_{j=1}^{N_0} \big(p_0(x_j) - \hat{p}(x_j,0;\theta)\big)^2, \quad (7a)$$

$$\mathcal{L}_r = \frac{1}{N_r}\sum_{j=1}^{N_r} \big(\mathcal{D}[\hat{p}(x_j,t_j;\theta)]\big)^2. \quad (7b)$$

The loss function in Eq. (6) quantifies the deviation of the true and approximate solutions in terms of the initial condition ($\mathcal{L}_0$) and the infinitesimal variation over space and time ($\mathcal{L}_r$). The parameters of $\hat{p}(x,t;\theta)$ are learned by minimizing the loss function, i.e., $\theta^* = \arg\min \mathcal{L}$.

We note that overfitting can arise if the training samples in Eqs. (7) do not sufficiently cover the domain Ω. To address this, we rely on common sampling practices in PINNs (e.g., uniformly sampled [Sirignano and Spiliopoulos, 2018]) along with an adaptive sampling method in Lu et al. [2021] to improve sampling efficiency (see Appendix C for details).

Assumption 1 $\hat{p} : \Omega \to \mathbb{R}$ *is assumed to be smooth.*

Assumption 1 is present because $\hat{p}$ is trained by the physics-informed loss in Eq. (6), in which the second term $\mathcal{L}_r$ requires the computation of the first and second derivatives with respect to time and space, respectively. To satisfy Assumption 1, smooth activation functions (e.g., Tanh and $\mathrm{Softplus}$) can be used in the architecture of $\hat{p}(x,t;\theta)$. While $\hat{p}$ here is real-valued, one can further ensure $\hat{p} \geq 0$ using non-negative activation functions (e.g., exponent or squared) for the last layer.

The weights w_0 and w_r in Eq. 6 balance the initial-condition and PDE-residual terms. Although tuning these weights can affect the speed of convergence in practice, the training convergence does not rely on finding optimal weights [Shin et al., 2020, Mishra and Molinaro, 2023].

We emphasize that this method of training requires only the initial PDF $p_0(x)$ and differential operator $\mathcal{D}$, allowing loss evaluation on unlimited space-time samples. This key distinction sets PINNs apart from data-driven learning, which relies on (limited) data of (unavailable) true solution p.

Recursive Learning of Approximation Error Given trained $\hat{p}$, we show that its approximation error can be characterized as a series of approximate solutions to PDEs. Specifically, we define the error as

$$e_1(x,t) := p(x,t) - \hat{p}(x,t). \tag{8}$$

Note that FP-PDE operator $\mathcal{D}$ is a linear operator; hence, by applying it to e_1, we obtain:

$$\mathcal{D}[e_1] = \mathcal{D}[p - \hat{p}] = \mathcal{D}[p] - \mathcal{D}[\hat{p}].$$

As $\mathcal{D}[p] = 0$, we can see that the error is essentially related to the residue of $\mathcal{D}[\hat{p}]$. Then, we can define the governing PDE of $e_1(x,t)$ as

$$\mathcal{D}[e_1] + \mathcal{D}[\hat{p}] = 0 \ \text{ s.t. } \ e_1(x,0) = p_0(x) - \hat{p}(x,0). \tag{9}$$

Hence, using a similar approach as in Eqs. (6) and (7), a PINN can be trained to approximate $e_1(x,t)$ using its governing physics in Eq. (9). Based on this, we can define the i-th error and its associated approximation in a recursive manner.

Definition 1 (i-th error and approximation) *Let $e_0 := p$ and $\hat{e}_0 := \hat{p}$. For $i \geq 1$, we define the i-th error*

$$e_i(x,t) := e_{i-1}(x,t) - \hat{e}_{i-1}(x,t),$$

where each $\hat{e}_i$ is a smooth and bounded function constructed by PINN to approximate e_i. Each e_i is the solution to the recursive PDE

$$\mathcal{D}[e_i(x,t)] + \sum_{j=1}^{i} \mathcal{D}[\hat{e}_{j-1}(x,t)] = 0 \quad s.t. \\ e_i(x,0) = e_{i-1}(x,0) - \hat{e}_{i-1}(x,0).$$

By the construction in Definition 1, the approximation error $e_1(x,t)$, for every choice of $n \geq 0$, is given by

$$\begin{aligned} e_1(x,t) &= p(x,t) - \hat{p}(x,t) \\ &= \sum_{i=1}^{n} \hat{e}_i(x,t) + e_{n+1}(x,t). \end{aligned} \tag{10}$$

Although this recursive procedure estimates the unknown approximation error e_1, it does not directly provide a worst-case error bound. This is because, regardless of how many $\hat{e}_i$, $i = 1, 2, \ldots, n$, are constructed, an unquantified error term e_{n+1} always remains. To address this, we present our error bound theory in the next section.

4 ARBITRARY TIGHT ERROR BOUND

Here, we derive upper bounds for the approximation error e_1, specifically, for the right-hand side of Eq. (10). We show that, by training just two PINNs under certain sufficient conditions, the series can be bounded with arbitrary precision. All proofs for the lemmas and theorems are provided in Appendices A.2–A.6.

First, we express how well $\hat{e}_i$ approximates the i-th error e_i by defining the *relative approximation* factor $\alpha_i(t)$ as

$$\alpha_i(t) := \frac{\max_{x \in X'} |e_i(x,t) - \hat{e}_i(x,t)|}{\max_{x \in X'} |\hat{e}_i(x,t)|}. \tag{11}$$

Recall from Definition 1 that $e_i - \hat{e}_i = e_{i+1}$. Hence, Eq. (11) can be written in a recursive form as

$$\max_{x \in X'} |e_{i+1}(x,t)| = \alpha_i(t) \max_{x \in X'} |\hat{e}_i(x,t)|, \tag{12}$$

which relates the unknown $(i+1)$-th error to the i-th error approximation. Now, let $e_i^*(t)$ and $\hat{e}_i^*(t)$ denote the maximum of $e_i(x,t), \hat{e}_i(x,t)$ over X', respectively, i.e.,

$$e_i^*(t) := \max_{x \in X'} |e_i(x,t)|, \tag{13a}$$

$$\hat{e}_i^*(t) := \max_{x \in X'} |\hat{e}_i(x,t)|. \tag{13b}$$

Recall that each $\hat{e}_i(x,t)$ can be represented using a PINN. Hence, it is safe to assume that the absolute value of its upper-bound is strictly greater than zero.

Assumption 2 *Assume that, for all $1 \leq i < n$, $\hat{e}_i^*(t) > 0$.*

Then, the following lemma upper-bounds the approximation error $e_1(x,t)$ using $\hat{e}_i^*(t)$.

Lemma 1 *Consider the approximation error $e_1(x,t) = p(x,t) - \hat{p}(x,t)$ in Eq. (10) with $n \geq 2$, and the upper-bounds $\hat{e}_i^*(t)$ for $1 \leq i < n$ in Eq. (13). Define ratio*

$$\gamma_{\frac{i+1}{i}}(t) := \frac{\hat{e}_{i+1}^*(t)}{\hat{e}_i^*(t)}. \tag{14}$$

Then, under Assumption 2, it holds that, $\forall x \in X'$,

$$|e_1(x,t)| \le \hat{e}_1^*(t)\Big(1 + \sum_{m=2}^{n}\prod_{i=1}^{m-1}\gamma_{\frac{i+1}{i}}(t) + \frac{e_{n+1}^*}{\hat{e}_{n-1}^*}\prod_{i=1}^{n-2}\gamma_{\frac{i+1}{i}}(t)\Big) \quad (15)$$

Next, we derive an upper- and lower-bound for the ratio $\gamma_{\frac{i+1}{i}}(t)$ in Eq. (15) using $\alpha_i(t)$.

Lemma 2 *If the relative approximation factors $\alpha_i(t) < 1$ for all $2 \le i < n$, then*

$$\frac{\alpha_{i-1}(t)}{1+\alpha_i(t)} \le \gamma_{\frac{i}{i-1}(t)} \le \frac{\alpha_{i-1}(t)}{1-\alpha_i(t)}. \quad (16)$$

Lemma 2 establishes the relationship between ratio $\gamma_{\frac{i}{i-1}}$ and relative approximation factors α_i under condition $\alpha_i < 1$. Intuitively, this condition holds when $\hat{e}_i$ approximates e_i reasonably well (see Eq. (11)). Lastly, we show that under certain conditions on α_1 and α_2, an ordering over $\gamma_{\frac{2}{1}}, \gamma_{\frac{3}{2}}, \ldots, \gamma_{\frac{i}{i-1}}$ can be achieved.

Lemma 3 *If, for all $t \in T'$,*

$$0 < \alpha_1(t) < 1, \quad (17a)$$
$$0 < \alpha_2(t) < 1 - \alpha_1(t), \quad (17b)$$
$$\alpha_2(t)(1+\alpha_2(t)) < \alpha_1(t)^2, \quad (17c)$$

then there exist feasible $0 \le \alpha_i(t) < 1$ for $2 < i < n$ such that

$$\gamma_{\frac{i}{i-1}}(t) < \gamma_{\frac{2}{1}}(t) < 1. \quad (18)$$

The intuition behind Lemma 3 is that if $\hat{e}_1$ and $\hat{e}_2$ are trained to certain accuracy (satisfying Conditions 17), then there exist feasible $\hat{e}_3, \hat{e}_4, \ldots, \hat{e}_{n-1}$ such that the ratios $\gamma_{\frac{3}{2}}, \gamma_{\frac{4}{3}}, \ldots, \gamma_{\frac{n-1}{n-2}}$ are upper bounded by $\gamma_{\frac{2}{1}} < 1$. Equipped with Lemmas 1-3, we can state our main result, which is an upper-bound on the approximation error of $\hat{p}$. Specifically, the following theorem shows that the approximation error bound in Lemma 1 becomes a geometric series as $n \to \infty$ under Conditions 17; hence, solving Problem 1.

Theorem 1 (Second-order error bound) *Consider Problem 1 and two approximate error functions $\hat{e}_1(x,t), \hat{e}_2(x,t)$ constructed by Definition 1 that satisfy Conditions 17. Then,*

$$|p(x,t) - \hat{p}(x,t)| \le B_2(t) = \hat{e}_1^*(t)\Big(\frac{1}{1-\gamma_{\frac{2}{1}}(t)}\Big), \quad (19)$$

where $\hat{e}_1^*(t)$ is defined in Eq. (13)*, and $\gamma_{\frac{2}{1}}(t) = \hat{e}_2^*(t)/\hat{e}_1^*(t)$.*

The above theorem shows that the second-order error bound $B_2(t)$ can be obtained by training only two PINNs that approximate the first two errors e_1, e_2 according to Definition 1 and that satisfy Conditions 17. In fact, using these two PINNs, it is possible to construct an arbitrary tight B_2 as stated below.

Theorem 2 (Arbitrary tightness) *Given Problem 1 and tolerance $\epsilon \in (0, \infty)$ on the error bound, an error bound $B_2(t)$ in Theorem 1 can be obtained by training two approximate error functions $\hat{e}_1(x,t)$ and $\hat{e}_2(x,t)$ through physics-informed learning such that*

$$B_2(t) - \max_{x \in X'} |e_1(x,t)| < \epsilon. \quad (20)$$

The proof of Theorem 2 is based on the observation that $\gamma_{\frac{2}{1}} \to 0$ when (i) $\hat{e}_1(x,t) \to e_1(x,t)$ and (ii) $\hat{e}_2(x,t) \to e_2(x,t)$. Then, according to Eq. (19), $B_2(t) \to \hat{e}_1^*(t)$, which itself $\hat{e}_1^*(t) \to e_1^*(t)$ under (i). By the theoretical convergence of PINNs [Shin et al., 2020, Mishra and Molinaro, 2023], $\hat{e}_1$ and $\hat{e}_2$ can be made arbitrary well; thus B_2 can be arbitrary tight. This result is important because it shows that arbitrary tightness can be achieved without the need for training infinite number of PINNs, i.e., $\hat{e}_i$, $i = 1, 2, \ldots$

Remark 2 *The construction of $B_2(t)$ in Theorem 1 only requires the values of $\hat{e}_1^*(t)$ and $\gamma_{\frac{2}{1}}(t)$ which are obtained from the known functions $\hat{e}_1(x,t), \hat{e}_2(x,t)$. Checking for α_1 and α_2 conditions can be performed a posterior.*

Feasibility Analysis Now, we analyze the feasibility for $\hat{e}_1$ and $\hat{e}_2$ satisfying Conditions 17 in Theorem 1. Specifically, Condition 17a on α_1 indicates that $\hat{e}_1$ must be learned well enough so that the magnitude of its maximum approximation error is less than its own maximum magnitude (see Eq. (11)). By fixing α_1, Conditions 17b-17c on α_2 require $\hat{e}_2$ to approximate e_2 more accurately than the approximation of e_1 by $\hat{e}_1$. These conditions are feasible in principle by the same convergence argument above. However, there are some practical challenge as discussed below.

Practical Challenge The challenge to construct $B_2(t)$ stems from the condition that $\hat{e}_2$ needs to approximate e_2 far more accurately than $\hat{e}_1$ approximates e_1, making training a PINN for $\hat{e}_2$ extremely difficult. Additionally, since the explicit values of α_1 and α_2 are unknown, there is no clear criterion for determining when to stop training $\hat{e}_1$ and $\hat{e}_2$. To address this, we provide a method for verifying the condition on α_1 and derive a bound that depends only on this condition below.

5 FIRST-ORDER ERROR BOUND

Here, we introduce a first-order error bound using a single PINN, overcoming the challenge of obtaining B_2. We also

provide an implicit formula to determine training termination. Additionally, we discuss the feasibility of this approach, outline our training scheme, and explore relevant extensions. Complete proofs can be found in Appendices A.7 and A.8.

By considering Eq. (10) with $n = 2$ and using Lemmas 1 and 2, we can derive the first-order error bound, as stated in the following corollary.

Corollary 1 (First-order error bound) *Consider Problem 1, and let $\hat{e}_1$ be trained such that $\alpha_1(t) < 1$ for all $t \in T'$. Then*

$$|p(x,t) - \hat{p}(x,t)| < B_1(t) = 2\hat{e}_1^*(t). \tag{21}$$

Note that the first-order error bound $B_1(t)$ can be at most twice as large as the arbitrary tight second-order bound $B_2(t)$ in Theorem 1, but it offers significant practical advantages. Firstly, the second-order bound $B_2(t)$ requires training a second PINN $\hat{e}_2$ after training $\hat{e}_1$. However, achieving the required accuracy for $\hat{e}_2$ in practice is quite challenging. In contrast, the first-order bound in Corollary 1 relies solely on the approximation provided by a single PINN, $\hat{e}_1$. Secondly, the condition $\alpha_1(t) < 1$ can be verified during $\hat{e}_1$ training using properties of the FP-PDE, as detailed below.

Checking $\alpha_1 < 1$ Condition From the definition of $\alpha_1(t)$ in Eq. (11), it suffices to bound the unknown term $|e_1(x,t) - \hat{e}_1(x,t)|$ for all $(x,t) \in \Omega$ to check for $\alpha_1(t) < 1$. We do this by using three constants: the first two constants are related to PDE stability and quadrature rules [Mishra and Molinaro, 2023], and the third constant comes from Sobolev embedding theorem [Hunter and Nachtergaele, 2001, Theorem 12.71][Mizuguchi et al., 2017].

The first constant C_{pde} is related to the *stability* of the first error PDE, which is defined as

$$\|e_1 - \hat{e}_1\|_Z \leq C_{pde}\|(\mathcal{D}[e_1] + \mathcal{D}[\hat{p}]) - (\mathcal{D}[\hat{e}_1] + \mathcal{D}[\hat{p}])\|_Y,$$

where $Z = W^{k,q}$ norm , $Y = L^s$ norm, $1 \leq s, q < \infty$, and $k \geq 0$. Note that since $e_1, \hat{e}_1$ and $(\mathcal{D}[e_1] + \mathcal{D}[\hat{p}]) - (\mathcal{D}[\hat{e}_1] + \mathcal{D}[\hat{p}]) = 0 - (\mathcal{D}[\hat{e}_1] + \mathcal{D}[\hat{p}])$ are bounded , such constant C_{pde} exists.

