MODELING UNKNOWN STOCHASTIC DYNAMICAL SYSTEM VIA AUTOENCODER

ZHONGSHU XU, YUAN CHEN, QIFAN CHEN AND DONGBIN XIU*

Abstract. We present a numerical method to learn an accurate predictive model for an unknown stochastic dynamical system from its trajectory data. The method seeks to approximate the unknown flow map of the underlying system. It employs the idea of autoencoder to identify the unobserved latent random variables. In our approach, we design an encoding function to discover the latent variables, which are modeled as unit Gaussian, and a decoding function to reconstruct the system's future states. Both the encoder and decoder are expressed as deep neural networks (DNNs). Once the DNNs are trained by the trajectory data, the decoder serves as a predictive model for the unknown stochastic system. Through an extensive set of numerical examples, we demonstrate that the method is able to produce long-term system predictions by using short bursts of trajectory data. It is also applicable to systems driven by non-Gaussian noises.

Key words. Data-driven modeling, stochastic dynamical systems, deep neural networks, autoencoder

MSC codes. 60H10, 60H35, 62M45, 65C30

1. Introduction.

Designing data-driven methods to discover unknown physical systems has attracted an increasing amount of attention recently. The goal is to discover the fundamental laws or equations behind the measurement data, in order to construct an effective predictive model for the unknown dynamics. Most of the existing methods are developed for learning deterministic dynamical systems. These include SINDy ([3]), physics-informed neural networks (PINNs) ([23, 24]), Fourier neural operator (FNO) ([16]), computational graph completion ([21]), sparsity promoting methods ([25, 26, 14]), flow map learning (FML) ([22, 8]), to name a few.

For a stochastic system, noises in data, along with the unobservability of the inherent stochasticity in the system, pose significant challenges in learning the system. Most of the existing methods focus on learning Itô type stochastic differential equations (SDEs). These methods employ techniques such as Gaussian process ([30, 2, 9, 20]), polynomial approximations ([28, 15]), deep neural networks (DNNs) [5, 29, 6, 31], etc. More recently, a stochastic extension of the deterministic FML approach ([22, 8]) was proposed in [7], where generative model such as GANs (generative adversarial networks) was employed.

The focus, as well as the contribution, of this paper, is on the development of a new generative model for data-driven modeling of unknown stochastic dynamical systems. Similar to the work of [7], the new method also employs the broad framework of FML, where a time-stepper is constructed using short bursts of measurement data to provide reliable long-term predictions of the unknown system. The use of GANs in [7], albeit effective, is computationally challenging. This is because, by its mathematical formulation, GANs are intrinsically difficult to train to high accuracy, which is critical for long-term system prediction. To circumvent to the computational difficulty, we adopt the concept of autoencoder and propose a novel stochastic FML (sFML) for unknown SDEs. Autoencoder is a type of DNN used to learn effective codings of unlabelled data and is widely used in problems such as classification [11], principal component analysis (PCA) [13], image processing [27, 4], etc. It has also

[*] E-mail addresses: {xu.4202,chen.11050,chen.11010,xiu.16}@osu.edu. Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA. Funding: This work was partially supported by AFOSR FA9550-22-1-0011.

shown promises in scientific computing for problems including operator learning [19], forward and backward SDEs [32, 31, 12], uncertainty quantification [17], etc. A standard autoencoder learns two functions: an encoding function that learns the latent variables of the input data, and a decoding function that recreates the input data from the encoded representation. In the proposed method, we design an encoding function to extract the hidden latent random variables from the noisy trajectory data of the unknown SDE, and a decoding function to reconstruct the future state data given the current state data and the latent random variables learned from the encoder. Both the encoder and the decoder are expressed as DNNs. The training of the encoder is carried out by enforcing the latent variables to be unit Gaussian independent of the current state data. By doing so, we effectively assume that the unknown SDEs can be expressed, or approximated, by an Itô type SDE driven by a Wiener process. The training loss of the entire sFML autoencoder consists of two parts: a reconstruction loss for the decoder and a distributional loss for the encoder to enforce the latent variables to be unit Gaussian. An important sub-sampling strategy of the training data is also developed to ensure that the latent variables are independent of the current state variables. Upon successful training of the entire DNN autoencoder, the decoder becomes a predictive model for long-term system predictions. An extensive set of numerical examples, including learning unknown SDEs with non-Gaussian noises, are presented to demonstrate the effectiveness of the proposed autoencoder sFML approach. Our code is accessible at https://github.com/AtticusXu/Modeling-Unknown-Stochastic-Dynamical-System-via-Autoencoder

2. Setup and preliminaries.

Let us consider a d-dimensional ($d \geq 1$) stochastic process, $\mathbf{x}_t := \mathbf{x}(\omega, t) : \Omega \times [0, T] \mapsto \mathbb{R}^d$, $d \geq 1$, driven by an unknown stochastic differential equation (SDE), where Ω is the event space and $T > 0$ a finite time horizon. We are interested in constructing an accurate numerical model for the unknown SDE by using measurement data of $\mathbf{x}_t$.

2.1. Assumption.

We assume the stochastic process $\mathbf{x}_t$ is time-homogenous, (cf. [18], Chapter 7), in the sense that for any time $\Delta \geq 0$,

$$\mathbb{P}(\mathbf{x}_{s+\Delta}|\mathbf{x}_s = \mathbf{x}) = \mathbb{P}(\mathbf{x}_\Delta|\mathbf{x}_0 = \mathbf{x}), \qquad s \geq 0. \tag{2.1}$$

More specifically, $\mathbf{x}_t$ is a time-homogeneous Itô diffusion driven by

$$d\mathbf{x}_t = \mathbf{b}(\mathbf{x}_t)dt + \sigma(\mathbf{x}_t)d\mathbf{W}_t, \tag{2.2}$$

where $\mathbf{W}_t$ is m-dimensional Brownian motion, and $\mathbf{b} : \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times m}$, $m \geq 1$, satisfy appropriate conditions (e.g., Lipschitz continuity). However, we assume that no information about the equation (2.2) is available: $\mathbf{b}$ and σ are not known, the Brownian motion $\mathbf{W}_t$ is not observable, and even its dimension m is not known.

2.2. Data.

We assume that $N_T \geq 1$ solution trajectories of $\mathbf{x}_t$ are observed over discrete time instances,

$$\mathbf{x}\left(t_0^{(i)}\right), \mathbf{x}\left(t_1^{(i)}\right), \ldots, \mathbf{x}\left(t_{L_i}^{(i)}\right), \qquad i = 1, ..., N_T, \tag{2.3}$$

where (L_i+1) is the length of the i-th observation sequence. For simplicity, we assume a constant time lag among the time instances and trajectory length, i.e. $t_n^{(i)} - t_{n-1}^{(i)} \equiv \Delta$, for any $n, i \geq 1$, and $L_i \equiv L \geq 1$, for all i.

In the proposed modeling approach, the time instances are not required, thanks to the time homogeneity condition. We therefore assume the trajectory data take the following form

$$\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}, \dots, \mathbf{x}_L^{(i)}, \qquad i = 1, ..., N_T, \tag{2.4}$$

which follows the same format as in (2.3), except the time variables are not required.

2.3. Objective and Related Work. Our goal is to use the trajectory data (2.4) to construct an iterative predictive model

$$\widetilde{\mathbf{x}}_{n+1} = \mathbf{G}_\Delta(\widetilde{\mathbf{x}}_n; \mathbf{z}_n), \qquad \mathbf{z}_n \in \mathbb{R}^{n_z}, \quad n \geq 0, \tag{2.5}$$

where $\mathbf{z}_n$ is a n_z-dimensional standard random vector with $n_z \geq 1$. When given an initial condition $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$, the model (2.5) produces a trajectory that approximates the true trajectory in distribution, i.e.,

$$(\widetilde{\mathbf{x}}_0, \widetilde{\mathbf{x}}_1, \dots, \widetilde{\mathbf{x}}_N) \overset{d}{\approx} (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N), \tag{2.6}$$

for any finite N.

The numerical model (2.5) takes the form of a time stepper. It follows the data-driven modeling framework of flow map learning (FML) that was first proposed in [22] for deterministic dynamical systems. (See [8] for a review.) For modeling unknown stochastic systems, the work of [7] proposed a stochastic flow map learning (sFML) model

$$\mathbf{G}_\Delta(\mathbf{x}; \mathbf{z}) = \mathbf{D}_\Delta(\mathbf{x}) + \mathbf{S}_\Delta(\mathbf{x}; \mathbf{z}),$$

where $\mathbf{D}_\Delta$ is a deterministic sub-map and $\mathbf{S}_\Delta$ a stochastic sub-map. The deterministic sub-map approximates the conditional mean of the system, and the stochastic sub-map takes the form of generated adversarial networks (GANs) to accomplish the approximation goal (2.6). (See [7] for details.) Though effective, GANs are known to be difficult to train to high accuracy, which is critical for achieving long-term predictive accuracy of the sFML model (2.5).

2.4. Contribution. In this paper, we propose a new construction of the sFML predictive model (2.5) that offers more robust DNN training and better accuracy control. This is accomplished by employing the concept of autoencoder. A standard autoencoder learns two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. In our work, we focus on the training data set (2.4) and choose adjacent pairs $(\mathbf{x}_n, \mathbf{x}_{n+1})$. We then train an encoding function to identify the unobserved (hidden) stochastic component:

$$(\mathbf{x}_n, \mathbf{x}_{n+1}) \to \mathbf{z}_n,$$

and a decoding function to reconstruct the trajectory

$$(\mathbf{x}_n, \mathbf{z}_n) \to \widetilde{\mathbf{x}}_{n+1},$$

such that $\widetilde{\mathbf{x}}_{n+1} \approx \mathbf{x}_{n+1}$.

By properly imposing conditions on the latent stochastic component $\mathbf{z}_n$ and defining a training loss function, we demonstrate that the autoencoder sFML approach can yield a highly effective predictive model (2.5). Compared with the approach in [7], the current autodecoder sFML approach can also effectively discover the true dimensions of unobserved stochastic component of the system via the learning of the latent variable $\mathbf{z}_n$. This provides a more detailed and quantitative understanding of the unknown stochastic dynamical system. Moreover, our extensive numerical experimentations suggest that the autoencoder sFML approach allows more robust DNN training and yields higher accuracy in the corresponding predictive models.

