SCALABLE LEARNING OF DRIFT AND DIFFUSION IN HIGH-DIMENSIONAL STOCHASTIC DIFFERENTIAL EQUATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Stochastic differential equations (SDEs) provide a fundamental framework for modeling high-dimensional stochastic dynamical systems. Their drift and diffusion functions govern the deterministic dynamics and stochastic fluctuations of the underlying system, respectively. Learning these functions from observed data is crucial for understanding and predicting system behavior. Despite the importance of high-dimensional SDEs ($d \gg 100$) in practical applications, existing methods mostly rely on prior knowledge of the functional forms or are difficult to scale to the high-dimensional case because of prohibitive time and space complexity. To address these limitations, we propose KoopSDE, an algorithm that uses the stochastic Koopman operator to learn drift and diffusion functions from a single discretely observed trajectory without prior knowledge of functional forms. To address the curse of dimensionality, we incorporate Hutchinson trace estimation (HTE), reducing the time and space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. We also establish a set of theoretical guarantees for KoopSDE. Experiments demonstrate that KoopSDE can learn $100,000$-dimensional SDEs on a single NVIDIA A800 GPU. This work opens up the possibility for learning the drift and diffusion functions of high-dimensional SDEs.

1 INTRODUCTION

Stochastic fluctuations are ubiquitous in real-world dynamical systems, interacting with deterministic dynamics to produce complex random behavior (Pavliotis, 2014). Stochastic differential equations (SDEs) provide a universal modeling framework for such systems. In many practical applications, system states are inherently high-dimensional. Low-dimensional SDEs are often oversimplified in these cases. High-dimensional SDEs become essential tools for modeling such systems.

Consider a d-dimensional autonomous stochastic differential equation defined on a complete filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 0}, \mathbb{P})$:

$$dX_t = \boldsymbol{b}(X_t)\, dt + \boldsymbol{\sigma}(X_t)\, dB_t, \quad t \geq 0, \tag{1}$$

where $X_t \in \mathcal{S} \subseteq \mathbb{R}^d$ is the d-dimensional stochastic state process, $\mathcal{S}$ is the state space, and B_t is an r-dimensional standard Brownian motion. $\boldsymbol{b} : \mathcal{S} \to \mathbb{R}^d$ and $\boldsymbol{\sigma} : \mathcal{S} \to \mathbb{R}^{d\times r}$ are unknown coefficients. We denote the associated diffusion matrix by $\mathbf{D}(\boldsymbol{x}) := \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{\sigma}(\boldsymbol{x})^\top$. The drift function $\boldsymbol{b}(\boldsymbol{x})$ governs the deterministic state dynamics, while the diffusion function $\mathbf{D}(\boldsymbol{x})$ characterizes the covariance structure of the stochastic perturbations. The problem is how to learn $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$ from discrete trajectory $(\boldsymbol{x}_l)_{l=0}^n$. Existing methods mostly rely on prior knowledge of functional forms, or suffer from high computational costs that prevent scaling to the high-dimensional case ($d \gg 100$).

Existing learning methods can be divided into two categories according to the availability of prior knowledge. Parametric methods assume known parametric forms of $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$ (Elerian et al., 2001; Singleton, 2001; Aït-Sahalia, 2002; Hurn et al., 2007; Riseth & Taylor-King, 2017; Wang et al., 2022), whereas nonparametric methods such as numerical methods (Davis & Buffett, 2022) and deep learning methods (Ren et al., 2024; Seifner et al., 2025) learn unknown functional forms directly from data. A critical challenge for both categories of methods is their inability to scale to high-dimensional SDEs, due to algorithmic design or high time and space complexity.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison of related works. *Prior-Free* indicates whether a method requires prior knowledge of the functional forms, while *Scalable Dimension* denotes the typical range of dimensions that the method can handle.

Work	Hurn et al. (2007)	Ruttor et al. (2013)	Batz et al. (2016)	Riseth et al. (2017)	Wang et al. (2022)
Prior-Free	✗	✗	✗	✗	✗
Scalable Dimension	$d \sim 10$	$d \sim 10$	$d \sim 10$	$d = 1$	$d \sim 10$
Work	Feng et al. (2023)	Davis et al. (2022)	Ren et al. (2024)	Seifner et al. (2025)	**KoopSDE (Ours)**
Prior-Free	✗	✓	✓	✓	✓
Scalable Dimension	$d \sim 10$	$d \sim 10$	$d \sim 10^2$	$d < 3$	$\boldsymbol{d \sim 10^4}$

This paper proposes a novel algorithm—KoopSDE, which learns $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$ directly from the discrete trajectory of SDEs. By treating the SDE in Eq. (1) as an autonomous stochastic dynamical system, we construct a two-stage learning framework based on the stochastic Koopman operator. To tackle the curse of dimensionality caused by directly computing Hessian matrices of eigenfunctions in the high-dimensional case, we incorporate Hutchinson Trace Estimation (HTE) (Hutchinson, 1989). Our main contributions are as follows:

- **Scalable method.** KoopSDE learns $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$ from $(\boldsymbol{x}_l)_{l=0}^n$ without prior knowledge of functional forms. Our proposed Diff-Hess trace estimator reduces the time and space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, enabling KoopSDE to scale effectively to high-dimensional SDEs.
- **Theoretical guarantees.** We prove the unbiasedness of the Diff-Hess trace estimator and analyze its variance. We establish the asymptotic consistency of the eigenvalue estimator and show that KoopSDE preferentially learns eigenfunctions corresponding to the largest eigenvalues. We also derive explicit oracle bounds for the eigen-loss and the generator-loss, clarifying how network capacity and sample size affect the learning accuracy.
- **Experimental validation.** We evaluate KoopSDE on four high-dimensional SDEs using relative drift and diffusion errors and peak GPU memory. Experiments on an NVIDIA A800 GPU extend to $100{,}000$ dimensions, with relative drift and diffusion errors of 11.40% and 8.68%.

