Learning Stochastic Dynamical System via Flow Map

Yuan Chen   Dongbin Xiu

The Ohio State University

July 26, 2023

USNCCM 17

Outline

Introduction

Setting and Objectives

Setting: consider a stochastic dynamical system

$$\frac{d\mathbf{x}_t}{dt}(\omega) = \mathbf{f}(\mathbf{x}_t, \omega), \qquad \mathbf{x}_0(\omega) \sim \mathbb{P}_0, \tag{1}$$

where ω is the random sample, $\mathbf{f}$ is **unknown**.

Data: observed solution trajectory data of (1):

$$\mathbf{X}^{(i)} = \left(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}, ..., \mathbf{x}_L^{(i)}\right), \qquad i = 1, \dots, N_T, \tag{2}$$

where $\mathbf{x}_k^{(i)} = \mathbf{x}(t_k^{(i)})$, with $t_0^{(i)} \leq t_1^{(i)} \leq ... \leq t_L^{(i)}$, for each i.

Goal: construct numerical predictions $\widetilde{\mathbf{x}}_n$:

$$\widetilde{\mathbf{x}}_n \overset{d}{\approx} \mathbf{x}(t_n), \qquad \text{for any new } \mathbf{x}_0. \tag{3}$$

- $\widetilde{\mathbf{x}}_n$: prediction of numerical model;
- $\mathbf{x}(t_n)$: true dynamics.

Setting and Objectives

The problem we are considering is a generalized form of:

- Itô SDE
$$\mathbf{f}(\mathbf{x}_t, \omega) = \mathbf{a}(\mathbf{x}_t) + \mathbf{b}(\mathbf{x}_t)\dot{\mathbf{W}}_t(\omega). \tag{4}$$
- "Itô SDE" with non-Gaussian noise
$$\mathbf{f}(\mathbf{x}_t, \omega) = \mathbf{a}(\mathbf{x}_t) + \mathbf{b}(\mathbf{x}_t)\boldsymbol{\eta}(\omega), \tag{5}$$
where $\boldsymbol{\eta}$ can be scaled non-Gaussian random noise.
- $\mathbf{f}$ to be other composite forms for $\mathbf{x}_t$ and ω
- Discrete stochastic dynamics, time series data, etc

Methodology

Stochastic Flow Map

Assumption: time-homogeneous ([Øksendal, 2003]): for any $\Delta \geq 0$,

$$\mathbb{P}(\mathbf{x}_{s+\Delta} | \mathbf{x}_s) = \mathbb{P}(\mathbf{x}_\Delta | \mathbf{x}_0), \qquad s \geq 0. \tag{6}$$

We call the Δ-shift map from $\mathbf{x}_s$ to $\mathbf{x}_{s+\Delta}$ **"stochastic flow map" $\mathbf{G}_\Delta$**.

$$\mathbf{x}_{s+\Delta} = \mathbf{G}_\Delta(\mathbf{x}_s) \tag{7}$$

Goal: Construct a numerical model $\widetilde{\mathbf{G}}_\Delta(\mathbf{x})$:

$$\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) \overset{d}{\approx} \mathbf{G}_\Delta(\mathbf{x}). \tag{8}$$

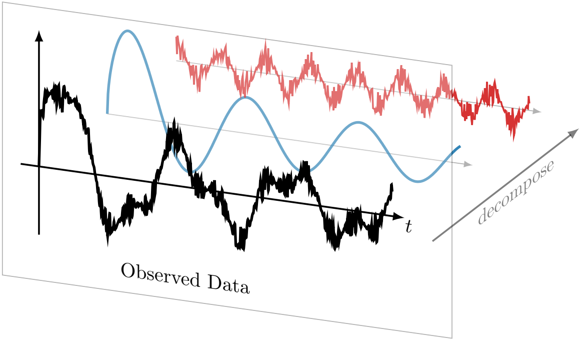
So $\widetilde{\mathbf{x}}_{n+1} = \widetilde{\mathbf{G}}_\Delta(\widetilde{\mathbf{x}}_n)$.

General Framework

The evolution of $\mathbf{x}_t$ can be separated into:

$$\mathbf{x}_{n+1} = \mathbf{G}_\Delta(\mathbf{x}_n) = \mathbb{E}\left[\mathbf{x}_{n+1}|\mathbf{x}_n\right] + \mathbf{x}'_{n+1} := \mathbf{D}_\Delta(\mathbf{x}_n) + \mathbf{S}_\Delta(\mathbf{x}_n), \quad (9)$$

where we call $\mathbf{D}_\Delta$ "deterministic sub-map" and $\mathbf{S}_\Delta$ "stochastic sub-map".



