Contents lists available at ScienceDirect

Journal of Computational Physics

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

Learning stochastic dynamical system via flow map operator [☆]

Yuan Chen, Dongbin Xiu [*]

Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA

ARTICLE INFO

Keywords:
Data driven modeling
Stochastic dynamical systems
Deep neural networks
Generative adversarial networks
Stochastic differential equations

ABSTRACT

We present a numerical framework for learning unknown stochastic dynamical systems using measurement data. Termed stochastic flow map learning (sFML), the new framework is an extension of flow map learning (FML) that was developed for learning deterministic dynamical systems. For learning stochastic systems, we define a stochastic flow map that is a superposition of two sub-flow maps: a deterministic sub-map and a stochastic sub-map. The stochastic training data are used to construct the deterministic sub-map first, followed by the stochastic sub-map. The deterministic sub-map takes the form of residual network (ResNet), similar to the work of FML for deterministic systems. For the stochastic sub-map, we employ a generative model, particularly generative adversarial networks (GANs) in this paper. The final constructed stochastic flow map then defines a stochastic evolution model that is a weak approximation, in term of distribution, of the unknown stochastic system. A comprehensive set of numerical examples are presented to demonstrate the flexibility and effectiveness of the proposed sFML method for various types of stochastic systems.

1. Introduction

In the recent years, data-driven modeling of unknown dynamical systems has become a prominent area of research among the scientific computing community. Methods for modeling various problems have been developed. See, for example, [5,29,33,24,21,35,36,32], among many others. Most of the work focused on deterministic problems, as modeling stochastic systems using noisy data poses additional challenges.

For modeling of stochastic systems, most of the existing efforts focus on learning Itô-type stochastic differential equations (SDE). These include Gaussian process approximation ([46,1,10,28]), Gaussian mixture distribution model ([20]), deep neural networks (DNNs) model ([12,11,18,44,48]), polynomial approximations ([38,23]), sparsity promoting learning ([4]), reconstruction of the corresponding Fokker-Planck equations ([14]), to name a few. Methods based on physics-informed neural networks (PINNs) were also developed. For example, the work of [43,7] seeks to recover the evolution of the probability density function of the SDEs, while the work of [6] leverages PINNs to recover the governing SDE with transition pathway data.

The focus, as well as the contribution, of this paper is on the development of a numerical framework for data driven modeling of general stochastic dynamical systems. More specifically, the method presented in this paper is applicable to the learning of a wide class of stochastic systems beyond Itô-type SDEs and Gaussian noises. To accomplish this, we propose a method to learn a stochastic flow map from the data. This is an extension of the deterministic flow map learning (FML), which was first developed to learn

☆ Funding: This work was partially supported by AFOSR FA9550-22-1-0011.

* Corresponding author.

E-mail addresses: chen.11050@osu.edu (Y. Chen), xiu.16@osu.edu (D. Xiu).

https://doi.org/10.1016/j.jcp.2024.112984
Received 5 May 2023; Received in revised form 25 March 2024; Accepted 31 March 2024

Available online 8 April 2024

unknown autonomous dynamical systems ([31]), in conjunction with residual network (ResNet), and was later extended to non-autonomous systems ([30]), systems with missing variables ([15]), as well as partial differential equations ([41,8]). For stochastic systems, the mapping between the solutions at different time instances, i.e., the flow map, is perturbed by an external stochastic inputs, which are unknown and often modeled via Wiener process. Consequently, the flow map can not be approximated via any deterministic function.

The proposed stochastic flow map learning (sFML) method seeks to construct a stochastic flow map consisting of two components: a deterministic sub-flow map and a stochastic sub-flow map. Using the noisy training data, we first construct the deterministic sub-map by fitting a deterministic function to the data. This is accomplished via the use of ResNet, in the same manner as FML for learning deterministic dynamical systems ([31]). This ResNet fitting obviously results in non-negligible mismatch to the data, as a deterministic function can not fit noisy data perfectly (unless one grossly over-parameterize the function). We then model the mismatch, which is the noisy component of the training data, using a stochastic sub-map in the form of a generative model. In this paper, we explore the use of generative adversarial networks (GANs), which are a class of generative DNN models designed for generating data with the same distribution as that of the training data, cf., [17]. While successful for problems such as image generation ([3,37,22]), text generation ([47,34]), etc., GANs also showed promises in scientific computing problems. See, for example, inverse problems ([42,25]), SDEs ([43,7,44]), uncertainty quantification ([45]), to name a few. In this paper, we employ Wasserstein GANs with gradient penalty (WGAN-GP) ([2,19]) to construct the stochastic sub-map. Upon learning both the deterministic sub-map and stochastic sub-map, we obtain the complete stochastic flow map. Using this stochastic flow map an evolution operator, we define an iterative scheme in time as a predictive model of the unknown stochastic dynamical system. The resulting sFML model is a weak approximation, in distribution, of the unknown stochastic system, and allows us to analyze the long-term system behavior under different initial conditions (not in the training data set). The effectiveness of the approach is then demonstrated via a comprehensive set of numerical examples.

This article is organized as follows: In Section 2, we introduce the setup of the problem. In Section 3, we give a brief review of the related materials, including the deterministic FML and GANs. The detailed description of the proposed sFML method is presented in Section 4, followed by the numerical examples in Section 5. We then conclude the paper by a brief summary in Section 6.

2. Problem statement

We consider a general stochastic dynamical system,

$$\frac{d\mathbf{x}_t}{dt}(\omega) = \mathbf{f}(\mathbf{x}_t, \omega), \qquad \mathbf{x}_{t_0}(\omega) \sim \mathbb{P}_0, \tag{2.1}$$

where $\omega \in \Omega$, an event space in a properly defined probability space, represents the random inputs, $\mathbb{P}_0$ is the distribution for the initial condition $\mathbf{x}_{t_0}$, and the solution $\mathbf{x}_t := \mathbf{x}(\omega, t) : \Omega \times [0, T] \mapsto \mathbb{R}^d$, $d \geq 1$, is a d-dimensional stochastic process for some finite time $T > 0$. The right-hand-side (RHS) $\mathbf{f} : \mathbb{R}^d \times \Omega \mapsto \mathbb{R}^d$ is unknown. Consequently, the solution $\mathbf{x}_t$ can not be obtained by solving (2.1).

2.1. Assumptions

We assume the dynamics of (2.1) is well-defined and time-homogeneous (cf. [27]). That is, for any $\Delta \geq 0$, the following property holds:

$$\mathbb{P}(\mathbf{x}_{s+\Delta} | \mathbf{x}_s) = \mathbb{P}(\mathbf{x}_\Delta | \mathbf{x}_0), \qquad s \geq 0. \tag{2.2}$$

Let $\mathbf{\Phi}_{t,t_0}(\mathbf{x}_0) = \mathbf{x}(t + t_0)$ be the time independent flow of (2.1). More precisely,

$$\mathbf{\Phi} : (\mathbb{R}^d \times \mathbb{R}) \times \mathbb{R} \to \mathbb{R}^d \times \mathbb{R}; \qquad \mathbf{\Phi}((\mathbf{x}_0, t_0), t) = (\mathbf{\Phi}_{t,t_0}(\mathbf{x}_0), t + t_0). \tag{2.3}$$

The time homogeneous assumption (2.2) implies that the Δ-shift flow map from $\mathbf{x}_s$ to $\mathbf{x}_{s+\Delta}$ produces a conditional distribution that depends only on the time lag Δ but not on the time variable s. Therefore, hereafter we use the following simplified notation for the flow map: For $\Delta \geq 0$, $s \geq 0$,

$$\mathbf{G}_\Delta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d; \qquad \mathbf{x}_{s+\Delta} = \mathbf{G}_\Delta(\mathbf{x}_s). \tag{2.4}$$

Naturally, we have $\mathbf{G}_0 = \mathbf{I}d$, the identity function, and

$$\mathbf{G}_\Delta(\mathbf{x}_s) \sim \mathbb{P}(\mathbf{x}_{s+\Delta} | \mathbf{x}_s) = \mathbb{P}(\mathbf{x}_\Delta | \mathbf{x}_0).$$

Note that $\mathbf{G}_\Delta$ is equivalent to $\mathbf{\Phi}$ in (2.3). We merely suppressed its explicit dependence on the time variable, because its distribution is the conditional distribution that depends only on the time lag Δ. Hereafter we shall refer to $\mathbf{G}_\Delta$ the stochastic flow map.

Remark 2.1. If the right-hand-side of (2.1) takes the following form

$$\mathbf{f}(\mathbf{x}_t, \omega) = \mathbf{a}\left(\mathbf{x}_t\right) + \mathbf{b}\left(\mathbf{x}_t\right) \dot{\mathbf{W}}_t(\omega), \tag{2.5}$$

where $\mathbf{a}$ is the drift function, $\mathbf{b}$ the diffusion function, and $\mathbf{W}_t$ a multivariate Wiener process, we obtain the classical form of SDE, which is often written as

$$d\mathbf{x}_t = \mathbf{a}\left(\mathbf{x}_t\right) dt + \mathbf{b}\left(\mathbf{x}_t\right) d\mathbf{W}_t. \tag{2.6}$$

It satisfies time-homogeneous assumption (2.2). In this case, our basic assumption of $\mathbf{f}$ being unknown mounts to the drift $\mathbf{a}$ and the diffusion $\mathbf{b}$ being unknown. Note that the discussion in the paper does not require the unknown SDE (2.1) to be in the classical form (2.6).

2.2. Setup and objective

Our goal is to construct a numerical model for the system (2.1), whose governing equations are unknown, by using measurement data of $\mathbf{x}_t$, such that the dynamics of (2.1) can be accurately simulated by the constructed numerical model.

Suppose observation data of the system states $\mathbf{x}_t$ are available over a sequence of discrete time instances $0 \le t_0 < t_1 < \ldots$. For simplicity, we assume the time instances are uniformly distributed with a constant time lag, $\Delta = t_n - t_{n-1}, \forall n \ge 1$. Suppose we have observation data of $N \ge 1$ number of solution trajectories. That is, for each $i = 1, \ldots, N$, we have the following solution sequence

$$\mathbf{x}\left(t_0^{(i)}\right), \mathbf{x}\left(t_1^{(i)}\right), \ldots, \mathbf{x}\left(t_{L_i}^{(i)}\right), \tag{2.7}$$

where $(L_i + 1)$ is the length of the sequence. This is the i-th solution trajectory, in the sense that it contains the solution of (2.1) from the initial condition $\mathbf{x}(t_0^{(i)})$. For notational convenience, hereafter we assume each of the trajectories is of the same length, i.e., $L_i \equiv L, \forall i$.

Under the time-homogeneous assumption (2.2), the differences between the time instances are important, as opposed to the actual time values. Therefore, we denote the available data as a set of $N \ge 1$ solution trajectories of length $(L+1)$,

$$\mathbf{X}^{(i)} = \left(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_L^{(i)}\right), \qquad i = 1, \ldots, N, \tag{2.8}$$

where $\mathbf{x}_k^{(i)} = \mathbf{x}(t_k^{(i)})$, $1 \le i \le N$, $0 \le k \le L$, and the time variables $t_k^{(i)}$ are neglected. We remark that this is done for more than just notational convenience. It implies that the actual time variables do not need to be recorded in the data set during data collection. It makes our proposed method "coordinate free" and have a wider applicability in practice. In another word, each data sequence $\mathbf{X}^{(i)}$, $i = 1, \ldots, N$, can be considered as the solutions of the system (2.1) over time instances $t_k = k\Delta$, $k = 0, \ldots, L$, with an initial condition $\mathbf{x}_0^{(i)}$.

Given a training data set consisting sufficiently large number $N > 1$ trajectory data (2.8), our goal is to construct a numerical stochastic flow map $\widetilde{\mathbf{G}}_\Delta$ as an approximation to the true (and unknown) stochastic flow map $\mathbf{G}_\Delta$ (2.4). More specifically, we seek

$$\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) \overset{d}{\approx} \mathbf{G}_\Delta(\mathbf{x}), \tag{2.9}$$

where $\overset{d}{\approx}$ stands for approximation in distribution. The numerical flow map $\widetilde{\mathbf{G}}_\Delta$ thus defines a numerical model such that given an initial condition $\widetilde{\mathbf{x}}_0$,

$$\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{G}}_\Delta(\widetilde{\mathbf{x}}_n), \qquad n \ge 0, \tag{2.10}$$

where we use subscript n to denote the solution at time $t_n = n\Delta$ hereafter. The numerical model (2.10) becomes a weak approximation to the true unknown system (2.1), in the sense that, given an arbitrary initial condition $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0 \sim \mathbb{P}_0$,

$$\widetilde{\mathbf{x}}_n \overset{d}{\approx} \mathbf{x}_n, \qquad n = 1, 2, \ldots. \tag{2.11}$$

Naturally, the condition (2.9) can be not accomplished by any deterministic function. We therefore propose to develop a stochastic flow map learning method to construct $\widetilde{\mathbf{G}}_\Delta$.

3. Preliminaries

The proposed stochastic flow map learning (sFML) method is an extension of the flow map learning (FML) method for deterministic systems. It involves the use of generative models. More specifically in this paper, we focus on the use of generated adversarial networks (GANs). Here, we briefly review the ideas of the deterministic FML method and GANs.

3.1. Deterministic flow map learning

Consider an autonomous system,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d, \tag{3.1}$$

where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is not known. Its flow map is a mapping $\mathbf{\Phi} : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ such that $\mathbf{\Phi}(\mathbf{x}(t), s) = \mathbf{x}(t+s)$. Over the uniform time stencils with the constant time lag Δ we consider in this paper, the flow map defines the evolution of the solution

$$\mathbf{x}_{n+1} = \mathbf{\Phi}_\Delta(\mathbf{x}_n), \tag{3.2}$$

where we move the parameter Δ into the subscript. When the governing equation (3.1) is unknown, this evolutionary equation is also unknown.

The FML method developed in [31] utilizes data to learn the evolutionary relation (3.2) by constructing an approximate flow map $\widetilde{\mathbf{\Phi}} \approx \mathbf{\Phi}_\Delta$ such that

$$\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{\Phi}}_\Delta(\widetilde{\mathbf{x}}_n) \tag{3.3}$$

is an accurate approximation to (3.2). This is accomplished by training the approximate evolution (3.3) to match the training data set (2.8), which are deterministic data in this case.

Consider the i-th trajectory data from (2.8), $i = 1, \dots, N$. Using $\mathbf{x}_0^{(i)}$ as the initial condition, i.e., $\widetilde{\mathbf{x}}_0^{(i)} = \mathbf{x}_0^{(i)}$, the approximate flow map evolution equation (3.3) generates a trajectory,

$$\widetilde{\mathbf{X}}^{(i)} = \left(\widetilde{\mathbf{x}}_0^{(i)}, \widetilde{\mathbf{x}}_1^{(i)}, \dots, \widetilde{\mathbf{x}}_L^{(i)}\right). \tag{3.4}$$

The approximate flow map $\widetilde{\mathbf{\Phi}}_\Delta$ is then determined by minimizing the mean squared loss

$$\min \sum_{i=1}^{N} \left\| \widetilde{\mathbf{X}}^{(i)} - \mathbf{X}^{(i)} \right\|_F^2, \tag{3.5}$$

where the Frobenius norm becomes vector 2-norm for scalar systems with $d = 1$.