2 Related Work

Parameter estimation with prior knowledge. Classical methods in this category can be broadly divided into likelihood-based approaches and methods based on matching sample characteristics (Hurn et al., 2007). Likelihood-based methods approximate the transition density to construct the likelihood function, including discrete and simulated maximum likelihood (Pedersen, 1995; Shoji & Ozaki, 1997), Markov chain Monte Carlo (Elerian et al., 2001; Eraker, 2001), finite-difference schemes (Jensen & Poulsen, 2002), and Hermite polynomial expansions (Aït-Sahalia, 2002). Methods based on matching sample characteristics estimate parameters by matching moments or characteristic functions, such as the generalized method of moments (Hansen & Scheinkman, 1993) and the characteristic function method (Singleton, 2001; Chacko & Viceira, 2003).

Riseth & Taylor-King (2017) employs Extended Dynamic Mode Decomposition (EDMD) to compute a transition matrix and thereby obtain a parameterized matrix approximation of the Koopman operator for parameter estimation. Ruttor et al. (2013) and Batz et al. (2016) assume known $\mathbf{D}(\boldsymbol{x})$ and focus on nonparametric estimation of $\boldsymbol{b}(\boldsymbol{x})$. More recently, Wang et al. (2022) and Feng et al. (2023) employed neural networks to estimate parameters of SDEs driven by Lévy noise and fractional Brownian motion, respectively. While these methods offer complete estimation frameworks for low-dimensional SDEs ($d \sim 10$), they rely on prior assumptions and face prohibitive time and space complexity in the high-dimensional case.

Prior-Free function learning. Methods in this category learn the drift and diffusion directly from data without prior knowledge of functional forms. Sparse Identification of Nonlinear Dynamics (SINDy) learns stochastic drift and diffusion through sparse regression over a prescribed function library (Boninsegna et al., 2018). Davis & Buffett (2022) handle non-uniformly sampled sequences via bivariate kernel density estimation. Dynamical Gaussian mixture approximation (DynGMA) learns unknown SDE coefficients using neural networks and a dynamical Gaussian mixture likelihood (Zhu & Li, 2024). Sparse neural network (SparseNN) learns drift and diffusion functions through empirical

loss minimization (Ren et al., 2024). FIM-SDE, based on the Transformer architecture, employs meta-learning on large-scale synthetic SDE data to enable in-context learning and rapid fine-tuning for low-dimensional SDEs (Seifner et al., 2025). Similarly, their time and space complexity limits high-dimensional scalability. In contrast, KoopSDE addresses both the high-dimensional scalability challenge and the prior-knowledge requirement (Table 1).

Curse of dimensionality and stochastic algorithms. The curse of dimensionality refers to computational and storage challenges in high-dimensional problems (Bellman, 1957). A series of stochastic algorithms has been introduced to mitigate this. HTE estimates the trace of a matrix using stochastic quadratic forms involving random vectors, thereby avoiding explicit computation and storage of the full matrix. It has been applied in various fields (Lai et al., 2022; Hu et al., 2024a; Shi et al., 2024; Hu et al., 2025). Related work also includes Physics-Informed Neural Networks (PINNs) for solving partial differential equations (PDEs) (Raissi et al., 2019). To scale PINNs to high-dimensional PDEs, Stochastic Dimension Gradient Descent (SDGD) randomly samples input dimensions for backpropagation to reduce computational cost (Hu et al., 2024b). In our ablation studies, we also evaluate the performance of KoopSDE when using SDGD.

3 Preliminaries and Discussions

3.1 Problem Setting

We study the nonparametric learning problem for the SDE in Eq. (1). The drift $\boldsymbol{b}(\boldsymbol{x})$ and diffusion $\mathbf{D}(\boldsymbol{x})$ are unknown, and no parametric forms are prescribed. For any initial condition $X_0 = \boldsymbol{x}_0$, we assume that Eq. (1) admits a unique strong solution $\{X_t\}_{t\geq 0}$. We refer to this solution as the diffusion process and assume that it admits an invariant probability measure μ. The available data consist of a single discretely sampled trajectory $(\boldsymbol{x}_l)_{l=0}^n$ with sampling interval τ. Our objective is to learn $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$ from this trajectory.

3.2 Stochastic Koopman Operator

The diffusion process $\{X_t\}_{t\geq 0}$ generates an autonomous stochastic dynamical system, which can be characterized by the stochastic Koopman operator. Let $L^2(\mathcal{S}, \mu)$ denote the space of square-integrable functions, abbreviated as L^2. The stochastic Koopman operator $\mathcal{K}^t : L^2 \to L^2$ is defined for any $f \in L^2$ by

$$(\mathcal{K}^t f)(\boldsymbol{x}) = \mathbb{E}\big[f(X_t) \mid X_0 = \boldsymbol{x}\big], \quad \boldsymbol{x} \in \mathcal{S}.$$

The autonomy of Eq. (1) implies that $\{X_t\}_{t\geq 0}$ is a time-homogeneous Markov process (Stroock & Varadhan, 2007; Oksendal, 2013). Consequently, $\{\mathcal{K}^t\}_{t\geq 0}$ forms a semigroup satisfying $\mathcal{K}^0 = \mathcal{I}$ (the identity operator) and $\mathcal{K}^{t+s} = \mathcal{K}^t\mathcal{K}^s$ for all $t, s \geq 0$.

For a general observable $f \in L^2$, the action of $\mathcal{K}^t$ does not generally admit a tractable closed-form characterization. We therefore focus on the Koopman eigenfunction space, where the evolution of each eigenfunction is explicitly characterized by a scalar exponential factor. Specifically, if a scalar $\lambda \in \mathbb{R}$ and a nonzero function $\phi \in L^2$ are such that

$$(\mathcal{K}^t\phi)(\boldsymbol{x}) = e^{\lambda t}\phi(\boldsymbol{x}), \tag{2}$$

they are called the Koopman eigenvalue and eigenfunction. For further discussion of stochastic Koopman operators, see Chapter 6 of Mauroy et al. (2020).