The second constant $C_{quad} > 0$ is related to the deviation between integral and its approximation with finite samples. Let $\mathcal{I} = \int_\Omega \Big(\mathcal{D}[\hat{e}_1(x,t)] + \mathcal{D}[\hat{p}(x,t)]\Big)dxdt$ be the integral of interest, and $\bar{\mathcal{I}} = \sum_{j=1}^N w_j \Big(\mathcal{D}[\hat{e}_1(x_j,t_j)] + \mathcal{D}[\hat{p}(x_j,t_j)]\Big)$ be its associated approximation, where $\{(x_j,t_j)_j\}_{j=1}^N \in \Omega$ is a set of N quadrature points, and $w_j \in \mathbb{R}_{>0}$ are weights according to the quadrature rules. Then C_{quad} is defined such that, for some $\beta > 0$, $\left|\mathcal{I} - \bar{\mathcal{I}}\right| \leq C_{quad}N^{-\beta}$.

The third constant C_{embed} from Sobolev embedding theorem is defined as

$$\|e_1(x,t) - \hat{e}_1(x,t)\|_\infty \leq C_{embed}\|e_1(x,t) - \hat{e}_1(x,t)\|_{W^{1,q}}.$$

Constant C_{embed} exists because $e_1(x,t)$ and $\hat{e}_1(x,t)$ are bounded (per Definition 1), and the first derivatives of $e_1(x,t)$ and $\hat{e}_1(x,t)$ are also bounded over the considered domain of Problem 1. With these constants, we propose an implicit checking formula for $\alpha_1(t) < 1$.

Proposition 1 (Checking $\alpha_1(t) < 1$) *Let $\{(x_j,t_j)_j\}_{j=1}^N \in \Omega$ be N space-time samples based on quadrature rules, $\hat{e}_1(x,t)$ be the first error approximation, and $\mathcal{L}^{(1)}$ be the physics-informed loss of $\hat{e}_1(x,t)$ evaluated on the set $\{(x_j,t_j)_j\}_{j=1}^N$. Then for some $q \geq 2$ and $\beta > 0$, $\alpha_1(t) < 1$ for all $t \in T'$ if*

$$C_{embed}C_{pde}\Big(\mathcal{L}^{(1)} + C_{quad}^{\frac{1}{q}}N^{\frac{-\beta}{q}}\Big) < \min_t \hat{e}_1^*(t). \tag{22}$$

By Proposition 1, it is clear that as the training loss decreases ($\mathcal{L}^{(1)} \to 0$) with sufficiently large number of samples ($N \to \infty$), the left-hand side of Eq. (22) approaches zero. Hence, condition $\alpha_1 < 1$ can be satisfied as validated in our numerical evaluations.

Note that, for the constants in Proposition 1, it is sufficient to have upper bounds. Specifically, Mishra and Molinaro [2023] show a method of over-estimating C_{pde}. Constant C_{embed} depends on the domain geometry [Mizuguchi et al., 2017]. Also note that a sufficiently large N can ensure $C_{quad}N^{-\beta} \ll 1$.

Training Scheme for First-Order Error Bound Guided by Proposition 1, our goal is to train $\hat{e}_1$ to achieve sufficiently small loss. By the PINN loss in Eq. (6) and the PDE of the first error in Eqs. (8) and (9), the training loss of $\hat{e}_1$ is:

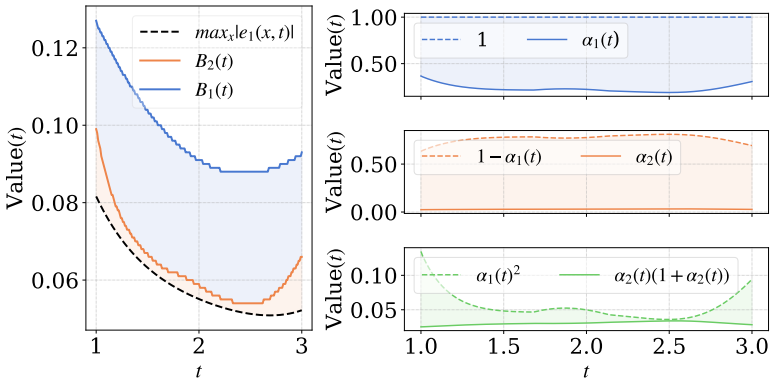
$$\mathcal{L}^{(1)} = w_0\mathcal{L}_0^{(1)} + w_r\mathcal{L}_r^{(1)}, \tag{23a}$$

$$\mathcal{L}_0^{(1)} = \frac{1}{N_0}\sum_{k=1}^{N_0}\big(p_0(x_k) - \hat{p}(x_k,0) - \hat{e}_1(x_k,0)\big)^2, \tag{23b}$$

$$\mathcal{L}_r^{(1)} = \frac{1}{N_r}\sum_{k=1}^{N_r}\big(\mathcal{D}[\hat{e}_1(x_k,t_k)] + \mathcal{D}[\hat{p}(x_k,t_k)]\big)^2. \tag{23c}$$

By Eq. (23b)–(23c), we see that training $\hat{e}_1$ requires inputs from neural network $\hat{p}$ and its derivatives $\mathcal{D}[\hat{p}]$. This could lead to difficult training for $\hat{e}_1$ if the input $\mathcal{D}[\hat{p}]$ is highly oscillating even when $\hat{p}$ is smooth by construction [Zhao et al., 2023]. To address this issue, we implement a regularization loss to prevent rapid changes in $\mathcal{D}[\cdot]$ of the first PINN $\hat{p}$. Specifically, we train $\hat{p}$ by adding the following regularization loss to Eq. (6):

$$\mathcal{L}_\nabla = \frac{1}{N_r}\sum_{k=1}^{N_r}\|\nabla\Big(\mathcal{D}[\hat{p}(x_k,t_k)]\Big)\|_2^2, \tag{24}$$



(a) $e_1^*(t) < B_2(t) < B_1(t)$ (b) Satisfy Conditions 17.

Figure 1: second-order error bound $B_2(t)$ on 1D Linear system

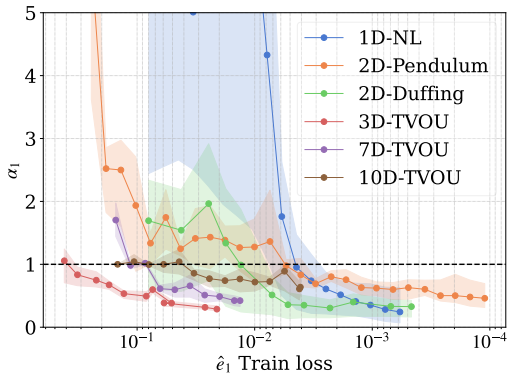


Figure 2: $\alpha_1(t),\ t \in T'$ vs train loss of $\hat{e}_1$, for all first-order error bound experiments.

where ∇ is the gradient operator and $\|\cdot\|_2$ is the L2 norm. We note that this regularization loss does not violate the paradigm of physics-informed learning, because as the residual of $\mathcal{D}[\cdot] \to 0$ for all $(x,t) \in \Omega$, the gradient of the differential residual $\nabla(\mathcal{D}[\cdot])$ also converges to zero. In fact, this regularization is investigated in Yu et al. [2022] to improve training stability of PINNs. Note that such gradient regularization loss in Eq. (24) is not applied to the training of $\hat{e}_1$ in Eqs. (23) because $\mathcal{D}[\hat{e}_1]$ is not used to train subsequent error functions. A detailed description of our training scheme is provided in Appendix C.

Remark 3 *Finally, we note that, while the presented approach focuses on FP-PDE and training an approximate PDF $\hat{p}$ and bounding its error, the only essential requirement is that the FP-PDE operator $\mathcal{D}[\cdot]$ is linear. Therefore, this approach naturally extends to other linear PDEs subject to initial and boundary conditions (e.g., Dirichlet and Neumann conditions). We illustrate this in a case study in Appendix A.9, B.6, and C.8.*

6 EXPERIMENT

We demonstrate our proposed error bound approach via several numerical experiments. First, we illustrate a synthetic second-order error bound on a simple linear system. Then, we present our main applications of the first-order error bound for non-trivial systems including nonlinear, chaotic, and high dimensional SDEs. The details of these systems are provided in Appendix B. For all the experiments, fully connected neural networks are used for both $\hat{p}(x,t)$ and $\hat{e}_1(x,t)$. Throughout all experiments, we employ a fixed weighting scheme for training (see Appendix C). More sophisticated weight-tuning strategies (e.g., Basir and Senocak [2023] and references therein) have been shown to enhance PINN accuracy and could further improve the results reported here. The code implemented in Python and Pytorch is available on GitHub [Kong, 2025]. All the experiments are conducted on a MacBook Pro with Apple M2 processor and 24GB RAM.

Second-order error bound illustration We consider a FP-PDE for a 1D linear SDE. This simple system has analytical PDF $p(x,t)$, which allows us to synthesize $\hat{e}_2(x,t)$ and validate second-order error bound $B_2(t)$. Specifically, we first train two PINNs in sequence: one $\hat{p}(x,t)$ and the other for $\hat{e}_1(x,t)$. Due to the practical challenge discussed in Section 4, we synthesize $\hat{e}_2(x,t)$ from the analytical solution and the trained PINNs, i.e., $\hat{e}_2(x,t) = p(x,t) - \hat{p}(x,t) - \hat{e}_1(x,t) + \delta(x,t)$, where $\delta(x,t)$ is a chosen sinusoidal perturbation such that Conditions 17 are satisfied. With the learned $\hat{e}_1(x,t)$ and synthesized $\hat{e}_2(x,t)$, we construct second- and first-order error bound $B_2(t)$ and $B_1(t)$ from Eqs. (19) and (21), respectively. Fig. 1a validates that both $B_2(t)$ and $B_1(t)$ upper-bound the worst-case error for all time. Furthermore, it shows that $B_2(t)$ is tighter than $B_1(t)$, and the relative tightness $B_1(t)/B_2(t)$ is at most $1.63 < 2$ as predicted by Corollary 1. Fig. 1b visualizes the satisfaction of the sufficient conditions on $\alpha_1(t)$ and $\alpha_2(t)$ in Eqs. (17) for all $t \in T'$. These results validate the soundness of our second-order error bound under the proposed conditions, and its tightness relative to the first-order bound.

First-order error bound application We apply the approach in Section 5 to construct error bound $B_1(t)$ for several FP-PDEs that do not have closed-form solutions. First, we consider FP-PDEs associated with nonlinear SDEs (see rows labeled as Nonlinear in Table 1). We note that 2D Duffing Oscillator is from Anderson and Farazmand [2024] and exhibits chaotic behaviors. Due to complexity of these systems, highly detailed Monte Carlo (M.C.) simulations with extremely small time steps and very large numbers of samples were necessary to generate high-fidelity 'ground-truth' PDF distributions at specific discrete time instances (see Appendix B). The final set of experiments consider high-dimensional (up to 10-D) time-varying Ornstein-Uhlenbeck (OU) processes (see rows labeled as High Dimensional in Table 1). We note that running M.C. to obtain 'ground-truth'

Table 1: First-order error bound results categorized into three groups: (1) Linear, where p is obtained analytically, (2) Nonlinear, where 'true' p is obtained by Monte-Carlo (M.C.) since no analytical solutions exist, and (3) High Dimensional, where 'true' p is obtained by semi-analytical numerical integration. Here, time p reports the computation time **in seconds** to obtain the 'true' p via M.C., time $\hat{p}$ and $\hat{e}_1$ are the training time for PINNs **in seconds**, $\alpha_1^{\max} := \max_t \alpha_1(t)$, α_1 reports the mean and standard deviation of $\alpha_1(t)$ over t. $\text{Gap}^{\min} := \min_t((B_1(t) - e_1^*(t))/\max_x p(x,t))$ is the minimum gap (over time) between the error bound and maximum error normalized by the true solution, and $B_1^{\mathcal{N}}$ reports the average and standard deviation (over time) of the normalized error bound $B_1(t)/\max_x p(x,t)$.

Category	**Experiment**	**time** p	**time** $\hat{p}$	**time** $\hat{e}_1$	$\alpha_1^{\max}$	α_1	$\textbf{Gap}^{\min}$	$B_1^{\mathcal{N}}$
Linear	1D Linear	Analytical	5	17	0.37	0.23 ± 0.04	7e-2	$0.18 \pm 6\text{e-}3$
Nonlinear	1D Nonlinear	37634 (M.C.)	1031	1840	0.63	0.24 ± 0.16	2e-2	$0.12 \pm 6\text{e-}2$
	2D Inverted Pendulum	45409 (M.C.).	1054	4484	0.69	0.46 ± 0.14	2e-2	$0.14 \pm 7\text{e-}2$
	2D Duffing Oscillator	39421 (M.C.)	358	3727	0.42	0.32 ± 0.11	8e-2	$0.21 \pm 3\text{e-}2$
High Dimensional	3D Time-varying OU	Semi-Analytical	267	943	0.34	0.29 ± 0.03	2e-2	$0.05 \pm 1\text{e-}2$
	7D Time-varying OU	Semi-Analytical	478	1475	0.49	0.43 ± 0.05	6e-2	$0.18 \pm 6\text{e-}2$
	10D Time-varying OU	Semi-Analytical	803	5954	0.83	0.64 ± 0.16	5e-2	$0.19 \pm 8\text{e-}2$

PDF for fully nonlinear and high-dimensional systems is not tractable. Hence, we choose these time-varying OU because they are non-trivial (i.e., no closed-form solutions) but allow us to efficiently estimate accurate $p(x,t)$ via semi-analytical method that numerically integrates the Gaussian distributions by exploiting the time-varying linear dynamics [Särkkä and Solin, 2019]. Thus, unlike our PINN-based method or standard M.C. simulations, this semi-analytical approach cannot handle general nonlinear dynamics.

Table 1 summarizes the results on all systems. The metrics include computation time for p and training times for $\hat{p}(x,t)$ and $\hat{e}_1$ PINNs in seconds. Also, we show training feasibility of $\hat{e}_1(x,t)$ by reporting $\alpha_1(t)$ and its maximum value over time ($\alpha^{\max}$). To show soundness of error bound B_1, we calculate the minimum gap $\text{Gap}^{\min}$ between B_1 and the true error, i.e., positive $\text{Gap}^{\min}$ indicate correctness of B_1. To assess the tightness of $B_1(t)$, we report the statistics of its magnitude normalized by $\max_x p(x,t)$, denoted by $B_1^{\mathcal{N}}$. Overall, the results show: (i) *soundness & feasibility:* B_1 correctly bounds the approximation error ($\text{Gap}^{\min} > 0$) and the learning condition is satisfied ($\alpha_1^{\max} < 1$) though it becomes increasingly challenging to meet as dimensionality grows, (ii) *efficiency:* the time comparisons between p vs $\hat{p}$ illustrate significant speedup (two orders of magnitude: $38\times$ to $65\times$) in obtaining accurate PDF with PINNs for systems that do not have analytical solutions (nonlinear systems), and (iii) *scalability:* our PINN method is able to scale to 10-dimensional systems with fairly tight error bounds across all cases (on average 5%-21% with respect to the true solution).