The work of [31] proposed the use of residual network (ResNet)

$$\widetilde{\mathbf{\Phi}}_\Delta = \mathbf{I} + \mathbf{N}, \tag{3.6}$$

where $\mathbf{I}$ is the identity operator and $\mathbf{N} : \mathbb{R}^d \to \mathbb{R}^d$ stands for the mapping operator of a standard feedforward fully connected DNN. The minimization of the mean squared loss thus becomes a minimization problem for the hyper parameters defining the DNN. Once the training is completed, one obtains a predictive model for the underlying unknown dynamical system, such that, for a given initial condition $\mathbf{x}_0$,

$$\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{x}}_n + \mathbf{N}(\widetilde{\mathbf{x}}_n), \qquad n = 0, 1, \dots, \tag{3.7}$$

where $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$. This framework has shown to be highly effective and accurate for many systems ([31,30,15]).

3.2. Generative adversarial networks

GANs (cf. [16]) are widely used, among many other tasks, to generate data with a desired target distribution. Consider a target distribution $\mathbb{P}_r$, with observed data $\mathbf{x}_{\mathtt{data}} \sim \mathbb{P}_r$. Let $\mathbf{z}$ be random variables with a fixed known distribution $\mathbb{P}_\mathbf{z}$. For example, a normal distribution. In GANs, one defines a neural network, called generator, $G(\mathbf{z}, \theta_g)$ with its hyper parameter set θ_g to map $\mathbf{z}$ into a new distribution $\mathbb{P}_{\theta_g}$. The goal is to tune the parameter θ_g such that the output distribution matches the target distribution $\mathbb{P}_r$. At the same time, one defines another neural network, called discriminator, $D(\mathbf{x}, \theta_d)$ with its hyper parameter set θ_d to assign a score to the input $\mathbf{x}$ such that it is able to distinguish whether $\mathbf{x}$ is from the target distribution $\mathbb{P}_r$ or the "fake" distribution $\mathbb{P}_{\theta_g}$.

The two networks D and G are trained by solving a zero-sum two-player game. The generator G aims to "fool" the discriminator D and let it believe the samples of $G(\mathbf{z})$ are from the target distribution $\mathbb{P}_r$. At the same time, the discriminator D aims to distinguish the samples from $G(\mathbf{z})$ and the real $\mathbb{P}_r$. The optimization problem can be written in the following minimax form:

$$\min_G \max_D L_{\mathtt{GANs}}(D, G) := \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_r}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[\log(1 - D(G(\mathbf{z})))]. \tag{3.8}$$

In this paper, we adopt the popular WGANs ([2]), which trains the discriminator to approximate Wasserstein-1 distance between the real and fake samples. The definition of Wasserstein-p distance is:

$$W_p(\mu, \nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \mathbb{E}_{(x,y)\sim\gamma} d(x, y)^p \right)^{1/p}, \qquad p \geq 1, \tag{3.9}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings of μ and ν, and $d(\cdot, \cdot)$ is a metric. More specifically, we employ WGANs with gradient penalty (WGANs-GP), which introduces a penalty on the gradient of the discriminator to enhance numerical stability ([19]). The loss function for WGANs-GP is:

$$L_{\mathtt{WGANs-GP}}(D, G) = \mathop{\mathbb{E}}_{\tilde{\mathbf{x}}\sim\mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathbb{P}_r}[D(\mathbf{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(\left\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\right\|_2 - 1\right)^2\right], \tag{3.10}$$

where the distribution $\mathbb{P}_{\hat{\mathbf{x}}}$ is defined to sample uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and $\mathbb{P}_g$, and $\lambda \geq 0$ is a penalty constant.

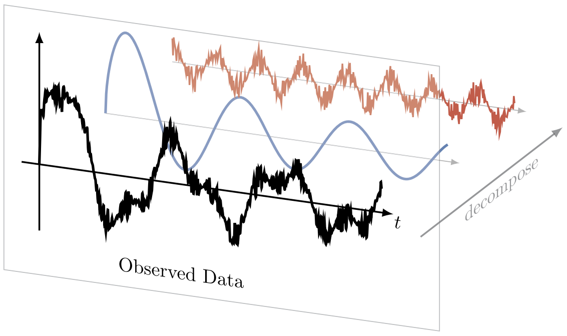

Fig. 1. An illustration of the decomposition of stochastic dynamics into the superposition of a smooth dynamics, governed by the conditional expectation, and a noisy dynamics.

4. Stochastic flow map learning

We now discuss the proposed stochastic flow map learning (sFML) method. The method consists of two steps: (1) construction of a deterministic sub-flow map to model the averaged dynamics of the unknown system; and (2) construction of a stochastic sub-flow map to model the noisy dynamics of the unknown system. The two sub-flow maps are learned from the same training data set (2.8).

4.1. Method description

The solution of the unknown system (2.1) is governed by the (unknown) stochastic flow map $\mathbf{G}_\Delta$ (2.4). Its evolution over two consecutive time steps can be decomposed into two parts:

$$\mathbf{x}_{n+1} = \mathbf{G}_\Delta(\mathbf{x}_n) = \mathbb{E}\left[\mathbf{x}_{n+1}|\mathbf{x}_n\right] + \mathbf{x}'_{n+1}, \tag{4.1}$$

where the conditional mean $\mathbb{E}\left[\mathbf{x}_{n+1}|\mathbf{x}_n\right]$ is a deterministic function of $\mathbf{x}_n$ and $\mathbf{x}'_{n+1}$ contains the stochastic component. Therefore, we write the stochastic flow map as a composition of two parts,

$$\mathbf{G}_\Delta = \mathbf{D}_\Delta + \mathbf{S}_\Delta, \tag{4.2}$$

where

$$\mathbf{D}_\Delta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d = \mathbb{E}(\mathbf{x}_{n+1}|\mathbf{x}_n), \qquad \mathbf{S}_\Delta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \tag{4.3}$$

are the deterministic part and stochastic part, respectively. Hereafter, we refer to $\mathbf{D}_\Delta$ as deterministic sub-map, and $\mathbf{S}_\Delta$ as stochastic sub-map. Obviously, neither $\mathbf{D}_\Delta$ nor $\mathbf{S}_\Delta$ is known. Fig. 1 illustrates the idea of the decomposition. Note that, by following (4.1), (4.2) is a nature expression of all stochastic processes.

Our proposed learning method seeks to construct numerical approximations to $\mathbf{D}_\Delta$ and $\mathbf{S}_\Delta$ by using the trajectory data (2.8). That is, we seek to construct a numerical stochastic flow map

$$\widetilde{\mathbf{G}}_\Delta = \widetilde{\mathbf{D}}_\Delta + \widetilde{\mathbf{S}}_\Delta, \tag{4.4}$$

where $\widetilde{\mathbf{D}}_\Delta$ and $\widetilde{\mathbf{S}}_\Delta$ are the approximation to $\mathbf{D}_\Delta$ and $\mathbf{S}_\Delta$, respectively. Specifically, we seek $\widetilde{\mathbf{G}}$ to be an approximation of $\mathbf{G}$ in distribution, in the sense that, for any $\mathbf{x} \in \mathbb{R}^d$,

$$\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) \overset{d}{\approx} \mathbf{G}_\Delta(\mathbf{x}). \tag{4.5}$$

4.2. Learning deterministic sub-map

Given the trajectory data (2.8), we first construct a numerical deterministic sub-map

$$\widetilde{\mathbf{D}}_\Delta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \tag{4.6}$$

as an accurate approximation of $\mathbf{D}_\Delta$. This is accomplished by finding a deterministic operator such that it generates deterministic trajectories that are the best fit to the stochastic training data (2.8).

Let us consider the i-th trajectory data from (2.8), $i = 1, \ldots, N$. Using $\mathbf{x}_0^{(i)}$ as the initial condition, the deterministic sub-map generates a deterministic trajectory,

$$\overline{\mathbf{X}}^{(i)} = \left(\bar{\mathbf{x}}_0^{(i)}, \bar{\mathbf{x}}_1^{(i)}, \ldots, \bar{\mathbf{x}}_L^{(i)}\right), \tag{4.7}$$

where $\bar{\mathbf{x}}_0^{(i)} = \mathbf{x}_0^{(i)}$, and

$$\bar{\mathbf{x}}_{n+1}^{(i)} = \widetilde{\mathbf{D}}_\Delta\left(\bar{\mathbf{x}}_n^{(i)}\right), \qquad n = 0, \dots, L-1. \tag{4.8}$$

The deterministic sub-map in our method is defined as the one that minimizes the total mis-fit between the deterministic trajectories $\bar{\mathbf{X}}^{(i)}$ and the stochastic training trajectories $\mathbf{X}^{(i)}$. More specifically, we use the common mean squared loss as the mis-fit measure and find $\widetilde{\mathbf{D}}_\Delta$ to accomplish

$$\min \sum_{i=1}^{N} \left\| \overline{\mathbf{X}}^{(i)} - \mathbf{X}^{(i)} \right\|_F^2, \tag{4.9}$$

where the Frobenius norm becomes vector 2-norm for scalar systems with $d = 1$.

To make the minimization problem tractable, we confine the search for $\widetilde{\mathbf{D}}_\Delta$ in a finite dimensional space. This is accomplished by employing a class of functions parameterized by a finite number of parameters. For low-dimensional dynamical systems, one may employ a polynomial space. See, for example, [40,26]. For more general systems whose dimensions d may not be small, DNNs have become a more popular choice. In this paper, we also utilize DNN as the deterministic sub-map. Similarly to the work of deterministic FML ([31]), we employ ResNet structure and define

$$\widetilde{\mathbf{D}}_\Delta(\cdot;\Theta) := \mathbf{I} + \mathbf{N}_\Delta(\cdot;\Theta), \tag{4.10}$$

where $\mathbf{I}$ is the identity matrix of size $(d \times d)$ and $\mathbf{N}_\Delta(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^d$ is the mapping operator of a connected feedforward DNN with its hyperparameter set Θ. Note that Θ is a finite set, whose cardinality is determined by the width and depth of the DNN. The minimization problem (4.9) thus becomes a minimization problem for the DNN parameter Θ, i.e.,

$$\min_\Theta \sum_{i=1}^{N} \sum_{n=1}^{L} \left\| \mathbf{x}_n^{(i)} - \widetilde{\mathbf{D}}_\Delta^{[n]}\left(\mathbf{x}_0^{(i)};\Theta\right) \right\|_2^2, \tag{4.11}$$

where $\widetilde{\mathbf{D}}_\Delta^{[n]}$ stands for n times composition of the operator $\widetilde{\mathbf{D}}_\Delta$. Upon solving the minimization problem, the hyperparameters Θ in the DNN are fixed. We thus neglect Θ for notational convenience, unless confusion arises otherwise.

Note that in the loss function (4.11), the case of $L > 1$ corresponds to the well-known multi-step loss, which is known to improve forecast accuracy for time series analysis, cf. [39,9,13]. It has also been used in FML for improving numerical stability of long-term predictions ([15,8]). Setting $L = 1$, it is straightforward to see that the loss function (4.11) ensures that, for sufficiently large number of samples $N \gg 1$, we have

$$\widetilde{\mathbf{D}}_\Delta(\mathbf{x}_0) \approx \mathbb{E}(\mathbf{x}_1|\mathbf{x}_0), \tag{4.12}$$

which implies, upon using the time homogeneous assumption (2.2),

$$\widetilde{\mathbf{D}}_\Delta(\mathbf{x}_n) \approx \mathbf{D}_\Delta(\mathbf{x}_n) = \mathbb{E}(\mathbf{x}_{n+1}|\mathbf{x}_n). \tag{4.13}$$

Remark 4.1. We remark that the decomposition of the stochastic flow map $\widetilde{\mathbf{G}}_\Delta$ into the deterministic sub-map and stochastic sub-map in (4.4), by following (4.2), is mainly for computational purpose. From a mathematical point of view, one can directly learn the stochastic flow map $\widetilde{\mathbf{G}}_\Delta$ from data via a generative model, in the same way of learning our stochastic sub-map in the next section. The purpose of learning the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ is to ensure that the stochastic learning part has mean value close to zero, which improves (sometimes drastically) the performance of the generative model. Consequently, the learning of the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ does not require high accuracy.

4.3. Learning stochastic sub-map

Once the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ is constructed, we proceed to construct the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$ in the numerical stochastic flow map (4.4), as an approximation to the exact stochastic sub-map $\mathbf{S}_\Delta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ in (4.3). Note that as the stochastic component of the flow map $\mathbf{G}_\Delta$, $\mathbf{S}_\Delta$ is subject to random input that is a function of the time lag Δ and unknown. In order to approximate this unknown random input, we define a generative model that incorporates an explicit random input. More precisely, we define

$$\widetilde{\mathbf{S}}_\Delta : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{n_s} \to \mathbb{R}^d, \qquad n_s \geq 1, \tag{4.14}$$

where n_s defines the stochastic dimension of the stochastic sub-map. Note that the true stochastic dimension of the sub-map $\mathbf{S}_\Delta$ is unknown. Upon choosing a finite number $n_s \geq 1$, we seek to construct the numerical stochastic flow map

$$\widetilde{\mathbf{G}}_\Delta = \widetilde{\mathbf{D}}_\Delta + \widetilde{\mathbf{S}}_\Delta : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{n_s} \to \mathbb{R}^d \tag{4.15}$$

as a weak approximation (in distribution) to the true (unknown) stochastic flow map $\mathbf{G}_\Delta$ (2.4). Note that at this point the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ is fixed, as it is already constructed by the procedure in the previous section. The only task is to construct the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$.

To construct the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$, we again utilize the training data set (2.8). For each of the i-th trajectory data in the set, $i = 1, \dots, N$, we use $\mathbf{x}_0^{(i)}$ as the initial condition and generate

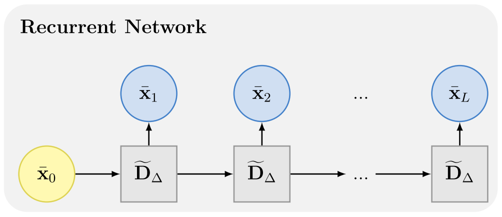

Fig. 2. An illustration of the recurrent network structure for the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$.

$$\widehat{\mathbf{X}}^{(i)} = \left(\hat{\mathbf{x}}_0^{(i)}, \hat{\mathbf{x}}_1^{(i)}, ..., \hat{\mathbf{x}}_L^{(i)}\right), \tag{4.16}$$

where $\hat{\mathbf{x}}_0^{(i)} = \mathbf{x}_0^{(i)}$, and

$$\hat{\mathbf{x}}_{n+1}^{(i)} = \widetilde{\mathbf{G}}_\Delta\left(\hat{\mathbf{x}}_n^{(i)}, \mathbf{z}_n^{(i)}\right), \qquad n = 0, \dots, L-1, \tag{4.17}$$

where $\mathbf{z}_n^{(i)} \in \mathbb{R}^{n_s}$ are i.i.d. random variables following a prescribed known distribution. For example, unit Gaussian.