3.3 Infinitesimal Generator

We next introduce the infinitesimal generator, a key linear operator that links $\boldsymbol{b}(\boldsymbol{x})$, $\mathbf{D}(\boldsymbol{x})$, and eigenfunction $\phi(\boldsymbol{x})$. For the semigroup $\{\mathcal{K}^t\}_{t\geq 0}$, define

$$\mathcal{D}_{\mathcal{L}} = \left\{ f \in L^2 : \exists g \in L^2 \ \text{ s.t. } \lim_{t\to 0^+} \left\| \frac{\mathcal{K}^t f - f}{t} - g \right\|_{L^2} = 0 \right\}.$$

According to Theorem 3.2.6 in Applebaum (2009), $\mathcal{D}_{\mathcal{L}}$ is a dense linear subspace of L^2. A linear operator $\mathcal{L}$ can be defined such that for any $f \in \mathcal{D}_{\mathcal{L}}$, $\mathcal{L}f = \lim_{t\to 0^+} \frac{\mathcal{K}^t f - f}{t}$. This operator $\mathcal{L}$ is

called the infinitesimal generator of the semigroup $\{\mathcal{K}^t\}_{t\geq 0}$, or equivalently of $\{X_t\}_{t\geq 0}$. An explicit expression of $\mathcal{L}$ follows from Itô's formula. For any twice continuously differentiable function $f \in \mathcal{C}^2(\mathbb{R}^d) \cap \mathcal{D}_{\mathcal{L}}$,

$$\mathcal{L}f = \boldsymbol{b} \cdot \nabla f + \frac{1}{2}\operatorname{tr}(\mathbf{D}\mathbf{H}_f), \tag{3}$$

where ∇ is the gradient operator and $\mathbf{H}_f = (\frac{\partial^2 f}{\partial x_i \partial x_j})_{d\times d}$ is the Hessian matrix of f. See Appendix B.1 for a proof. $\mathcal{K}^t$ describes the average evolution of f over a finite interval t, while the generator $\mathcal{L}$ captures its instantaneous rate of change. Furthermore, applying the definition of $\mathcal{L}$ to Eq. (2) yields

$$\mathcal{L}\phi = \frac{d}{dt}\left(e^{\lambda t}\phi\right)\Big|_{t=0} = \lambda e^{\lambda t}\phi\Big|_{t=0} = \lambda\phi. \tag{4}$$

Combining Eq. (4) with Eq. (3) yields a constraint linking the Koopman eigenfunction $\phi(\boldsymbol{x})$ to the drift $\boldsymbol{b}(\boldsymbol{x})$ and diffusion $\mathbf{D}(\boldsymbol{x})$. Together with Eq. (2), this constraint forms the basis of KoopSDE.

4 METHODOLOGY

This section presents KoopSDE as a two-stage method for learning high-dimensional SDEs. The first stage learns Koopman eigenfunctions and eigenvalues via Eq. (2), while the second estimates the drift and diffusion functions using Eqs. (3)-(4).

4.1 LEARNING THE EIGENFUNCTIONS AND EIGENVALUES

Parameterization of eigenfunctions. Given the discrete trajectory $(\boldsymbol{x}_l)_{l=0}^n$ with time step τ, we consider N real-valued eigenfunctions $\phi_k : \mathcal{S} \to \mathbb{R}$ of $\mathcal{K}^\tau$ with corresponding real eigenvalues λ_k, where $k = 1, \ldots, N$. Define the eigenfunction vector $\Phi : \mathcal{S} \to \mathbb{R}^N$ and the diagonal matrix $\mathbf{\Lambda}$ as

$$\Phi(\boldsymbol{x}) := [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_N(\boldsymbol{x})]^\top, \quad \mathbf{\Lambda} := \operatorname{diag}(e^{\lambda_1\tau}, \ldots, e^{\lambda_N\tau}) \in \mathbb{R}^{N\times N}.$$

Applying Eq. (2) componentwise and stacking the resulting relations gives

$$(\mathcal{K}^\tau\Phi)(\boldsymbol{x}_l) = \mathbf{\Lambda}\Phi(\boldsymbol{x}_l),\ l = 0, \ldots, n. \tag{5}$$

Eq. (5) describes the dynamic evolution in the eigenfunction space. To learn Φ, we construct a dynamic loss based on this relation. By the universal approximation theorem (Hornik et al., 1989; Lu et al., 2017), we parameterize Φ with a Multi-Layer Perceptron (MLP) as

$$\widetilde{\Phi}(\boldsymbol{x}; \boldsymbol{\theta}_\Phi) = \left[\widetilde{\phi}_1(\boldsymbol{x}; \boldsymbol{\theta}_\Phi), \widetilde{\phi}_2(\boldsymbol{x}; \boldsymbol{\theta}_\Phi), \ldots, \widetilde{\phi}_N(\boldsymbol{x}; \boldsymbol{\theta}_\Phi)\right]^\top,$$

where $\boldsymbol{\theta}_\Phi$ represents the trainable parameters of the MLP. Note that we do not directly parameterize $\mathbf{\Lambda}$ by treating the eigenvalues $\lambda_1, \ldots, \lambda_N$ as trainable parameters. From Eq. (2), it follows that

$$\int_{\mathcal{S}} (\mathcal{K}^\tau\phi_k)\phi_k d\mu = \int_{\mathcal{S}} e^{\lambda_k\tau}\phi_k^2 d\mu = e^{\lambda_k\tau}\int_{\mathcal{S}} \phi_k^2 d\mu.$$

Since $\phi_k \in L^2 \setminus \{0\}$, it holds that $\int_{\mathcal{S}} \phi_k^2 d\mu > 0$. A functional dependent on ϕ_k can be defined as

$$e^{\lambda_k\tau} = \left[\int_{\mathcal{S}} (\mathcal{K}^\tau\phi_k)\phi_k d\mu\right]\left[\int_{\mathcal{S}} \phi_k^2 d\mu\right]^{-1} =: \nu(\phi_k). \tag{6}$$