3. Autoencoder Stochastic Flow Map Learning.

In this section, we describe the details of the proposed autoencoder sFML approach for modeling unknown stochastic dynamical systems. We first present the mathematical motivation, then discuss the numerical algorithm, particularly the DNN structure and loss function, followed by a discussion of several important numerical implementation details.

3.1. Mathematical Motivation.

For the training data set (2.4), we consider its data pairs, separated by the constant time step Δ,

$$(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}), \quad (\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}), \quad \dots, \quad (\mathbf{x}_{L-1}^{(i)}, \mathbf{x}_L^{(i)}), \qquad i = 1, \dots, N_T,$$

which contain a total of $M = N_T L$ such data pairs. Since the stochastic process is time-homogeneous, we rename each pair using indices 0 and 1, and re-organize the training data set into a set of such pairwise data entries,

$$\left(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}\right), \qquad i = 1, \dots, M, \qquad M = N_T L. \tag{3.1}$$

In other words, the training data set is re-arranged into M numbers of very short trajectories of only two entries. Each of the i-th trajectory, $i = 1, \dots, M$, starts with an “initial condition” $\mathbf{x}_0^{(i)}$ and ends a single time step Δ later at $\mathbf{x}_1^{(i)}$.

Since the process $\mathbf{x}_t$ follows the (unknown) time-homogeneous Ito diffusion (2.2), we have

$$\mathbf{x}_1 = \mathbf{x}_0 + \int_0^\Delta \mathbf{b}(\mathbf{x}_s) ds + \int_0^\Delta \sigma(\mathbf{x}_s) d\mathbf{W}_s, \tag{3.2}$$

where the training data (3.1) can be considered as sample paths of the process. This is the basis of our modeling principle: Given the training data (3.1), there exists a standard Gaussian random variable $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_{n_z})$, $n_z \geq 1$, independent of $\mathbf{x}_0$, and a function $\mathbf{G}_\Delta$ such that

$$\mathbf{x}_1^{(i)} = \mathbf{x}_0^{(i)} + \mathbf{G}_\Delta(\mathbf{x}_0^{(i)}, \mathbf{z}^{(i)}), \qquad i = 1, \dots, M. \tag{3.3}$$

Here, the standard Gaussian random variable $\mathbf{z}$ is the “hidden” latent variable we seek to learn via the encoding function. It represents the increment jump of the Brownian motion $\mathbf{W}_t$ over Δ, which follows a Gaussian distribution. Since the dimension of $\mathbf{W}_t$ is assumed to be unknown, the “true” dimension of $\mathbf{z}$, n_z, is also unknown.

Once the hidden variable $\mathbf{z}$ is learned by the encoding function, the unknown function $\mathbf{G}_\Delta$ will be learned by a decoding function. This is the design principle of our autodecoder sFML modeling of the SDE behind the data $\mathbf{x}_t$.

3.2. Method Description.

Our autoencoder sFML method consists of two functions: an encoding function $\mathbf{E}_\Delta$ and a decoding function $\mathbf{D}_\Delta$:

$$\begin{aligned} &\mathbf{E}_\Delta : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}^{n_\mathbf{z}}, \quad \text{(Encoder)} \\ &\mathbf{D}_\Delta : \mathbb{R}^d \times \mathbb{R}^{n_\mathbf{z}} \mapsto \mathbb{R}^d. \quad \text{(Decoder)} \end{aligned} \tag{3.4}$$

Both functions are realized via DNNs, which are to be trained via the training data set (3.1) with properly defined loss functions.

3.2.1. Encoding Function. The encoding function $\mathbf{E}_\Delta$ in (3.4) takes a data pair from the training data set (3.1) and returns a random vector of dimension n_z,

$$\mathbf{z}^{(i)} = \mathbf{E}_\Delta(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}), \qquad i = 1, \dots, M. \tag{3.5}$$

The goal is to enforce two conditions: (i) $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_{n_z})$, unit Gaussian distribution of dimension n_z, where $\mathbf{I}_{n_z}$ is the identity matrix of size $n_z \times n_z$; and (ii) $\mathbf{z}$ is independent of $\mathbf{x}_0$. Since both conditions are on the probabilistic properties of $\mathbf{z}$, we consider sets of the output of the encoder. Specifically, we divide the entire output set $\{\mathbf{z}^{(i)}, i = 1, \dots, M\}$ of M samples into $n_B > 1$ "batches" of N samples with $n_B N = M$. (For notational simplicity, we assume each batch contains an equal N number of samples.)

Let us consider one batch of N samples, $B = \left\{\mathbf{z}^{(i)}, i = 1, \dots, N\right\}$. We seek to enforce the distribution of the batch to be n_z-dimensional standard unit Gaussian distribution by minimizing a distributional loss function:

$$\mathcal{L}_D(B) = \mathcal{L}_{\text{distance}}(B) + \tau \cdot \mathcal{L}_{\text{moment}}(B), \tag{3.6}$$

where $\mathcal{L}_{\text{distance}}$ measures a statistical distance between the distribution of the batch B and the distribution of n_z-dimensional standard Gaussian, and $\mathcal{L}_{\text{moment}}$ measures the deviation of the moments of B to those of n_z-dimensional standard Gaussian, and $\tau > 0$ is a penalty parameter. Through our extensive numerical experimentation, we found that Renyi-entropy-based distance is suitable for $\mathcal{L}_{\text{distance}}$, and it is necessary to include the moment loss $\mathcal{L}_{\text{moment}}$ to ensure Gaussianity of the latent variable $\mathbf{z}$. The technical details of $\mathcal{L}_{\text{distance}}$ and $\mathcal{L}_{\text{moment}}$ are in Section 3.3.2 and 3.3.3, respectively.

While the minimization of the distributional loss (3.6) can force the batch samples to follow the unit normal distribution, it is not sufficient to ensure the samples are independent of $\mathbf{x}_0$. To enforce the independence condition, we recognize that the samples $\mathbf{z}^{(i)}$ in the batch B are computed by the encoder via (3.5). Consequently, sampling $\mathbf{z}^{(i)}$ is determined by sampling $\mathbf{x}_0^{(i)}$ in the training data set (3.1) via

$$B = \left\{\mathbf{z}^{(i)} = \mathbf{E}_\Delta\left(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}\right)\right\}_{i=1}^{N}, \tag{3.7}$$

Therefore, to enforce the samples $\mathbf{z}^{(i)}$ in the batch B to have the standard unit Gaussian distribution *and* be independent of $\mathbf{x}_0$, we propose the following sub-sampling principle for the encoding function construction:

> For each batch B (3.7), let $\left\{\mathbf{x}_0^{(i)}\right\}_{i=1}^{N}$ be sampled from an arbitrary non-Gaussian distribution out of the training data set (3.1), the encoding function (3.5) is determined by minimizing the distributional loss function (3.6) over the batch B.

The principle is applied to each of the $n_B > 1$ batches. This ensures that $\mathbf{z}$ follows the standard unit Gaussian distribution regardless of the distribution of $\mathbf{x}_0$. Subsequently, this promotes independence between $\mathbf{z}$ and $\mathbf{x}_0$. The effective way to "arbitrarily" sample $\mathbf{x}_0$ is discussed in detail in Section 3.3.1.

Remark 3.1. The proposed principle is critical in promoting independence between $\mathbf{z}$ and $\mathbf{x}_0$. Consider, for example, a case when the unknown SDE follows a simple Itô SDE (2.2) with a constant diffusion. In this case, if $\{\mathbf{x}_0^{(i)}\}$ are sampled following a Gaussian distribution, then $\{\mathbf{x}_1^{(i)}\}$ will follow a Gaussian distribution as well. Consequently, the encoding function $\mathbf{E}_\Delta$ can be defined as a linear combination of $\mathbf{x}_0$ and $\mathbf{x}_1$ to return a standard Gaussian variable $\mathbf{z}$. Such an encoder would satisfy a successful minimization of the distributional loss (3.6) but produce a latent variable $\mathbf{z}$ that is dependent on $\mathbf{x}_0$.

3.2.2. Decoding Function. The decoding function $\mathbf{D}_\Delta$ in (3.4) seeks to replicate $\mathbf{x}_1$ by taking inputs from $\mathbf{x}_0$ and the latent standard normal variable $\mathbf{z}$ identified by the encoder. Specifically, consider the same batch samples B in (3.7). We define the decoding function as

$$\widetilde{\mathbf{x}}_1^{(i)} = \mathbf{D}_\Delta(\mathbf{x}_0^{(i)}, \mathbf{z}^{(i)}), \qquad i = 1, \dots, N. \tag{3.8}$$

The decoder is then constructed by minimizing mean squared error (MSE) loss function

$$\mathcal{L}_{MSE}(B) = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{x}_1^{(i)} - \widetilde{\mathbf{x}}_1^{(i)} \right\|_2^2. \tag{3.9}$$

3.2.3. DNN Structure and Loss Function. Both the encoder (3.5) and decoder (3.8) are constructed using DNNs. In this paper, they both take the form of standard fully connected feedforward networks. The complete DNN structure is shown in Figure 1. The encoder takes the inputs of $\mathbf{x}_0$ and $\mathbf{x}_1$ from the training data set (3.1) and produces the output $\mathbf{z}$; the decoder takes $\mathbf{z}$, along with $\mathbf{x}_0$ and produces the output $\widetilde{\mathbf{x}}_1$. The training of the DNN is conducted by minimizing the following loss function averaged over the n_B batches:

$$\mathcal{L} = \frac{1}{n_B}\sum_{j=1}^{n_B} \mathcal{L}(B_j) = \frac{1}{n_B}\sum_{j=1}^{n_B} \left[\mathcal{L}_{MSE}(B_j) + \lambda \cdot \mathcal{L}_D(B_j)\right], \tag{3.10}$$

where $\mathcal{L}_D$ is the distributional loss (3.6), $\mathcal{L}_{MSE}$ the MSE loss (3.9), and $\lambda > 0$ a scaling parameter.