We plot $\alpha_1(t)$ distributions (over time) vs training loss of $\hat{e}_1$ in Fig. 2 to show that the condition in Corollary 1 is satisfied for all systems as training loss decreases. We visualize the error bounds for some representative cases in Fig. 3. Specifically, Fig. 3a plots the PDF $p(x,t)$ and its PINN approximation $\hat{p}(x,t)$ of the 1D Nonlinear experiment. Note that $\hat{p}(x,t)$ is a continuous surface over time and space, while $p(x,t)$ is illustrated by curves at discrete time instances according to those used in Monte-Carlo simulation. Fig. 3b shows the 'true' error $e_1(x,t)$ at discrete time instances and its PINN approximation $\hat{e}_1(x,t)$ as a continuous surface of the 1D Nonlinear experiment. Observe that both $\hat{p}$ and $\hat{e}$ closely approximate p and e, respectively, over all space and time, respectively. In addition, Fig. 3c shows the PINN-learned density $\hat{p}$ alongside the GMM approximation $\tilde{p}_{\text{GM}}$ (125 Gaussian mixtures integrated over $\Delta t = 0.001$ time step). We include the GMM—a common tool for uncertainty propagation in nonlinear dynamics [Archambeau et al., 2007, Terejanu et al., 2008, Vittaldev et al., 2016, Figueiredo et al., 2024]—as a classical alternative. Compared to the PINN $\hat{p}$, observe that $\tilde{p}_{GM}$ gradually deviates from p as time increases. More importantly, the true PDF p lies within the first-order error bound B_1 (illustrated by green regions) of approximate PDF $\hat{p}$, while $\tilde{p}_{GM}$ does not provide rigorous error bound. For the 2D Duffing Oscillator experiment, Fig. 3d plots the time evolution of the true error, PINN error approximation, and the first-order error bound. Observe that the magnitude of the error bound does not necessarily grow over time, suggesting that the error bound may remain tight even over extended horizons. Figs. 3e and 3f visualize PINNs results and the constructed B_1 of the 2D Inverted Pendulum experiment at a given time ($t = 3$). Observe that both $\hat{p}$ and $\hat{e}_1$ closely approximate the unknown complicated distributions of p and e_1, respectively. For the 3D Time-varying OU, Figs. 3g and 3h visualize the true error e_1 and PINN error $\hat{e}_1$ at $t = 1$, showing good approximation (i.e., $\alpha_1 < 1$) for constructing the first-order error bound B_1. Lastly, PINN predictions $\hat{u}$ and $\hat{e}_1$ for the 1D Heat PDE (Fig. 4) closely match the true solution u and error e_1 across space and time, respectively, demonstrating straightforward extension to other linear PDEs (Remark 3). See Appendix C for complete training details and additional results of the conducted experiments.

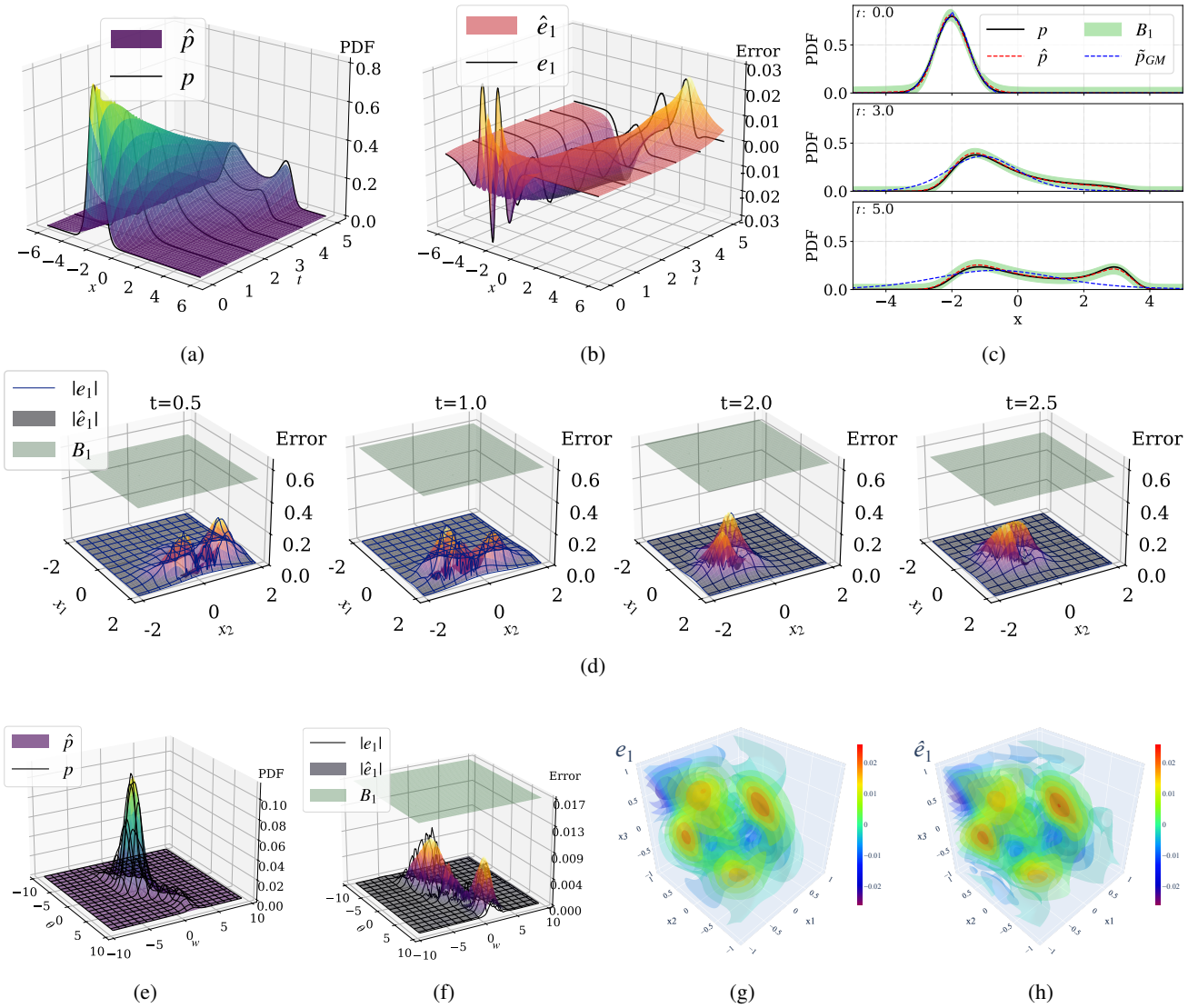
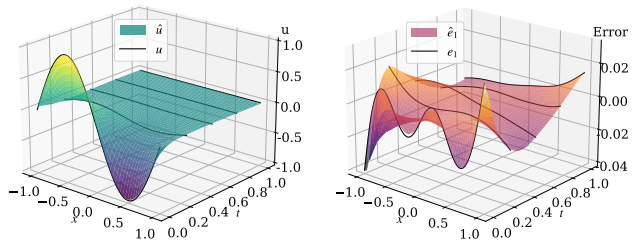


Figure 3: First-order error bound results. (a)-(c) 1D Nonlinear PDF p vs PINN $\hat{p}$, error e_1 vs error PINN $\hat{e}_1$, and error bound B_1 compared to the classical Gaussian mixture method $\hat{p}_{GM}$, illustrated at three time points. (d) 2D Duffing Oscillator true error $|e_1|$, error PINN $|\hat{e}_1|$, and error bound $B_1 \geq |e_1|$ over time. (e)-(f) 2D Inverted Pendulum PDF p, PINN $\hat{p}$, true error $|e_1|$, error PINN $|\hat{e}_1|$, and error bound $B_1 \geq |e_1|$ at $t = 3$. (g)-(h) 3D Time-varying OU error e_1 and error PINN $\hat{e}_1$ at $t = 1$.



(a) $\hat{u}$ vs u (b) $\hat{e}_1$ vs e_1

Figure 4: 1D Heat PINN solution $\hat{u}(x,t)$ and error $\hat{e}_1(x,t)$ v.s. true solution $u(x,t)$ and error $e_1(x,t)$.

7 CONCLUSION

We proposed a physics-informed learning method to approximate the PDF evolution governed by FP-PDE and bound its error using recursively learned PINN-based error functions. We proved that only two error terms are needed for arbitrarily tight bounds and introduced a more efficient first-order bound requiring just one error function, reducing computation while providing clear termination criteria. Our results validate the bounds' correctness and demonstrate significant computational speedups over Monte Carlo methods. We trained the solution and error PINNs separately, but joint training may work better, which is left for future work.

Acknowledgements

We acknowledge the anonymous reviewers for their constructive feedback. This material is based upon work supported by the Air Force Research Laboratory/RVSW under Contract No. FA945324CX026. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Research Laboratory/RVSW.

References

William Anderson and Mohammad Farazmand. Fisher information and shape-morphing modes for solving the fokker–planck equation in higher dimensions. *Applied Mathematics and Computation*, 467:128489, 2024.

Cedric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. Gaussian process approximations of stochastic differential equations. In *Gaussian Processes in Practice*, pages 1–16. PMLR, 2007.

Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine learning*, 14:115–133, 1994.

Shamsulhaq Basir and Inanc Senocak. An adaptive augmented lagrangian method for training physics and equality constrained artificial neural networks. *arXiv preprint arXiv:2306.04904*, 2023.

Suman Chakravorty. A homotopic galerkin approach to the solution of the fokker-planck-kolmogorov equation. In *2006 American Control Conference*, pages 6–pp. IEEE, 2006.

Tim De Ryck and Siddhartha Mishra. Error analysis for physics-informed neural networks (pinns) approximating kolmogorov pdes. *Advances in Computational Mathematics*, 48(6):79, 2022a.

Tim De Ryck and Siddhartha Mishra. Generic bounds on the approximation error for physics-informed (and) operator learning. *Advances in Neural Information Processing Systems*, 35:10945–10958, 2022b.

Tim De Ryck, Ameya D Jagtap, and Siddhartha Mishra. Error estimates for physics-informed neural networks approximating the navier–stokes equations. *IMA Journal of Numerical Analysis*, 44(1):83–119, 2024.

AN Drozdov and M Morillo. Solution of nonlinear fokker-planck equations. *Physical Review E*, 54(1):931, 1996.

Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Eduardo Figueiredo, Andrea Patane, Morteza Lahijanian, and Luca Laurenti. Uncertainty propagation in stochastic systems via mixture models with error quantification. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pages 328–335, 2024. doi: 10.1109/CDC56724.2024.10886416.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

John K Hunter and Bruno Nachtergaele. *Applied analysis*. World Scientific Publishing Company, 2001.

Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving for high-dimensional committor functions using artificial neural networks. *Research in the Mathematical Sciences*, 6:1–13, 2019.

Chun-Wei Kong. Pinn error repository. `https://github.com/aria-systems-group/pinn_pde/tree/release/uai2025`, 2025. Accessed: 2025-06-10.

Samuel Lanthaler, Siddhartha Mishra, and George E Karniadakis. Error estimates for deeponets: A deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6(1):tnac001, 2022.

Bo Lin and Weiqing Ren. Deep learning method for computing committor functions with adaptive sampling. *arXiv preprint arXiv:2404.06206*, 2024.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

Arif Masud and Lawrence A Bergman. Application of multiscale finite element methods to the solution of the fokker–planck equation. *Computer methods in applied mechanics and engineering*, 194(12-16):1513–1526, 2005.

Siddhartha Mishra and Roberto Molinaro. Estimates on the generalization error of physics-informed neural networks for approximating pdes. *IMA Journal of Numerical Analysis*, 43(1):1–43, 2023.

Makoto Mizuguchi, Kazuaki Tanaka, Kouta Sekine, and Shin'ichi Oishi. Estimation of sobolev embedding constant on a domain dividable into bounded convex domains. *Journal of inequalities and applications*, 2017: 1–18, 2017.

Kenshiro Oguri and Jay W McMahon. Robust spacecraft guidance around small bodies under uncertainty: Stochastic optimal control approach. *Journal of Guidance, Control, and Dynamics*, 44(7):1295–1313, 2021.

Bernt Øksendal. *Stochastic differential equations*. Springer, 2003.

Lorenzo Paiola, Giorgio Grioli, and Antonio Bicchi. On the evaluation of collision probability along a path. *IEEE Transactions on Robotics*, 2024.

Grigorios A Pavliotis. Stochastic processes and applications. *Texts in applied mathematics*, 60, 2014.

Lukas Pichler, Arif Masud, and Lawrence A Bergman. Numerical solution of the fokker–planck equation by finite difference and finite element methods—a comparative study. *Computational Methods in Stochastic Dynamics: Volume 2*, pages 69–85, 2013.

Yiran Qian, Zhongming Wang, and Shenggao Zhou. A conservative, free energy dissipating, and positivity preserving finite difference scheme for multi-dimensional nonlocal fokker–planck equation. *Journal of Computational Physics*, 386:22–36, 2019.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

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.

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

Zezheng Song, Maria K Cameron, and Haizhao Yang. A finite expression method for solving high-dimensional committor problems. *SIAM Journal on Scientific Computing*, 47(1):C1–C21, 2025.

BF Spencer and LA Bergman. On the numerical solution of the fokker-planck equation for nonlinear stochastic systems. *Nonlinear Dynamics*, 4:357–372, 1993.

Armin Tabandeh, Neetesh Sharma, Leandro Iannacone, and Paolo Gardoni. Numerical solution of the fokker–planck equation using physics-based mixture models. *Computer Methods in Applied Mechanics and Engineering*, 399: 115424, 2022.

Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D Scott. Uncertainty propagation for nonlinear dynamic systems using gaussian mixture models. *Journal of guidance, control, and dynamics*, 31(6):1623–1633, 2008.

Francisco Ureña, Luis Gavete, Ángel García Gómez, Juan José Benito, and Antonio Manuel Vargas. Nonlinear fokker-planck equation solved with generalized finite differences in 2d and 3d. *Applied Mathematics and Computation*, 368:124801, 2020.

Vivek Vittaldev, Ryan P Russell, and Richard Linares. Spacecraft uncertainty propagation using gaussian mixture models and polynomial chaos expansions. *Journal of Guidance, Control, and Dynamics*, 39(12):2615–2626, 2016.

Yejiang Yang, Tao Wang, Jefferson P Woolard, and Weiming Xiang. Guaranteed approximation error estimation of neural networks and model modification. *Neural Networks*, 151:61–69, 2022.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural networks*, 94:103–114, 2017.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393: 114823, 2022.

Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks. *arXiv preprint arXiv:2307.11833*, 2023.

Error Bounds for Physics-Informed Neural Networks in Fokker-Planck PDEs (Supplementary Material)

Chun-Wei Kong[1] **Luca Laurenti**[2] **Jay McMahon**[1] **Morteza Lahijanian**[1]

[1]Dept. of Aerospace Eng. Sciences, University of Colorado Boulder
[2]Center for Systems and Control , Delft University of Technology

A PROOFS

A.1 DERIVATION OF DEFINITION 1

Let $e_1(x,t) = p(x,t) - \hat{p}(x,t)$ and initialize $e_0(x,t) := p(x,t)$ and $\hat{e}_0(x,t) = \hat{p}(x,t)$. Then, Eq. (9) becomes Definition 1 for $i = 1$:

$$\mathcal{D}[e_1(x,t)] + \mathcal{D}[\hat{e}_0(x,t)] = 0, \quad \text{subject to } e_1(x,0) = e_0(x,0) - \hat{e}_0(x,0).$$

For $i = 2$, we define $e_2(x,t) := e_1(x,t) - \hat{e}_1(x,t)$ and obtain $\mathcal{D}[e_2(x,t)] = \mathcal{D}[e_1(x,t)] - \mathcal{D}[\hat{e}_1(x,t)]$ (because $\mathcal{D}[\cdot]$ is a linear operator). Since $\hat{e}_1 \neq e_1$, a residual will remain such that

$$\mathcal{D}[\hat{e}_1] + \mathcal{D}[\hat{e}_0] := r_1 \neq 0.$$

Hence, we have the recursive PDE for $i = 2$ (omitting x and t for simplicity of presentation):

$$\mathcal{D}[e_2] = \mathcal{D}[e_1] - \mathcal{D}[\hat{e}_1] = (-\mathcal{D}[\hat{e}_0]) - (-\mathcal{D}[\hat{e}_0] + r_1) = -r_1 \implies \mathcal{D}[e_2] + r_1 := \mathcal{D}[e_2] + \sum_{j=1}^{2} \mathcal{D}[\hat{e}_{j-1}] = 0.$$

The derivation recursively follows for $i > 2$. In addition, following PINNs training in Eq. (6), the training loss of each $\hat{e}_i$ is:

$$\mathcal{L}^{(i)} = w_0 \mathcal{L}_0^{(i)} + w_r \mathcal{L}_r^{(i)}, \ w_0, w_r \in \mathbb{R}_{>0}, \tag{25a}$$

$$\mathcal{L}_0^{(i)} = \frac{1}{N_0} \sum_{k=1}^{N_0} \Big(e_i(x_k, 0) - \hat{e}_i(x_k, 0) \Big)^2, \tag{25b}$$

$$\mathcal{L}_r^{(i)} = \frac{1}{N_r} \sum_{k=1}^{N_r} \Big(\mathcal{D}[\hat{e}_i(x_k, t_k)] + \sum_{j=1}^{i} \mathcal{D}[\hat{e}_{j-1}(x_k, t_k)] \Big)^2. \tag{25c}$$