We then minimize the distance between $\left\{\widehat{\mathbf{X}}^{(i)}\right\}_{i=1}^N$ and $\left\{\mathbf{X}^{(i)}\right\}_{i=1}^N$, where the distance function is chosen to measure probability distribution. Since $\widetilde{\mathbf{D}}_\Delta$ in $\widetilde{\mathbf{G}}_\Delta$ is fixed, the minimization procedure thus determines the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$, which in turn becomes a minimization problem for the hyperparameters after we express $\widetilde{\mathbf{S}}_\Delta$ in a parameterized family of functions.

In this paper, we adopt GANs for the construction of the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$. Specifically, we use DNN to represent $\widetilde{\mathbf{S}}_\Delta$, i.e.,

$$\widetilde{\mathbf{S}}_\Delta := \widetilde{\mathbf{S}}_\Delta(\mathbf{x}, \mathbf{z}; \Theta), \tag{4.18}$$

where $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{z} \in \mathbb{R}^{n_s}$, and Θ represents the hyperparameters in the DNN. Here $\mathbf{z}$ is a random vector within known distribution. In our examples, we use i.i.d. unit normal distribution. The dimension of $\mathbf{z}$, $n_s \geq 1$, is a choice made by user, in order to approximate the true probability space Ω whose dimension is unknown. Consequently, the stochastic flow map $\widetilde{\mathbf{G}}_\Delta = \widetilde{\mathbf{G}}_\Delta(\mathbf{x}, \mathbf{z}; \Theta)$ by using (4.15).

By using the GANs approach described in Section 3.2, the generator is the stochastic flow map $\widetilde{\mathbf{G}}_\Delta(\mathbf{x}, \mathbf{z}; \Theta)$, which generates the "fake" data sequence $\left\{\widehat{\mathbf{X}}^{(i)}\right\}_{i=1}^N$ in (4.16). The generator is expected to minimize the distance between the "fake" data $\widehat{\mathbf{X}} := \left\{\widehat{\mathbf{X}}^{(i)}\right\}_{i=1}^N$ (4.16) and the training data $\mathbf{X} := \left\{\mathbf{X}^{(i)}\right\}_{i=1}^N$ (2.8). To achieve this goal, a discriminator network $\boldsymbol{C}$, which is another DNN with its own hyperparameters, is trained to distinguish the two sets of data by approximating the Wasserstein-1 distance (3.9). For the network training, we adopt the WGANs with gradient penalty (WGANs-GP), whose loss function (3.10) takes the following form in our case:

$$L(\boldsymbol{C}, \widetilde{\mathbf{G}}_\Delta) = \mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim \widehat{\mathbf{X}}}[\boldsymbol{C}(\tilde{\mathbf{x}})] - \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathbf{X}}[\boldsymbol{C}(\mathbf{x})] + \lambda \mathop{\mathbb{E}}_{\hat{\mathbf{x}} \sim \mathbb{P}(\widehat{\mathbf{X}}, \mathbf{X})}\left[\left(\left\|\nabla_{\hat{\mathbf{x}}} \boldsymbol{C}(\hat{\mathbf{x}})\right\|_2 - 1\right)^2\right], \tag{4.19}$$

where $\mathbb{P}(\widehat{\mathbf{X}}, \mathbf{X})$ is uniform sampling distribution along the straight lines between pairs of points from $\widehat{\mathbf{X}}$ and $\mathbf{X}$, and $\lambda \geq 0$ is a penalty constant.

4.4. DNN structures and algorithm

In this section, we summarize the sFML algorithm and its corresponding DNN structures.

Given the training data set (2.8), the sFML algorithm consists of the following two steps:

(1) Constructing the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ by minimizing the loss function (4.11). The multi-step loss (when $L > 1$) requires recurrent use of the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$, as illustrated in Fig. 2.

(2) Constructing the stochastic sub-map $\widetilde{\mathbf{S}}_\Delta$, which defines the generator $\widetilde{\mathbf{G}}_\Delta = \widetilde{\mathbf{D}}_\Delta + \widetilde{\mathbf{S}}_\Delta$. Subsequently, the generator generates the "fake" data $\widehat{\mathbf{X}}$ in (4.16) via (4.17). In the spirit of ResNet, we write (4.17) in an equivalent form

$$\hat{\mathbf{x}}_{n+1}^{(i)} = \hat{\mathbf{x}}_n^{(i)} + \widetilde{\mathbf{G}}_\Delta\left(\hat{\mathbf{x}}_n^{(i)}, \mathbf{z}_n^{(i)}\right), \qquad n = 0, \dots, L-1, \quad i = 1, \dots, N, \tag{4.20}$$

where we use the same notation $\widetilde{\mathbf{G}}_\Delta$ here to refer to the core DNN part of the operation. Therefore, $\widetilde{\mathbf{G}}_\Delta$ corresponds to the difference $\hat{\mathbf{y}}_n = \hat{\mathbf{x}}_n - \hat{\mathbf{x}}_{n-1}$, $n = 1, \dots, L$.

The generator is illustrated on the left of Fig. 3. The right of Fig. 3 illustrates the discriminator $\boldsymbol{C}$, which is a standard DNN that computes a score $\mathbf{s}$ to the "fake" data. It also computes a score for the training data $\mathbf{X}$. These are then used in the minimization of the loss function (4.19).

The algorithm of the sFML method is summarized in Algorithm 4.1.

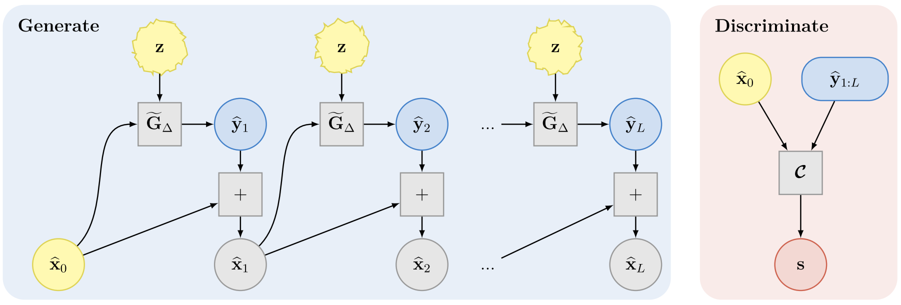

Fig. 3. Left: An illustration of the generator in the stochastic flow map. Right: Discriminator that computes a score for the data.

Algorithm 4.1 Learning SDE via GANs.

Input: Trajectory data of SDE: $\{(\mathbf{x}_0, \mathbf{y}_1, ..., \mathbf{y}_L)^{(i)}\}_{i=1}^N$; Deterministic model $\widetilde{\mathbf{D}}_\Delta$.

Parameters: Recurrent length L; Noise input dimension Z of generator; Layers and neurons each layer, Adam optimizer hyper-parameters β_1, β_2, base learning rate l_r of DNN $\widetilde{\mathbf{S}}_\Delta$, C with training parameter $\Theta_{\mathbf{S}}$ and Θ_C; Iterations of training critic per generator iteration $n_{\texttt{ct}}$; Batch size B and number of batches $n_B = N/B$ per epoch; Number of training epochs n_E; Gradient penalty constant λ.

Note: In the following algorithm, we apply operator $\widetilde{\mathbf{D}}$, $\widetilde{\mathbf{S}}$ and C by rows when imputing a matrix.

```
1: for $n = 1, 2, ..., n_E$ do
2:     for $i = 1, 2, ..., n_B$ do
3:         Slice $i$th group of data $\{(\mathbf{x}_0, \mathbf{y}_1, ..., \mathbf{y}_L)^{(i)}\}_{i=1+(i-1)B}^{iB}$;  $\hat{\mathbf{x}}_0 \leftarrow \mathbf{x}_0$;
4:         for $j = 0, 1, ..., L-1$ do
5:             Sample $B$ $Z$-dimensional random vectors $\mathbf{z} = \{\mathbf{z}_k\}_{k=1}^B$, $\mathbf{z}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
6:             $\hat{\mathbf{y}}_{j+1} \leftarrow \widetilde{\mathbf{D}}_\Delta(\hat{\mathbf{x}}_j) - \hat{\mathbf{x}}_j + \widetilde{\mathbf{S}}_\Delta(\hat{\mathbf{x}}_j, \mathbf{z})$;
7:             $\hat{\mathbf{x}}_{j+1} \leftarrow \hat{\mathbf{x}}_j + \hat{\mathbf{y}}_{j+1}$;                ▷ Generate fake SDE data
8:         end for
9:         for $k = 1, 2, ..., B$ do
10:            Sample $n_B$ random numbers $\epsilon = \{\epsilon_k\}_{k=1}^{n_B}$, $\epsilon_k \sim \mathcal{U}(0, 1)$;
11:            $\widetilde{\mathbf{y}}_{1:L}^{(k)} \leftarrow \epsilon_k \mathbf{y}_{1:L}^{(k)} + (1-\epsilon_k)\hat{\mathbf{y}}_{1:L}^{(k)}$;
12:            $P_C^{(k)} \leftarrow \left(\left\| \nabla_{(\mathbf{x}_0^{(k)}, \widetilde{\mathbf{y}}_{1:L}^{(k)})} C(\mathbf{x}_0^{(k)}, \widetilde{\mathbf{y}}_{1:L}^{(k)}) \right\|_2 - 1\right)^2$;
13:            $L_C^{(k)} \leftarrow C(\mathbf{x}_0^{(k)}, \hat{\mathbf{y}}_{1:L}^{(k)}) - C(\mathbf{x}_0^{(k)}, \mathbf{y}_{1:L}^{(k)}) + \lambda P_C^{(k)}$;
14:        end for
15:        $\Theta_C \leftarrow \texttt{Adam}(\frac{1}{B}\sum_{k=1}^B L_C^{(k)}, l_r, \beta_1, \beta_2)$;                ▷ Update Critic
16:        if $\texttt{mod}(n_B, n_{\texttt{ct}}) = 0$ then
17:            for $k = 1, 2, ..., B$ do
18:                $L_{\mathbf{S}}^{(k)} \leftarrow -C(\mathbf{x}_0^{(k)}, \hat{\mathbf{y}}_{1:L}^{(k)})$;
19:            end for
20:            $\Theta_{\mathbf{S}} \leftarrow \texttt{Adam}(\frac{1}{B}\sum_{k=1}^B L_{\mathbf{S}}^{(k)}, l_r, \beta_1, \beta_2)$;                ▷ Update Generator
21:        end if
22:    end for
23: end for
```

4.5. *Model prediction and analysis*

Upon satisfactory training of the DNNs, we obtain a predictive sFML model for the underlying unknown SDE. For a given initial condition $\mathbf{x}_0$,

$$\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{G}}_\Delta\left(\widetilde{\mathbf{x}}_n, \mathbf{z}_n\right) = \widetilde{\mathbf{D}}_\Delta(\widetilde{\mathbf{x}}_n) + \widetilde{\mathbf{S}}_\Delta(\widetilde{\mathbf{x}}_n, \mathbf{z}_n), \qquad n = 0, 1, 2 \dots, \tag{4.21}$$

with $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$. The model can be marched forward in time far beyond the length of the training data.

Note that the sFML model resembles the form of the classical SDE (2.6), in case that the underlying unknown SDE is in the form of (2.6). In fact, we can re-write the sFML model as

$$\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{x}}_n + \hat{\mathbf{a}}(\widetilde{\mathbf{x}}_n)\Delta + \hat{\mathbf{b}}(\widetilde{\mathbf{x}}_n)\delta\mathbf{W}_n, \tag{4.22}$$

where

$$\hat{\mathbf{a}}(\mathbf{x}) = \frac{\mathbb{E}_{\mathbf{z}}(\widetilde{\mathbf{G}}_\Delta(\mathbf{x}, \mathbf{z}) - \mathbf{x})}{\Delta}, \qquad \hat{\mathbf{b}}(\mathbf{x}) = \frac{\mathrm{Std}_{\mathbf{z}}(\widetilde{\mathbf{G}}_\Delta(\mathbf{x}, \mathbf{z}))}{\sqrt{\Delta}}, \tag{4.23}$$

and can be considered the effective drift and diffusion, respectively. Therefore, if the underlying unknown true SDE is of the classical form (2.6), we expect the effective drift and diffusion (4.23) of the sFML model (4.21) to be good approximations of the true drift and diffusion, respectively.

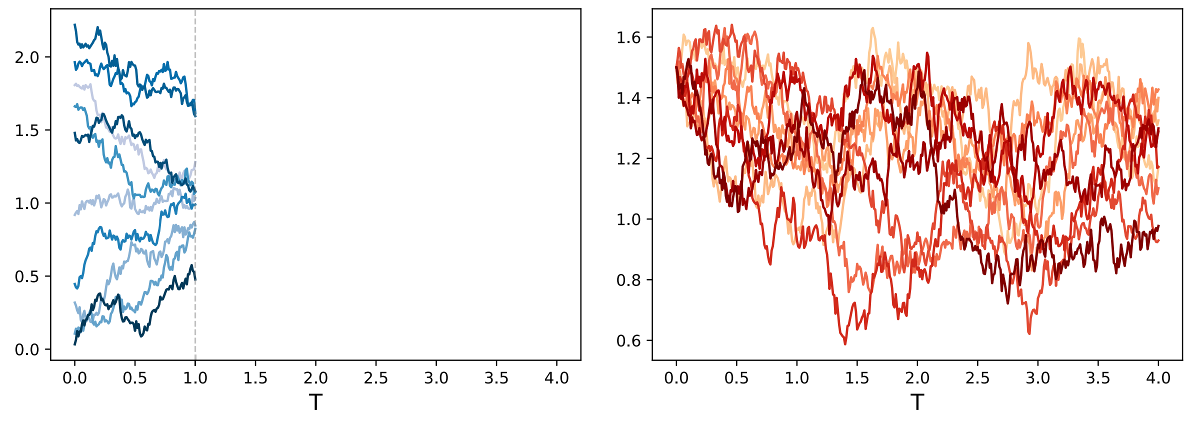

Fig. 4. Solutions of the OU process. Left: samples of the training data; Right: simulated samples by the sFML model for up to $T = 4.0$, with an initial condition $x_0 = 1.5$.

5. Numerical results

In this section, we present several numerical examples to demonstrate the performance of the proposed sFML method. The examples cover the following cases:

- Linear SDEs. These include an Ornstein-Uhlenbeck (OU) process and a geometric Brownian motion;
- Nonlinear SDEs. These include SDEs with exponential and trigonometric drift or diffusions, as well as the well-known stochastic double-well potential problem;
- SDEs with non-Gaussian noise, including exponential distribution and lognormal distribution.
- Two-dimensional SDEs, which include a 2-dimensional OU process and a stochastically driven harmonic oscillator.

For all of these problems, the exact SDEs are known. They are used to generate the training data sets, which are generated by solving the SDEs with Euler-Maruyama method, with initial conditions uniformly distributed in a region (specified later for each example). The SDEs are solved using a time step $\Delta = 0.01$ for 100 steps, from which we randomly choose a length $L = 40$ sequence to form our training data set (2.8). In other words, the training data are solution sequences of length $T = 0.4$. Our training data set (2.8) consists of $N = 10,000$ such trajectories.