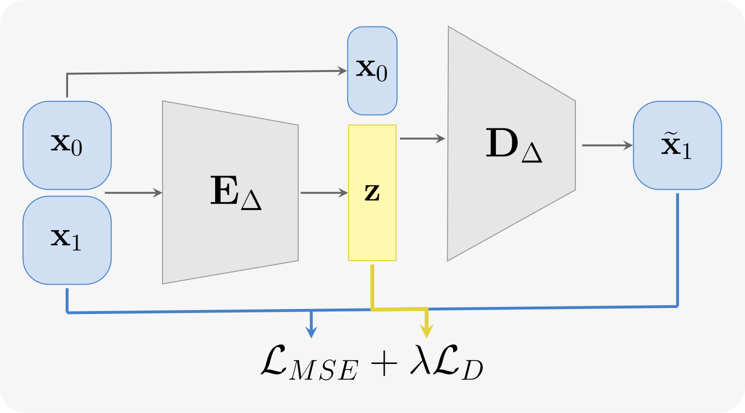

FIG. 1. *An illustration of the network structure and training loss for the proposed autoencoder sFML method.*

3.3. Algorithm Detail. In this section, we discuss a few important technical details for the implementation of the proposed autoencoder sFML method.

3.3.1. Sub-sampling Strategy. As discussed in Section 3.2.1, it is critical to select each of the n_B batches by sampling $\mathbf{x}_0$ in an arbitrary way out of the training data set (3.1). To accomplish this, we propose the following sampling strategy: consider all the samples of the entire training data set (3.1)

$$\left\{\mathbf{x}_0^{(i)}, i = 1, \dots, M\right\}.$$

(Since $\mathbf{x}_0$ and $\mathbf{x}_1$ appear in pairs, samples of $\mathbf{x}_1$ are immediately determined once samples of $\mathbf{x}_0$ are chosen.) To construct $n_B > 1$ batches, each of which contains N samples ($M = Nn_B$), we proceed as follows:

- Randomly choose n_B samples of $\mathbf{x}_0$ from (3.1) using uniform distribution;
- For each of the chosen $\mathbf{x}_0$ samples, find its $(N-1)$ nearest neighbor points to form a batch with N samples.

The procedure is carried out at the beginning of each epoch. This is to ensure the "arbitrariness" of the distribution of $\mathbf{x}_0$ in the batches. An illustrative example of sampling of 6 batches is shown in Figure 2. The distribution of the batches of $\mathbf{x}_0$ are shown on the left. They are quite arbitrary and clearly non-Gaussian. For reference, the corresponding distributions of $\mathbf{x}_1$ are shown on the right – they are close to Gaussian as expected. If the encoder $\mathbf{E}_\Delta$ can produce unit Gaussian variable $\mathbf{z}$ under such kind of arbitrarily distributed $\mathbf{x}_0$, when the distributions of $\mathbf{x}_0$ are arbitrarily different across different batches and arbitrarily changed at the beginning of each training iteration (i.e., epoch), it is reasonable to state that the output unit Gaussian $\mathbf{z}$ shall be independent of $\mathbf{x}_0$.

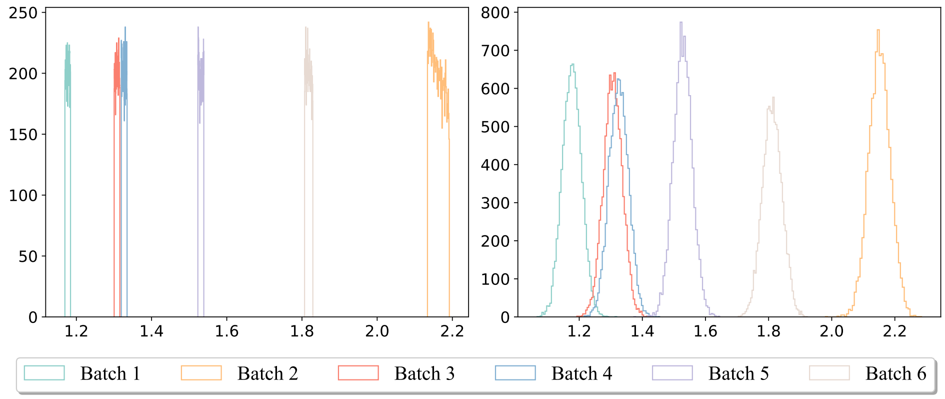

FIG. 2. *An illustration of the proposed batch sampling: Left: histograms of 6 batches of sampled* $\mathbf{x}_0$*'s; Right: histograms of the corresponding* $\mathbf{x}_1$*'s.*

3.3.2. Statistical Distance. There exist a variety of statistical distances that one can use in defining $\mathcal{L}_{\text{distance}}$ in the distributional loss (3.6). Through numerical experimentation, we have found Renyi-entropy-based distance ([1]) to be highly effective. The Renyi-entropy-based distance first estimates the probability density distribution of the samples using kernel density estimator (KDE) (c.f. [10]) and then computes the L^2 distance between the KDE estimated distribution to the target distribution. In our case, consider a batch $B = \left\{\mathbf{z}^{(i)}, i = 1, \dots, N\right\}$ with N samples. Its

KDE distribution estimation is

$$\hat{f}_B(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}\kappa_\sigma\left(\mathbf{z}^{(i)} - \mathbf{y}\right),$$

where κ_σ is a kernel function with bandwidth $\sigma > 0$. The Renyi-entropy-based distance loss is defined as

$$\mathcal{L}_{\text{distance}}(B) = \left\|\hat{f}_B - f_{\mathcal{N}}\right\|_2, \tag{3.11}$$

where $f_{\mathcal{N}}$ is the probability distribution function of n_z-dimensional unit Gaussian.

Mathematically speaking, minimization of this loss function to zero value shall enforce the distribution of the batch B to follow the unit Gaussian. In practice, however, this can not be accomplished, due to the discrete nature of the batch B and its finite number of samples, along with the limited capability of (stochastic) optimization method used during the training. (No optimization method can achieve zero loss value.) Our numerical experimentation suggests it is necessary to incorporate a measure of the statistical moments to enhance the learning.

3.3.3. Moment Loss Function. To enhance the performance of the training and further enforce the batch B to follow unit Gaussian distribution, we introduce moment loss $\mathcal{L}_{\text{moment}}(B)$ in the distributional loss (3.6). More specifically, we seek to enforce the marginal moments of the batch $B = \left\{\mathbf{z}^{(i)}, i = 1, \dots, N\right\}$ to match those of the unit Gaussian, for up to the 6th central moment.

We first consider the case of $n_z = 1$. The moment loss function takes the form

$$\mathcal{L}_{\text{moment}}(B) = \sum_{j=1}^{J}\frac{1}{c_j}\left[\widehat{\mu}^{(j)}(B) - \mu^{(j)}(P_{\mathcal{N}})\right]^2, \tag{3.12}$$

wherefor each $1 \le j \le J$, $\widehat{\mu}^{(j)}(B)$ is the j-th central moment of the batch B, $\mu^{(j)}(P_{\mathcal{N}})$ the j-th central moment of unit Gaussian, $c_j > 0$ a scaling parameter, and J the highest moment chosen. We set $J = 6$ in all of our numerical testing and found it to be a robust choice.

For unit Gaussian, we have $\mu^{(1)} = \mu^{(3)} = \mu^{(5)} = 0$, $\mu^{(2)} = 1$, $\mu^{(4)} = 3$, and $\mu^{(6)} = 15$. After extensive testing, we advocate the use of the following scaling parameters:

$$c_1 = 1.0, \quad c_2 = 1.0, \quad c_3 = 2.0, \quad c_4 = 3.0, \quad c_5 = 8.0, \quad c_6 = 15.0. \tag{3.13}$$

For multi-dimensional case $n_z > 1$, the moments are written as n_z-dimensional vectors consisting of the marginal moments in each direction, and vector 2-norm is used in (3.12). We also introduce cross-correlation coefficients among all pairs of dimensions, which are zero for the unit Gaussian distribution. The moment loss for $n_z > 1$ is thus defined as

$$\mathcal{L}_{\text{moment}}(B) = \sum_{j=1}^{6}\frac{1}{c_j}\left\|\widehat{\mu}_N^{(j)}(B) - \mu^{(j)}(P_{\mathcal{N}})\right\|_2^2 + \frac{\nu}{K}\sum_{1\le j<k\le n_z}\left(\widehat{\rho}_{jk}(B)\right)^2, \tag{3.14}$$

where $\widehat{\rho}_{jk}(B)$, $1 \le j < k \le n_z$, are the cross-correlation coefficient of the batch B, $K = n_z(n_z - 1)/2$ the total number of the cross-correlation coefficients, and $\nu > 0$ a penalty parameter. Upon extensive numerical experimentation, we suggest the use of $\nu = 2$.

4. Numerical Examples.

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

- Linear SDEs: Ornstein-Uhlenbeck (OU) process and geometric Brownian motion;
- Nonlinear SDEs: SDEs with exponential and trigonometric drift or diffusion, and SDE double-well potential;
- SDEs with non-Gaussian noise of exponential and lognormal distributions;
- Multi-dimensional SDEs: 2-dimensional and 5-dimensional OU processes. These examples demonstrate how the training procedure can automatically detect the correct dimension n_z for the latent random variable.

For all these problems, the true SDEs are used to generate training data sets, in the form of (3.1), as well as validation data to examine the accuracy of the prediction results by the learned sFML model. The knowledge of the true SDEs is otherwise not used anywhere in the model construction procedure. The trajectory data (2.4) are generated by solving the SDEs via Euler-Maruyama method, with $N_T = 10,000$ initial conditions uniformly distributed within a region (to be specified for each example) and a length $L = 100$ with a time step $\Delta = 0.01$. These N_T trajectories thus cover a time up to $T = 1.0$. They are subsequently reorganized via the procedure in Section 3.1, resulting in training data sets (3.1) with $M = 10^6$ data pairs. In all of the examples, we re-sample the training data sets into $n_B = 1,000$ batches, each of which contains $N = 10,000$ data pairs, by following the sub-sampling strategy from Section 3.3.1.

Regarding the DNN architecture, we employ fully connected feedforward DNNs for both the encoder and decoder. The encoder has 4 layers, each of which with 20 nodes. The first three layers utilize the eLU activation function. The decoder has the same structure as the encoder, except an identity operation is introduced to implement it in the ResNet structure. All the examples underwent training for up to 1,000 epochs.

To evaluate the efficacy of the method, we conduct system predictions using the learned sFML models for a time horizon typically up to $T = 5 \sim 500$, much longer than that of the training data (whose time horizon is up to $T = 1$.) We then compare the sFML prediction results against the ground truth obtained by solving the true SDEs. The comparisons include the following:

- Mean and standard deviation of the solution;
- Evolution of the solution probability distribution over time;
- The one-step conditional distribution $\mathbb{P}(\mathbf{x}_{n+1}|\mathbf{x}_n = \mathbf{x})$ for some arbitrarily chosen $\mathbf{x}$;
- The effective drift and diffusion functions reconstructed from the learned sFML models against the true drift and diffusion;
- The distribution of the latent variables obtained from the encoder against the true latent variables (unit Gaussian).