A.2 PROOF OF LEMMA 1

Proof 1 *From Definition 1, we have that, for all $x \in X'$,*

$$\begin{aligned} |p(x,t) - \hat{p}(x,t)| &= \left| \sum_{i=1}^{n} \hat{e}_i(x,t) + e_{n+1}(x,t) \right| \leq \sum_{i=1}^{n} |\hat{e}_i(x,t)| + |e_{n+1}(x,t)| \\ &\leq \sum_{i=1}^{n} \max_x |\hat{e}_i(x,t)| + \max_x |e_{n+1}(x,t)| := \sum_{i=1}^{n} \hat{e}_i^*(t) + e_{n+1}^*(t). \end{aligned}$$

From the definition of $\gamma_{\frac{i+1}{i}}$ in Eq. (14)*, we obtain (omitting t for simplicity of presentation)*

$$
\begin{aligned}
&|p(x,\cdot)-\hat{p}(x,\cdot)| \le \hat{e}_1^*\Big(1+\frac{\hat{e}_2^*}{\hat{e}_1^*}+\frac{\hat{e}_3^*}{\hat{e}_1^*}+\cdots+\frac{\hat{e}_n^*}{\hat{e}_1^*}+\frac{e_{n+1}^*}{\hat{e}_1^*}\Big)\\
&=\hat{e}_1^*\Big[1+\gamma_{\frac{2}{1}}+\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}+\cdots+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n}{n-1}})+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\gamma_{\frac{n}{n-1}}\frac{e_{n+1}^*}{\hat{e}_n^*})\Big]\\
&=\hat{e}_1^*\Big[1+\gamma_{\frac{2}{1}}+\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}+\cdots+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n}{n-1}})+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\frac{\hat{e}_n^*}{\hat{e}_{n-1}^*}\frac{e_{n+1}^*}{\hat{e}_n^*})\Big]\\
&=\hat{e}_1^*\Big[1+\gamma_{\frac{2}{1}}+\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}+\cdots+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n}{n-1}})+(\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\frac{e_{n+1}^*}{\hat{e}_{n-1}^*})\Big].
\end{aligned}
$$

A.3 PROOF OF LEMMA 2

Proof 2 *From Definition 1, we have, for $i \ge 0$,*

$$e_i(x,t)=\hat{e}_i(x,t)+e_{i+1}(x,t). \tag{26}$$

By taking the maximum on the absolute value of Eq. (26)*, we get*

$$\max_x|e_i(x,t)| \le \max_x|\hat{e}_i(x,t)|+\max_x|e_{i+1}(x,t)|. \tag{27}$$

Similarly, from Eq. (26)*, we obtain*

$$
\begin{aligned}
&\hat{e}_i(x,t)=e_i(x,t)-e_{i+1}(x,t) \implies\\
&\max_x|\hat{e}_i(x,t)| \le \max_x|e_i(x,t)|+\max_x|e_{i+1}(x,t)|.
\end{aligned} \tag{28}
$$

Now take $2 \le i < n$, and suppose the corresponding $\alpha_i(t) < 1$. Then, we can write the two inequalities in Eqs. (27) *and* (28) *with the definition of $\hat{e}_i^*(t)$ in Eq.* (13) *and the expression in Eq.* (12) *as*

$$
\begin{cases}
\alpha_{i-1}(t)\hat{e}_{i-1}^*(t) \le \hat{e}_i^*(t)+\alpha_i(t)\hat{e}_i^*(t)\\
\hat{e}_i^*(t) \le \alpha_{i-1}(t)\hat{e}_{i-1}^*(t)+\alpha_i(t)\hat{e}_i^*(t).
\end{cases} \tag{29}
$$

By rearranging Eq. (29)*, we obtain the lower and upper bounds of $\gamma_{\frac{i}{i-1}}(t)$:*

$$\frac{\alpha_{i-1}(t)}{1+\alpha_i(t)} \le \frac{\hat{e}_i^*(t)}{\hat{e}_{i-1}^*(t)}=\gamma_{\frac{i}{i-1}(t)} \le \frac{\alpha_{i-1}(t)}{1-\alpha_i(t)},\ 2 \le i < n, \tag{30}$$

which is well defined because the denominator $\hat{e}_{i-1}^* > 0$ by Assumption 2, and the (RHS) of Eq. (30) *is always $\ge$ the (LHS) of Eq.* (30) *if $0 \le \alpha_i(t) < 1$ for all $2 \le i < n$.*

A.4 PROOF OF LEMMA 3

Proof 3 *For simplicity of presentation, we omit writing the dependent variable t. Assume the conditions in Eq.* (17) *are satisfied; then it is true that $0 < \alpha_2 < 1$. Since both $\alpha_1, \alpha_2 < 1$, by Lemma 2 and Condition* (17b)*, we obtain*

$$\gamma_{\frac{2}{1}} \le \frac{\alpha_1}{1-\alpha_2} < 1,$$

proving the RHS of Eq. (18)*.*

For the LHS of Eq. (18)*, let $\alpha_i \le \alpha_2$ for all $2 < i < n$. Since $\alpha_2 < 1$, then by Lemma 2, we have*

$$\gamma_{\frac{i}{i-1}} \le \frac{\alpha_{i-1}}{1-\alpha_i} \le \frac{\alpha_{i-1}}{1-\alpha_2} \le \frac{\alpha_2}{1-\alpha_2}. \tag{31}$$

What remains is to show that RHS of Eq. (31) *is* $< \gamma_{\frac{2}{1}}$. *From Condition* (17c)*, we have*

$$\alpha_2(1+\alpha_2) < \alpha_1^2 \tag{32}$$
$$< \alpha_1(1-\alpha_2), \tag{33}$$

where Eq. (33) *holds by Condition* (17b)*. From Eq.* (33)*, we obtain*

$$\frac{\alpha_2}{1-\alpha_2} < \frac{\alpha_1}{1+\alpha_2}. \tag{34}$$

By combining Eqs. (31) *and* (34)*, we have*

$$\gamma_{\frac{i}{i-1}} < \frac{\alpha_1}{1+\alpha_2} < \gamma_{\frac{2}{1}},\ 2 < i < n.$$

A.5 PROOF OF THEOREM 1

Proof 4 *Take* $n \to \infty$ *for Lemma 1, and train* $\hat{e}_1$ *and* $\hat{e}_2$ *such that the sufficient conditions of Eq.* (17) *are met, therefore,* $\gamma_{\frac{3}{2}}, \gamma_{\frac{4}{3}}, \ldots, \gamma_{\frac{n-1}{n-2}} < \gamma_{\frac{2}{1}} < 1$ *by Lemma 3. Then we have*

$$\begin{aligned}
&|p(x,t) - \hat{p}(x,t)| \\
&\leq \hat{e}_1^* \lim_{n\to\infty} \Big(1 + \gamma_{\frac{2}{1}} + \gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}} + \cdots + \Big[\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\gamma_{\frac{n}{n-1}}\Big] + \Big[\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\frac{e_{n+1}^*}{\hat{e}_{n-1}^*}\Big]\Big) \\
&= \hat{e}_1^* \lim_{n\to\infty} \Big(1 + \gamma_{\frac{2}{1}} + \gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}} + \cdots + \Big[\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\frac{\hat{e}_n^*}{\hat{e}_{n-1}^*}\Big] + \Big[\gamma_{\frac{2}{1}}\gamma_{\frac{3}{2}}\ldots\gamma_{\frac{n-1}{n-2}}\frac{e_{n+1}^*}{\hat{e}_{n-1}^*}\Big]\Big) \\
&\leq \hat{e}_1^* \lim_{n\to\infty} \Big(1 + \gamma_{\frac{2}{1}} + \gamma_{\frac{2}{1}}^2 + \cdots + \gamma_{\frac{2}{1}}^{n-2} + \Big[\gamma_{\frac{2}{1}}^{n-2}\frac{\hat{e}_n^*}{\hat{e}_{n-1}^*}\Big] + \Big[\gamma_{\frac{2}{1}}^{n-2}\frac{e_{n+1}^*}{\hat{e}_{n-1}^*}\Big]\Big) \\
&= \Big[\hat{e}_1^* \lim_{n\to\infty} \Big(1 + \gamma_{\frac{2}{1}} + \gamma_{\frac{2}{1}}^2 + \cdots + \gamma_{\frac{2}{1}}^{n-2}\Big)\Big] + \Big[\hat{e}_1^* \lim_{n\to\infty} \Big(\gamma_{\frac{2}{1}}^{n-2}\frac{(\hat{e}_n^* + e_{n+1}^*)}{\hat{e}_{n-1}^*}\Big)\Big].
\end{aligned} \tag{35}$$

The first term in Eq. (35) *forms a geometric series, and the second term in Eq.* (35) *is zero as* n *goes to infinity, because* $\hat{e}_1^*, \hat{e}_{n-1}^*, \hat{e}_n^*, e_{n+1}^*$ *are bounded by construction and* $\hat{e}_{n-1}^* > 0$ *by Assumption 2. Hence,*

$$|p(x,t) - \hat{p}(x,t)| \leq \hat{e}_1^*\Big(\frac{1}{1-\gamma_{\frac{2}{1}}}(t)\Big) := B_2(t). \tag{36}$$

A.6 PROOF OF THEOREM 2

Proof 5 *We omit the time variable* t *in this proof for readability. By Definition 1, the maximum approximation error* $\max_x |e_1(x,\cdot)| := e_1^*$*. Using the relations of* $\hat{e}_1 = e_1 - e_2, \hat{e}_1^* \leq e_1^* + e_2^*$*, the error bound in Theorem 1 can be upper-bounded by*

$$B_2 = \hat{e}_1^*\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*}\Big) \leq (e_1^* + e_2^*)\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*}\Big). \tag{37}$$

Hence, the gap between B_2 *and the maximum approximation error* e_1^* *is upper-bounded by*

$$B_2 - e_1^* \leq e_1^*\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*} - 1\Big) + e_2^*\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*}\Big). \tag{38}$$

Now suppose $\hat{e}_1$ *approximates* e_1 *sufficiently well such that* $e_2(x,t) = e_1(x,t) - \hat{e}_1(x,t) := \delta(x,t)$*, where* $\delta(x,t)$ *denotes a sufficiently small function for all* $(x,t) \in \Omega$*. Furthermore, suppose* $\hat{e}_2$ *approximates* e_2 *sufficiently well such that* $\hat{e}_2(x,t) \to e_2(x,t) = \delta(x,t)$ *for all* $(x,t) \in \Omega$*. Define* $\delta^* := \max_x |\delta(x,\cdot)|$*, then* $\hat{e}_2^* \to \delta^*$*, and* $\delta^* \to 0$ *as* $\delta(x,t) \to 0$ *for all* $(x,t) \in \Omega$*. Consequently, the RHS of Eq.* (38)*, at the limit, becomes*

$$\begin{aligned}
&\lim_{\hat{e}_2^* \to \delta^*, \delta^* \to 0} \Big[e_1^*\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*} - 1\Big) + e_2^*\Big(\frac{1}{1-\hat{e}_2^*/\hat{e}_1^*}\Big)\Big] \\
&= \lim_{\delta^* \to 0} \Big[e_1^*\Big(\frac{1}{1-\delta^*/\hat{e}_1^*} - 1\Big) + \delta^*\Big(\frac{1}{1-\delta^*/\hat{e}_1^*}\Big)\Big] = \delta^*
\end{aligned} \tag{39}$$

Lastly, for every $\epsilon \in (0,\infty)$*, take* δ^* *to be smaller than* ϵ*, then the proof is completed.*

A.7 PROOF OF COROLLARY 1

Proof 6 *For all $t \in T'$, let $0 < \alpha_1(t) < 1$. First, $e_1 = \hat{e}_1 + e_2$ by Def. 1, which implies*

$$|e_1(x,t)| \leq \max_x |\hat{e}_1(x,t)| + \max_x |e_2(x,t)| \tag{40}$$

for all $x \in X'$. Then by $0 < \alpha_1 < 1$ and its definition in Eq. (12)*, we have*

$$\alpha_1 \max_x |\hat{e}_1(x,t)| := \max_x |e_1(x,t) - \hat{e}_1(x,t)| = \max_x |e_2(x,t)|.$$

Hence, Eq. (40) *becomes*

$$\begin{aligned} |e_1(x,t)| &\leq \max_x |\hat{e}_1(x,t)| + \alpha_1(t) \max_x |\hat{e}_1(x,t)| \\ &= \max_x |\hat{e}_1(x,t)|(1 + \alpha_1(t)) \\ &< 2\hat{e}_1^*(t) := B_1(t) \end{aligned}$$

It is clear that $B_1(t)$ is not arbitrary tight because of the constant 2.

A.8 PROOF OF PROPOSITION 1

Proof 7 *Let $x \in \mathbb{R}^n$. By [Mishra and Molinaro, 2023, theorem 2.6], we know*

$$\varepsilon_G := \|e_1 - \hat{e}_1\|_{W^{1,q}} \leq C_{pde}\mathcal{L}^{(1)} + C_{pde}C_{quad}^{\frac{1}{q}} N^{\frac{-\beta}{q}}, \tag{41}$$

where $\mathcal{L}^{(1)}$ is the training loss of $\hat{e}_1$, $C_{pde} > 0$ is the stability estimate of the first error PDE associated with the $W^{1,q}$ norm ($q \geq 2$), and $C_{quad}, \beta > 0$ are the constants according to the quadrature sampling points. By Definition 1, $e_2 = e_1 - \hat{e}_1$, and since $e_1(x,t), \hat{e}_1(x,t)$ and their first derivatives are bounded over the considered domain of Problem 1, we know there exists a universal embedding constant C_{embed} [Mizuguchi et al., 2017] such that

$$|e_2(x,t)| \leq C_{embed}\|e_2(x,t)\|_{W^{1,q}}. \tag{42}$$

Hence, we have

$$|e_2(x,t)| \leq C_{embed}\Big(C_{pde}\mathcal{L}^{(1)} + C_{pde}C_{quad}^{\frac{1}{q}} N^{\frac{-\beta}{q}}\Big). \tag{43}$$

Using the definition of $\alpha_1(t) := \frac{\max_x |e_2(x,t)|}{\hat{e}_1^*(t)}$, we obtain

$$\begin{aligned} \alpha_1(t) &\leq \frac{\max_x |e_2(x,t)|}{\min_t \hat{e}_1^*(t)} \\ &\leq \frac{1}{\min_t \hat{e}_1^*(t)} \Big[C_{embed}\Big(C_{pde}\mathcal{L}^{(1)} + C_{pde}C_{quad}^{\frac{1}{q}} N^{\frac{-\beta}{q}}\Big)\Big]. \end{aligned} \tag{44}$$

A.9 DERIVATION OF EXTENSION TO HEAT PDE WITH DIRICHLET BOUNDARY CONDITION

Here, we take heat equation for example. The governing partial differential equation of solution $u : \Omega = (\mathbb{R}^n \times [0, t_f]) \to \mathbb{R}$ is

$$\frac{\partial u(x,t)}{\partial t} = \Delta[u(x,t)],$$

subject to initial and Dirichlet boundary conditions

$$\begin{aligned} &u(x,0) = u_{ic}(x), \\ &u(x,t) = u_{bc}(x,t),\ (x,t) \in \partial\Omega, \end{aligned}$$

where $\partial\Omega$ is the boundary, and $\Delta[\cdot] := \sum_i^n \frac{\partial^2}{\partial x_i^2}[\cdot]$. Define the heat differential operator $\mathcal{D}_h[\cdot] := \frac{\partial}{\partial t}[\cdot] - \Delta[\cdot]$. By adding the boundary constraints into the training loss in Eq. (6), which is common in standard PINNs [Sirignano and Spiliopoulos,

2018], we can train $\hat{u}$ that approximates the solution u. Define the approximation error $e_1 = u - \hat{u}$, then a trained $\hat{u}(x,t)$ yields

$$\begin{aligned}
&\mathcal{D}_h[\hat{u}] = r_1(x,t),\\
&\hat{u}(x,0) = u_{ic}(x) - e_{1,ic}(x,0),\\
&\hat{u}(x,t) = u_{bc}(x,t) - e_{1,bc}(x,t),\ (x,t) \in \partial\Omega.
\end{aligned}$$

Apply the heat differential operator on the first error, we obtain

$$\begin{aligned}
&\mathcal{D}_h[e_1] + r_1 = 0,\\
&e_1(x,0) = u_{ic}(x) - e_{1,ic}(x,0),\\
&e_1(x,t) = e_{1,bc}(x,t),\ (x,t) \in \partial\Omega.
\end{aligned} \tag{45}$$

Compared Eq. (45) to Eq. (23), the only difference is the boundary condition on $\partial\Omega$. Thus, if an additional loss term regarding boundary condition $\mathcal{L}_{bc}$ is added into Eq. (25) to construct $\hat{e}_1$ (as well as other $\hat{e}_i$), and u is smooth and bounded, then the derivation of theorem 1 can be followed.