Consequently, $\mathbf{\Lambda} = \operatorname{diag}\big(\nu(\phi_1), \ldots, \nu(\phi_N)\big)$, and we parameterize it directly from $\widetilde{\Phi}$ without introducing additional trainable parameters as

$$\widetilde{\mathbf{\Lambda}} = \operatorname{diag}(\nu(\widetilde{\phi}_1), \ldots, \nu(\widetilde{\phi}_N)),$$

where $\nu(\widetilde{\phi}_k) = \left[\int_{\mathcal{S}} (\mathcal{K}^\tau\widetilde{\phi}_k)\widetilde{\phi}_k d\mu\right]\left[\int_{\mathcal{S}} \widetilde{\phi}_k^2 d\mu\right]^{-1}$. Based on Eq. (5), the dynamic loss function can be constructed entirely from $\widetilde{\Phi}$ as

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = \frac{1}{n}\sum_{l=0}^{n-1} \|\mathcal{K}^\tau\widetilde{\Phi}(\boldsymbol{x}_l; \boldsymbol{\theta}_\Phi) - \widetilde{\mathbf{\Lambda}}\widetilde{\Phi}(\boldsymbol{x}_l; \boldsymbol{\theta}_\Phi)\|^2.$$

When $\widetilde{\Phi} = \Phi$, Eq. (5) implies $\mathcal{L}_{\mathrm{dyn}} = 0$. Therefore, learning the eigenfunctions Φ is equivalent to minimizing the dynamic loss $\mathcal{L}_{\mathrm{dyn}}$. This process only requires optimizing the parameters $\boldsymbol{\theta}_\Phi$ of the eigenfunction network $\widetilde{\Phi}$.

Dynamic loss. The dynamic loss $\mathcal{L}_{\mathrm{dyn}}$ can be estimated based on $(\boldsymbol{x}_l)_{l=0}^n$. Specifically, we construct two design matrices as

$$\mathbf{Y}_0 := \left[\widetilde{\Phi}(\boldsymbol{x}_0;\boldsymbol{\theta}_\Phi)\cdots\widetilde{\Phi}(\boldsymbol{x}_{n-1};\boldsymbol{\theta}_\Phi)\right] \in \mathbb{R}^{N\times n}, \quad \mathbf{Y}_1 := \left[\widetilde{\Phi}(\boldsymbol{x}_1;\boldsymbol{\theta}_\Phi)\cdots\widetilde{\Phi}(\boldsymbol{x}_n;\boldsymbol{\theta}_\Phi)\right] \in \mathbb{R}^{N\times n}. \tag{7}$$

Since $(\mathcal{K}^\tau\widetilde{\Phi})(\boldsymbol{x}_l)$ is the conditional expectation of the next observable, $\widetilde{\Phi}(\boldsymbol{x}_{l+1})$ can be used as its sample approximation. Thus, $\mathcal{L}_{\mathrm{dyn}}$ can be approximated as

$$\frac{1}{n}\sum_{l=0}^{n-1}\|\widetilde{\Phi}(\boldsymbol{x}_{l+1};\boldsymbol{\theta}_\Phi) - \widetilde{\boldsymbol{\Lambda}}\widetilde{\Phi}(\boldsymbol{x}_l;\boldsymbol{\theta}_\Phi)\|^2 = \frac{1}{n}\|\mathbf{Y}_1 - \widetilde{\boldsymbol{\Lambda}}\mathbf{Y}_0\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Estimating $\mathcal{L}_{\mathrm{dyn}}$ requires estimating $\widetilde{\boldsymbol{\Lambda}}$. Note that the estimator of $\widetilde{\boldsymbol{\Lambda}}$ is essentially the diagonal matrix that minimizes $\|\mathbf{Y}_1 - \widetilde{\boldsymbol{\Lambda}}\mathbf{Y}_0\|_F^2$, which reduces to a linear least squares problem. That is, $\nu(\widetilde{\phi}_k)$ can be estimated as

$$\widehat{\nu}(\widetilde{\phi}_k) = \left[\sum_{l=0}^{n-1}\frac{\widetilde{\phi}_k(\boldsymbol{x}_{l+1};\boldsymbol{\theta}_\Phi)\widetilde{\phi}_k(\boldsymbol{x}_l;\boldsymbol{\theta}_\Phi)}{n}\right]\left[\sum_{l=0}^{n-1}\frac{\widetilde{\phi}_k^2(\boldsymbol{x}_l;\boldsymbol{\theta}_\Phi)}{n}\right]^{-1} = \frac{(\mathbf{Y}_0\mathbf{Y}_1^\top)_{kk}}{(\mathbf{Y}_0\mathbf{Y}_0^\top)_{kk}}. \tag{8}$$

The resulting diagonal estimate is $\widehat{\boldsymbol{\Lambda}} = \mathrm{diag}(\widehat{\nu}(\widetilde{\phi}_1),\ldots,\widehat{\nu}(\widetilde{\phi}_N))$. The dynamic loss function can be constructed as

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = \frac{1}{n}\|\mathbf{Y}_1 - \widehat{\boldsymbol{\Lambda}}\mathbf{Y}_0\|_F^2.$$

Orthogonality loss. The dynamic loss alone may cause $\widetilde{\Phi}$ to collapse to N identical functions or zero functions. We impose an orthogonality constraint on $\widetilde{\Phi}$ to prevent such collapse. The orthonormality of the eigenfunctions indicates that

$$\langle\Phi,\Phi\rangle_{L^2} = \int_{\mathcal{S}}\Phi\Phi^\top d\mu = \mathbf{I},$$

where $\mathbf{I}$ is the $N\times N$ identity matrix. Note that the matrix $\int_{\mathcal{S}}\Phi\Phi^\top d\mu$ can be approximated by the empirical Gram matrix $\frac{1}{n}\mathbf{Y}_0\mathbf{Y}_0^\top$. Thus, we construct the orthogonality loss

$$\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi) = \left\|\frac{1}{n}\mathbf{Y}_0\mathbf{Y}_0^\top - \mathbf{I}\right\|_F^2.$$

The total loss for eigenfunction learning combines the dynamic and orthogonality losses as

$$\mathcal{L}_{\mathrm{eigen}}(\boldsymbol{\theta}_\Phi) = \mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) + \alpha\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi),$$

where $\alpha > 0$ is the weight of $\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi)$. Then, minimizing $\mathcal{L}_{\mathrm{eigen}}(\boldsymbol{\theta}_\Phi)$ encourages the network to learn N approximately orthonormal eigenfunctions of $\mathcal{K}^\tau$ and their associated eigenvalues.