General Framework

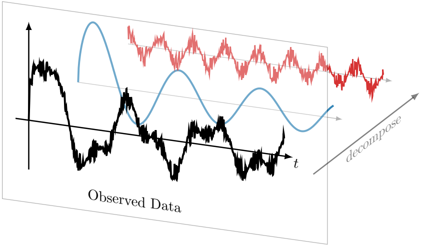
We propose to construct a numerical stochastic flow map model:

$$\widetilde{\mathbf{G}}_\Delta = \widetilde{\mathbf{D}}_\Delta + \widetilde{\mathbf{S}}_\Delta, \tag{10}$$

such that:

$$\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) \overset{d}{\approx} \mathbf{G}_\Delta(\mathbf{x}). \tag{11}$$

In our work, we choose Deep Neural Network (DNN) to construct the approximations $\widetilde{\mathbf{D}}_\Delta$ and $\widetilde{\mathbf{S}}_\Delta$.



Deterministic Sub-map

Setup: we are given N trajectory data with length L:

$$\mathbf{X}^{(i)} = \left(\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}, ..., \mathbf{x}_L^{(i)}\right), \qquad i = 1, \ldots, N, \tag{12}$$

Goal: find $\widetilde{\mathbf{D}}_\Delta : \mathbb{R}^d \mapsto \mathbb{R}^d$, such that $\widetilde{\mathbf{D}}_\Delta(\mathbf{x}_n) \approx \mathbf{D}_\Delta(\mathbf{x}_n) = \mathbb{E}\left[\mathbf{x}_{n+1}|\mathbf{x}_n\right]$.

We employ the deterministic flow map learning ([Qin et al., 2019]): let

$$\widetilde{\mathbf{D}}_\Delta(\cdot; \Theta) := \mathbf{I} + \mathbf{N}_\Delta(\cdot; \Theta), \tag{13}$$

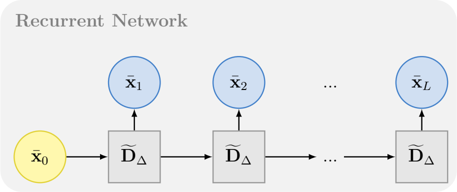
- $\mathbf{I}$: the identity operator;
- $\mathbf{N}_\Delta(\cdot; \Theta)$: mapping operator of a connected feedforward DNN with its hyperparameter set Θ.

Deterministic Sub-map

We could use the above-defined numerical operator to recurrently generate:

$$\overline{\mathbf{X}}^{(i)} = \left(\bar{\mathbf{x}}_0^{(i)}, \bar{\mathbf{x}}_1^{(i)}, ..., \bar{\mathbf{x}}_L^{(i)}\right), \qquad i = 1, \ldots, N. \tag{14}$$

where $\bar{\mathbf{x}}_0^{(i)} = \mathbf{x}_0^{(i)}$, and $\bar{\mathbf{x}}_{n+1}^{(i)} = \widetilde{\mathbf{D}}_\Delta\left(\bar{\mathbf{x}}_n^{(i)}; \Theta\right)$, $n = 0, \ldots, L-1$ recurrently.



Then we train the network by:

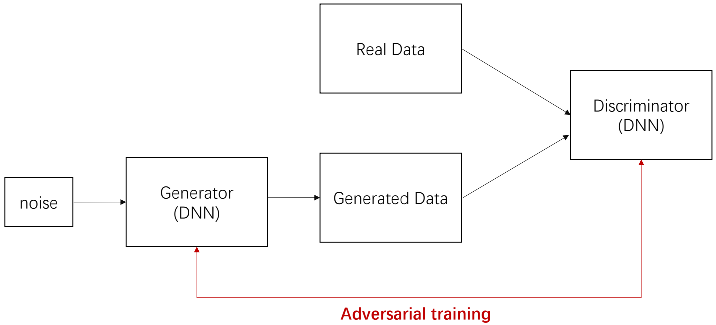
$$\Theta_{\widetilde{\mathbf{D}}} = \arg\min \sum_{i=1}^{N} \left\| \overline{\mathbf{X}}^{(i)} - \mathbf{X}^{(i)} \right\|_F^2. \tag{15}$$

Stochastic Sub-map

Recall: $\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) = \widetilde{\mathbf{D}}_\Delta(\mathbf{x}) + \widetilde{\mathbf{S}}_\Delta(\mathbf{x})$ with $\widetilde{\mathbf{D}}_\Delta$ pre-trained.