For the DNN structure, we use a simple fully connect feedforward DNN for both the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$ (4.10) (see Fig. 2) and the discriminator $\boldsymbol{C}$ in Fig. 3 in the generator. In most examples, we use 3 layers, where each layer has 20 nodes. The exceptions are the 2-dimensional examples, where each layer has 40 nodes. In the training process of the GANs, we follow Algorithm 4.1 and set $n_{\mathtt{ct}} = 5$, $\beta_1 = 0.5$, $\beta_2 = 0.999$, $l_r = 5 \times 10^{-5}$. All examples are trained up to 100,000 epochs.

To examine the performance of the constructed sFML models, we provide the following ways:

- Simulated trajectories and their mean and standard deviation, for visual comparisons as well as the basic statistics;
- Conditional distribution the stochastic flow map $\widetilde{\mathbf{G}}_\Delta$, in comparison to that of the true flow map $\mathbf{G}_\Delta$;
- Evolution of the predicted solution distribution at certain times beyond the training data time domain.
- Comparison of the effective drift and diffusion (4.23) against the true drift and diffusion to those of true SDEs.

5.1. Linear SDEs

For scalar linear SDEs, we present the results for learning an OU process and a geometric Brownian motion.

5.1.1. Ornstein–Uhlenbeck process

We first consider the following Ornstein–Uhlenbeck (OU) process,

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t, \tag{5.1}$$

where $\theta = 1.0$, $\mu = 1.2$, and $\sigma = 0.3$. The training data are generated by solving the SDE with initial conditions uniformly sampled from $\mathcal{U}(0, 0.25)$. Some of these samples are shown on the left of Fig. 4. Upon training the DNNs and creating the sFML model for the OU process, we generate simulated trajectories from the learned model for up to $T = 4.0$. On the right of Fig. 4, we show the sFML model simulation results with an initial condition $x_0 = 1.5$ for time up to $T = 4.0$. We also compute the mean and standard deviation of the sFML model prediction, averaged over 100,000 simulation samples. The results are shown in Fig. 5, along with the reference mean and variance from the true OU process. We observe good agreement between the learned sFML model and the true model. Note that the agreement goes beyond the time horizon of the training data by a factor of 4.

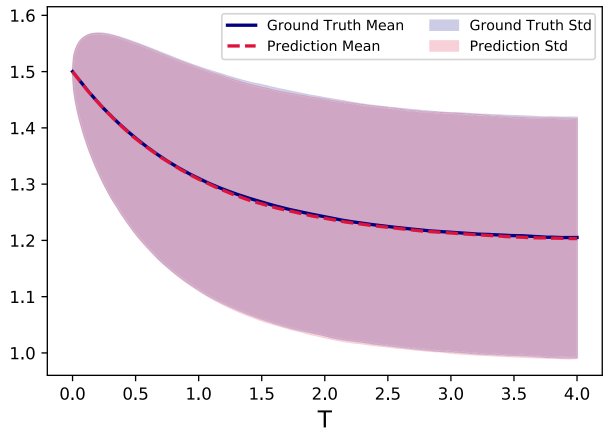

Fig. 5. Mean and standard deviation (STD) of the OU process by the sFML model.

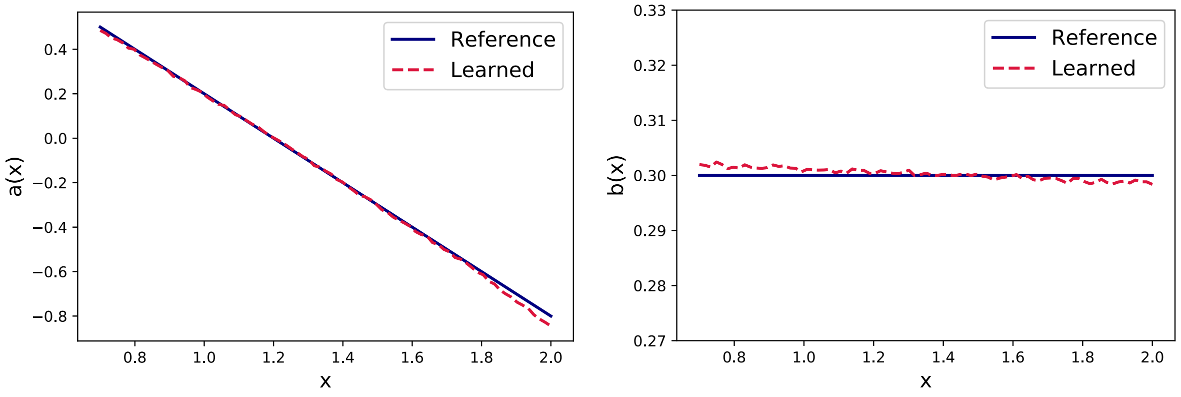

Fig. 6. Effective drift and diffusion of the OU process. Left: drift $a(x)$; Right: diffusion $b(x)$.

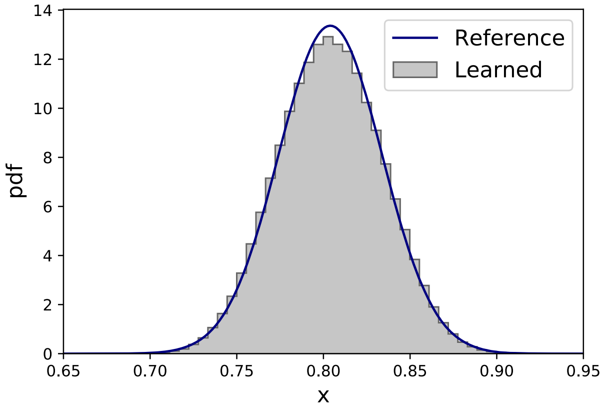

Fig. 7. OU process at steady state: comparison of the conditional distribution of the sFML model $\widetilde{\mathbf{G}}_\Delta(0.8)$ against that of the true model $\mathbf{G}_\Delta(0.8)$.

In Fig. 6, we show the recovered effective drift and diffusion functions for the OU process, by following the procedure (4.23) using 100,000 sFML model simulations. Comparing them to the true drift and diffusion, we observe good accuracy of the recovered functions, with relative errors on the order of 10^{-2}. Note that the discrepancy is relatively larger towards the end points of the comparison domain. This is expected because there are far less number of samples in these regions to gather highly accurate statistical results. Such a behavior is presented throughout all the examples in this paper. The OU process eventually reaches a steady state. In Fig. 7, we plot the histogram for the samples of $\widetilde{\mathbf{G}}_\Delta(0.8)$, as a comparison of the conditional distribution against that of the true stochastic flow map $\mathbf{G}_\Delta(0.8)$. Good agreement can be observed.

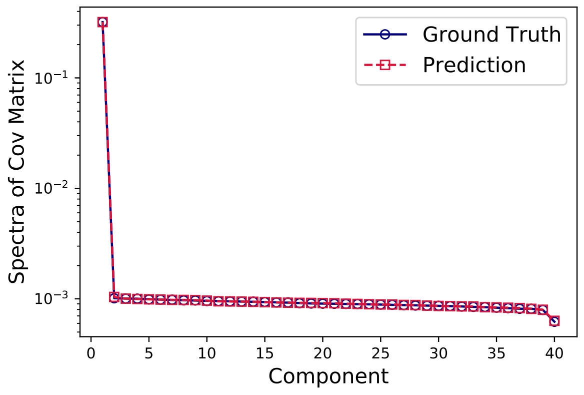

Fig. 8. OU process: comparison of covariance matrix spectra of the sFML model solution sequence ("Prediction") against that of the true solution ("Ground Truth").

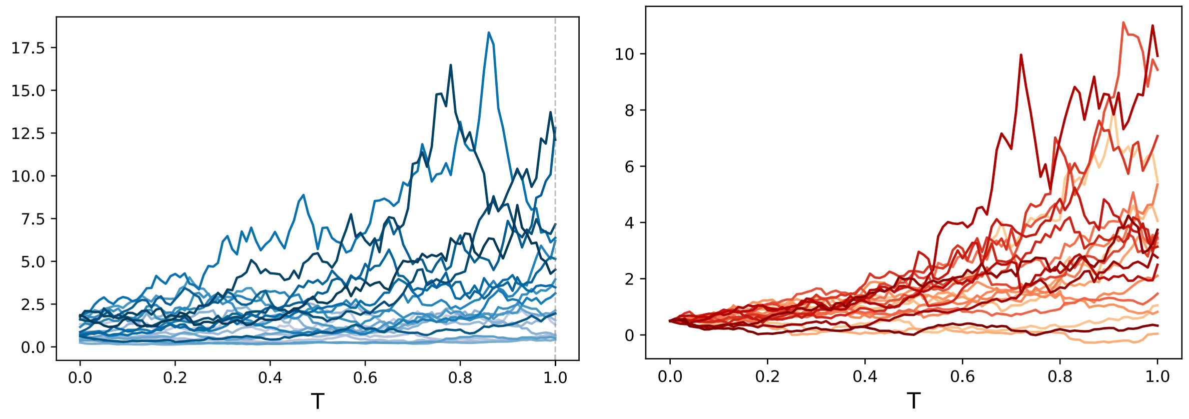

Fig. 9. Solutions of the geometric Brownian motion. Left: samples of true solution for the training data; Right: simulated samples by the sFML model with an initial condition $x_0 = 0.5$.

We conduct further examination of the solution time sequence produced by the sFML model. In Fig. 8, we present the spectra of the covariance function of the simulated sequences by the sFML model, along with that of the true solution sequence under the same initial condition. We observe good agreement between the sFML model and the true model.

5.1.2. *Geometric Brownian motion*

We now consider geometric Brownian motion, which, unlike the OU process, does not have steady state. In particular, we consider

$$dx_t = \mu x_t dt + \sigma x_t dW_t, \tag{5.2}$$

where μ and σ are constant parameters. We set $\mu = 2.0$, $\sigma = 1.0$ in our test.

On the left of Fig. 9, some samples of the true solutions for the training data are shown. These are generated from initial conditions uniformly distributed $\mathcal{U}(0,2)$. On the right of Fig. 9, we show the simulated solution samples by the trained sFML model with an initial condition $x_0 = 0.5$. The mean and STD of the solution comparison are shown in Fig. 10. Good agreement can be observed. Since this geometric Brownian motion grows exponentially fast over time, the simulation is stopped at $T = 1.0$.

The effective drift and diffusion functions recovered by the sFML model are shown in Fig. 11. Good agreement with the true drift and diffusion can be observed. The conditional distribution produced by the stochastic flow map $\widetilde{\mathbf{G}}_\Delta(x)$ is plotted at $x = 6$ (an arbitrary choice for illustration purpose). We note that it agrees quite well with the true distribution at $\mathbf{G}_\Delta(6)$, see also Fig. 12.

5.2. *Nonlinear SDE*

Similar to the work of [10], we present two Itô type SDEs with non-linear drift and diffusion functions.

5.2.1. *SDE with nonlinear diffusion*

We first consider a SDE with a nonlinear diffusion,

$$dx_t = -\mu x_t dt + \sigma e^{-x_t^2} dW_t, \tag{5.3}$$

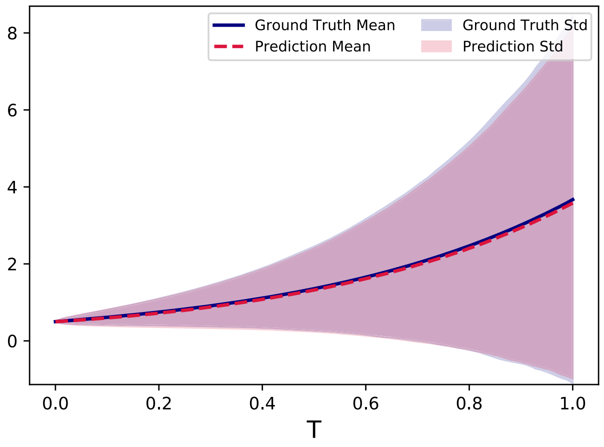

Fig. 10. Mean and standard deviation of the geometric Brownian motion by the sFML model.

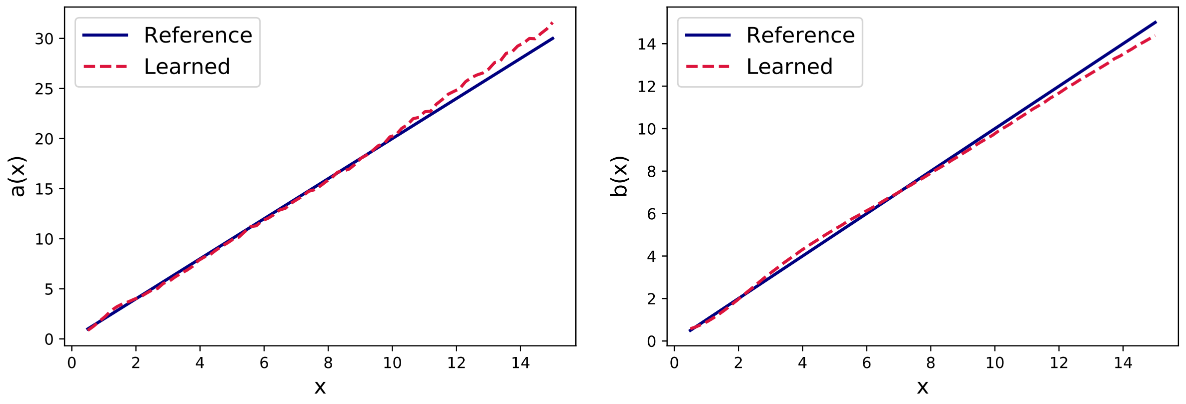

Fig. 11. Geometric Brownian motion. Left: recovery of the drift $a(x) = \mu x$; Right: recovery of the diffusion $b(x) = \sigma x$.

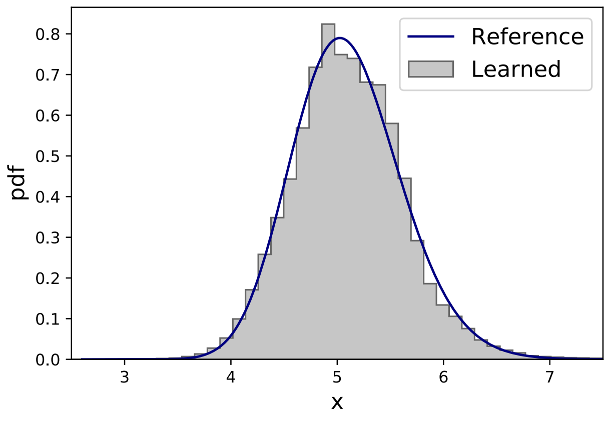

Fig. 12. Geometric Brownian motion: comparison of the conditional distribution $\mathbf{G}_\Delta(6)$ and $\widetilde{\mathbf{G}}_\Delta(6)$.

where μ and σ are constants. In this example, we set $\mu = 5$ and $\sigma = 0.5$. The SDE is solved with initial conditions from a uniform distribution $\mathcal{U}(-1, 1)$, for up to time $T = 1.0$. Some of the solution samples are shown on the left of Fig. 13. The simulated solutions of the learned sFML model are shown on the right of Fig. 13, with an initial condition $x_0 = -0.4$ and for time up to $T = 10$, for visual comparison. The evolutions of mean and STD of the sFML model are shown in Fig. 14, where good agreement with those of the true solution can be observed.