4.2 Learning the Drift and Diffusion Functions

Given the eigenfunctions $\widetilde{\Phi}$ learned in Section 4.1, we parameterize the drift $\boldsymbol{b}(\boldsymbol{x})$ and diffusion $\mathbf{D}(\boldsymbol{x})$ associated with Eq. (1) using separate MLPs $\widetilde{\boldsymbol{b}}(\boldsymbol{x};\boldsymbol{\theta}_b)$ and $\widetilde{\mathbf{D}}(\boldsymbol{x};\boldsymbol{\theta}_\sigma)$. In the second stage, we optimize only $\boldsymbol{\theta}_b$ and $\boldsymbol{\theta}_\sigma$ while keeping $\boldsymbol{\theta}_\Phi$ fixed.

Eq. (6) implies $\lambda_k = \tau^{-1}\ln\nu(\phi_k)$. Using the estimator in Eq. (8), we obtain $\widehat{\lambda}_k = \tau^{-1}\ln\widehat{\nu}(\widetilde{\phi}_k)$ and define the diagonal eigenvalue matrix $\widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}$ and generator matrix $\mathbf{L}$ as

$$\widehat{\boldsymbol{\Lambda}}_{\mathcal{L}} := \mathrm{diag}(\widehat{\lambda}_1,\ldots,\widehat{\lambda}_N) \in \mathbb{R}^{N\times N}, \quad \mathbf{L} := \left[\widetilde{\mathcal{L}}\widetilde{\Phi}(\boldsymbol{x}_0),\cdots,\widetilde{\mathcal{L}}\widetilde{\Phi}(\boldsymbol{x}_{n-1})\right] \in \mathbb{R}^{N\times n}. \tag{9}$$

For each $\widetilde{\phi}_k$, the parameterized generator is given by

$$\widetilde{\mathcal{L}}\widetilde{\phi}_k := \widetilde{\boldsymbol{b}}\cdot\nabla\widetilde{\phi}_k + \frac{1}{2}\,\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k), \tag{10}$$

where $\mathbf{H}_k = \left(\frac{\partial^2\widetilde{\phi}_k}{\partial x_i\partial x_j}\right)_{d\times d}$ is the Hessian matrix of $\widetilde{\phi}_k$. Equation (4) gives the ideal relation $\mathbf{L} = \widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}\mathbf{Y}_0$, which motivates the generator loss

$$\mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b,\boldsymbol{\theta}_\sigma) = \frac{1}{n}\left\|\mathbf{L} - \widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}\mathbf{Y}_0\right\|_F^2.$$

The drift and diffusion functions are jointly learned by minimizing $\mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b,\boldsymbol{\theta}_\sigma)$ to reduce the fitting error of the infinitesimal generator.

Diff-Hess trace estimator. Although $\mathbf{Y}_0$ and $\widehat{\mathbf{\Lambda}}_{\mathcal{L}}$ can be computed directly from Eq. (7) and Eq. (9), computing the generator matrix $\mathbf{L}$ via Eq. (10) requires computing the Hessian matrix $\mathbf{H}_k$ for each $\widetilde{\phi}_k$. When computing $\mathbf{H}_k$ via automatic differentiation in the high-dimensional case, we face the following main obstacles:

Memory: Storing $\mathbf{H}_k$ explicitly requires $\mathcal{O}(d^2)$ memory overhead. More critically, automatic differentiation for each second-order partial derivative must retain the full computational graph, yielding a memory complexity of $\mathcal{O}(d^2 \times |\boldsymbol{\theta}_\Phi|)$, where $|\boldsymbol{\theta}_\Phi|$ is the number of parameters in $\widetilde{\Phi}$.

Time: Computing $\mathbf{H}_k$ must perform a separate backpropagation for each dimension to compute the second-order derivatives. Each backpropagation itself requires $\mathcal{O}(d \times |\boldsymbol{\theta}_\Phi|)$ time cost, resulting in a total time complexity of $\mathcal{O}(d^2 \times |\boldsymbol{\theta}_\Phi|)$.

Therefore, the time and space complexity of computing $\mathbf{H}_k$ is $\mathcal{O}(d^2 \times |\boldsymbol{\theta}_\Phi|)$, which is significantly higher than the $\mathcal{O}(d \times |\boldsymbol{\theta}_\Phi|)$ required for computing the drift-gradient inner product $\widetilde{\boldsymbol{b}} \cdot \nabla \widetilde{\phi}_k$. This makes the direct computation of $\mathbf{H}_k$ the main bottleneck for scaling KoopSDE to high-dimensional SDEs ($d \gg 100$). In fact, we ultimately only need to compute the trace of $\widetilde{\mathbf{D}}\mathbf{H}_k \in \mathbb{R}^{d\times d}$ by Eq. (10), and Hutchinson trace estimator (HTE) provides an efficient approach for this.

Lemma 4.1. *(Hutchinson, 1989) For any matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$,*

$$\mathrm{tr}(\mathbf{A}) = \mathbb{E}_{\boldsymbol{z}}\big[\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z}\big],$$

where $\boldsymbol{z} \in \mathbb{R}^d$ is a random vector with i.i.d. components satisfying $\mathbb{E}[z_i] = 0$ and $\mathrm{Var}(z_i) = 1$.