Goal: construct a stochastic model $\widetilde{\mathbf{S}}_\Delta$, such that $\widetilde{\mathbf{G}}_\Delta(\mathbf{x}) \overset{d}{\approx} \mathbf{G}_\Delta(\mathbf{x})$.

In our work, we employ Wasserstein Generative Adversarial Networks (WGANs) [Arjovsky and Bottou, 2017] for distribution matching.



Stochastic Sub-map

Let DNN $\widetilde{\mathcal{S}}_\Delta(\mathbf{x}, \mathbf{z}; \Theta_{\widetilde{\mathbf{S}}}) : \mathbb{R}^d \times \mathbb{R}^{n_\mathbf{z}} \mapsto \mathbb{R}^d$ to approximate $\widetilde{\mathbf{S}}_\Delta$.

- **Generator**: $\widetilde{\mathcal{G}}_\Delta(\mathbf{x}, \mathbf{z}) = \widetilde{\mathbf{D}}_\Delta(\mathbf{x}) + \widetilde{\mathcal{S}}_\Delta(\mathbf{x}, \mathbf{z})$;
- **Discriminator**: $\mathcal{C}$, A DNN with parameter $\Theta_\mathcal{C}$;
- The increment **real** data:

$$\mathbf{Y}^{(i)} = \left(\mathbf{x}_0^{(i)}, \mathbf{y}_1^{(i)}, ..., \mathbf{y}_L^{(i)}\right), \qquad i = 1, \ldots, N, \tag{16}$$

  where $\mathbf{y}_k^{(i)} = \mathbf{x}_k^{(i)} - \mathbf{x}_{k-1}^{(i)}$, $k = 1, 2, ..., L$.
- The increment **fake** data:

$$\widehat{\mathbf{Y}}^{(i)} = \left(\hat{\mathbf{x}}_0^{(i)}, \hat{\mathbf{y}}_1^{(i)}, ..., \hat{\mathbf{y}}_L^{(i)}\right) = \left(\hat{\mathbf{x}}_0^{(i)}, \underline{\hat{\mathbf{x}}_1^{(i)} - \hat{\mathbf{x}}_0^{(i)}}, ..., \underline{\hat{\mathbf{x}}_L^{(i)} - \hat{\mathbf{x}}_{L-1}^{(i)}}\right), \tag{17}$$

  where $\hat{\mathbf{x}}_0^{(i)} = \mathbf{x}_0^{(i)}$, $\hat{\mathbf{x}}_{n+1}^{(i)} = \widetilde{\mathcal{G}}_\Delta\left(\hat{\mathbf{x}}_n^{(i)}, \mathbf{z}_n^{(i)}\right)$, $n = 0, \ldots, L-1$.

Stochastic Sub-map

Then we train the network by:

$$\min_{\widetilde{\mathbf{S}}} \max_{\mathcal{C}} \; \mathbb{E}[\mathcal{C}(\widehat{\mathbf{Y}})] - \mathbb{E}[\mathcal{C}(\mathbf{Y})] + \lambda \mathop{\mathbb{E}}_{\widetilde{\boldsymbol{Y}} \sim \mathbb{P}_{\widetilde{\boldsymbol{Y}}}} \left[\left(\left\| \nabla_{\widetilde{\boldsymbol{Y}}} \mathcal{C}(\widetilde{\boldsymbol{Y}}) \right\|_2 - 1 \right)^2 \right], \tag{18}$$

- $\mathbb{P}_{\widetilde{\boldsymbol{Y}}}$: uniformly distributed along straight lines between pairs of points sampled from $\widehat{\mathbf{Y}}$ and $\mathbf{Y}$;
- λ: non-negative penalty constant.

As indicated by the theory of WGANs [Arjovsky et al., 2017], the above minimization would minimize the Wasserstein-1 distance between the distribution of $\widehat{\mathbf{Y}}$ and $\mathbf{Y}$.