4.1. Linear SDEs.

We first present the results of learning an Ornstein Uhlenbeck (OU) process and a geometric Brownian motion.

4.1.1. OU Process.

The true SDE takes the following form,

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t, \tag{4.1}$$

where the parameters are set as $\theta = 1.0$, $\mu = 1.2$, and $\sigma = 0.3$.

For the training data, the initial points are sampled uniformly from interval $(0, 2.5)$. For the sFML model prediction, we fix the initial condition to be $x_0 = 1.5$

and generate $500,000$ prediction trajectories up to $T = 5$ to examine the solution statistics and distribution.

The mean and standard deviation of the predictions from the learned sFML model, along with the reference mean and standard deviation from the true SDE, are shown in Figure 3. Good agreement between the learned sFML model and the true model can be observed. Note that the agreement goes beyond the time horizon of the training data by a factor of 5.

We then recover the effective drift and diffusion of the sFML model. From Figure 4, we observe they closely approximate the reference true functions with relative errors of $O(10^{-3})$.

In Figure 5, we present two histograms to compare the solution distributions further. On the left, we show the comparison between the distribution of the latent variable against the true latent variable (unit Gaussian). On the right of the figure, we show the histogram of the decoder $\mathbf{D}_\Delta(1.5, z)$ with $z \sim N(0, 1)$. This represents the one-step conditional distribution $\mathbb{P}(\mathbf{x}_{n+1}|\mathbf{x}_n = 1.5)$. We observe a good agreement with the ground truth.

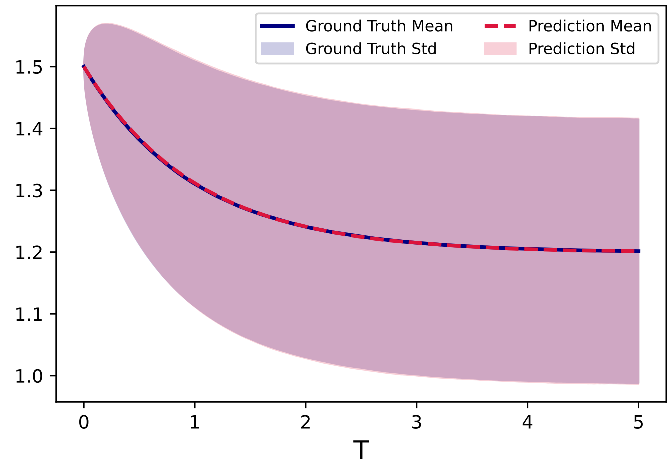

Fig. 3. *OU process (4.1): mean and standard deviation by the learned sFML model against the ground truth.*

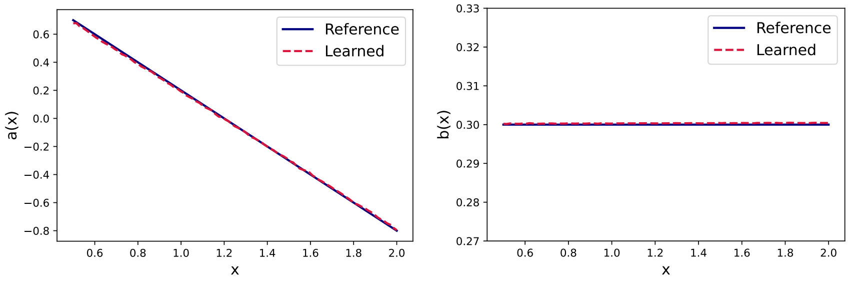

Fig. 4. *OU process (4.1): Learned and reference effective drift and diffusion of the sFML model. Left: drift $a(x) = 1.2 - x$; Right: diffusion $b(x) = 0.3$.*

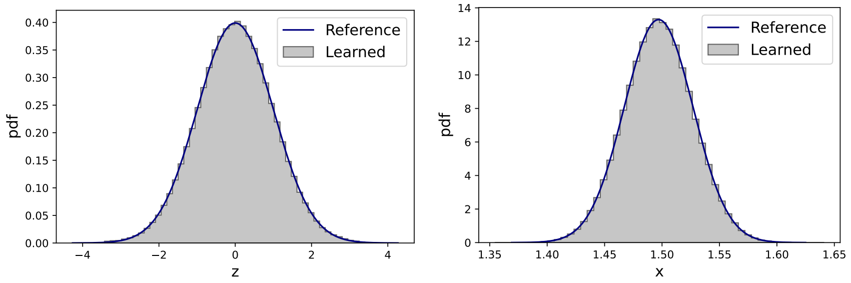

Fig. 5. *OU process* (4.1)*: Left: Distribution of the latent variable from the trained encoder* $\mathbf{E}_\Delta(x_0, x_1)$*,* $x_0 = 1.5$*, against unit Gaussian* $\mathcal{N}(0,1)$*; Right: one-step conditional distribution of the trained decoder* $\mathbf{D}_\Delta(1.5, z)$*,* $z \sim \mathcal{N}(0,1)$*, against the ground truth.*

4.1.2. Geometric Brownian Motion. We now consider a geometric Brownian motion (GBM)

$$dx_t = \mu x_t dt + \sigma x_t dW_t, \tag{4.2}$$

where the parameters are set as $\mu = 2$ and $\sigma = 1$.

For the training data, the initial points are uniformly drawn from the interval $(0, 2)$. For model prediction, we fix the initial condition at $x_0 = 0.5$ and produce $500,000$ prediction trajectories to examine the solution statistics. Due to the exponential growth of this particular geometric Brownian motion over time, the predictive simulations are stopped at $T = 1.0$.

The mean and standard deviation of predictions are shown in Figure 6. The recovered effective drift and diffusion functions by the sFML model are presented in Figure 7. To further validate the learned sFML model, we compare the outputs of the encoder and decoder against their corresponding references in terms of distributions, as displayed in Figure 8. Good agreement with the ground truth can be observed throughout these results.

4.2. Nonlinear SDEs. We now consider two Itô type SDEs with non-linear drift and diffusion functions, along with an SDE with a double well potential for long-term predictions.

4.2.1. SDE with nonlinear diffusion. We first consider an SDE with a nonlinear diffusion,

$$dx_t = -\mu x_t dt + \sigma e^{-x_t^2} dW_t, \tag{4.3}$$

where μ and σ are constants. In this example, we set $\mu = 5$ and $\sigma = 0.5$.

For the training data, the SDE is solved with initial conditions drawn uniformly from $(-1, 1)$. Upon constructing the sFML model, we conduct system prediction up to $T = 5.0$. The evolution of mean and STD of the learned sFML model, with a fixed initial condition $x_0 = -0.4$, are shown in Figure 9. The recovered effective drift and diffusion functions from the learned sFML model are plotted in Figure 10, and the comparisons between the encoder and decoder and their respective references in terms of distributions can be found in Figure 11. Again, good agreements can be observed between the learned sFML model and the ground truth.

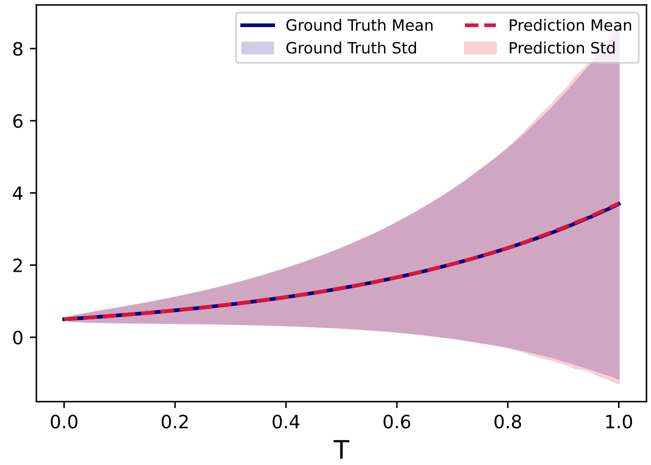

Fig. 6. *Geometric Brownian Motion* (4.2)*: mean and standard deviation by the sFML model.*

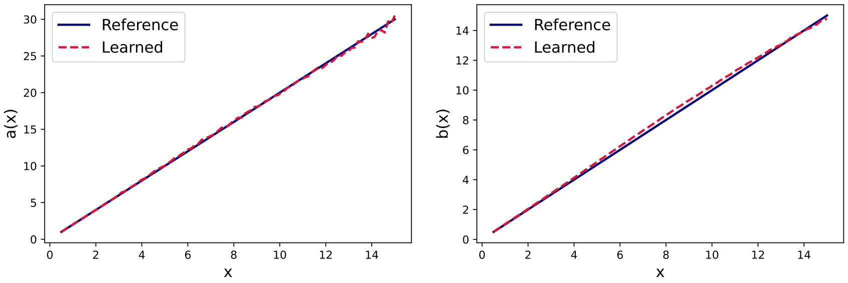

Fig. 7. *Geometric Brownian Motion* (4.2)*: Learned and reference effective drift and diffusion of the sFML model. Left: drift $a(x) = 2x$; Right: diffusion $b(x) = x$.*

4.2.2. Trigonometric SDE. We now consider the following non-linear SDE with trigonometric drift and diffusion,

$$dx_t = \sin(2k\pi x_t)dt + \sigma\cos(2k\pi x_t)dW_t, \tag{4.4}$$

where the constants are set at $k = 1$ and $\sigma = 0.5$.

The training data are generated with initial conditions uniformly distributed as $(0.35, 0.7)$. Upon training the sFML model, we carry out the predictions with an initial condition $x_0 = 0.6$ for time up to $T = 10.0$. The mean and standard deviation of the predictions from the learned sFML model are shown in Figure 12. As illustrated by Figure 13, the recovered effective drift and diffusion functions from the sFML model compare very well with the true drift and diffusion functions. From Figure 14, we also observe excellent agreement between the distributions from the encoder and decoder and the reference solutions. Note that the same example was considered in [7], where accuracy deterioration was observed over the prediction time. Here, we observe that the high accuracy of the sFML model maintains in a more robust manner over the prediction time.