Proof. See Appendix B.2. Applying Lemma 4.1 to $\widetilde{\mathbf{D}}\mathbf{H}_k$ gives

$$\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k) = \mathbb{E}_{\boldsymbol{z}}\big[\boldsymbol{z}^\top \widetilde{\mathbf{D}}\mathbf{H}_k \boldsymbol{z}\big] = \mathbb{E}_{\boldsymbol{z}}\big[\boldsymbol{z}^\top \widetilde{\mathbf{D}} \cdot \nabla\big(\boldsymbol{z}^\top \nabla \widetilde{\phi}_k\big)\big].$$

The second equality follows from $\mathbf{H}_k \boldsymbol{z} = \nabla\big(\boldsymbol{z}^\top \nabla \widetilde{\phi}_k\big)$. Indeed, $\mathbf{H}_k = \nabla^2 \widetilde{\phi}_k$ and $\boldsymbol{z}$ is independent of $\boldsymbol{x}$, so $\nabla\big(\boldsymbol{z}^\top \nabla \widetilde{\phi}_k\big) = (\nabla^2 \widetilde{\phi}_k)\boldsymbol{z} = \mathbf{H}_k \boldsymbol{z}$. We then sample m independent Rademacher vectors $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)} \in \mathbb{R}^d$, whose components satisfy $\mathbb{P}(z_i^{(l)} = 1) = \mathbb{P}(z_i^{(l)} = -1) = 1/2$. The Diff-Hess trace estimator is defined as

$$\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k) := \frac{1}{m} \sum_{l=1}^{m} \boldsymbol{u}_1^{(l)} \cdot \nabla\big(\boldsymbol{u}_2^{(l)}\big), \tag{11}$$

where $\boldsymbol{u}_1^{(l)} = \boldsymbol{z}^{(l)\top} \widetilde{\mathbf{D}}$ and $\boldsymbol{u}_2^{(l)} = \boldsymbol{z}^{(l)\top} \nabla \widetilde{\phi}_k$.

The core idea of Eq. (11) is to avoid explicitly computing $\mathbf{H}_k$ and instead compute a Hessian-vector product via two backpropagations. This avoids building a separate computational graph for each dimension, reducing the memory cost of graph from $\mathcal{O}(d^2 \times |\boldsymbol{\theta}_\Phi|)$ to $\mathcal{O}(md \times |\boldsymbol{\theta}_\Phi|)$ and matrix storage from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. The time complexity likewise decreases from $\mathcal{O}(d^2 \times |\boldsymbol{\theta}_\Phi|)$ to $\mathcal{O}(md \times |\boldsymbol{\theta}_\Phi|)$. Since the number of probes $m \ll d$ (we set $m = 10$ in Section 6), the Diff-Hess trace estimator $\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ reduces time and space complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$. Combining Eq. (10) and Eq. (11), we can efficiently compute the generator matrix $\mathbf{L}$ in the high-dimensional case. We provide the complete training algorithm of KoopSDE in Appendix A.1.

Remark 1. EDMD is a widely used and effective method for constructing finite-dimensional approximations of the Koopman operator (Williams et al., 2015). Its approximation quality depends on the choice of dictionary. Constructing a suitable dictionary often requires domain knowledge and manual design. KoopSDE instead learns Koopman eigenfunctions with neural networks and estimates the corresponding eigenvalues from data, without requiring a manually predefined dictionary.

Remark 2. The primary role of orthogonality loss $\mathcal{L}_{\mathrm{orth}}$ is to prevent $\widetilde{\Phi}(\boldsymbol{x}; \boldsymbol{\theta}_\Phi)$ from collapsing into N identical or zero eigenfunctions. Indeed, according to Eq. (4), each eigen-pair (ϕ, λ) imposes a constraint on $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$. Therefore, $\mathcal{L}_{\mathrm{orth}}$ ensures sufficient diversity among these constraints and enhances the effect of second-stage learning. The influencing mechanism of the number N is analyzed in detail in Appendix A.6.

5 Theoretical Analysis

In this section, we establish several theoretical properties for the KoopSDE algorithm.

Proposition 5.1. *The Diff-Hess trace estimator* $\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)$ *in Eq.* (11) *is unbiased, as*

$$\mathbb{E}_{\boldsymbol{z}}\left[\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)\right] = \mathrm{tr}(\widetilde{\mathbf{DH}}_k).$$

Among all admissible distributions satisfying the conditions of Lemma 4.1, the Rademacher distribution attains the minimum variance.

$$\mathrm{Var}_{\boldsymbol{z}\sim Rademacher}\left[\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)\right] = \min_P \mathrm{Var}_{\boldsymbol{z}\sim P}\left[\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)\right] = \frac{1}{m}\sum_{i<j}\left[(\widetilde{\mathbf{DH}}_k)_{ij} + (\widetilde{\mathbf{DH}}_k)_{ji}\right]^2.$$

Proof. See Appendix B.3. Proposition 5.1 provides a theoretical justification for choosing Rademacher random vectors in $\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)$. The variance of $\widehat{\mathrm{tr}}(\widetilde{\mathbf{DH}}_k)$ depends on the number of samples m and the off-diagonal entries of $\widetilde{\mathbf{DH}}_k$, and scales inversely with m. In particular, if $\widetilde{\mathbf{DH}}_k$ is a diagonal matrix, a single sample ($m = 1$) suffices to estimate $\mathrm{tr}(\widetilde{\mathbf{DH}}_k)$ exactly.

Theorem 5.2. *Suppose that the sampled Markov chain* $\{X_{l\tau}\}_{l\geq 0}$ *is ergodic with respect to* μ*. Then, for every nonzero real-valued Koopman eigenfunction* $\phi_k \in L^2(\mu)$ *and* μ*-almost every initial state* $\boldsymbol{x}_0 \in \mathcal{S}$*, the following almost-sure convergences hold:*

$$\widehat{\nu}(\phi_k) \xrightarrow{\text{a.s.}} e^{\lambda_k \tau}, \quad \widehat{\lambda}_k \xrightarrow{\text{a.s.}} \lambda_k.$$

Throughout this theorem, $\widehat{\lambda}_k = \tau^{-1}\ln\widehat{\nu}(\phi_k)$*. If, in addition,* $\phi_k \in L^4(\mu)$*, define*

$$s_k^2 := \int_{\mathcal{S}} \phi_k^2(\boldsymbol{x})\,\mu(d\boldsymbol{x}), \quad q_k := \int_{\mathcal{S}} \phi_k^2(\boldsymbol{x})\left[(\mathcal{K}^\tau\phi_k^2)(\boldsymbol{x}) - e^{2\lambda_k\tau}\phi_k^2(\boldsymbol{x})\right]\mu(d\boldsymbol{x}).$$