The recovered effective drift and diffusion functions are plotted in Fig. 15, whereas the conditional distribution of $\widetilde{\mathbf{G}}_\Delta(-0.3)$ is plotted in Fig. 16. Again, good agreement can be observed when compared to the reference solutions.

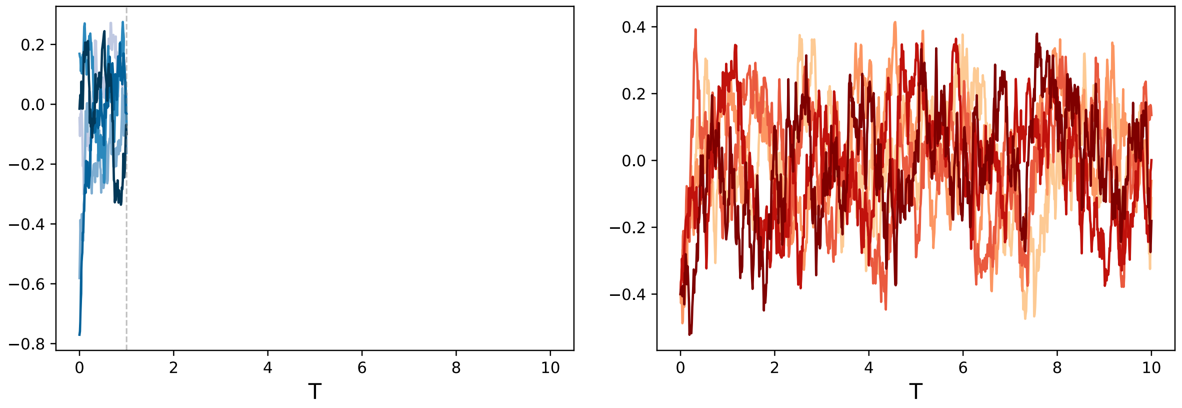

Fig. 13. SDE with nonlinear diffusion (5.3). Left: samples of the training data; Right: samples of the learned sFML model simulations for up to $T = 10.0$ with an initial condition $x_0 = -0.4$.

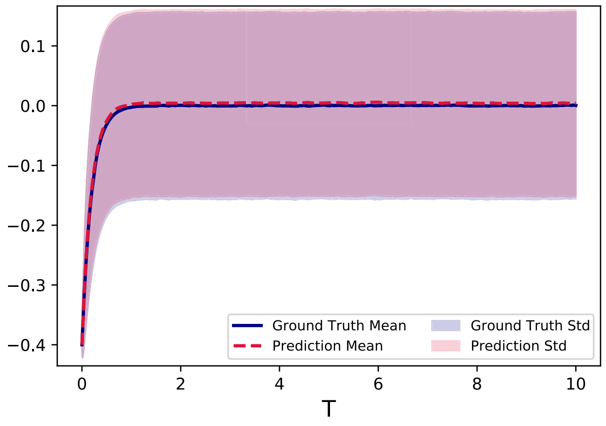

Fig. 14. Mean and standard deviation of the learned sFML model for (5.3).

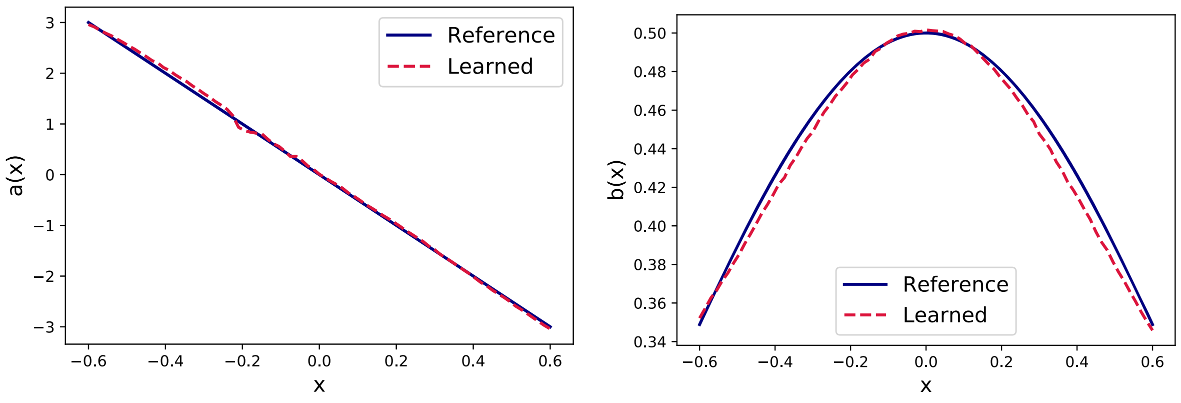

Fig. 15. Nonlinear SDE (5.3). Left: recovery of the drift $a(x) = -\mu x$; Right: recovery of the diffusion $b(x) = \sigma e^{-x^2}$.

5.2.2. *Trigonometric drift and diffusion*

We now consider the following non-linear SDE:

$$dx_t = \sin(2k\pi x_t)dt + \sigma\cos(2k\pi x_t)dW_t, \tag{5.4}$$

where the constant k and σ are set at $k = 1$ and $\sigma = 0.5$. The training data are generated with initial conditions uniformly distributed as $\mathcal{U}(0.35, 0.7)$ for up to time $T = 1.0$ as the training data. See the left of Fig. 17 for an example trajectory. Upon training the sFML model from the training data, we simulate the learned process for a longer time up to $T = 10$. A sample of the simulated trajectory

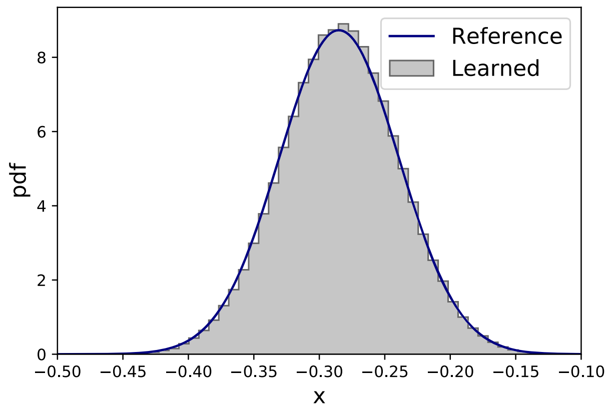

Fig. 16. Nonlinear SDE (5.3). Comparison of the conditional distribution for $\mathbf{G}_\Delta(-0.3)$ and $\widetilde{\mathbf{G}}_\Delta(-0.3)$.

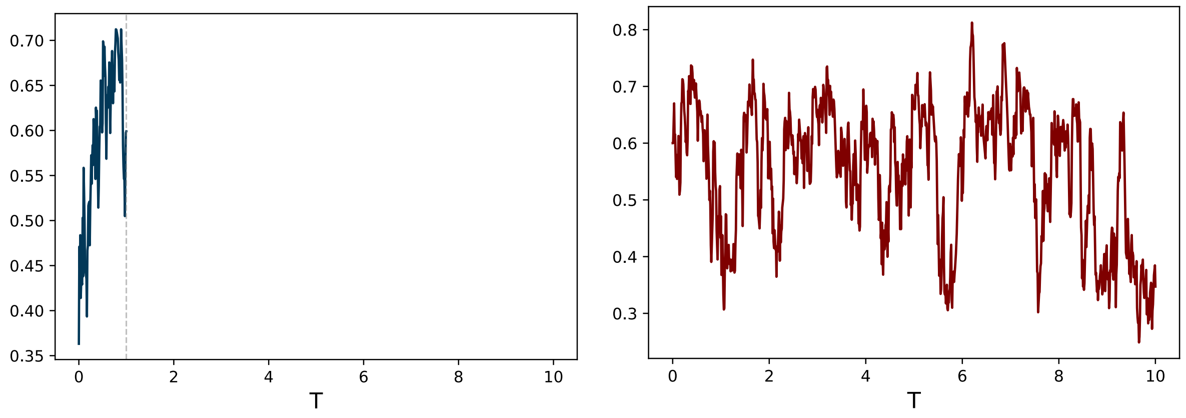

Fig. 17. Nonlinear SDE (5.4). Left: a sample trajectory used in the training data; Right: a sample trajectory of the simulated solution by the learned sFML model for time up to $T = 10.0$ with an initial condition $x_0 = 0.6$.

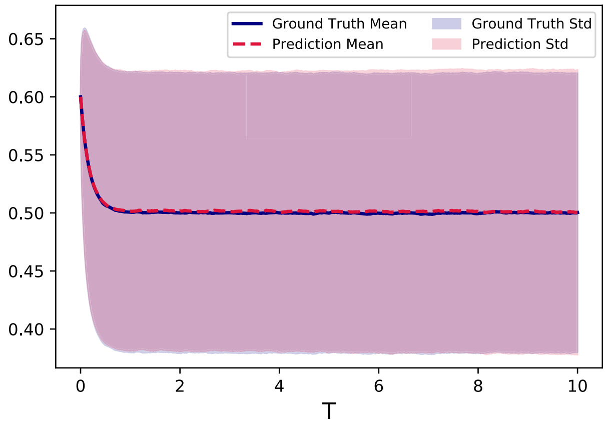

Fig. 18. Mean and standard deviation of the learned sFML model for (5.4).

is shown on the right of Fig. 17, under the initial condition $x_0 = 0.6$. The mean and standard deviation of the solutions are estimated using 100,000 sFML model simulations and plotted Fig. 18. They agree very well with those of the true solutions.

We then compute the effective drift and diffusion functions (4.23) from the learned sFML model and compare them to the true drift and diffusion functions. The comparison is shown in Fig. 19, where good agreement can be seen. The accuracy starts to deteriorate near the end points, as there are increasingly small number of samples towards the end points. Accuracy loss is thus expected. The conditional distribution by the learned sFML stochastic flow map $\widetilde{\mathbf{G}}_\Delta(x)$ is plotted for $x = 0.5$ (an arbitrary choice for illustration purpose). It agrees well with the distribution of the true stochastic flow map $\mathbf{G}_\Delta(0.5)$, see also Fig. 20.

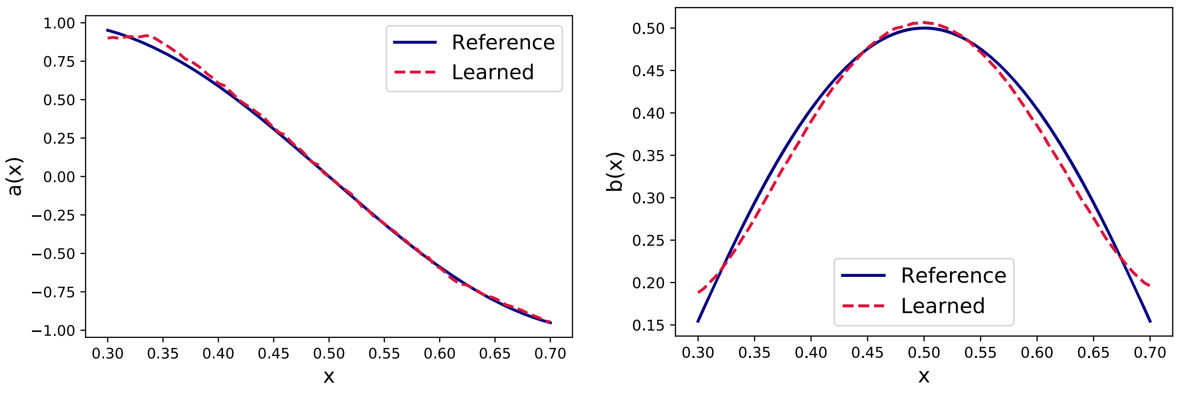

Fig. 19. Nonlinear SDE (5.4). Left: recovery of the drift $a(x) = \sin(2k\pi x)$; Right: recovery of the diffusion $b(x) = \sigma\cos(2k\pi x)$.

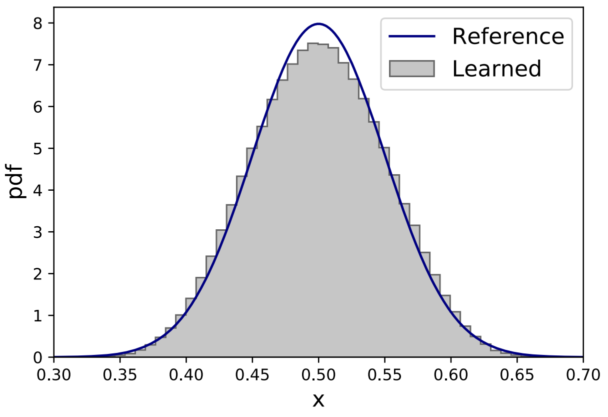

Fig. 20. Nonlinear SDE (5.4): comparison of the conditional distribution of $\mathbf{G}_\Delta(0.5)$ and $\widetilde{\mathbf{G}}_\Delta(0.5)$.

5.2.3. SDE with double well potential

We consider a one-dimensional SDE with a double well potential:

$$dx_t = (x_t - x_t^3)dt + \sigma dW_t, \tag{5.5}$$

where the constant σ is set as $\sigma = 0.5$. The stochastic driving term is sufficiently large to induce random solution transitions between the two stable states of $x = 1$ and $x = -1$.

The training data are generated by solving the true SDE with initial conditions uniformly sampled in $\mathcal{U}(-2.5, 2.5)$ for up to $T = 1.0$. A few such sample trajectories are shown on the left of Fig. 21. We emphasize that the trajectories in the training data set are so short that they do not contain transitions between the two stable states. Using these short bursts of training data, we train the sFML model and conduct long-term predictions. The results of two solution trajectories with an initial condition $x_0 = 1.5$ for time up to $T = 300$ are shown on the right of Fig. 21. We clearly observe the solution transition between the two stable states. The transitions occur with a small probability and over a time frame on the order of $O(10)$. Thus, the transitions are not observed in the training data, which span only $T = 0.4$. This result demonstrates the learning capability of sFML – the learned model is able to produce the correct solution behaviors even though they may not be present in the training data.

We then recover the drift and diffusion functions by computing the effective drift and diffusion functions (4.23). The results are shown in Fig. 22, where good agreement in the drift function is observed. Although the error in the constant diffusion function is visible, it is rather acceptable at about 2%. The evolution of the probability distribution of the solution is shown in Fig. 23, at various time levels $T = 0.5, 10.0, 30.0, 100.0$. It is started from an initial condition $x_0 = 1.5$ and computed by using 100,000 simulated trajectories by the sFML model. Excellent agreement with the reference solution from the true SDE is evident.

5.3. SDEs with non-Gaussian noises

The applicability of the proposed sFML approach is not restricted to modeling the classical SDE with Gaussian noises from Wiener process. To demonstrate this, we consider SDEs with non-Gaussian noises.