B SYSTEM CONFIGURATIONS

Here, we report system configurations of all the conducted numerical experiments.

B.1 1D LINEAR

We consider an 1D linear system (Ornstein-Uhlenbech process) in Pavliotis [2014, Eq. 4.19]

$$dx = -\beta x dt + \sqrt{D} dw.$$

From the analytical solution in Pavliotis [2014, Eq. 4.22]:

$$p(x,t|x_0) = \sqrt{\frac{\beta}{2\pi D(1-e^{-2\beta t})}} \exp\Big(-\frac{\beta(x - x_0 e^{-\beta t})^2}{2D(1-e^{-2\beta t})}\Big),$$

we set the initial distribution as $p_0(x) = p(x, t=1|x_0 = 1)$, and the system parameters are $\beta = D = 0.2$. The computation domain is $t \in [1,3]$ and $x \in [-6,6]$.

B.2 1D NONLINEAR

We consider an 1D nonlinear system

$$dx = (ax^3 + bx^2 + cx + d)dt + edw.$$

The initial distribution is Gaussian

$$p_0(x) \sim \mathcal{N}(\mu, \sigma^2).$$

The computation domain is $t \in [0,5]$ and $x \in [-6,6]$, and the system parameters are $a = -0.1, b = 0.1, c = 0.5, d = 0.5, e = 0.8, \mu = -2$ and $\sigma = 0.5$. Since there is no analytical solution, the "true" PDF $p(x,t)$ is obtained by extensive Monte-Carlo simulation using Euler Scheme with small integration time step $\Delta t = 0.0005$ and 10^8 samples.

B.3 2D INVERTED PENDULUM

We consider 2D nonlinear system of inverted pendulum

$$\begin{aligned}
dx_1 &= x_2 dt + B_{11} dw,\\
dx_2 &= -\frac{g}{l}\sin(x_1)dt + B_{22} dw,
\end{aligned}$$

where x_1 is the angle, and x_2 is the angular rate. The initial distribution is multivariate Gaussian

$$p_0(x) \sim \mathcal{N}(\mu, \Sigma).$$

The computation domain is $t \in [0, 5]$ and $x_1, x_2 \in [-3\pi, 3\pi]$, and the system parameters are $g = 9.8, l = 9.8, B_{11} = B_{22} = 0.5, \mu = [0.5\pi, 0]^T$ and $diag([0.5, 0.5]$. Similarly, the "true" PDF $p(x, t)$ is obtained by Monte-Carlo simulation using Euler Scheme with integration time step $\Delta t = 0.01$ and 2×10^7 samples.

B.4 2D DUFFING OSCILLATOR

We consider chaotic dynamics of a 2D duffing oscillator in Anderson and Farazmand [2024]

$$\begin{aligned} dx_1 &= x_2 dt \\ dx_2 &= (1.0x_1 - 0.2x_2 - 1.0x_1^3)dt + \frac{1}{\sqrt{20}} dw. \end{aligned}$$

The initial distribution is multivariate Gaussian

$$p_0(x) \sim \mathcal{N}(\mu, \Sigma).$$

The computation domain is $t \in [0, 2.5]$ and $x_1, x_2 \in [-2, 2]$, and the system parameters are $\mu = [1, 1]^T, \Sigma = diag([0.05, 0.05])$. Again, extensive Monte-Carlo simulations using Euler Scheme are carried out to estimate the "true" PDF $p(x, t)$, with integration time step $\Delta t = 0.005$ and 2×10^7 samples.

B.5 HIGH-DIMENSIONAL TIME-VARYING OU

The generic dynamics of high-dimensional time-varying Ornstein-Uhlenbech is

$$dx = \Big(A_n + \Delta A_n(t)x\Big)dt + B_n dw,$$

where $x, w \in R^n$. The initial distribution is multivariate Gaussian

$$p(x, 0) \sim \mathcal{N}(\mu_n, \Sigma_n).$$

Since there is no closed-form solution as discussed in Sec. XXX, we use Euler forward numerical integration with $\Delta t = 0.0001$ time step to obtain the "true" PDF. The computation domain is $t \in [0, 1]$ and $x \in [-1, 1]^n$. Below, we summarize the system parameters of 3D, 7D, and 10D experiments. For $n = 3$, we set

$$\begin{aligned} A_3 &= diag([0.3, 0.3, 0.3]) \\ \Delta A_3(t) &= \sin(t) \begin{bmatrix} 0 & 0 & -0.1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ B_3 &= diag([0.05, 0.05, 0.05]) \\ \mu_3 &= [-0.2, 0.2, 0.0]^T \\ \Sigma_3 &= diag([0.1, 0.1, 0.1]). \end{aligned}$$

For $n = 7$, we set

$$A_7 = \begin{bmatrix} 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.3 & 0 \\ -0.01 & 0 & 0 & 0 & 0 & 0 & 0.3 \end{bmatrix}$$

$$\Delta A_7(t) = \cos(t) \begin{bmatrix} 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_7 = diag([0.05, 0.05, 0.05, 0.05, 0.05, 0.05])$$

$$\mu_7 = [0, 0, 0, 0, 0, 0, 0]^T$$

$$\Sigma_7 = diag([0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12]).$$

For $n = 10$, we set

$$A_{10} = \begin{bmatrix} 0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & 0.03 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.06 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.21 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.02 & 0 & 0.3 \end{bmatrix}$$

$$\Delta A_{10}(t) = \sin(t) \begin{bmatrix} 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{10} = diag([0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05])$$

$$\mu_{10} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

$$\Sigma_{10} = diag([0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12]).$$

B.6 1D HEAT

We consider a one-dimensional heat equation as in Appendix A.9:

$$u_t - u_{xx} = 0,$$

subject to initial and Dirichlet boundary conditions

$$u_0(x) = -\sin(\pi x),$$
$$u(\pm 1, t) = 0, \forall t.$$

The computation domain is $t \in [0, 1]$ and $x \in [-1, 1]$. For this particular setting, analytical solution exists

$$u(x, t) = -\sin(\pi x) \exp^{-\pi^2 t}.$$

C TRAINING CONFIGURATIONS AND ADDITIONAL RESULTS

Here we first discuss the general training setting across all experiments. Then we discuss each experiment in details and present additional results and visualizations. In terms of training scheme, we use Adam optimizer for all experiments. For the systems in Section 6, we use the regularization technique discussed in Section 5 to train the PDF PINN $\hat{p}$. We also employ adaptive sampling to make training of both $\hat{p}$ and the error PINN $\hat{e}_1$ more efficient (see Lu et al. [2021] for detailed explanation). As for the 1D Linear system in Section 6, we simply sample random space-time points at every training iteration. For the architecture of the neural network, we use simple fully-connected feed-forward neural networks for both the solution PINN $\hat{p}$ and the error PINN $\hat{e}_1$. Information of the number of hidden layers, neurons, and activation functions for each experiment is provided below.

C.1 1D LINEAR EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 2 and 3. For each training iteration, $N_0 = N_r = 500$ space-time points are uniformly sampled as in Eq. 25, with weights $w_0 = 1$ and $w_r = |T| = 2$. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 5. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 6, and the constructed (and synthesized) error bounds at some time instances are visualized in Fig. 7.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	32	Softplus
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	32	Softplus
Hidden Layer 2 $\rightarrow$ Output Layer	Fully Connected	1	Softplus

Table 2: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	32	Softplus
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	32	Softplus
Hidden Layer 2 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 3: Neural network architecture and hyper-parameters of $\hat{e}_1$

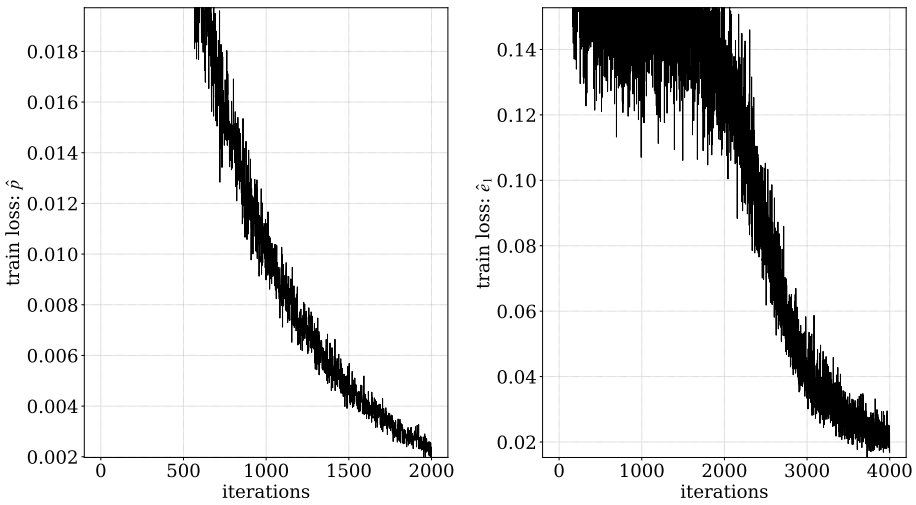
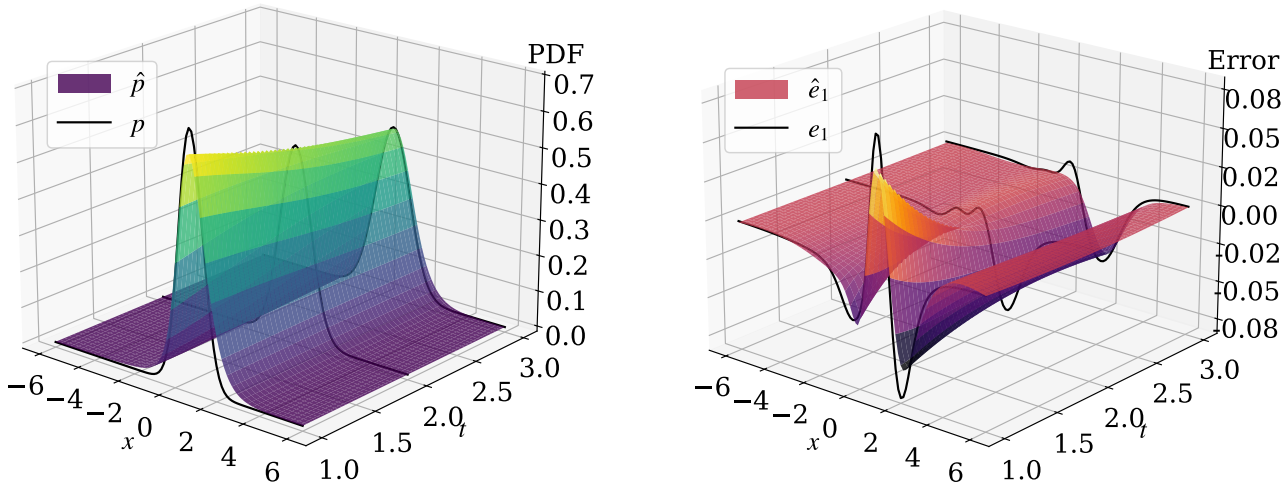


Figure 5: Training losses of $\hat{p}$ and $\hat{e}_1$



(a) $\hat{p}$ vs p

(b) $\hat{e}_1$ vs e_1

Figure 6: Trained PINNs $\hat{p}(x,t)$ and $\hat{e}_1(x,t)$ v.s. true PDF $p(x,t)$ and error $e_1(x,t)$ for all x and t.

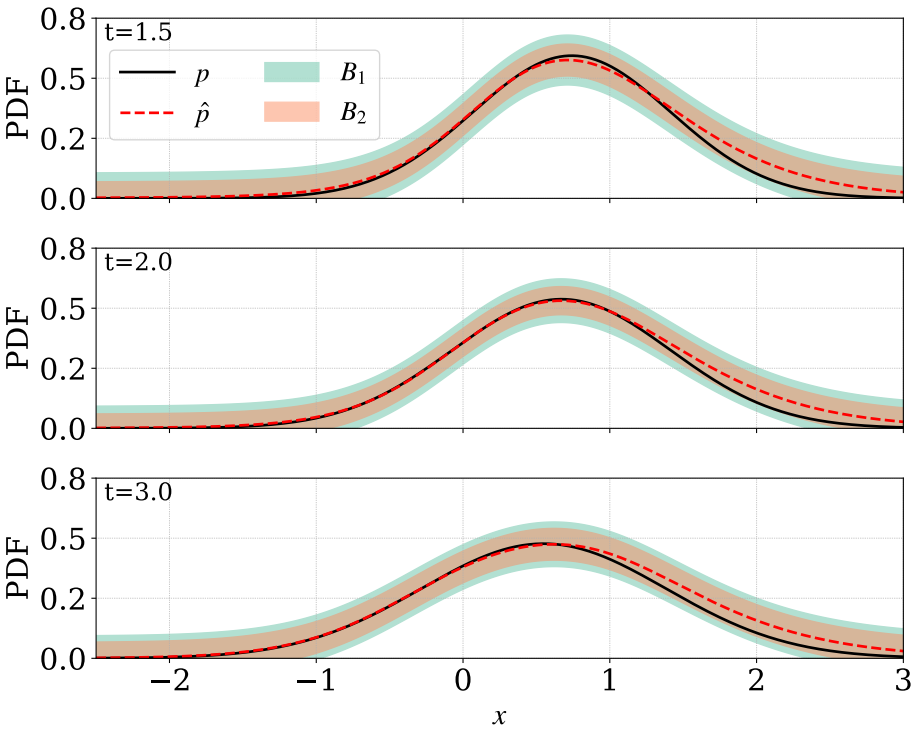


Figure 7: p, $\hat{p}$, B_1, and B_2 at some t.