Then $s_k^2 > 0$*,* $q_k \in [0,\infty)$*, and the following convergence-in-distribution results hold:*

$$\sqrt{n}\left(\widehat{\nu}(\phi_k) - e^{\lambda_k\tau}\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{q_k}{s_k^4}\right), \quad \sqrt{n}\left(\widehat{\lambda}_k - \lambda_k\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{q_k}{\tau^2 e^{2\lambda_k\tau} s_k^4}\right).$$

Proof. See Appendix B.4. Theorem 5.2 establishes strong consistency and a parametric $\sqrt{n}$ asymptotic law for both estimators. This result justifies estimating $e^{\lambda_k\tau}$ and λ_k from ϕ_k in KoopSDE. In practical experiments, as long as the approximation error of $\widetilde{\phi}_k$ is sufficiently small, this limit result still ensures that $\widehat{\nu}(\widetilde{\phi}_k)$ and $\widehat{\lambda}_k$ satisfy the favorable asymptotic property.

Lemma 5.3. *If the orthogonality loss* $\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi) = 0$*, i.e., the parameterized eigenfunctions* $\widetilde{\Phi}(\boldsymbol{x};\boldsymbol{\theta}_\Phi)$ *are orthonormal on the discrete trajectory* $(\boldsymbol{x}_l)_{l=0}^n$*, then the dynamic loss can be expressed as*

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = N - \sum_{k=1}^N e^{2\widehat{\lambda}_k\tau}.$$

Proof. See Appendix B.5.

Corollary 5.4. *Under the conditions of Lemma 5.3, during the first-stage learning, the neural network* $\widetilde{\Phi}(\boldsymbol{x};\boldsymbol{\theta}_\Phi)$ *preferentially learns the eigenfunctions corresponding to the* N *largest eigenvalues of the stochastic Koopman operator* $\mathcal{K}^\tau$*.*

Proof. Let $\mathcal{Q} = \sum_{k=1}^N e^{2\widehat{\lambda}_k\tau}$*. By Lemma 5.3, minimizing* $\mathcal{L}_{\mathrm{dyn}}$ *is equivalent to maximizing* $\mathcal{Q}$*. Since* $e^{2\lambda\tau}$ *increases monotonically with* λ*,* $\mathcal{Q}$ *is maximized when* $(\widehat{\lambda}_k)_{k=1}^N$ *are the* N *largest eigenvalues of* $\mathcal{K}^\tau$*. Without loss of generality, let* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ *denote these eigenvalues, so that* $\widetilde{\Phi}$ *preferentially learns the eigenfunctions corresponding to* $(\lambda_k)_{k=1}^N$*.*

Corollary 5.4 reflects the intrinsic principle of modal decomposition: the eigenvalues of $\mathcal{K}^\tau$ govern the evolution rates of the corresponding modes. For decaying modes, a larger eigenvalue implies slower decay, making the mode more dominant in the long-term evolution of the system. By prioritizing eigenfunctions with the largest eigenvalues in the first stage, KoopSDE naturally concentrates on the most dynamically influential eigenspace, providing a basis for second-stage learning.

Additional oracle bounds for the two learning stages are provided in Appendices B.6 and B.7. Theorem B.1 bounds the expected empirical eigenfunction-learning loss in terms of transition noise, neural approximation error, and finite-trajectory error, while Theorem B.2 decomposes the empirical generator loss into the learned eigenpair defect, drift and diffusion approximation errors, and finite-sample trace-estimation variance.

6 EXPERIMENTS

This section validates the effectiveness of KoopSDE on four high-dimensional SDEs. Due to the page limit, only the key results are reported in the main text. The complete experimental configuration is detailed in Appendix A.4. All experiments were conducted on a single NVIDIA A800 GPU (80 GiB) with PyTorch (Paszke et al., 2017).

SDEs. We consider an Ornstein–Uhlenbeck (OU) process, a stochastic Lotka–Volterra system, a Galerkin fluid system, and a damped swing network. Under the parameters specified in Appendix A.2, these autonomous systems admit unique global strong solutions and invariant probability measures. We evaluate eight dimensions ranging from 10 to 100,000, using three random seeds and an observation interval of $\tau = 0.1$.

Baselines. Given the lack of validated baselines specifically designed for function learning in high-dimensional SDEs, we adapt several existing methods to the setting in Section 3.1. Our adaptations of SparseNN (Ren et al., 2024), DynGMA (Zhu & Li, 2024), and SINDy (Boninsegna et al., 2018) retain their respective core ideas. We also include the classical Euler–Gaussian quasi-maximum likelihood method (EM-qMLE) and two EDMD baselines, EDMD-Polynomial and EDMD-RFF, with polynomial and random Fourier dictionaries, respectively (Williams et al., 2015; Rahimi & Recht, 2007). Details of these adaptations are provided in Appendix A.3. To evaluate the effectiveness of the Diff-Hess trace estimator $\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$, we consider the following ablation variants.

- **KoopSDE (vanilla):** Computes $\mathbf{H}_k$ and $\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ exactly.
- **KoopSDE (SDGD):** Approximates $\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ using SDGD (Hu et al., 2024b), retaining approximately 15% of the Hessian entries at each iteration to balance efficiency and accuracy.
- **KoopSDE (HTE):** Uses our proposed Diff-Hess trace estimator $\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$.

These variants differ only in how the trace is computed. KoopSDE (HTE) uses $m = 10$ unless otherwise specified. The algorithmic principles of SDGD are detailed in Appendix A.3.

Table 2: Mean peak allocated GPU memory at $\tau = 0.1$. OOM indicates that the memory requirement exceeds 80 GiB. NaN indicates numerical failure due to nonfinite values.