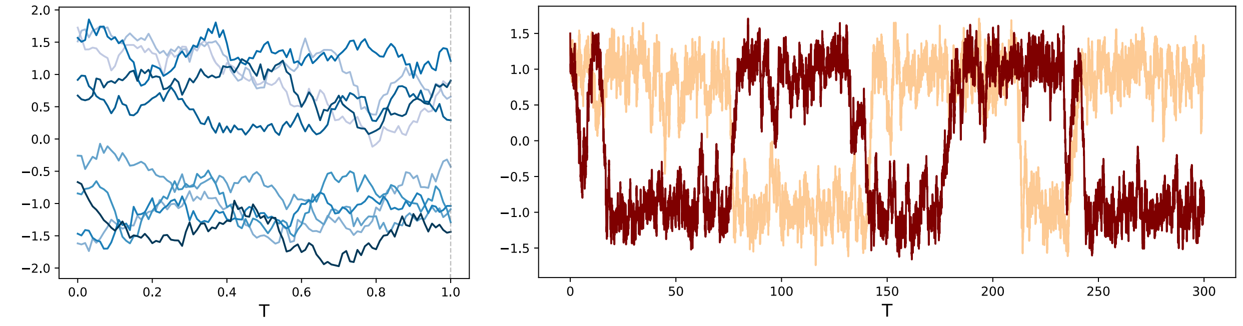

Fig. 21. SDE with double well potential (5.5). Left: samples of the training date trajectories for $T = 1.0$ (Note that there are no solution transitions.); Right: solution trajectories by the learned sFML model for time $T = 300.0$, with an initial condition $x_0 = 1.5$. Note the state transitions occur over time.

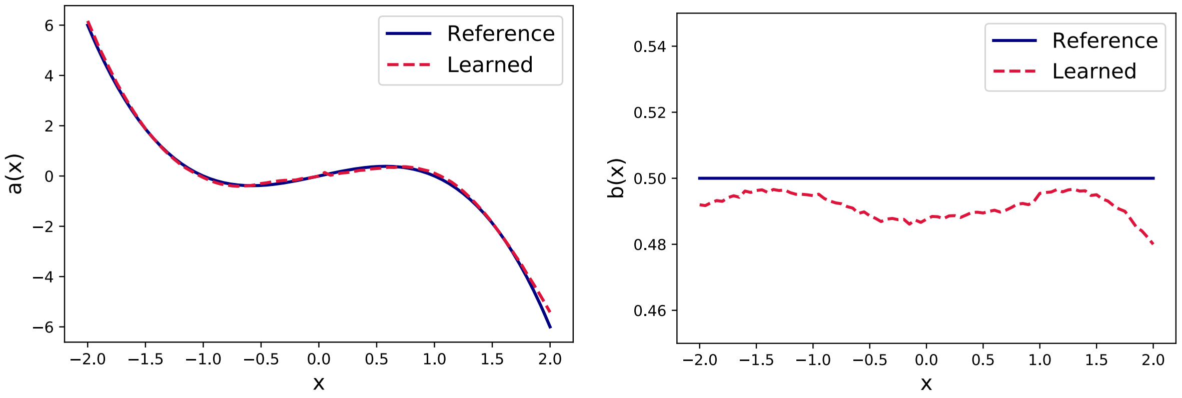

Fig. 22. SDE with double well potential (5.5). Left: recovery of the drift $a(x) = x - x^3$; Right: recovery of diffusion $b(x) = 0.5$.

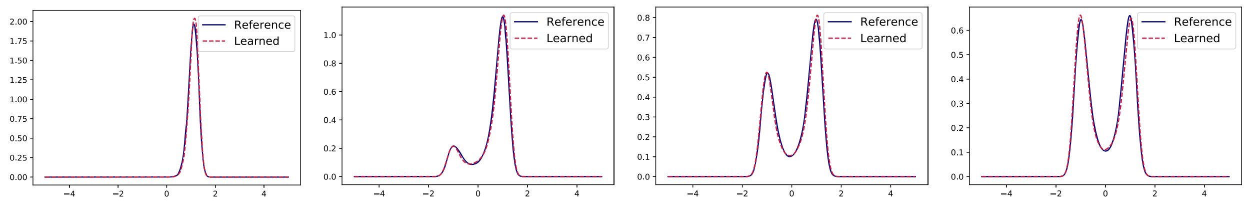

Fig. 23. SDE with double well potential (5.5). The predicted solution pdf's with an initial condition $x_0 = 1.5$ against those of the reference true solution at time $T = 0.5$, 10, 30 and 100 (from left to right).

5.3.1. Noises with exponential distribution

Let us consider a SDE with exponentially distributed noise inputs,

$$dx_t = \mu x_t dt + \sigma \sqrt{dt}\eta_t, \qquad \eta_t \sim \text{Exp}(1), \tag{5.6}$$

where η_t has an exponential pdf $f_\eta(x) = e^{-x}$, $x \geq 0$, and the constants are set as $\mu = -2.0$ and $\sigma = 0.1$.

We plot a few samples of the training data in the left of Fig. 24 up to $T = 1.0$. Some of the simulated trajectory samples by the learned sFML model are shown are on the right of Fig. 24 for up to $T = 5.0$. The mean and standard deviation of the sFML prediction are shown on the left of Fig. 25, where good agreement with those of the truth can be observed. On the right of Fig. 25, we show the conditional distribution by the sFML model generator $\widetilde{\mathbf{G}}_\Delta(x)$ at $x = 0.34$ (an arbitrary choice for demonstration). It agrees with the reference solution $\mathbf{G}_\Delta(0.34)$ generally well, with visible discrepancy at the left end of the domain. The reference distribution is exponential and has a clear cut-off, whereas the sFML solution exhibits a small tail. This is not surprising because the sFML generative model is built upon stochastic input $\mathbf{z}$ with the standard Gaussian distribution. Presumably the small discrepancy in the left tail can be eliminated if one uses exceedingly high order accuracy in the approximation. This is difficult to achieve with DNN training and we did not pursue it further.

Upon moving the non-zero mean of the exponential distribution out of the noise term, we can write the SDE in the form of the classical SDE (2.6) and obtain the drift and diffusion as

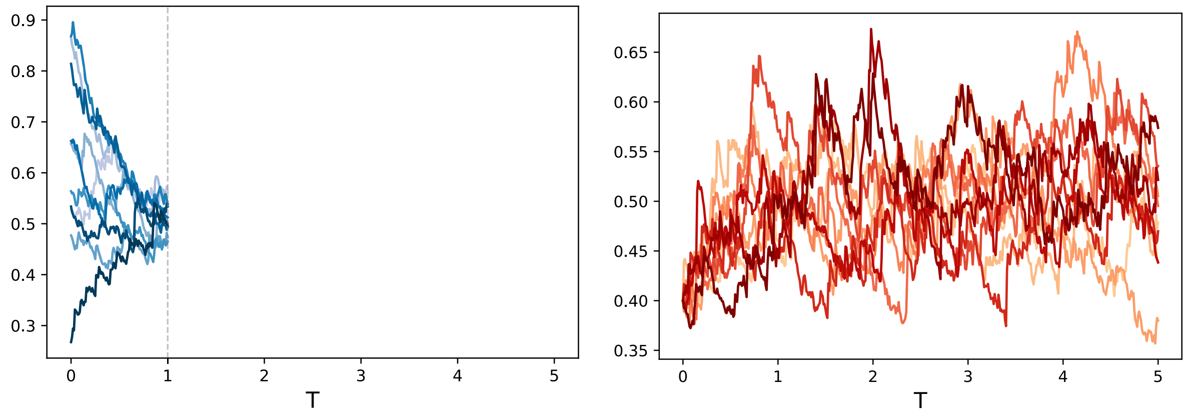

Fig. 24. SDE with exponential noise. Left: samples of the training data trajectories; Right: Samples of the sFML model simulations with an initial condition $x_0 = 0.4$ for up to $T = 5.0$.

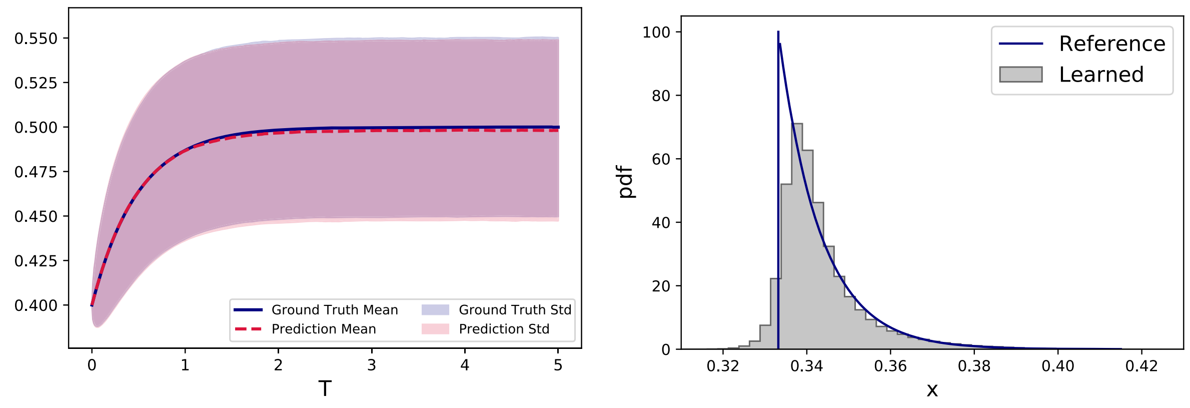

Fig. 25. SDE with exponential noise. Left: mean and standard deviation of the sFML model predictions; Right: conditional distribution by the sFML model $\widetilde{\mathbf{G}}_\Delta(x)$ against that of the true model $\mathbf{G}_\Delta(x)$ at $x = 0.34$.

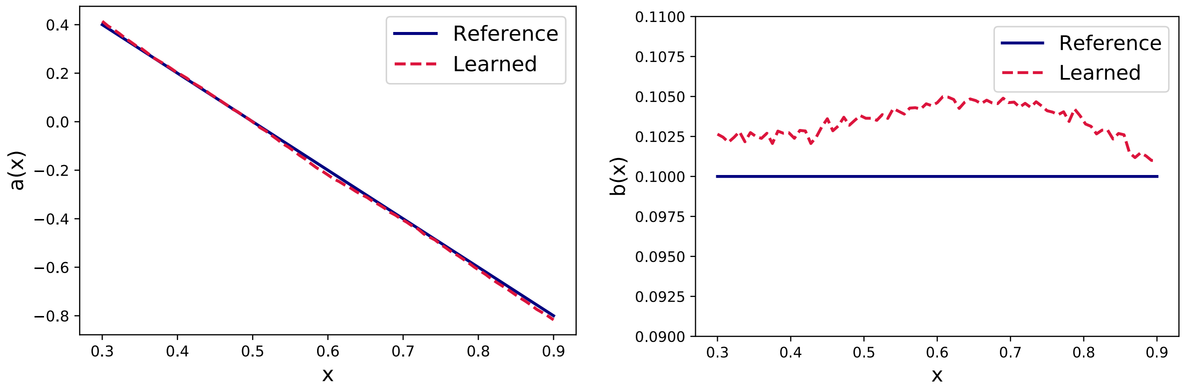

Fig. 26. SDE with exponential noise. Left: Recovery of the drift $a(x) = \mu x + \sigma/\sqrt{\Delta}$; Right: recovery of the diffusion $b(x) = \sigma$.

$$\begin{aligned} a(x_n) &= \mathbb{E}_\omega\left(\frac{x_{n+1}-x_n}{\Delta}\middle| x_n\right) = \mu x_n + \frac{\sigma}{\sqrt{\Delta}}, \\ b(x_n) &= \mathrm{Std}_\omega\left(\frac{x_{n+1}-x_n}{\Delta}|x_n\right) = \sigma. \end{aligned} \tag{5.7}$$

The effective drift and diffusion recovered by the sFML model are shown in Fig. 26, where we observe very good agreement with the reference truth.

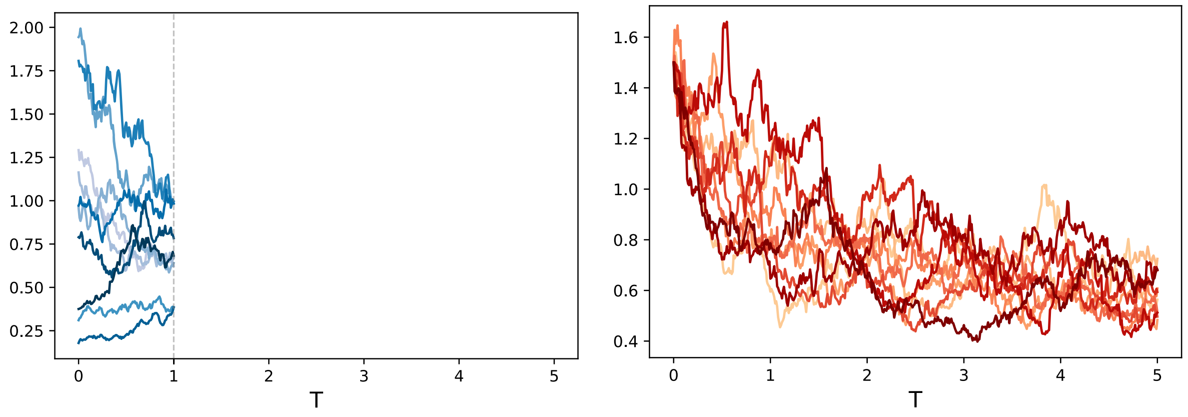

Fig. 27. SDE with lognormal distribution. Left: Samples of the training trajectory data; Right: Simulation sample of the trained sFML model with an initial condition $x_0 = 1.5$.

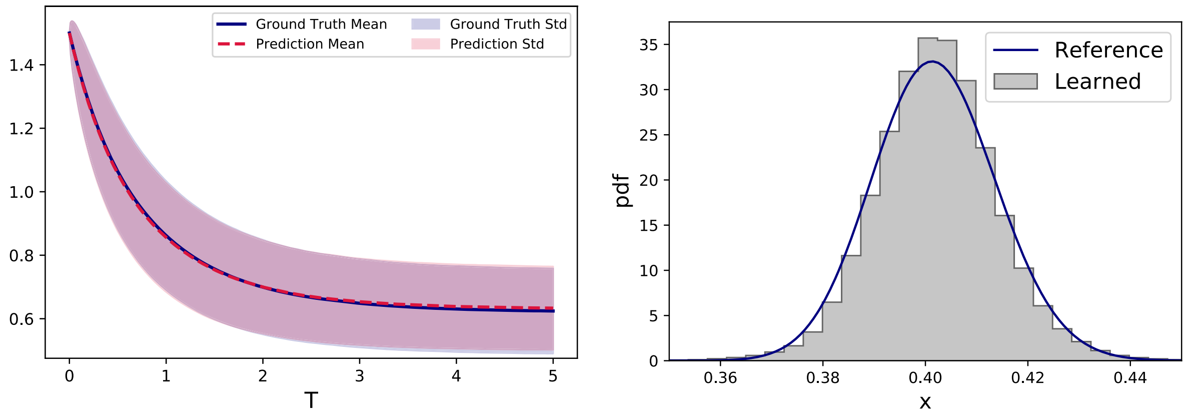

Fig. 28. SDE with lognormal distribution. Left: mean and standard deviation of the sFML model prediction with the initial condition $x_0 = 1.5$; Right: conditional probability by the sFML generator $\widetilde{\mathbf{G}}_\Delta(x)$ at $x = 0.4$.