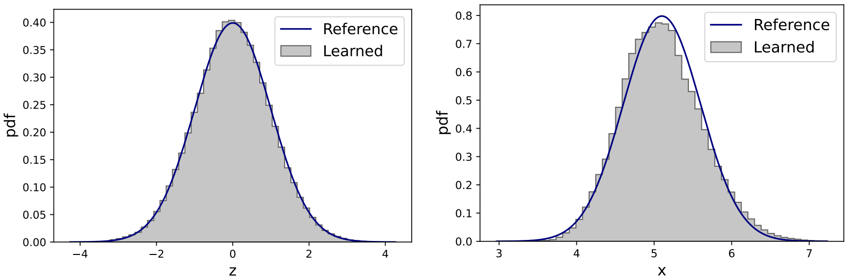

FIG. 8. *Geometric Brownian Motion* (4.2)*: Left: distribution of the latent variable from the trained encoder* $\mathbf{E}_\Delta(x_0, x_1)$*,* $x_0 = 5.0$ *against unit Gaussian; Right: one-step conditional distribution of the trained decoder* $\mathbf{D}_\Delta(5.0, z)$*,* $z \sim \mathcal{N}(0, 1)$*, against the ground truth.*

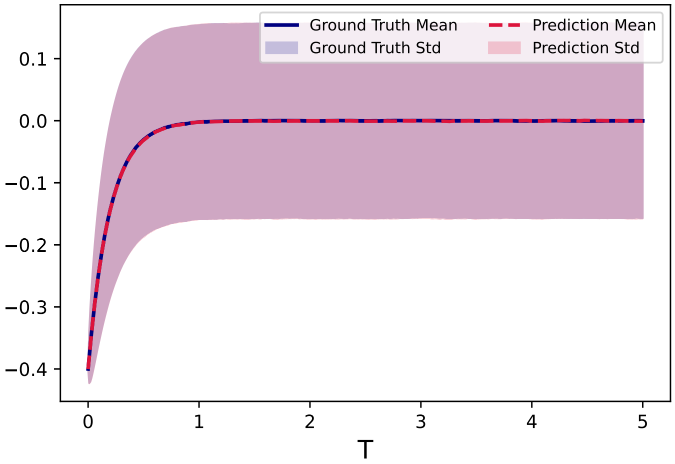

FIG. 9. *SDE with nonlinear diffusion* (4.3)*: mean and standard deviation of the sFML model.*

4.2.3. SDE with Double Well Potential. We consider an SDE with a double well potential:

$$dx_t = (x_t - x_t^3)dt + \sigma dW_t, \tag{4.5}$$

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

The training data are generated by solving the true SDE with initial conditions uniformly sampled in $(-2.5, 2.5)$ up to $T = 1.0$.

Upon constructing the sFML model, we conduct system prediction for up to $T = 500.0$. This is a long-term simulation well beyond the time window of the training data ($T = 1.0$). One sample prediction trajectory with an initial condition $x_0 = 1.5$ is shown in Figure 15, where we clearly observe the random switching between the two stable states $x = \pm 1$. The switching time is on the order of $O(100)$ and well outside the training data time window ($T = 1$). Due to the random transitions between the two stable states, the probability distribution of the solution becomes bi-modal over time

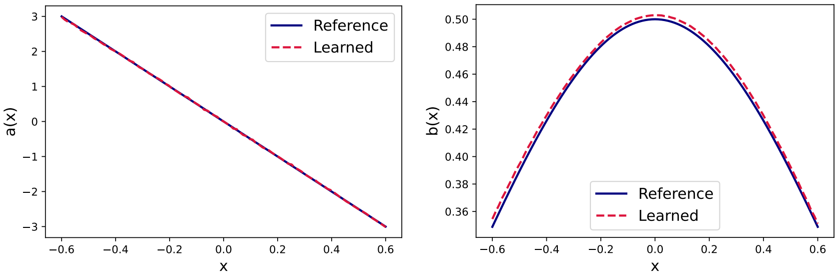

Fig. 10. *SDE with nonlinear diffusion* (4.3)*: Learned and reference effective drift and diffusion of the sFML model. Left: drift* $a(x) = -5x$*; Right: diffusion* $b(x) = 0.5e^{-x^2}$.

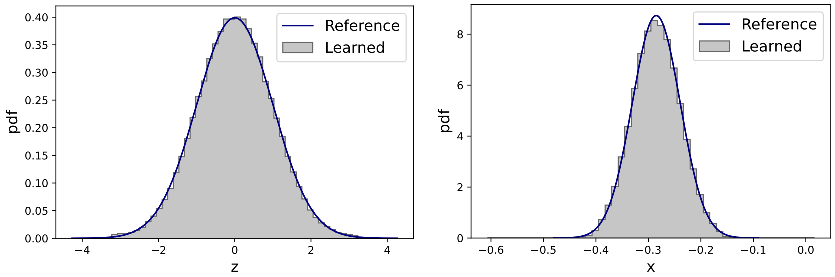

Fig. 11. *SDE with nonlinear diffusion* (4.3)*: Left: distribution of the latent variable from the trained encoder* $\mathbf{E}_\Delta(x_0, x_1)$*,* $x_0 = -0.3$*; Right: one-step conditional distribution of the trained decoder* $\mathbf{D}_\Delta(-0.3, z)$*,* $z \sim \mathcal{N}(0,1)$*, against the ground truth.*

and reaches a stable state asymptotically. To verify this, we conduct ensemble sFML predictions (from the same initial condition $x_0 = 1.5$) of $500,000$ sample realizations to collect the solution statistics. The probability distribution of the solutions by the sFML model can be seen from Figure 16, at time levels $T = 0.5, 10.0, 30.0, 100.0$. We observe excellent agreement between the sFML model predictions and the ground truth, for such a long-term simulation. We also recover the effective drift and diffusion functions of the sFML model and observe their good agreement with the ground truth in Figure 17.

4.3. SDEs with Non-Gaussian Noise. We now present the proposed sFML approach for modeling SDEs driven by non-Gaussian stochastic processes.

4.3.1. Noise with Exponential Distribution. We consider an SDE with exponentially distributed noise,

$$dx_t = \mu x_t dt + \sigma\sqrt{dt}\ \eta_t, \qquad \eta_t \sim \text{Exp}(1), \tag{4.6}$$

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 $\sigma = 0.1$.

The training data are generated by solving the true SDE with initial conditions uniformly sampled in $(0.2, 0.9)$ for up to $T = 1.0$. Upon constructing the sFML model, we report the simulation result with an initial condition $x_0 = 0.34$ for up to $T = 5.0$. The mean and standard deviation of the predictions are shown on the left of Figure 18. The one-step conditional distribution by the sFML model through the trained decoder $\mathbf{D}_\Delta(x, z)$, $z \sim \mathcal{N}(0,1)$, is shown for an arbitrarily chosen location at $x = 0.34$,

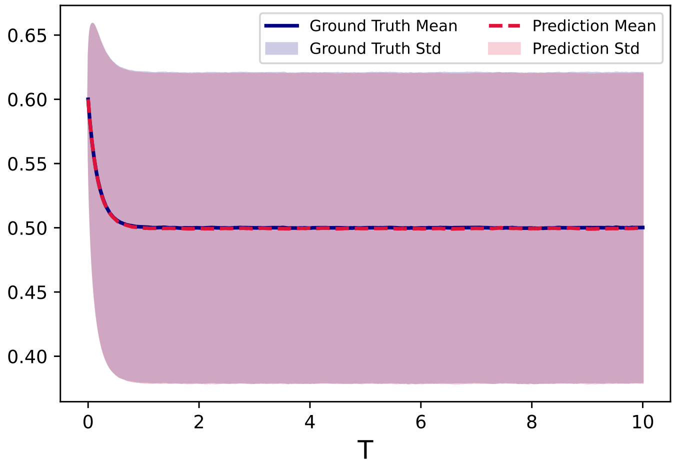

FIG. 12. *Trigonometric SDE (4.4): Mean and standard deviation of the sFML model.*

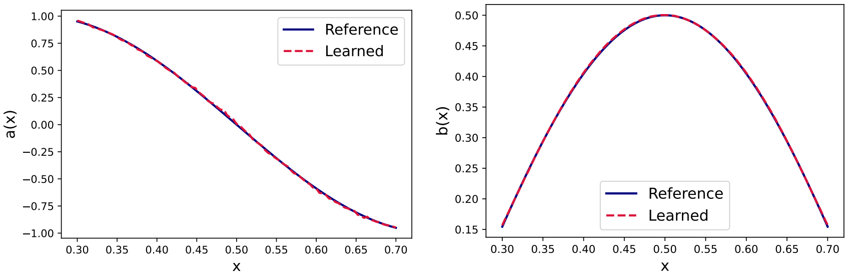

FIG. 13. *Trigonometric SDE (4.4): Learned and reference effective drift and diffusion of the sFML model. Left: drift $a(x) = \sin(2k\pi x)$; Right: diffusion $b(x) = \cos(2k\pi x)$.*

on the right of Figure 18. We observe excellent agreement between the learned sFML model prediction and the reference solution.

The exact drift and diffusion can be computed from the SDE by extracting the non-zero mean of the exponential distribution out of the noise term. We obtain

$$\begin{aligned} a\left(x_n\right) &= \mathbb{E}\left(\frac{x_{n+1}-x_n}{\Delta}\middle| x_n\right) = \mu x_n + \frac{\sigma}{\sqrt{\Delta}}, \\ b\left(x_n\right) &= \operatorname{Std}\left(\frac{x_{n+1}-x_n}{\Delta}\middle| x_n\right) = \sigma. \end{aligned} \tag{4.7}$$

The effective drift and diffusion recovered from the sFML model are shown in Figure 19, where good agreement with the reference solution can be seen.