C.2 1D NONLINEAR EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 4 and 5. The training starts from $N_0 = N_r = 1000$ uniformly distributed samples for both $\hat{p}$ and $\hat{e}_1$. We regularize the training of $\hat{p}$ by setting the weights $w_0 = 1$ and $w_r = w_\nabla = |T| = 5$. For $\hat{e}_1$, the weights are $w_0 = 1, w_r = |T|$, and $w_\nabla = 0$. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 8. Note that the periodic spikes are not due to unstable training. Instead they are due to the adaptive sampling scheme that periodically adds space-time points at which the residual values are large. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 9; the constructed error bounds at some time instances can be seen in Fig. 3a.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	50	Softplus
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	50	Softplus
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	50	Softplus
Hidden Layer 3 $\rightarrow$ Output Layer	Fully Connected	1	Softplus

Table 4: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	50	GeLU
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	50	GeLU
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	50	GeLU
Hidden Layer 3 $\rightarrow$ Hidden Layer 4	Fully Connected	50	GeLU
Hidden Layer 4 $\rightarrow$ Hidden Layer 5	Fully Connected	50	GeLU
Hidden Layer 5 $\rightarrow$ Hidden Layer 6	Fully Connected	50	GeLU
Hidden Layer 6 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 5: Neural network architecture and hyper-parameters of $\hat{e}_1$

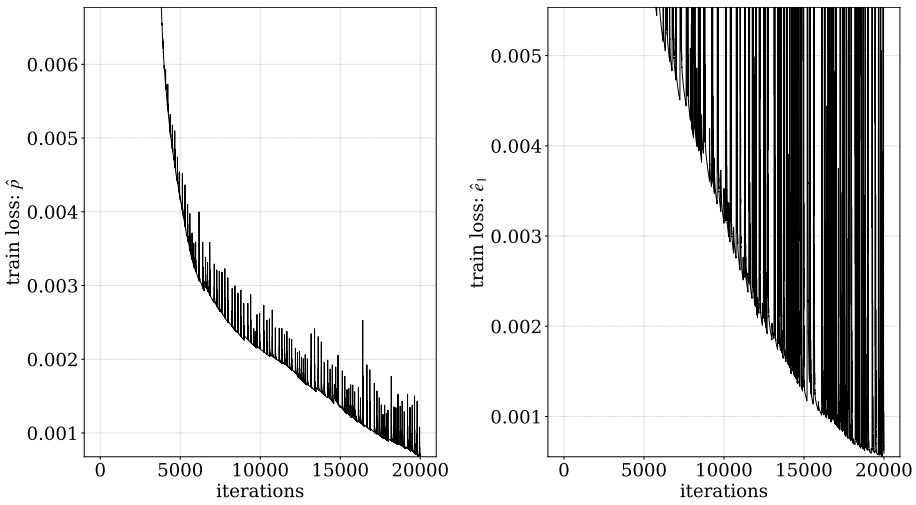
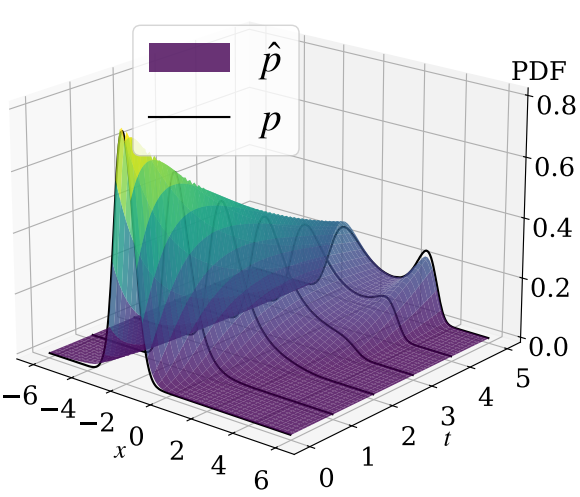
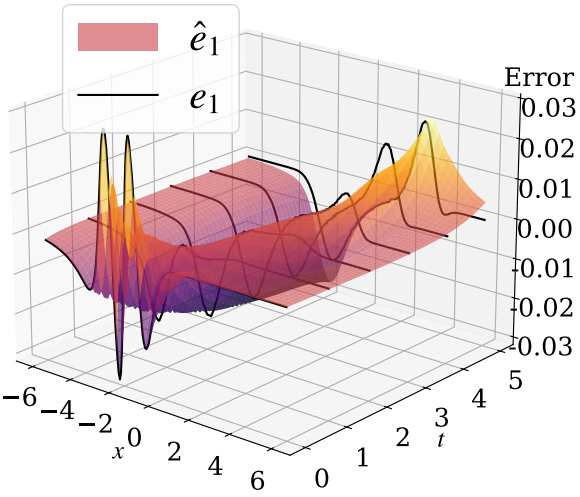


Figure 8: Training losses of $\hat{p}$ and $\hat{e}_1$



(a) $\hat{p}$ vs p



(b) $\hat{e}_1$ vs e_1

Figure 9: Trained PINNs $\hat{p}(x,t)$ and $\hat{e}_1(x,t)$ v.s. true PDF $p(x,t)$ and error $e_1(x,t)$ for all x and t.

C.3 2D INVERTED PENDULUM EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 6 and 7. As in Appendix C.2, the training starts from $N_0 = N_r = 1000$ uniformly distributed samples for both $\hat{p}$ and $\hat{e}_1$. We regularize the training of $\hat{p}$ by setting the weights $w_0 = 1$ and $w_r = w_\nabla = |T| = 5$. For $\hat{e}_1$, the weights are $w_0 = 1, w_r = |T|$, and $w_\nabla = 0$. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 10. Again, the periodic spikes in training loss are due to the adaptive sampling scheme that periodically adds space-time points, which becomes more effective as the system dimension grows. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 11. The constructed tight error bounds at some time instances are visualized in Fig. 12.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	50	Softplus
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	50	Softplus
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	50	Softplus
Hidden Layer 3 $\rightarrow$ Output Layer	Fully Connected	1	Softplus

Table 6: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	50	Softplus
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	50	Softplus
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	50	Softplus
Hidden Layer 3 $\rightarrow$ Hidden Layer 4	Fully Connected	50	Softplus
Hidden Layer 4 $\rightarrow$ Hidden Layer 5	Fully Connected	50	Softplus
Hidden Layer 5 $\rightarrow$ Hidden Layer 6	Fully Connected	50	Softplus
Hidden Layer 6 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 7: Neural network architecture and hyper-parameters of $\hat{e}_1$

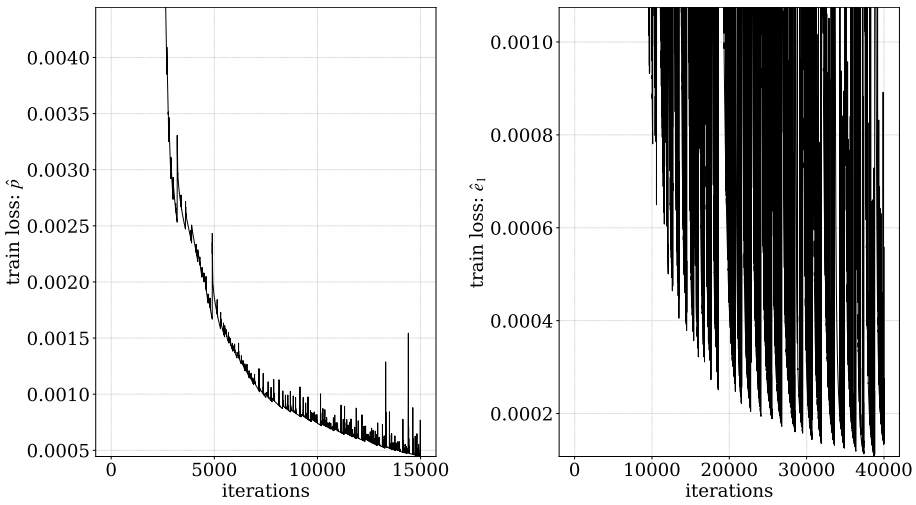
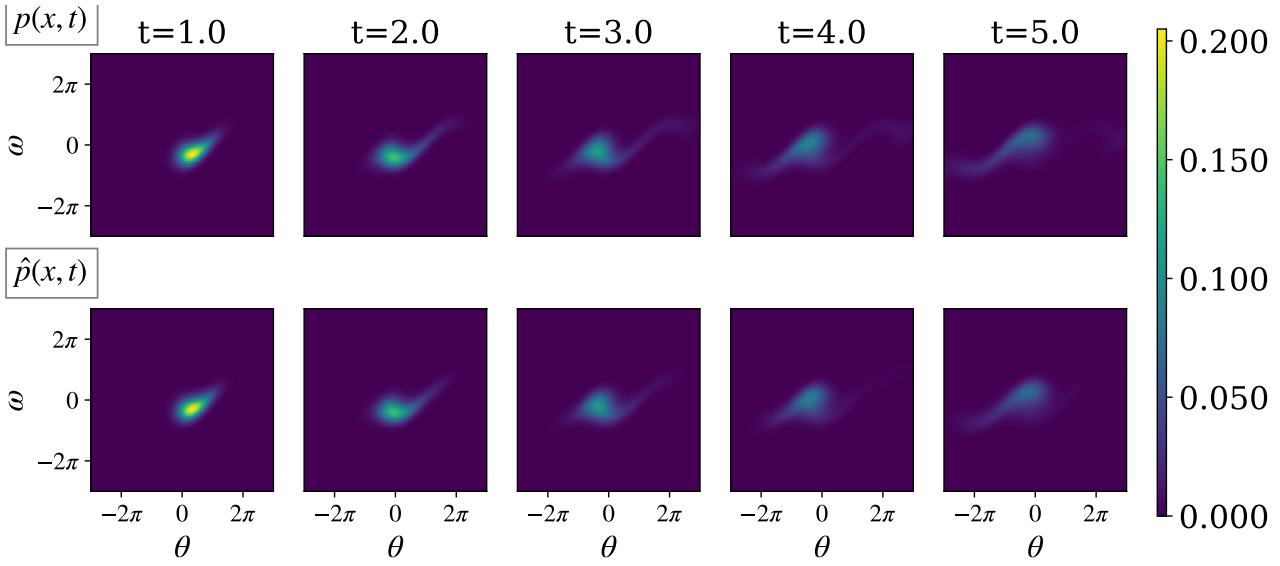
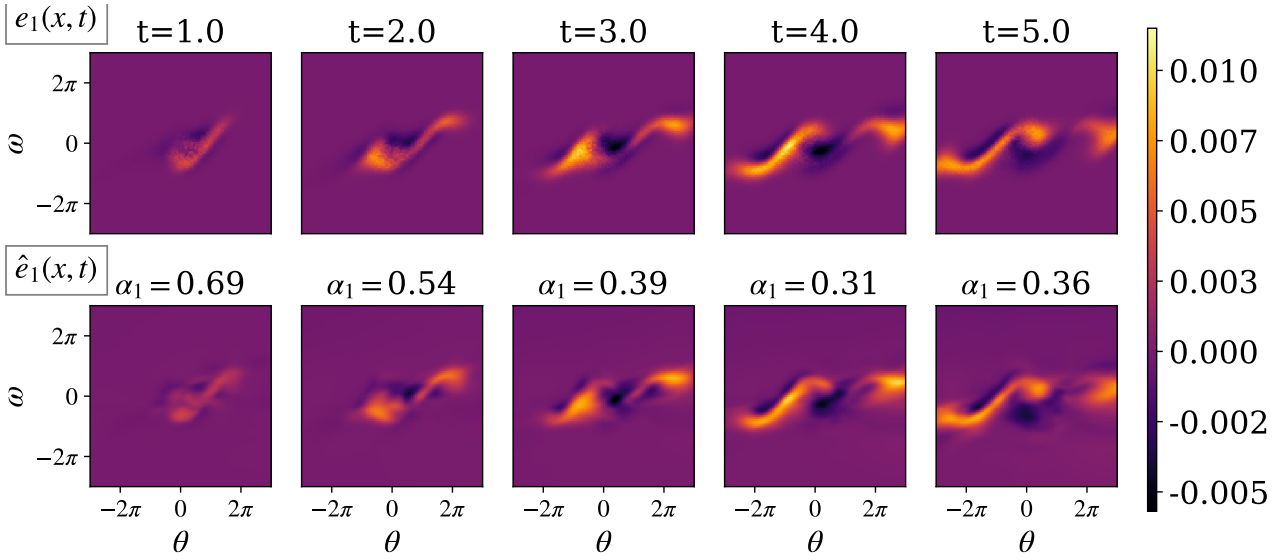


Figure 10: Training losses of $\hat{p}$ and $\hat{e}_1$



(a) $\hat{p}$ vs p



(b) $\hat{e}_1$ vs e_1

Figure 11: Trained PINNs $\hat{p}(\theta, \omega, t)$ and $\hat{e}_1(\theta, \omega, t)$ v.s. true PDF $p(\theta, \omega, t)$ and error $e_1(\theta, \omega, t)$ for all θ, ω at some t.

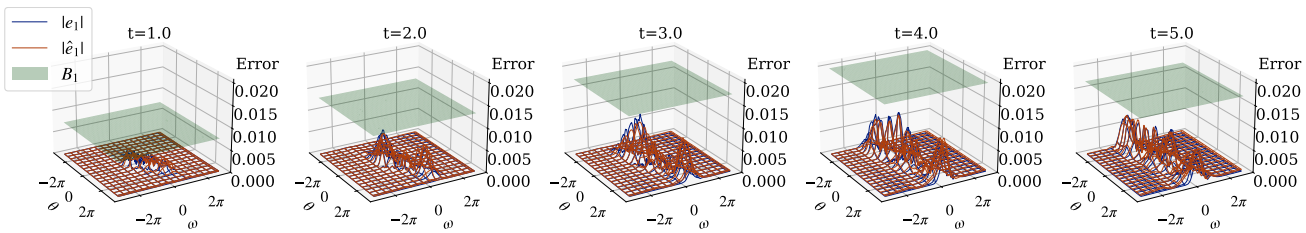


Figure 12: $|e_1|$, $|\hat{e}_1|$, and B_1at some t.

C.4 2D DUFFING OSCILLATOR EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 8 and 9. The training starts from $N_0 = N_r = 1000$ samples for both $\hat{p}$ and $\hat{e}_1$. Half of these samples are drawn uniformly, and the other half follow the normal distribution specified by the initial condition. We regularize the training of $\hat{p}$ by setting the weights $w_0 = 1$ and $w_r = w_\nabla = |T| = 5$. For $\hat{e}_1$, the weights are $w_0 = 1, w_r = |T|$, and $w_\nabla = 0$. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 13. Again, the periodic spikes in training loss are due to the adaptive sampling scheme that periodically adds space-time points, which becomes more effective as the system dimension grows. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 14. The constructed tight error bounds at some time instances are visualized in Fig. 15.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	60	GeLU
Hidden Layer 1 → Hidden Layer 2	Fully Connected	60	GeLU
Hidden Layer 2 → Hidden Layer 3	Fully Connected	60	GeLU
Hidden Layer 3 → Hidden Layer 4	Fully Connected	60	GeLU
Hidden Layer 4 → Hidden Layer 5	Fully Connected	60	GeLU
Hidden Layer 5 → Output Layer	Fully Connected	1	Softplus

Table 8: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	100	GeLu
Hidden Layer 1 → Hidden Layer 2	Fully Connected	100	GeLu
Hidden Layer 2 → Hidden Layer 3	Fully Connected	100	GeLu
Hidden Layer 3 → Hidden Layer 4	Fully Connected	100	GeLu
Hidden Layer 4 → Hidden Layer 5	Fully Connected	100	GeLu
Hidden Layer 5 → Output Layer	Fully Connected	1	N/A