5.3.2. *Noise with lognormal distribution*

We now consider the following stochastic system

$$d \log x_t = (\log m - \theta \log x_t)dt + \sigma dW_t, \tag{5.8}$$

where m, θ and σ are parameters. This is effectively an OU process after taking an exponential operation. Its dynamics can be solved by using Euler method with the following scheme:

$$x_{n+1} = m^\Delta x_n^{1-\theta\Delta} \eta_n^{\sigma\sqrt{\Delta}}, \qquad \eta_n \sim \text{Lognormal}(0, 1). \tag{5.9}$$

In our test, we take $m = 1/\sqrt{e}$, $\theta = 1.0$ and $\sigma = 0.3$. It is obvious that the conditional distribution from the stochastic flow map $\mathbf{G}_\Delta(x)$ follows lognormal distribution for any x. The training data are generated with initial condition uniformly sampled from $\mathcal{U}(0.1, 2.0)$ up to time $T = 1.0$. See the left of Fig. 27 for some samples. Upon training the sFML model, we conduct system prediction for time up to $T = 5.0$. Some simulation samples with an initial condition $x_0 = 1.5$ are shown on the right of Fig. 27. The mean and standard deviation from the sFML model prediction, as well as the distribution of $\widetilde{\mathbf{G}}_\Delta$, are shown in Fig. 27. Good agreement with the truth reference solutions can be observed, see also Fig. 28.

Like in the previous example, we rewrite this SDE in the form of the classical SDE (2.6) and obtain

$$\begin{aligned} a(x_n) &= \ln\left[\left(\mathbb{E}_\omega\left(\frac{x_{n+1}}{x_n}\Big|x_n\right)\right)^{1/\Delta}\right] = \ln(m x_n^{-\theta}) + \frac{\sigma^2}{2}, \\ b(x_n) &= \text{Std}_\omega\left(x_{n+1}|x_n\right) = \sqrt{e^{\sigma^2\Delta} - 1}(m e^{\sigma^2/2})^\Delta (1 - \theta\Delta) x_n. \end{aligned} \tag{5.10}$$

From the learned sFML model, we estimate the effective drift and diffusion via

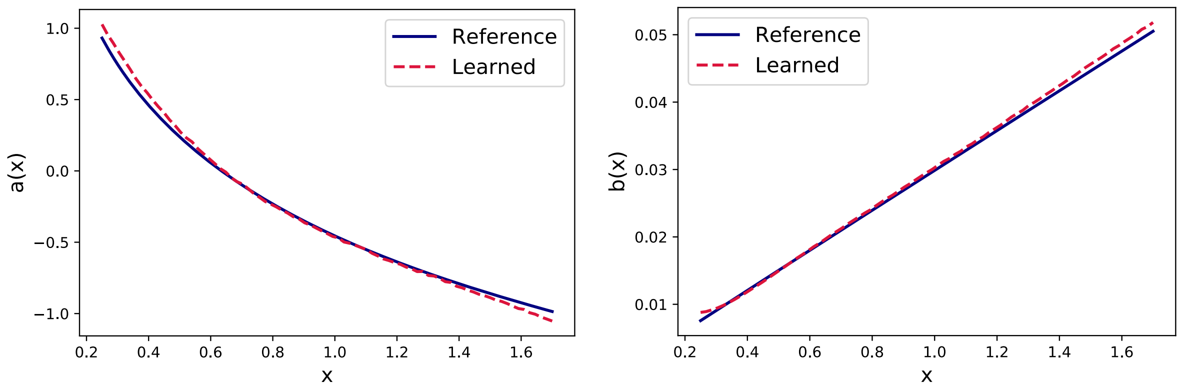

Fig. 29. SDE with lognormal distribution. Left: recovery of the drift $a(x)$; Right: recovery of conditional standard deviation $b(x)$. (See (5.10) for the reference solution.)

$$\hat{a}(x) = \ln\left[\left(\mathbb{E}_\omega\left(\frac{\widetilde{\mathbf{G}}_\Delta(x,\omega)}{x}\right)\right)^{1/\Delta}\right], \qquad \hat{b}(x) = \mathrm{Std}_\omega(\widetilde{\mathbf{G}}_\Delta(x,\omega)). \tag{5.11}$$

The comparison of functions for $a(x)$ and $b(x)$ are shown in Fig. 29, where we observe good agreement between the recovered functions by the learned sFML model and the reference (5.10).

5.4. Two-dimensional SDEs

In this section, we present examples of learning two-dimensional SDE systems.

5.4.1. Two-dimensional Ornstein–Uhlenbeck process

We first consider a 2-dimensional OU process,

$$d\mathbf{x}_t = \mathbf{B}\mathbf{x}_t dt + \boldsymbol{\Sigma} d\mathbf{W}_t, \tag{5.12}$$

where $\mathbf{x}_t = (x_1, x_2) \in \mathbb{R}^2$ are the state variables, $\mathbf{B}$ and $\boldsymbol{\Sigma}$ are (2×2) are matrices. In our test, we set

$$\mathbf{B} = \begin{pmatrix} -1 & -0.5 \\ -1 & -1 \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}.$$

Our training data are generated with random initial conditions uniformly sampled from $\mathcal{U}([-4,4]\times[-3,3])$, for a termination time up to $T = 1.0$. In the top row of Fig. 30, a few sample training data trajectories are shown. Upon training the sFML model, we conduct system predictions for time up to $T = 5$. In the bottom row of Fig. 30, we show a few prediction trajectories by the trained sFML model, with an initial condition of $\mathbf{x}_0 = (0.3, 0.4)$. The mean and the standard deviation of the model prediction are shown Fig. 31, where we observe very good agreement with the reference solutions. To examine the conditional probability distribution generated by the learned sFML model, we present both the joint probability distribution and its marginal distributions by the sFML generator $\widetilde{\mathbf{G}}_\Delta(\mathbf{x})$ at $\mathbf{x} = (0,0)$, along with the true distribution $\mathbf{G}_\Delta$ for comparison. The results are in Fig. 32, where good agreement with the true conditional distribution can be seen.

5.4.2. Stochastic oscillator

In this example, we consider the following stochastic oscillator,

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = -x_1 + \sigma \dot{W}_t. \end{cases} \tag{5.13}$$

Here σ is the perturbation parameter and set as $\sigma = 0.1$ in our test. Our training data are generated with initial conditions uniformly sampled from $\mathcal{U}([-1.5,1.5]\times[-1.5,1.5])$, for time up to $T = 1.0$. Some of the training data trajectories are shown on the left of Fig. 33, for visualization purpose. For the sFML model training, we use 100 recurrent steps for 5,000 epochs for the deterministic sub-map $\widetilde{\mathbf{D}}_\Delta$. For the stochastic sub-map, we use GANs the 4-layer, each of which with 80 nodes. Once trained, we conduct system prediction using the learned sFML model for time up to $T = 6.5$. Some sample trajectories are shown on the right of Fig. 33, for an initial condition of $\mathbf{x}_0 = (0.3, 0.4)$.

The mean and standard deviation of the system prediction by the learned sFML model are shown in Fig. 34, along with those of the true solution. Good accuracy of the model prediction is observed. In Fig. 35, we present the marginal distributions of the conditional probability distribution produced by the learned stochastic flow map $\widetilde{\mathbf{G}}_\Delta(\mathbf{x})$ at $\mathbf{x} = (-0.5, -0.5)$, in comparison with those of the true conditional distribution from $\mathbf{G}_\Delta$. In this particularly case, the marginal distribution in x_1 is a Dirac measure, as the equation of x_1 is deterministic. We observe that the learned sFML model produces very accurate prediction for such an extreme case.

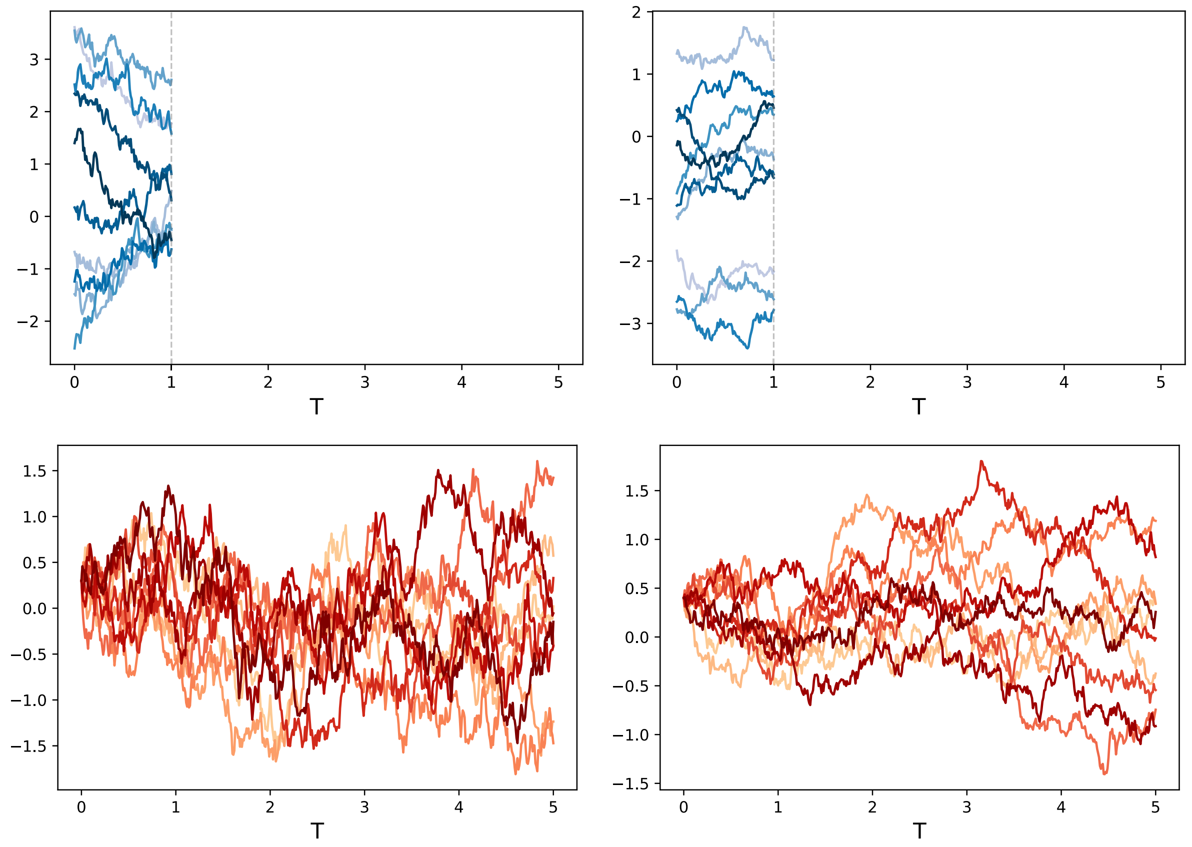

Fig. 30. Learning the 2d OU process (5.12). Top row: samples of the training data for x_1 (left) and x_2 (right); Bottom row: The sFML model prediction trajectories for x_1 (left) and x_2 (right), with an initial condition $\mathbf{x}_0 = (0.3, 0.4)$.

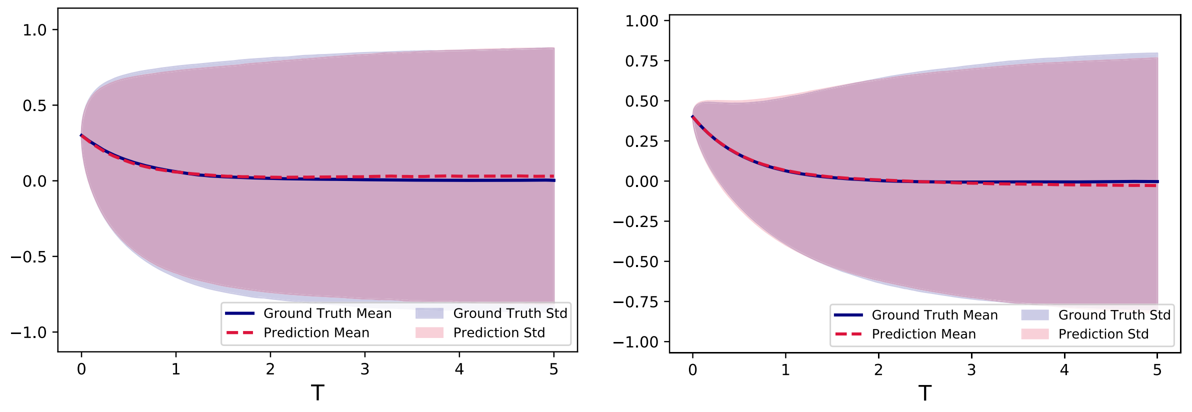

Fig. 31. Solution mean and standard deviation of the sFML model prediction for the 2d OU process (5.12). Left: x_1; Right: x_2.

6. Conclusions

In this paper, we proposed a general numerical framework for modeling unknown stochastic system by using observed trajectory data. Our method is based on constructing an approximation of the underlying flow map of the unknown stochastic system. The constructed flow map consists of two parts: a deterministic sub-map and a stochastic sub-map, both of which are learned by using the same observation data set. While a standard feedforward DNN is used to learn the deterministic sub-map, which predicts the conditional mean of the underlying dynamics, a generative DNN is used to learn the stochastic sub-map, which captures the noisy dynamics. In this paper, we employ GANs as the stochastic sub-map generator. By using a comprehensive set of numerical examples, we demonstrated that the proposed approach is highly effective and accurate to model a variety of stochastic systems. The learned generative model is able to conduct long-term system predictions beyond the time domain of the training data. More research is

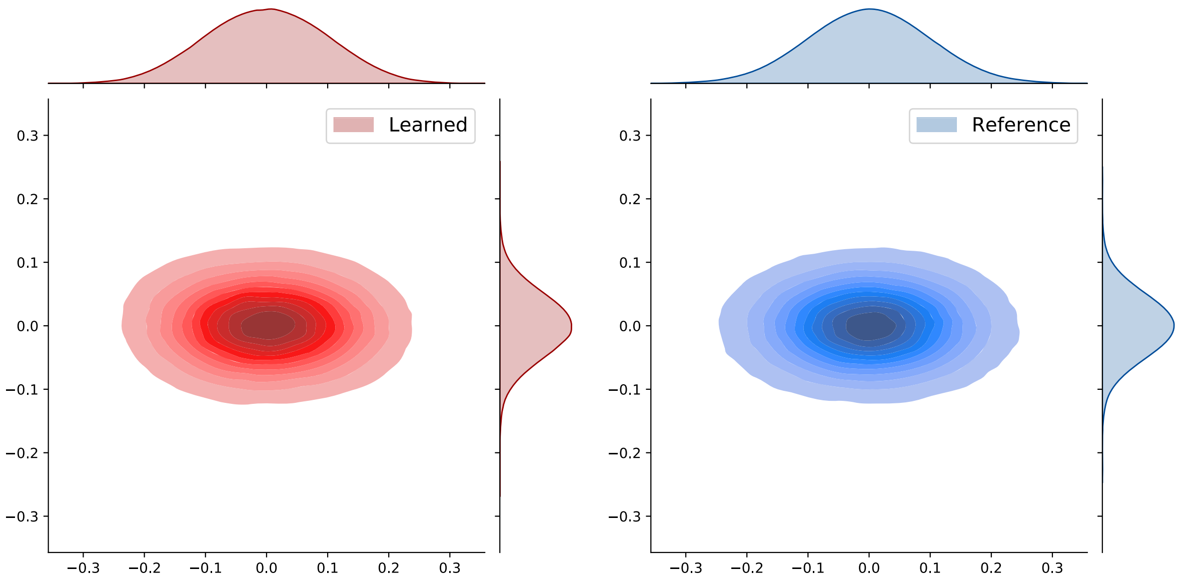

Fig. 32. Learning the 2d OU process (5.12).: conditional probability distribution from the learned sFML generator $\widetilde{\mathbf{G}}_\Delta(\mathbf{x})$ (left) and from the true generator $\mathbf{G}_\Delta(\mathbf{x})$ (right) at $\mathbf{x} = (0, 0)$.