4.3.2. SDE with lognormally distributed noise. We now consider the following stochastic system

$$d\log x_t = \left(\log m - \theta \log x_t\right) dt + \sigma dW_t, \tag{4.8}$$

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 the Euler-Maruyama

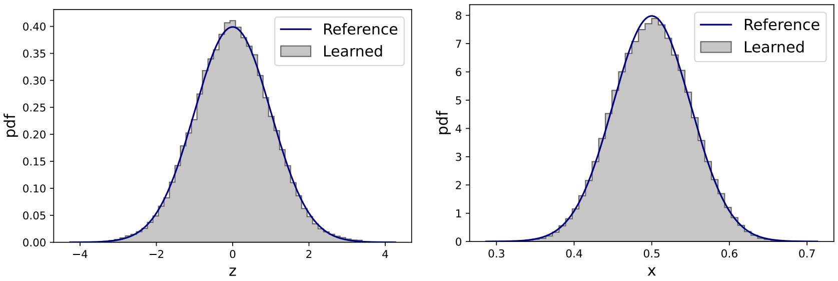

FIG. 14. *Trigonometric SDE* (4.4)*: Left: distribution of the latent variable of the trained encoder* $\mathbf{E}_\Delta(x_0, x_1)$*,* $x_0 = 0.5$ *against unit Gaussian; Right: one-step conditional distribution of the trained decoder* $\mathbf{D}_\Delta(0.5, z)$*,* $z \sim \mathcal{N}(0,1)$*, against the ground truth.*

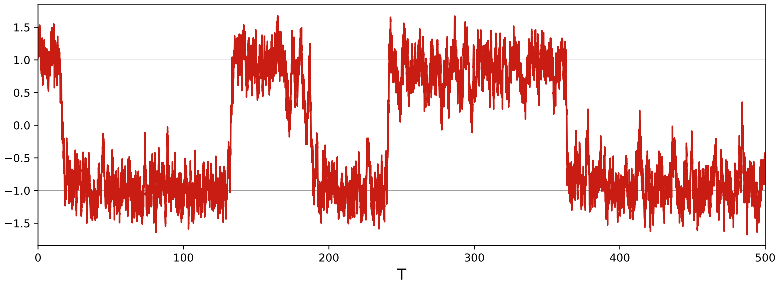

FIG. 15. *SDE with Double Well Potential* (4.5)*: Solution trajectory for time up to* $T = 500$ *from initial condition* $x_0 = 1.5$*, simulated by the learned sFML model. The random switching between the two stable states* $x = \pm 1$ *is clearly visible.*

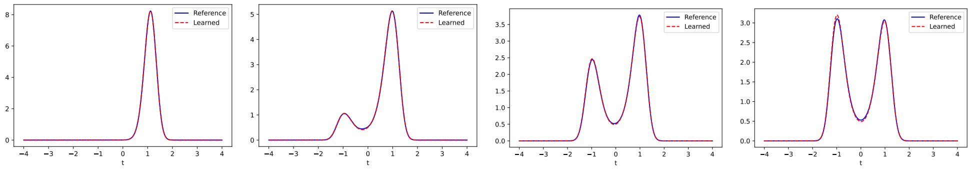

FIG. 16. *SDE with Double Well Potential* (4.5)*: Evolution of the solution PDF of the learned sFML model with an initial condition* $x_0 = 1.5$*, at time* $T = 0.5$*,* 10*,* 30*, and* 100 *(from left to right).*

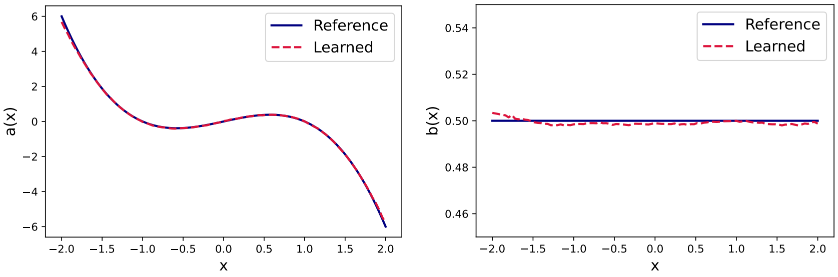

FIG. 17. *SDE with Double Well Potential* (4.5)*: Learned and reference effective drift and diffusion of the sFML model. Left: recovery of the drift* $a(x) = x - x^3$*; Right: recovery of diffusion* $b(x) = 0.5$*.*

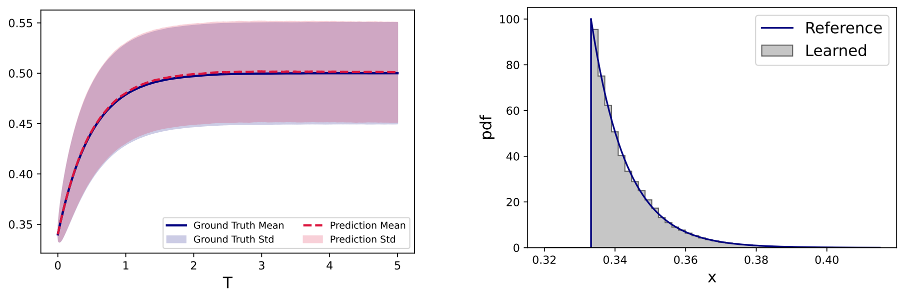

Fig. 18. *SDE with exponentially distributed noise* (4.6)*: Left: mean and standard deviation of the sFML model predictions; Right: one-step conditional probability by the trained decoder* $\mathbf{D}_\Delta(x,z)$*,* $z \sim \mathcal{N}(0,1)$*, against that of the true model at* $x = 0.34$*.*

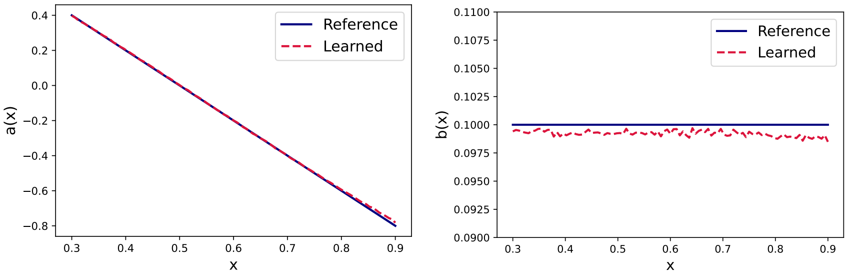

Fig. 19. *SDE with exponentially distributed noise* (4.6)*: Learned and reference effective drift and diffusion of the sFML model. Left: recovery of the drift* $a(x) = \mu x + \sigma/\sqrt{\Delta}$*; Right: recovery of the diffusion* $b(x) = \sigma$*. See* (4.7) *for the reference solution.*

method with the following scheme:

$$x_{n+1} = m^\Delta x_n^{1-\theta\Delta} \eta_n^{\sigma\sqrt{\Delta}}, \qquad \eta_n \sim \text{Lognormal}(0,1). \tag{4.9}$$

Here we set $m = 1/\sqrt{e}$, $\theta = 1.0$, $\sigma = 0.3$, and generate the training data with initial conditions uniformly sampled from $(0.2, 0.9)$ for up to time $T = 1.0$. Upon training the sFML model, we conduct system predictions with the sFML model for time up to $T = 5.0$. The mean and standard deviation from the predictions, as well as the one-step conditional distribution, are shown in Figure 20. Good agreement with the reference solutions from the true SDE can be observed.

Similar to the previous example, we can rewrite this SDE in the form of the classical SDE (2.2) and obtain its true drift and diffusion,

$$\begin{aligned} a(x_n) &= \ln\left[\left(\mathbb{E}\left(\frac{x_{n+1}}{x_n}\middle| x_n\right)\right)^{1/\Delta}\right] = \ln(m x_n^{-\theta}) + \frac{\sigma^2}{2}, \\ b(x_n) &= \text{Std}\left(x_{n+1}|x_n\right) = \sqrt{e^{\sigma^2\Delta} - 1}(m e^{\sigma^2/2})^\Delta (1-\theta\Delta) x_n. \end{aligned} \tag{4.10}$$

On the other hand, from the learned sFML model, we can recover its effective drift and diffusion via

$$\hat{a}(x) = \ln\left[\left(\mathbb{E}_z\left(\frac{\widetilde{\mathbf{G}}_\Delta(x,z)}{x}\right)\right)^{1/\Delta}\right], \qquad \hat{b}(x) = \text{Std}_z(\widetilde{\mathbf{G}}_\Delta(x,z)). \tag{4.11}$$

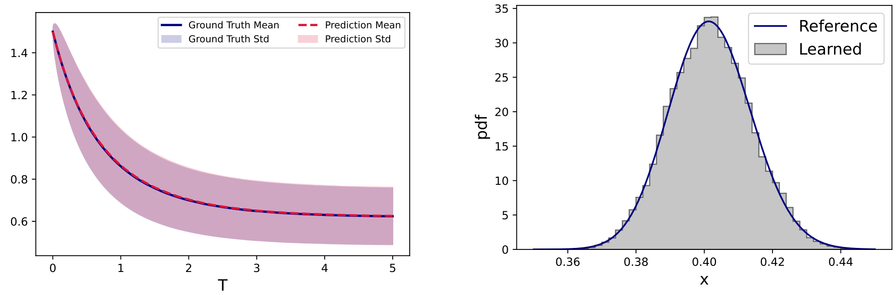

Fig. 20. *SDE with lognormally distributed noise* (4.8)*. Left: mean and standard deviation of the sFML model prediction with the initial condition $x_0 = 1.5$; Right: one-step conditional probability by the trained decoder $\mathbf{D}_\Delta(0.5, z)$, $z \sim \mathcal{N}(0, 1)$, against the true model at $x = 0.4$.*

The learned drift and diffusion are plotted in Figure 21, where we observe excellent agreement with the reference solution.

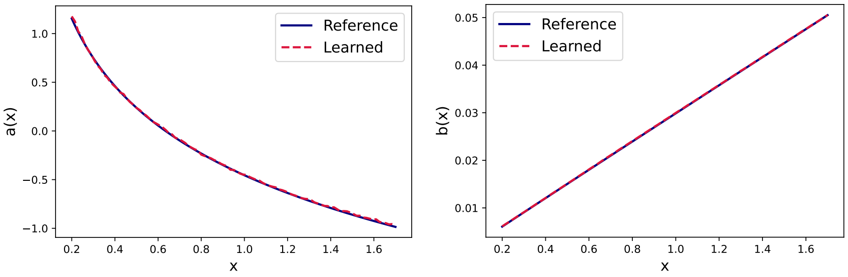

Fig. 21. *SDE with lognormally distributed noise* (4.8)*: Learned and reference effective drift and diffusion of the sFML model. Left: recovery of the drift $a(x)$; Right: recovery of conditional standard deviation $b(x)$; (See* (4.10) *for the reference solution.)*

4.4. Multi-Dimensional Examples.

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

4.4.1. Two-dimensional Ornstein–Uhlenbeck process.

We first consider a two-dimensional OU process

$$d\mathbf{x}_t = \mathbf{B}\mathbf{x}_t dt + \mathbf{\Sigma}\, d\mathbf{W}_t, \tag{4.12}$$

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

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

Our training data are generated with random initial conditions uniformly sampled from $(-4, 4) \times (-3, 3)$, for a time up to $T = 1.0$. For validation, we fix an initial condition of $\mathbf{x}_0 = (0.3, 0.4)$ and evolve the learned sFML model for up to $T = 5$.