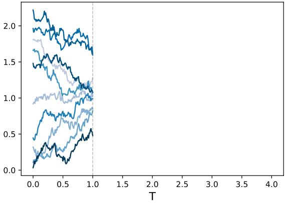
Numerical Examples

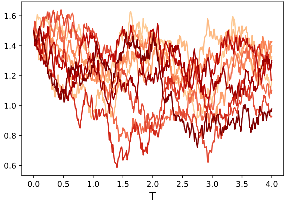
Ornstein–Uhlenbeck process

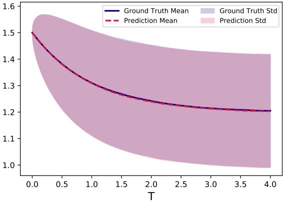
We consider the 1D OU process, in the following form

$$dx_t = \theta(\mu - x_t)dt + \sigma dW_t, \tag{19}$$

The training data, simulated test data and "mean-std vs time" plot for $\theta = 1.0$, $\mu = 1.2$, and $\sigma = 0.3$:



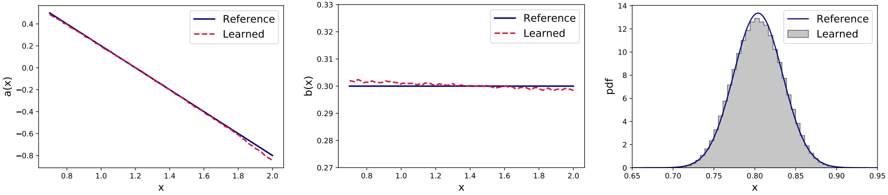




To examine the accuracy of sFM learning, we propose the following estimator:

$$\hat{a}(x) = \frac{\mathbb{E}_z(\widetilde{\mathcal{G}}_\Delta(x,z) - x)}{\Delta}, \quad \hat{b}(x) = \frac{\mathrm{Std}_z(\widetilde{\mathcal{G}}_\Delta(x,z))}{\sqrt{\Delta}} \tag{20}$$

Note that $\hat{a}(x)$ and $\hat{b}(x)$ are Monte Carlo estimators of drift and diffusion $a(x)$ and $b(x)$. The comparison and illustration of sFM $\mathbf{G}_\Delta(0.8)$:

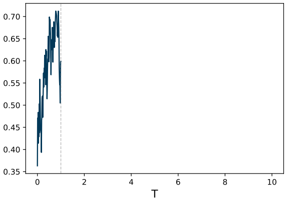


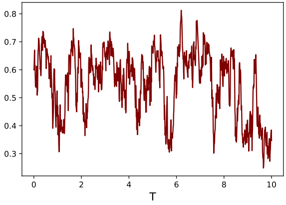
Nonlinear Stochastic Dynamics

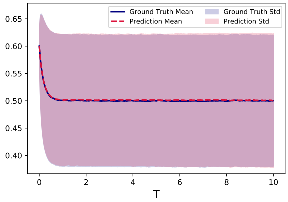
We consider the following nonlinear stochastic dynamical system inspired by [Darcy et al., 2022]

$$dx_t = \sin(2k\pi x_t)dt + \sigma \cos(2k\pi x_t)dW_t, \tag{21}$$

The training data, simulated test data and "mean-std vs time" plot for $k = 1$ and $\sigma = 0.5$:

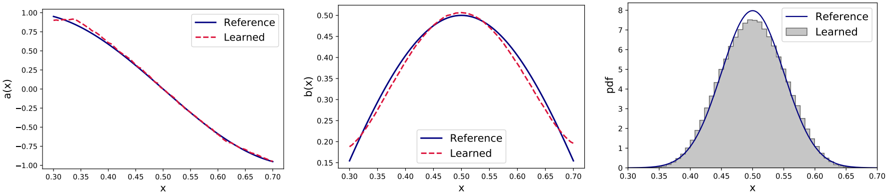






Nonlinear Stochastic Dynamics

The comparison of sFMs:

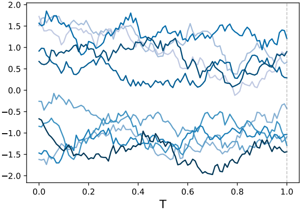


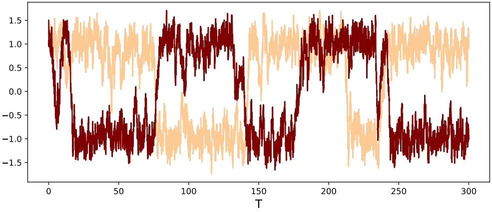
Double Well Potential

We consider the classical double well potential system, in the following form

$$dx_t = (x_t - x_t^3)dt + \sigma dW_t, \tag{22}$$

where we take $\sigma = 0.5$. The training data, simulated test data plots:

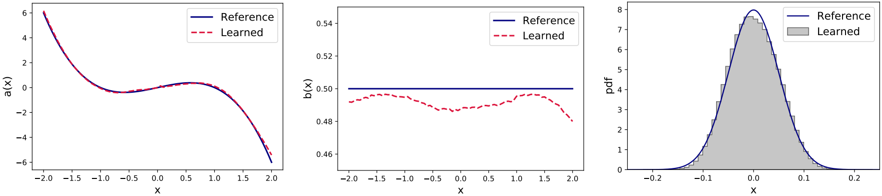




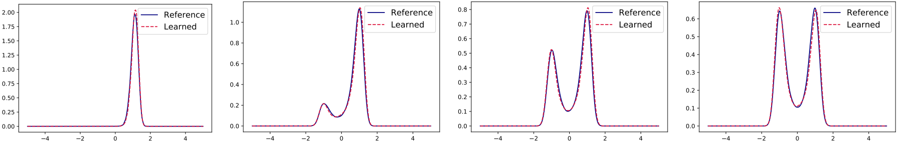
Videos on this site.

Double Well Potential

Estimation of drift, diffusion functions and $\mathbf{G}_{\Delta}(0)$:



The estimated density functions at time point $T = 0.5$, 10.0, 30.0 and 100.0:

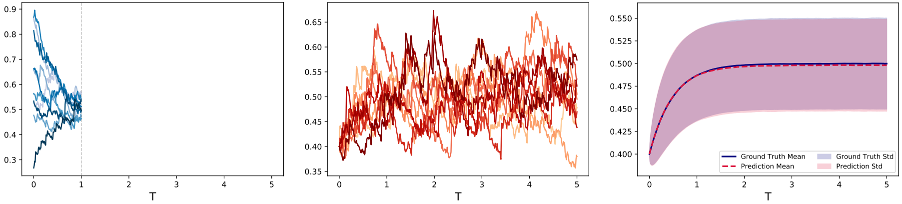


Non-Gaussian Driven Dynamics

We consider the following non-Gaussian driven SDE

$$dx_t = \mu x_t dt + \sigma\sqrt{dt}\eta, \quad \eta \sim \text{Exp}(1), \tag{23}$$

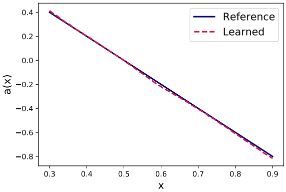
where we have density function of η to be $f_\eta(x) = e^{-x}$, $x \geq 0$ and take $\mu = -2.0$, $\sigma = 0.1$. The training data, simulated test data and "mean-std vs time" plot:

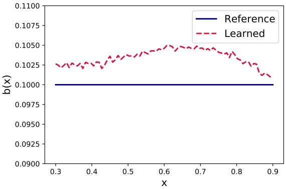


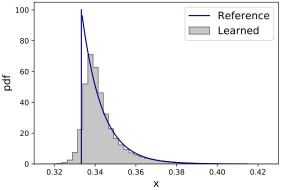
To measure the accuracy of our method, we define the scaled growth and deviation functions $a(x_n)$ and $b(x_n)$ as:

$$
\begin{aligned}
a(x_n) &= \mathbb{E}_z\left(\frac{x_{n+1}-x_n}{\Delta}\middle| x_n\right) = \mu x_n + \frac{\sigma}{\sqrt{\Delta}}, \\
b(x_n) &= \mathrm{Std}_z\left(\frac{x_{n+1}-x_n}{\Delta}\middle| x_n\right) = \sigma.
\end{aligned}
\tag{24}
$$

We have the following results for a, b and $\mathbf{G}_\Delta(0.34)$:





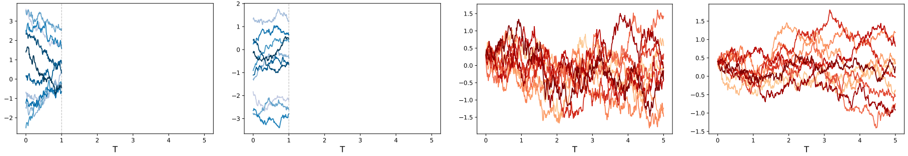


Two-Dimensional OU Process

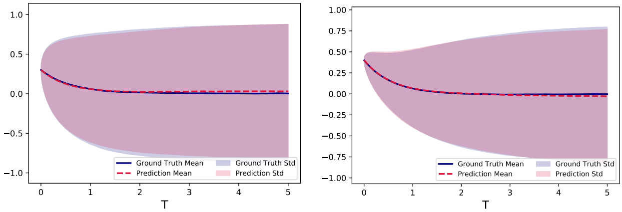
We consider a 2D OU process, in the following form

$$d\mathbf{x}_t = \begin{pmatrix} -1 & -0.5 \\ -1 & -1 \end{pmatrix} \mathbf{x}_t dt + \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix} d\mathbf{W}_t, \tag{25}$$