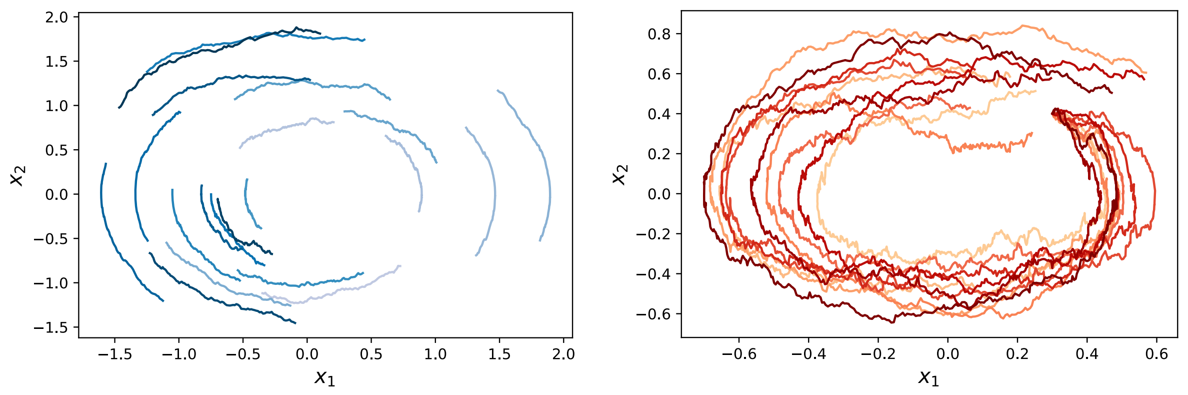

Fig. 33. Stochastic oscillator (5.13). Left: samples of the phase portrait of the training trajectory data (up to $T = 1.0$); Right: phase portrait of samples of the sFML prediction solution for up to $T = 6.5$, with the initial condition $\mathbf{x}_0 = (0.3, 0.4)$.

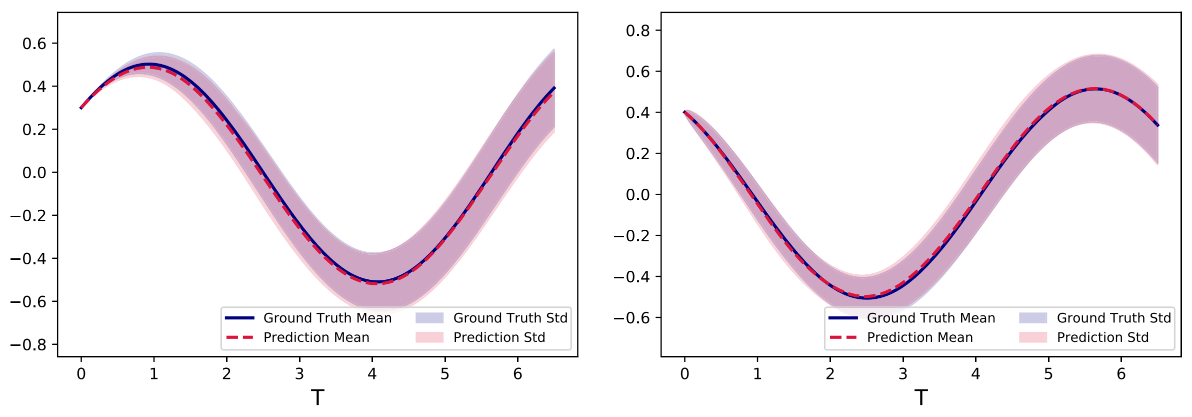

Fig. 34. Mean and standard deviation of the stochastic oscillator (5.13). Left: x_1; Right: x_2.

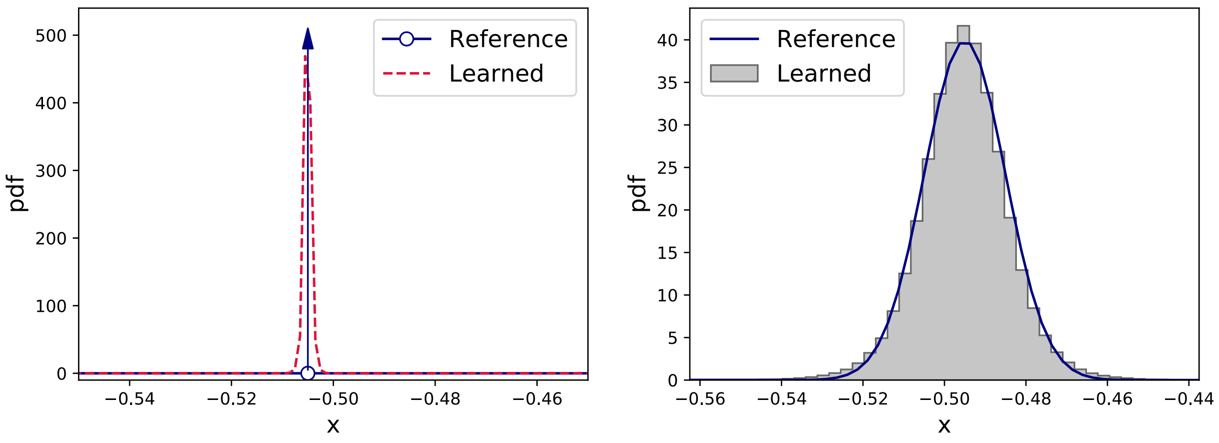

Fig. 35. Stochastic oscillator (5.13): marginal distributions for x_1 (left) and x_2 (right) generated by the learned model $\widetilde{\mathbf{G}}_\Delta(\mathbf{x})$ and the true model $\mathbf{G}_\Delta(\mathbf{x})$ at $\mathbf{x} = (-0.5, -0.5)$.

needed to further examine and understand the proposed method. For example, other generative models such as normalizing flow can be used for the construction of the stochastic sub-map, as GANs are known to be difficult to train. Also, in the current setting, our method is restricted to data with a constant time step Δt. The learning of stochastic systems also requires a large number of data. These issues will be studied and reported in future work.

CRediT authorship contribution statement

Yuan Chen: Investigation, Software, Validation, Writing – original draft. **Dongbin Xiu:** Methodology, Project administration, Writing – review & editing, Conceptualization, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Dongbin Xiu reports financial support was provided by Air Force Office of Scientific Research FA9550-22-1-0011. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] C. Archambeau, D. Cornford, M. Opper, J. Shawe-Taylor, Gaussian process approximations of stochastic differential equations, in: N.D. Lawrence, A. Schwaighofer, J. Quiñonero Candela (Eds.), Gaussian Processes in Practice, Bletchley Park, UK, 12–13 Jun 2007, PMLR, in: Proceedings of Machine Learning Research, vol. 1, 2007, pp. 1–16, https://proceedings.mlr.press/v1/archambeau07a.html.
[2] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: D. Precup, Y.W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, PMLR, 06–11 Aug 2017, in: Proceedings of Machine Learning Research, vol. 70, 2017, pp. 214–223, https://proceedings.mlr.press/v70/arjovsky17a.html.
[3] D. Berthelot, T. Schumm, L. Metz, Began: boundary equilibrium generative adversarial networks, arXiv preprint, arXiv:1703.10717, 2017.
[4] L. Boninsegna, F. Nüske, C. Clementi, Sparse learning of stochastic dynamical equations, J. Chem. Phys. 148 (2018) 241723.
[5] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (2016) 3932–3937.
[6] X. Chen, J. Duan, J. Hu, D. Li, Data-driven method to learn the most probable transition pathway and stochastic differential equation, Phys. D, Nonlinear Phenom. 443 (2023) 133559.
[7] X. Chen, L. Yang, J. Duan, G.E. Karniadakis, Solving inverse stochastic problems from discrete particle observations using the Fokker–Planck equation and physics-informed neural networks, SIAM J. Sci. Comput. 43 (2021) B811–B830.
[8] Z. Chen, V. Churchill, K. Wu, D. Xiu, Deep neural network modeling of unknown partial differential equations in nodal space, J. Comput. Phys. 449 (2022) 110782.
[9] G. Chevillon, D. Hendry, Non-parametric direct multi-step estimation for forecasting economic progresses, Int. J. Forecast. 21 (2005) 201–218.
[10] M. Darcy, B. Hamzi, G. Livieri, H. Owhadi, P. Tavallali, One-shot learning of stochastic differential equations with data adapted kernels, Phys. D, Nonlinear Phenom. 444 (2023) 133583.
[11] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, I.G. Kevrekidis, Learning effective stochastic differential equations from microscopic simulations: linking stochastic numerics to deep learning, Chaos, Interdiscip. J. Nonlinear Sci. 33 (2023) 023121.

[12] N. Dridi, L. Drumetz, R. Fablet, Learning stochastic dynamical systems with neural networks mimicking the Euler-Maruyama scheme, in: 2021 29th European Signal Processing Conference (EUSIPCO), IEEE, 2021, pp. 1990–1994.
[13] P. Franses, R. Legerstee, A unifying view on multi-step forecasting using an autoregression, J. Econ. Surv. 24 (2009) 389–401.
[14] R. Friedrich, J. Peinke, M. Sahimi, M.R.R. Tabar, Approaching complexity by stochastic methods: from biological systems to turbulence, Phys. Rep. 506 (2011) 87–162.
[15] X. Fu, L.-B. Chang, D. Xiu, Learning reduced systems via deep neural networks with memory, J. Mach. Learn. Model. Comput. 1 (2020) 97–118.
[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc., 2014, https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (2020) 139–144.
[18] Y. Gu, J. Harlim, S. Liang, H. Yang, Stationary density estimation of Itô diffusions using deep learning, SIAM J. Numer. Anal. 61 (2023) 45–82, https://doi.org/10.1137/21M1445363.
[19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A.C. Courville, Improved training of Wasserstein gans, Adv. Neural Inf. Process. Syst. 30 (2017).
[20] S. Infante, C. Luna, L. Sánchez, A. Hernández, Approximations of the solutions of a stochastic differential equation using Dirichlet process mixtures and Gaussian mixtures, Stat. Optim. Inf. Comput. 4 (2016) 289–307.
[21] S.H. Kang, W. Liao, Y. Liu, Ident: identifying differential equations with numerical time evolution, J. Sci. Comput. 87 (2021) 1–27.
[22] S. Kazeminia, C. Baur, A. Kuijper, B. van Ginneken, N. Navab, S. Albarqouni, A. Mukhopadhyay, Gans for medical image analysis, Artif. Intell. Med. 109 (2020) 101938.
[23] Y. Li, J. Duan, A data-driven approach for discovering stochastic dynamical systems with non-Gaussian Lévy noise, Phys. D, Nonlinear Phenom. 417 (2021) 132830.
[24] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2021, https://openreview.net/forum?id=c8P9NQVtmnO.
[25] J. Liu, Z. Long, R. Wang, J. Sun, B. Dong, Rode-net: learning ordinary differential equations with randomness from data, arXiv preprint, arXiv:2006.02377, 2020.
[26] Z. Long, Y. Lu, B. Dong, Pde-net 2.0: learning pdes from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[27] B. Øksendal, Stochastic Differential Equations, in Stochastic Differential Equations, Springer, 2003, pp. 65–84.
[28] M. Opper, Variational inference for stochastic differential equations, Ann. Phys. 531 (2019) 1800233.
[29] H. Owhadi, Computational graph completion, Res. Math. Sci. 9 (2022) 27.
[30] T. Qin, Z. Chen, J.D. Jakeman, D. Xiu, Data-driven learning of nonautonomous systems, SIAM J. Sci. Comput. 43 (2021) A1607–A1624.
[31] T. Qin, K. Wu, D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. 395 (2019) 620–635.
[32] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv preprint, arXiv:1801.01236, 2018.
[33] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[34] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, Generative adversarial text to image synthesis, in: International Conference on Machine Learning, PMLR, 2016, pp. 1060–1069.
[35] H. Schaeffer, S.G. McCalla, Sparse model selection via integral terms, Phys. Rev. E 96 (2017) 023302.
[36] H. Schaeffer, G. Tran, R. Ward, Extracting sparse high-dimensional dynamics from limited data, SIAM J. Appl. Math. 78 (2018) 3279–3295.
[37] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, B. Catanzaro, High-resolution image synthesis and semantic manipulation with conditional gans, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8798–8807.
[38] Y. Wang, H. Fang, J. Jin, G. Ma, X. He, X. Dai, Z. Yue, C. Cheng, H.-T. Zhang, D. Pu, et al., Data-Driven Discovery of Stochastic Differential Equations, Engineering, 2022.
[39] A. Weiss, A. Andersen, Estimating time series models using the relevant forecast evaluation criterion, J. R. Stat. Soc., Ser. A, Stat. Soc. 147 (1984) 484.
[40] K. Wu, D. Xiu, Numerical aspects for approximating governing equations using data, J. Comput. Phys. 384 (2019) 200–221.
[41] K. Wu, D. Xiu, Data-driven deep learning of partial differential equations in modal space, J. Comput. Phys. 408 (2020) 109307.
[42] K. Xu, E. Darve, Solving inverse problems in stochastic models using deep neural networks and adversarial training, Comput. Methods Appl. Mech. Eng. 384 (2021) 113976.
[43] L. Yang, C. Daskalakis, G.E. Karniadakis, Generative ensemble regression: learning particle dynamics from observations of ensembles with physics-informed deep generative models, SIAM J. Sci. Comput. 44 (2022) B80–B99.
[44] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, SIAM J. Sci. Comput. 42 (2020) A292–A317.
[45] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, J. Comput. Phys. 394 (2019) 136–152.
[46] C. Yildiz, M. Heinonen, J. Intosalmi, H. Mannerstrom, H. Lahdesmaki, Learning stochastic differential equations with Gaussian processes without gradient matching, in: 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2018, pp. 1–6.
[47] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, D.N. Metaxas, Stackgan: text to photo-realistic image synthesis with stacked generative adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5907–5915.
[48] Y. Zhu, Y.-H. Tang, C. Kim, Learning stochastic dynamics with statistics-informed neural network, J. Comput. Phys. 474 (2023) 111819.