The mean and the standard deviation of the sFML model predictions are shown in Figure 22, 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, generated by the learned decoder $\mathbf{D}_\Delta(\mathbf{x}, \mathbf{z})$, $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I}_2)$, at $\mathbf{x} = (0, 0)$. The results are in Figure 23, where good agreement with the true conditional distribution can be seen.

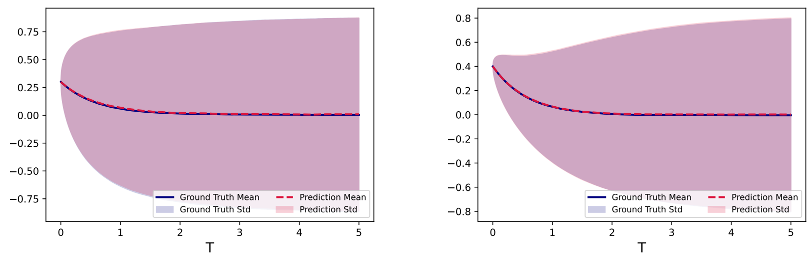

FIG. 22. *2D OU process (4.12): Mmean and standard deviation of the sFML model prediction with the initial condition $x_1 = 0.3, x_2 = 0.4$. Left: x_1; Right: x_2.*

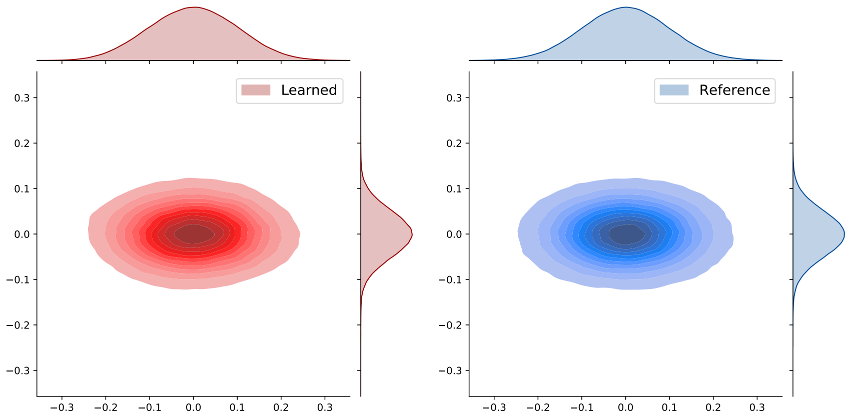

FIG. 23. *2D OU process (4.12): one-step conditional probability distribution from the learned decoder $\mathbf{D}_\Delta$ (left) and the ground truth (right), at $\mathbf{x} = (0, 0)$.*

4.4.2. Five-dimensional Ornstein–Uhlenbeck process. We now consider a 5-dimensional OU process, driven by Wiener processes of varying dimensions

$$d\mathbf{x}_t = \mathbf{B}\mathbf{x}_t dt + \mathbf{\Sigma}\ d\mathbf{W}_t, \tag{4.13}$$

where $\mathbf{x}_t = (x_1, \ldots, x_5) \in \mathbb{R}^5$ are the state variables, $\mathbf{B}$ and $\mathbf{\Sigma}$ are (5×5) are matrices. While the $\mathbf{B}$ matrix is set as

$$\mathbf{B} = \begin{pmatrix} 0.2 & 1.0 & 0.2 & 0.4 & 0.2 \\ -1.0 & 0.0 & 0.2 & 0.8 & -1.0 \\ 0.2 & 0.2 & -0.8 & -1.2 & 0.2 \\ -0.6 & 0.0 & 1.2 & -0.2 & 0.6 \\ 0.2 & 0.2 & 0.6 & 0.4 & 0.0 \end{pmatrix},$$

we choose the following 5 different cases for $\mathbf{\Sigma}$, whose ranks vary from 1 to 5. The objective is to examine the learning capability of the proposed sFML method when the underlying true stochastic dimension is unknown. The 5 cases we choose are:

$$\mathbf{\Sigma_1} = \operatorname{diag}(0,\, 0,\, 1,\, 0,\, 0)$$

$$\mathbf{\Sigma_2} = \operatorname{diag}(0,\, 0.8,\, 0,\, 0,\, -0.8)\,, \quad \mathbf{\Sigma_3} = \begin{pmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{\Sigma_4} = \begin{pmatrix} 0.7 & 0 & -0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0.6 & 0.2 & -0.1 \\ 0 & 0 & 0.1 & -0.6 & 0.2 \\ 0 & 0 & 0 & 0.3 & 0.8 \end{pmatrix}, \mathbf{\Sigma_5} = \begin{pmatrix} 0.8 & 0.2 & 0.1 & -0.3 & 0.1 \\ -0.3 & 0.6 & 0.1 & 0 & -0.1 \\ 0.2 & -0.1 & 0.9 & 0.1 & 0.2 \\ 0.1 & 0.1 & -0.2 & 0.7 & 0 \\ -0.1 & 0.1 & 0.1 & -0.1 & 0.5 \end{pmatrix},$$

where the indices are chosen such that $\operatorname{rank}(\mathbf{\Sigma}_k) = k$, $k = 1, \ldots, 5$.

To generate the training data, we use random initial conditions drawn uniformly from the hypercube $(-4, 4)^5$ and solve the system up to $T = 1.0$. For the test data, we set the initial condition to $\mathbf{x}_0 = (0.3, -0.2, -1.7, 2.5, 1.4)$ and evolve the learned sFML for up to $T = 5$.

To construct the sFML model, a key parameter is the dimension n_z of the latent variable $\mathbf{z}$. It needs to match the true dimension of the random process driven by the unknown stochastic system. Since the true random dimension is unknown, we propose to construct a sequence of sFML models, starting with a small random dimension n_z, for example, $n_z = 1$. We then train more sFML models with progressively increasing values of n_z. During this process, we monitor the MSE losses (3.9) of the models. The results of this procedure are tabulated in Table 1. We observe that when the latent variable dimension n_z matches the true random dimension, there is a significant drop of two to three orders in magnitude in the MSE losses. In practice, this can be used as a guideline to determine the proper dimension of the latent variable $\mathbf{z}$.

n_z	$\mathbf{\Sigma}_1$	$\mathbf{\Sigma}_2$	$\mathbf{\Sigma}_3$	$\mathbf{\Sigma}_4$	$\mathbf{\Sigma}_5$
1	3.5×10^{-7}	1.1×10^{-3}	1.8×10^{-3}	2.3×10^{-3}	3.3×10^{-3}
2	5.1×10^{-7}	6.4×10^{-7}	7.2×10^{-4}	1.4×10^{-3}	1.9×10^{-3}
3	3.3×10^{-7}	5.0×10^{-7}	7.1×10^{-7}	5.3×10^{-4}	8.8×10^{-4}
4	3.8×10^{-7}	5.4×10^{-7}	4.1×10^{-7}	7.2×10^{-7}	4.5×10^{-4}
5	7.2×10^{-7}	9.4×10^{-7}	6.3×10^{-7}	6.1×10^{-7}	6.2×10^{-7}

TABLE 1

5D OU process (4.13): The MSE loss (3.9) of the sFML models with latent variable dimension $n_z = 1, \ldots, 5$ with varying stochastic dimensions $k = 1, \ldots, 5$, rank$(\mathbf{\Sigma}_k) = k$. Note the significant drops of the MSE losses from $n_z < k$ to $n_z \geq k$.

The learned sFML model predictions of the solution mean and standard deviation of each component are shown in Figure 24, for all five cases by using the matched latent variable dimension n_z. We observe excellent agreements with the reference solutions.

For the one-step conditional probability distribution, we plot its marginal distributions for each of the components, $x_1, \ldots, x_5$, obtained by the decoder of the learned sFML model, in Figure 25 (from left column to right), for each of the five

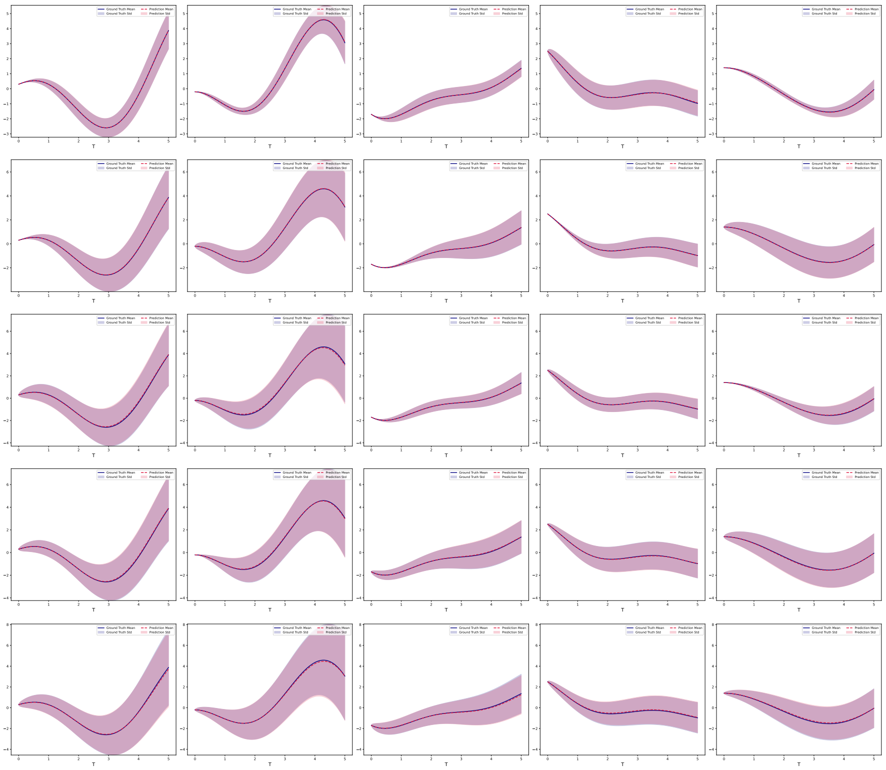

FIG. 24. *5D OU process (4.13): Mean and standard deviation of the sFML model prediction for $x_1, \dots, x_5$ (left column to right), for the five different cases $\mathbf{\Sigma}_1, \dots, \mathbf{\Sigma}_5$ (top row to bottom).*

cases $\mathbf{\Sigma}_1, \dots, \mathbf{\Sigma}_5$ (from top row to bottom). We observe excellent agreement with the reference solutions. For the k-th case, $k = 1, \dots, 5$, since $\text{rank}(\mathbf{\Sigma}_k) = k$, the true random dimension is k. Consequently, there are $(5 - k)$ non-random, i.e., deterministic, components. We clearly observe this in the marginal distributions, where the deterministic components exhibit Delta function distribution. We observe that the learned sFML models accurately reflect this and produce the correct Delta function distributions in each of the five cases.