Table 9: Neural network architecture and hyper-parameters of $\hat{e}_1$

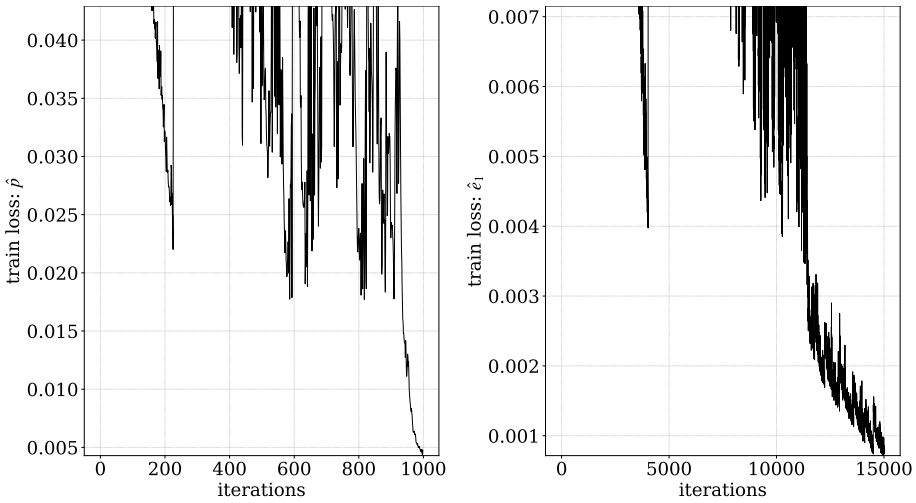
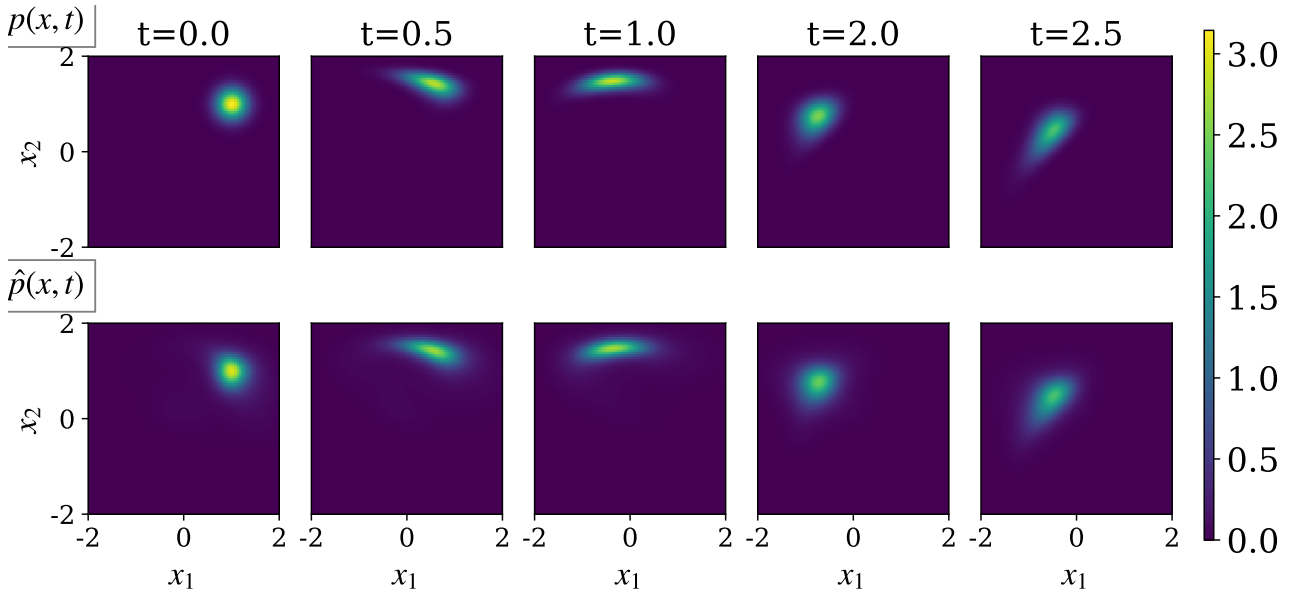
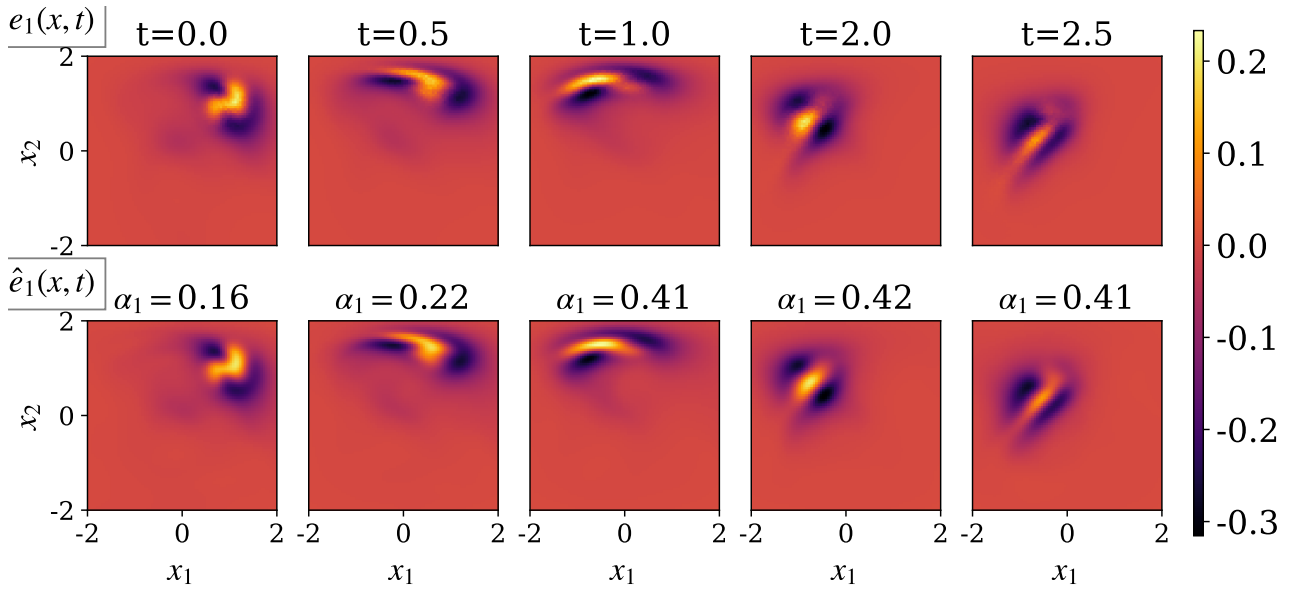


Figure 13: Training losses of $\hat{p}$ and $\hat{e}_1$



(a) $\hat{p}$ vs p



(b) $\hat{e}_1$ vs e_1

Figure 14: Trained PINNs $\hat{p}(x_1, x_2, t)$ and $\hat{e}_1(x_1, x_2, t)$ v.s. true PDF $p(x_1, x_2, t)$ and error $e_1(x_1, x_2, t)$ for all x_1, x_2 at some t.

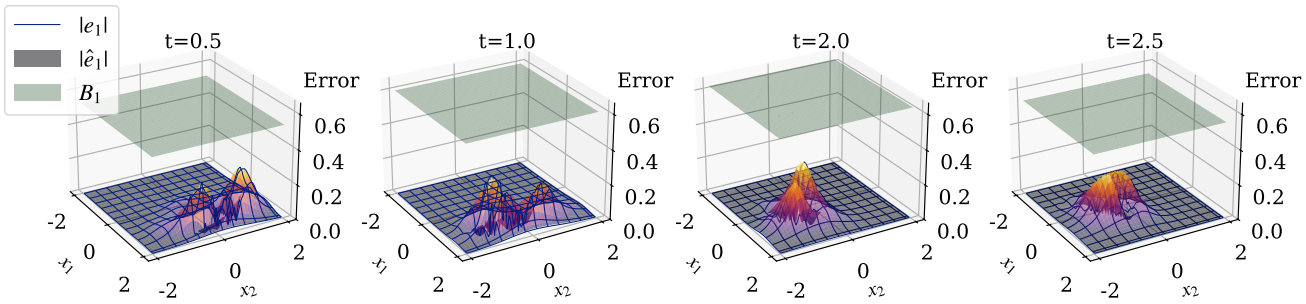


Figure 15: $|e_1|$, $|\hat{e}_1|$, and B_1at some t.

C.5 3D TIME-VARYING OU EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 10 and 11. For training $\hat{p}$, we begins with $N_0 = N_r = 2000$ samples. Half of these samples are drawn uniformly, and the other half follow the normal distribution specified by the initial condition. During training, we gradually add samples using adaptive sampling. For training $\hat{e}_1$, we begins with $N_0 = N_r = 300$ samples (with same distributions as training $\hat{p}$), and gradually add samples using adaptive sampling. The weights of training both $\hat{p}$ and $\hat{e}_1$ are $w_0 = 1, w_r = |T| = 1$, and $w_\nabla = 0$ without regularization. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 16. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 17 and 18.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	32	GeLU
Hidden Layer 1 → Hidden Layer 2	Fully Connected	32	GeLU
Hidden Layer 2 → Hidden Layer 3	Fully Connected	32	GeLU
Hidden Layer 3 → Hidden Layer 4	Fully Connected	32	GeLU
Hidden Layer 4 → Hidden Layer 5	Fully Connected	32	GeLU
Hidden Layer 5 → Output Layer	Fully Connected	1	Softplus

Table 10: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	32	GeLu
Hidden Layer 1 → Hidden Layer 2	Fully Connected	32	GeLu
Hidden Layer 2 → Hidden Layer 3	Fully Connected	32	GeLu
Hidden Layer 3 → Hidden Layer 4	Fully Connected	32	GeLu
Hidden Layer 4 → Hidden Layer 5	Fully Connected	32	GeLu
Hidden Layer 5 → Output Layer	Fully Connected	1	N/A

Table 11: Neural network architecture and hyper-parameters of $\hat{e}_1$

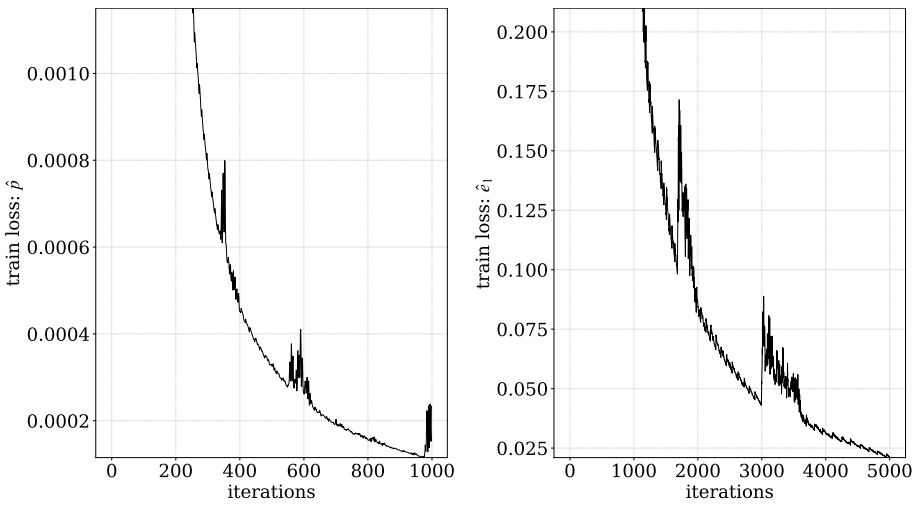
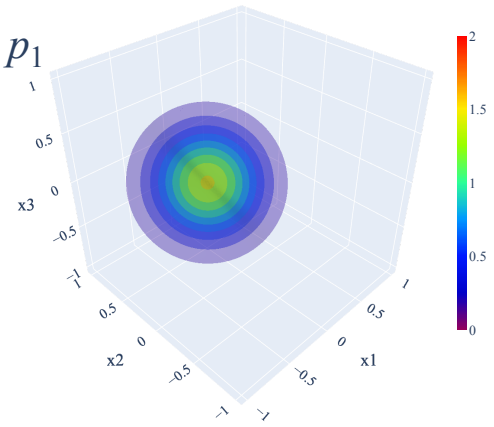
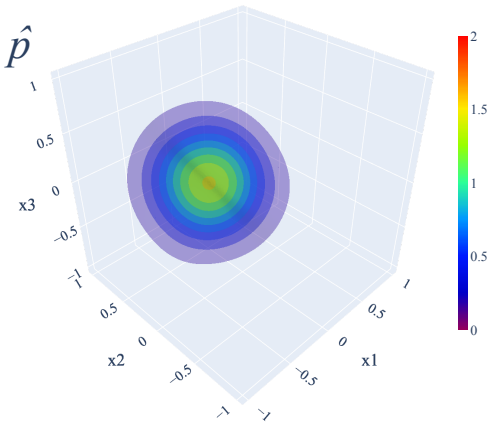


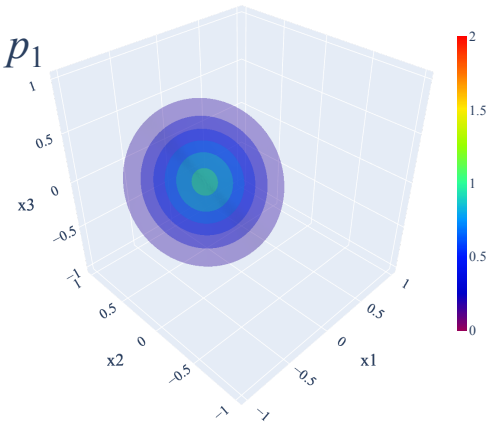
Figure 16: Training losses of $\hat{p}$ and $\hat{e}_1$



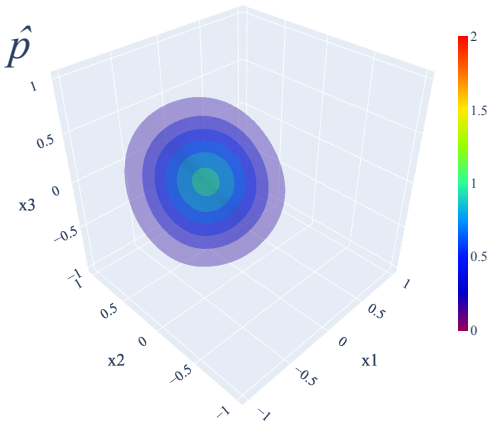
(a) $p(x, t = 0.2)$



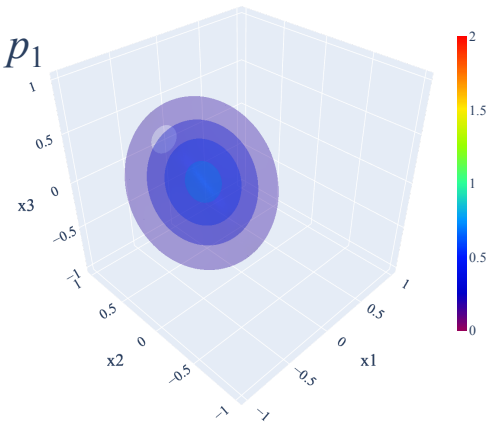
(b) $\hat{p}(x, t = 0.2)$



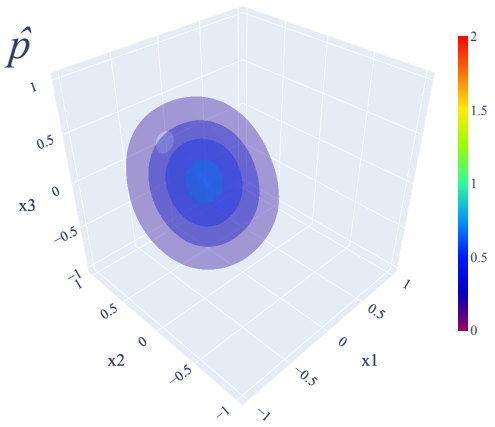
(c) $p(x, t = 0.6)$



(d) $\hat{p}(x, t = 0.6)$

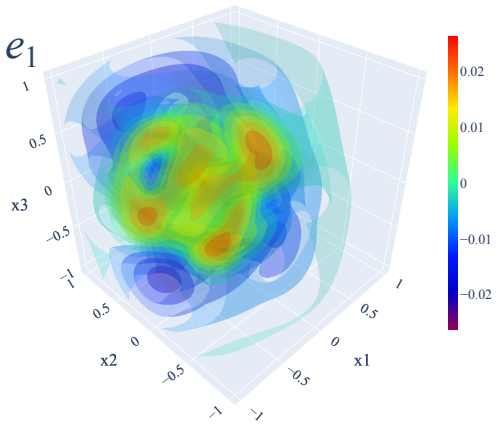


(e) $p(x, t = 1.0)$

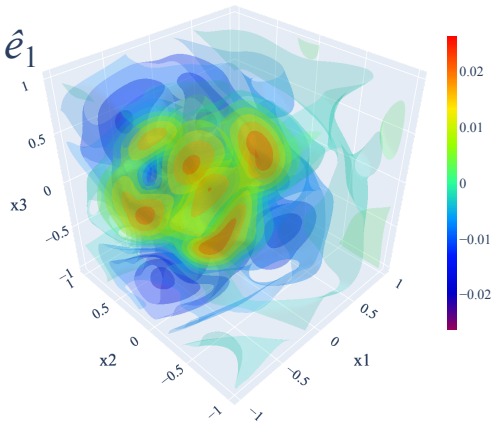


(f) $\hat{p}(x, t = 1.0)$

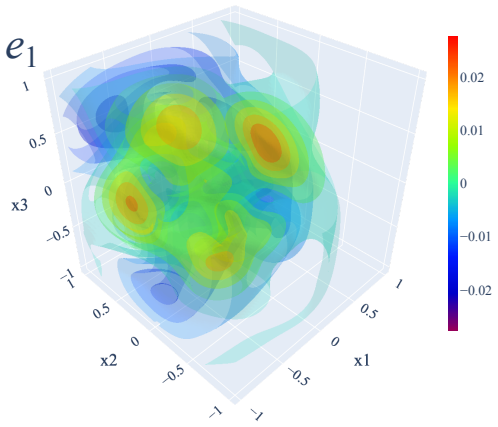
Figure 17: Trained PINNs $\hat{p}(x, t)$ v.s. true PDF $p(x, t)$ for all x at some t.