The training data, and simulated test data:

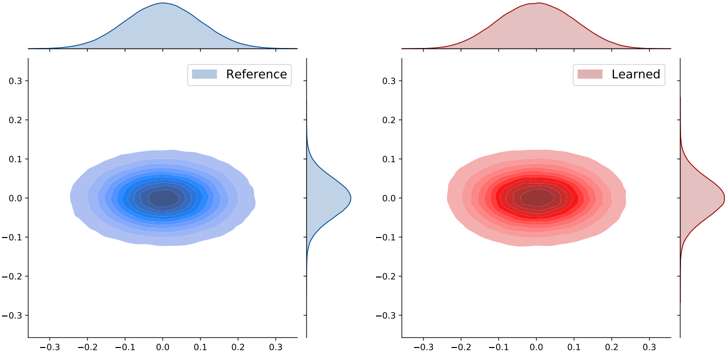


The "mean-std vs time" plot:



Two-Dimensional OU Process

The comparison of $\mathbf{G}_\Delta(0,0)$:

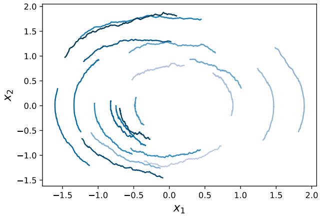


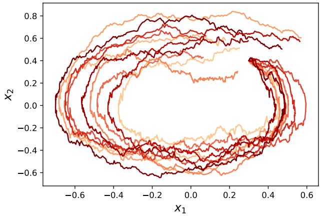
Stochastic Oscillator

Another 2D example is the stochastic oscillator, in the following form

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = -x_1 + \sigma \dot{W}_t. \end{cases} \tag{26}$$

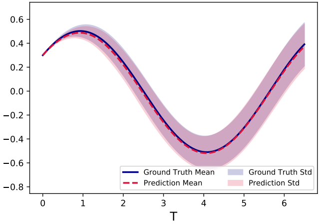
The training data, simulated test data (in the phase space):

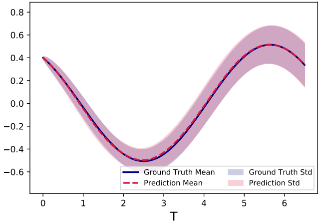




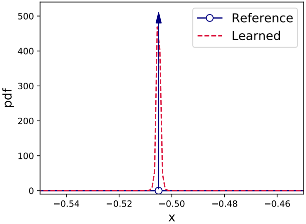
Stochastic Oscillator

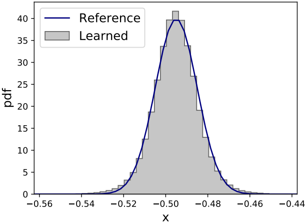
The "mean-std vs time" plot:





The comparison of $\mathbf{G}_{\Delta}(-0.5, -0.5)$:





Ending

Details and more examples can be found:

Y. Chen, D. Xiu. Learning Stochastic Dynamical System via Flow Map Operator, (2023), *arXiv: 2305.03874*.

Thanks for your attention!

Reference

Reference I

Arjovsky, M. and Bottou, L. (2017).

Towards principled methods for training generative adversarial networks.

arXiv preprint arXiv:1701.04862.

Arjovsky, M., Chintala, S., and Bottou, L. (2017).

Wasserstein generative adversarial networks.

In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Reference II

Darcy, M., Hamzi, B., Livieri, G., Owhadi, H., and Tavallali, P. (2022).

One-shot learning of stochastic differential equations with computational graph completion.

arXiv preprint arXiv:2209.12086.

Øksendal, B. (2003).

Stochastic differential equations.

In *Stochastic differential equations*, pages 65–84. Springer.

Reference III

Qin, T., Wu, K., and Xiu, D. (2019).

Data driven governing equations approximation using deep neural networks.

Journal of Computational Physics, 395:620–635.