5. Conclusion. In this paper, we presented a numerical method for modeling unknown stochastic dynamical systems using trajectory data. The method is based on learning the underlying stochastic flow map, which is parameterized by Gaussian latent variables via an autoencoder architecture. The encoding function recovers the unobserved Gaussian random variables and the decoding function reconstructs the trajectory data. Loss functions are carefully designed to ensure the autoencoder sFML model accurately learns the stochastic components of the unknown system. By using a comprehensive set of numerical examples, we demonstrate that the proposed sFML model is highly effective and able to produce long-term system predictions well beyond the time horizon covered by the training data sets. The method can also automatically identify the true stochastic dimension of the unknown system. This aspect is important for practical applications and will be studied more rigorously in future work, beyond the academic example investigated in this paper.

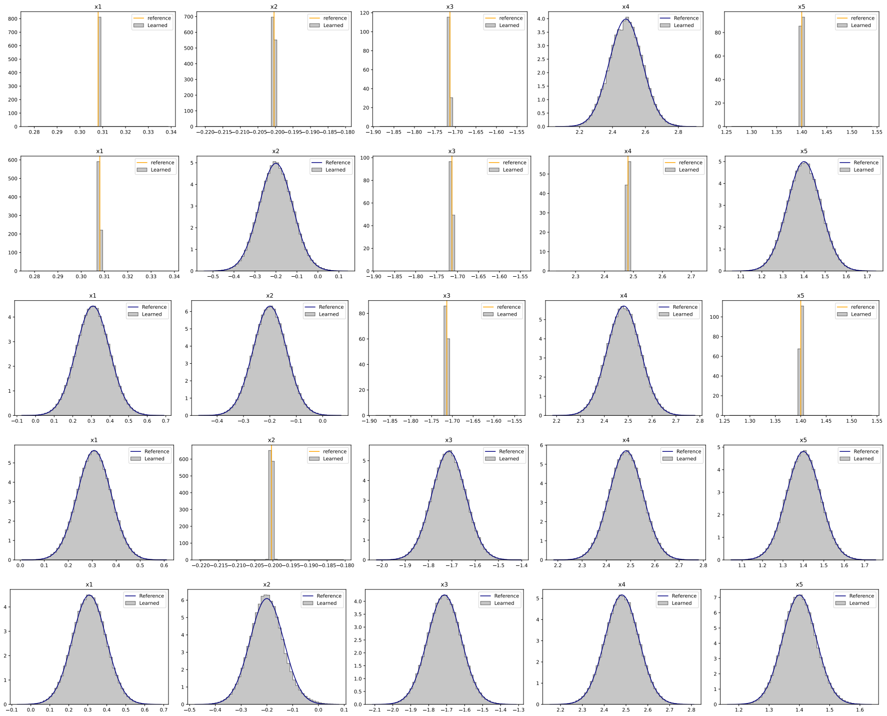

Fig. 25. *5D OU process* (4.13)*: One-step conditional probability distribution comparison: marginal distribution for* $x_1, \ldots, x_5$ *(left column to right), for the five different cases* $\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_5$ *(top row to bottom). Note the deterministic components with the Delta function distributions in each case.*

REFERENCES

[1] N. H. Anderson, P. Hall, and D. M. Titterington, *Two-sample test statistics for measuring discrepancies between two multivariate probability density functions using kernel-based density estimates*, J. Multivariate Anal., 50 (1994), pp. 41–54, https://doi.org/10.1006/jmva.1994.1033.

[2] C. Archambeau, D. Cornford, M. Opper, and J. Shawe-Taylor, *Gaussian process approximations of stochastic differential equations*, in Gaussian Processes in Practice, N. D. Lawrence, A. Schwaighofer, and J. Quiñonero Candela, eds., vol. 1 of Proceedings of Machine Learning Research, Bletchley Park, UK, 12–13 Jun 2007, PMLR, pp. 1–16, https://proceedings.mlr.press/v1/archambeau07a.html.

[3] S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proc. Natl. Acad. Sci. USA, 113 (2016), pp. 3932–3937, https://doi.org/10.1073/pnas.1517384113.

[4] A. Buades, B. Coll, and J. M. Morel, *A review of image denoising algorithms, with a new one*, Multiscale Model. Simul., 4 (2005), pp. 490–530, https://doi.org/10.1137/040616024.

[5] X. Chen, J. Duan, J. Hu, and D. Li, *Data-driven method to learn the most probable transition pathway and stochastic differential equation*, Phys. D, 443 (2023), pp. Paper No. 133559, 15, https://doi.org/10.1016/j.physd.2022.133559.

[6] X. Chen, L. Yang, J. Duan, and 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), pp. B811–B830, https://doi.org/10.1137/20M1360153.

[7] Y. Chen and D. Xiu, *Learning stochastic dynamical system via flow map operator*, arXiv preprint arXiv:2305.03874, (2023).

[8] V. Churchill and D. Xiu, *Flow map learning for unknown dynamical systems: Overview, implementation, and benchmarks*, Journal of Machine Learning for Modeling and Computing,

4 (2023), pp. 173–201.

[9] M. Darcy, B. Hamzi, G. Livieri, H. Owhadi, and P. Tavallali, *One-shot learning of stochastic differential equations with data adapted kernels*, Phys. D, 444 (2023), pp. Paper No. 133583, 18, https://doi.org/10.1016/j.physd.2022.133583.

[10] K. Dehnad, *Density estimation for statistics and data analysis*, Taylor & Francis, 1987.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[12] A. Hasan, J. M. Pereira, S. Farsiu, and V. Tarokh, *Identifying latent stochastic differential equations*, IEEE Transactions on Signal Processing, 70 (2022), pp. 89–104, https://doi.org/10.1109/TSP.2021.3131723.

[13] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science, 313 (2006), pp. 504–507, https://doi.org/10.1126/science.1127647.

[14] S. H. Kang, W. Liao, and Y. Liu, *IDENT: identifying differential equations with numerical time evolution*, J. Sci. Comput., 87 (2021), pp. Paper No. 1, 27, https://doi.org/10.1007/s10915-020-01404-9.

[15] Y. Li and J. Duan, *A data-driven approach for discovering stochastic dynamical systems with non-Gaussian Lévy noise*, Phys. D, 417 (2021), pp. Paper No. 132830, 12, https://doi.org/10.1016/j.physd.2020.132830.

[16] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, *Fourier neural operator for parametric partial differential equations*, in International Conference on Learning Representations, 2021, https://openreview.net/forum?id=c8P9NQVtmnO.

[17] M. Liu, D. Grana, and L. P. de Figueiredo, *Uncertainty quantification in stochastic inversion with dimensionality reduction using variational autoencoder*, GEOPHYSICS, 87 (2022), pp. M43–M58, https://doi.org/10.1190/geo2021-0138.1.

[18] B. Øksendal, *Stochastic differential equations*, in Stochastic differential equations, Springer, 2003, pp. 65–84.

[19] V. Oommen, K. Shukla, S. Goswami, R. Dingreville, and G. E. Karniadakis, *Learning two-phase microstructure evolution using neural operators and autoencoder architectures*, npj Computational Materials, 8 (2022), p. 190.

[20] M. Opper, *Variational inference for stochastic differential equations*, Ann. Phys., 531 (2019), pp. 1800233, 9, https://doi.org/10.1002/andp.201800233.

[21] H. Owhadi, *Computational graph completion*, Research in the Mathematical Sciences, 9 (2022), p. 27.

[22] T. Qin, K. Wu, and D. Xiu, *Data driven governing equations approximation using deep neural networks*, J. Comput. Phys., 395 (2019), pp. 620–635, https://doi.org/10.1016/j.jcp.2019.06.042.

[23] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707, https://doi.org/10.1016/j.jcp.2018.10.045.

[24] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Multistep neural networks for data-driven discovery of nonlinear dynamical systems*, arXiv preprint arXiv:1801.01236, (2018).

[25] H. Schaeffer and S. G. McCalla, *Sparse model selection via integral terms*, Phys. Rev. E, 96 (2017), pp. 023302, 7, https://doi.org/10.1103/physreve.96.023302.

[26] H. Schaeffer, G. Tran, and R. Ward, *Extracting sparse high-dimensional dynamics from limited data*, SIAM J. Appl. Math., 78 (2018), pp. 3279–3295, https://doi.org/10.1137/18M116798X.

[27] L. Theis, W. Shi, A. Cunningham, and F. Huszár, *Lossy image compression with compressive autoencoders*, arXiv preprint arXiv:1703.00395, (2017).

[28] Y. Wang, H. Fang, J. Jin, G. Ma, X. He, X. Dai, Z. Yue, C. Cheng, H.-T. Zhang, D. Pu, D. Wu, Y. Yuan, J. Gonçalves, J. Kurths, and H. Ding, *Data-driven discovery of stochastic differential equations*, Engineering, 17 (2022), pp. 244–252, https://doi.org/https://doi.org/10.1016/j.eng.2022.02.007.

[29] L. Yang, C. Daskalakis, and G. E. Karniadakis, *Generative ensemble regression: Learning particle dynamics from observations of ensembles with physics-informed deep generative models*, SIAM Journal on Scientific Computing, 44 (2022), pp. B80–B99, https://doi.org/10.1137/21M1413018.

[30] C. Yildiz, M. Heinonen, J. Intosalmi, H. Mannerstrom, and 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.

[31] J. Zhang, S. Zhang, and G. Lin, *MultiAuto-DeepONet: A multi-resolution autoencoder Deep-*

ONet for nonlinear dimension reduction, uncertainty quantification and operator learning of forward and inverse stochastic problems, arXiv preprint arXiv:2204.03193, (2022).

[32] W. Zhong and H. Meidani, *Pi-vae: Physics-informed variational auto-encoder for stochastic differential equations*, Computer Methods in Applied Mechanics and Engineering, 403 (2023), p. 115664, https://doi.org/10.1016/j.cma.2022.115664.