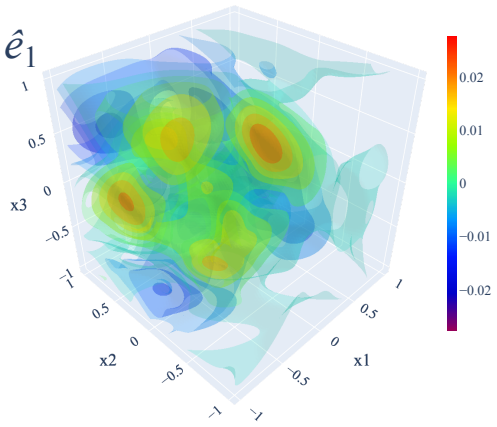
(a) $e_1(x, t = 0.2)$



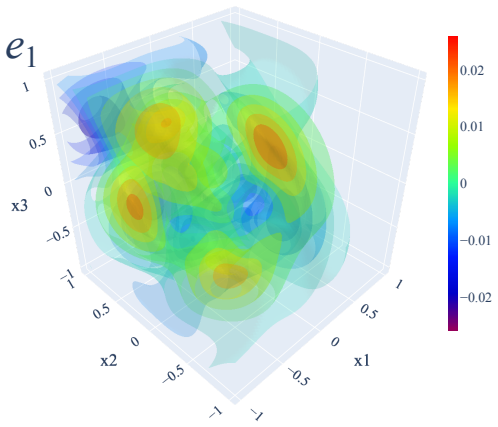
(b) $\hat{e}_1(x, t = 0.2)$



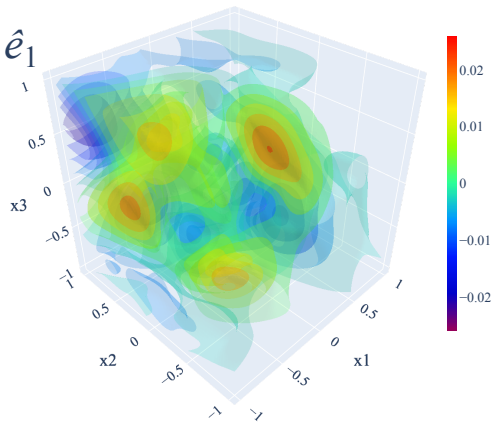
(c) $e_1(x, t = 0.6)$



(d) $\hat{e}_1(x, t = 0.6)$



(e) $e_1(x, t = 1.0)$



(f) $\hat{e}_1(x, t = 1.0)$

Figure 18: Trained PINNs $\hat{e}_1(x, t)$ v.s. true error $e_1(x, t)$ for all x at some t.

C.6 7D TIME-VARYING OU EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 12 and 13. For training $\hat{p}$, we begins with $N_0 = N_r = 2000$ samples. Half of these samples are drawn uniformly, and the other half follow the normal distribution specified by the initial condition. During training, we gradually add samples using adaptive sampling. For training $\hat{e}_1$, we begins with $N_0 = N_r = 300$ samples (with same distributions as training $\hat{p}$), and gradually add samples using adaptive sampling. The weights of training both $\hat{p}$ and $\hat{e}_1$ are $w_0 = 1, w_r = |T| = 1$, and $w_\nabla = 0$ without regularization. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 19.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	32	GeLU
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	32	GeLU
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	32	GeLU
Hidden Layer 3 $\rightarrow$ Hidden Layer 4	Fully Connected	32	GeLU
Hidden Layer 4 $\rightarrow$ Hidden Layer 5	Fully Connected	32	GeLU
Hidden Layer 5 $\rightarrow$ Output Layer	Fully Connected	1	Softplus

Table 12: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	32	GeLu
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	32	GeLu
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	32	GeLu
Hidden Layer 3 $\rightarrow$ Hidden Layer 4	Fully Connected	32	GeLu
Hidden Layer 4 $\rightarrow$ Hidden Layer 5	Fully Connected	32	GeLu
Hidden Layer 5 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 13: Neural network architecture and hyper-parameters of $\hat{e}_1$

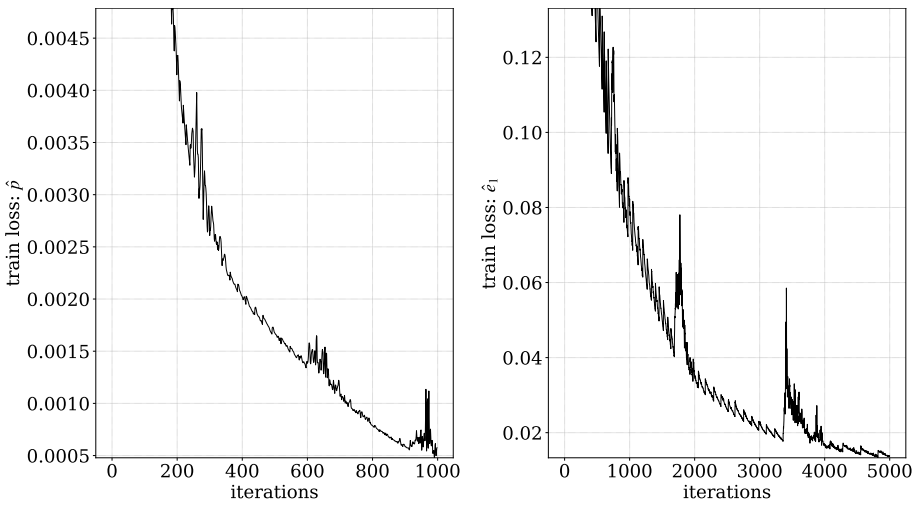


Figure 19: Training losses of $\hat{p}$ and $\hat{e}_1$

C.7 10D TIME-VARYING OU EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 14 and 15. For training $\hat{p}$, we begins with $N_0 = N_r = 600$ samples. Half of these samples are drawn uniformly, and the other half follow the normal distribution specified by the initial condition. During training, we gradually add samples using adaptive sampling. For training $\hat{e}_1$, we begins with $N_0 = N_r = 600$ samples (with same distributions as training $\hat{p}$), and gradually add samples using adaptive sampling. The weights of training both $\hat{p}$ and $\hat{e}_1$ are $w_0 = 1, w_r = |T| = 1$, and $w_\nabla = 0$ without regularization. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 20.

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	50	GeLU
Hidden Layer 1 → Hidden Layer 2	Fully Connected	50	GeLU
Hidden Layer 2 → Hidden Layer 3	Fully Connected	50	GeLU
Hidden Layer 3 → Hidden Layer 4	Fully Connected	50	GeLU
Hidden Layer 4 → Hidden Layer 5	Fully Connected	50	GeLU
Hidden Layer 5 → Hidden Layer 6	Fully Connected	50	GeLU
Hidden Layer 6 → Output Layer	Fully Connected	1	Softplus

Table 14: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer → Hidden Layer 1	Fully Connected	50	GeLu
Hidden Layer 1 → Hidden Layer 2	Fully Connected	50	GeLu
Hidden Layer 2 → Hidden Layer 3	Fully Connected	50	GeLu
Hidden Layer 3 → Hidden Layer 4	Fully Connected	50	GeLu
Hidden Layer 4 → Hidden Layer 5	Fully Connected	50	GeLu
Hidden Layer 5 → Hidden Layer 6	Fully Connected	50	GeLu
Hidden Layer 6 → Output Layer	Fully Connected	1	N/A

Table 15: Neural network architecture and hyper-parameters of $\hat{e}_1$

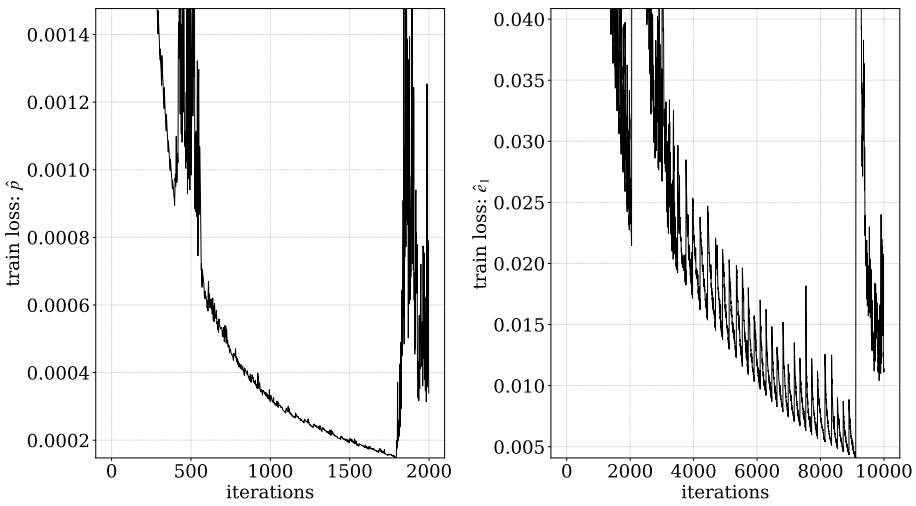


Figure 20: Training losses of $\hat{p}$ and $\hat{e}_1$

C.8 1D HEAT PDE EXPERIMENT

The neural networks of $\hat{p}$ and $\hat{e}_1$ are summarized in Table 16 and 17. For each training iteration, $N_0 = N_r = 500$ space-time points are uniformly sampled as in Eq. 25, with weights $w_0 = w_{bc} = 1$ and $w_r = |T| = 1$, where w_{bc} is the weight of the

Dirichlet boundary condition loss described in Appendix A.9. The training losses of $\hat{p}$ and $\hat{e}_1$ are illustrated in Fig. 21. The training results of $\hat{p}$ vs p and $\hat{e}_1$ vs e_t are shown in Fig. 22,

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	32	Tanh
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	32	Tanh
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	32	Tanh
Hidden Layer 3 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 16: Neural Network Architecture and Hyperparameters of $\hat{p}$

Layer Connection	**Type**	**# Neurons (Output)**	**Activation Function**
Input Layer $\rightarrow$ Hidden Layer 1	Fully Connected	50	Tanh
Hidden Layer 1 $\rightarrow$ Hidden Layer 2	Fully Connected	50	Tanh
Hidden Layer 2 $\rightarrow$ Hidden Layer 3	Fully Connected	50	Tanh
Hidden Layer 3 $\rightarrow$ Hidden Layer 4	Fully Connected	50	Tanh
Hidden Layer 4 $\rightarrow$ Hidden Layer 5	Fully Connected	50	Tanh
Hidden Layer 5 $\rightarrow$ Output Layer	Fully Connected	1	N/A

Table 17: Neural network architecture and hyper-parameters of $\hat{e}_1$

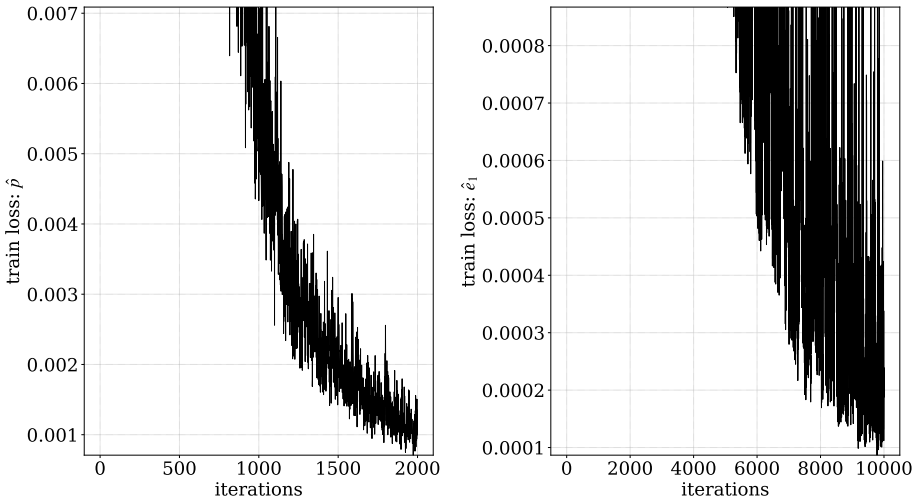
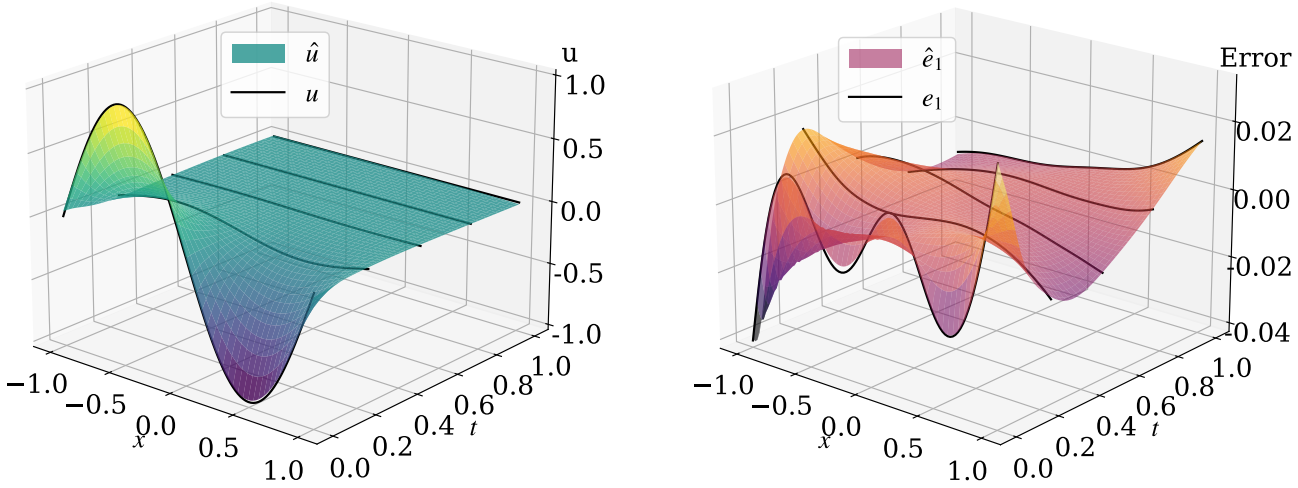


Figure 21: Training losses of $\hat{p}$ and $\hat{e}_1$



(a) $\hat{u}$ vs u

(b) $\hat{e}_1$ vs e_1

Figure 22: Trained PINNs $\hat{u}(x,t)$ and $\hat{e}_1(x,t)$ v.s. true solution $u(x,t)$ and error $e_1(x,t)$ for all x and t.

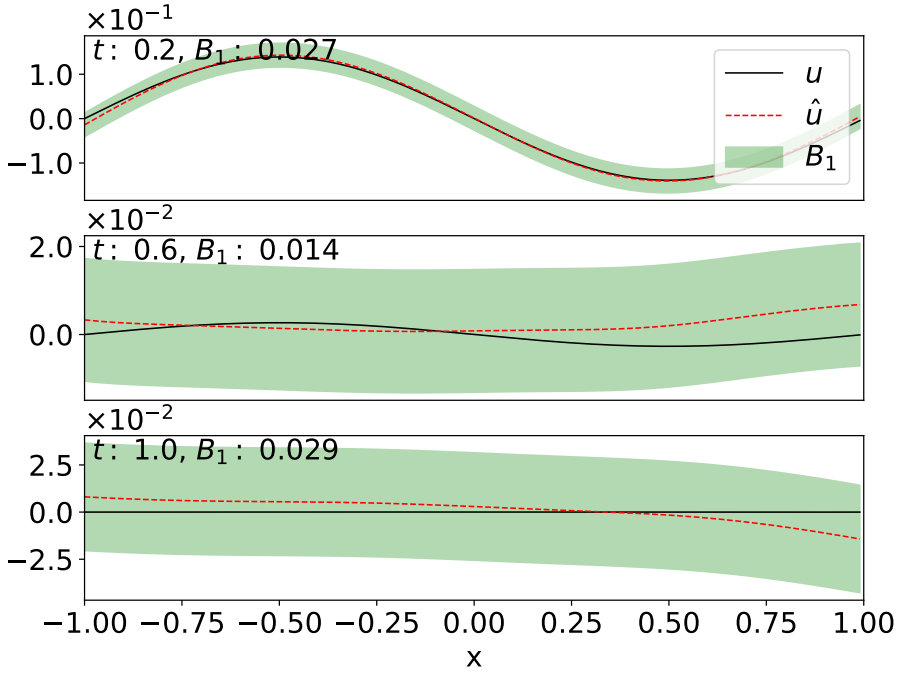


Figure 23: u, $\hat{u}$, and the error bound B_1 at some t.