Method	GPU Memory (GiB) (↓)							
	10 D	100 D	500 D	1K D	5K D	10K D	50K D	100K D
SparseNN	**0.02**	0.17	1.17	4.59	OOM	OOM	OOM	OOM
EM-qMLE	**0.02**	**0.13**	0.93	3.63	NaN	OOM	OOM	OOM
DynGMA	0.03	0.16	1.07	4.17	NaN	OOM	OOM	OOM
EDMD-RFF	0.05	0.30	1.06	4.11	OOM	OOM	OOM	OOM
EDMD-Polynomial	**0.02**	0.52	OOM	OOM	OOM	OOM	OOM	OOM
SINDy	0.08	OOM	OOM	OOM	OOM	OOM	OOM	OOM
KoopSDE (SDGD)	0.14	1.09	1.47	7.15	OOM	OOM	OOM	OOM
KoopSDE (vanilla)	0.14	2.66	3.55	17.81	OOM	OOM	OOM	OOM
KoopSDE (HTE)	0.14	0.38	**0.19**	**0.40**	**1.07**	**1.33**	**6.51**	**12.99**

Memory and scalability. Table 2 shows that KoopSDE (HTE) achieves the lowest reported GPU memory usage among the compared methods from $d = 500$ onward and scales to $d = 100{,}000$, demonstrating its high-dimensional scalability. Beyond fixed storage overhead, the high-dimensional measurements at a common batch size are consistent with approximately linear memory growth in d, increasing from 1.33 GiB at $d = 10{,}000$ to 12.99 GiB at $d = 100{,}000$. At $d = 1000$, KoopSDE (HTE) requires 0.40 GiB, compared with 17.81 GiB for KoopSDE (vanilla) and 7.15 GiB for KoopSDE (SDGD), corresponding to approximately $44.5\times$ and $17.9\times$ lower memory usage, respectively. These reductions demonstrate the effectiveness of the Diff-Hess trace estimator.

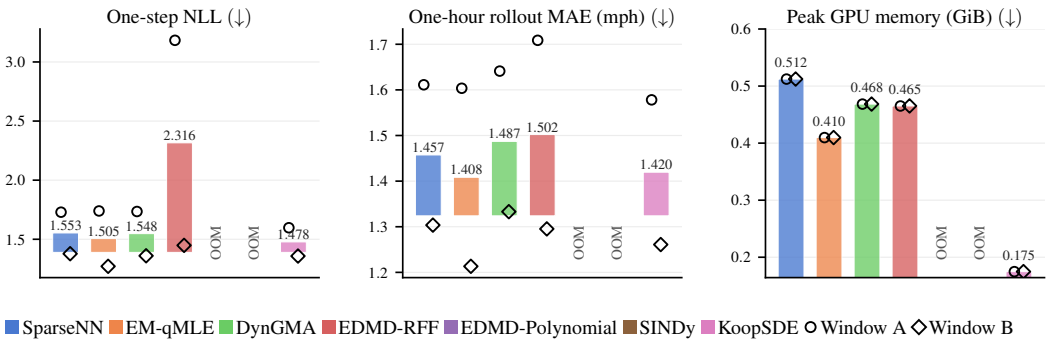

Figure 1: PEMS-BAY real-data evaluation of KoopSDE. Hollow circles and diamonds with black outlines denote Windows A and B, respectively. Bars show their unweighted mean. Lower values are better for all metrics. EDMD-Polynomial and SINDy retain recorded OOM status. Here, mph (miles per hour) is the unit of the average vehicle speed recorded by the traffic sensors.

Appendix A.5 provides the relative drift and diffusion error comparisons, including their system-level distributions, together with the observation-interval ablation at $\tau = 0.05$ and 0.25. The ablation study on the number of eigenfunctions N is provided in Appendix A.6.

Real-data validation on PEMS-BAY. We further evaluate KoopSDE on the PEMS-BAY traffic speed series with 325 sensors. We perform one-hour rollout forecasts from two preregistered time origins to cover different traffic conditions, reduce dependence on a single test window, and assess predictive stability. Because the data are sampled at 5-minute intervals, each test window contains 12 consecutive prediction steps, denoted as Window A and Window B, respectively. For each window, the model is trained using the history before the test origin and evaluated over the fixed subsequent hour, with the two window results averaged with equal weight. We report the one-step joint Gaussian NLL and the 12-step rollout MAE, averaging each metric equally across the two windows. The calculation formulas for both metrics are given in Appendix A.7. As shown in Figure 1, KoopSDE achieves an NLL of 1.478, a rollout MAE of 1.4196 mph, and a peak training memory of 0.1748 GiB. Among the successful baselines, KoopSDE achieves the lowest NLL, has a slightly higher MAE than EM-qMLE, and uses the least memory among the external successful baselines.

Appendix A.8 examines how the number of Hutchinson probes m affects the training speed and peak GPU memory of KoopSDE variants across dimensions. Appendix A.9 validates the learned drift, diffusion, eigenvalues, and simulated trajectories on a one-dimensional nonlinear OU process.

7 Conclusion

This paper introduces KoopSDE for learning the drift and diffusion functions of high-dimensional SDEs from discrete observations. While preserving information about stochastic evolution, KoopSDE provides a spectral representation of complex dynamical systems. Across four SDE benchmarks, KoopSDE scales to $100{,}000$ dimensions on an NVIDIA A800 GPU while maintaining moderate relative errors. Experiments on the PEMS-BAY real-world traffic-speed dataset further demonstrate the feasibility of applying the method to multivariate real-world time series. The goal of this work is not to replace existing classical methods for learning SDEs, but to explore a feasible approach for learning high-dimensional SDEs from discrete observations.

Future work. Porting the current implementation to high-performance numerical computing frameworks such as JAX may further improve the computational efficiency of KoopSDE and extend its applicability to higher-dimensional state spaces (Bradbury et al., 2018). In addition, the main contribution of this work lies at the algorithmic level. Future research could investigate more efficient architectures for parameterizing eigenfunctions. For systems whose state variables have an explicit spatial topology, graph neural networks could be used to model the eigenfunctions and exploit local dependencies and spatial structure among variables.

AI USE STATEMENT

In this work, we used generative AI tools for English-language editing, as well as programming assistance for plotting, synthetic-data generation, and utility scripts. We have not used generative AI tools to provide methodological ideas or mathematical proofs, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for implementation support and program debugging. All AI-assisted text, code, and mathematical expressions were reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Yacine Aït-Sahalia. Maximum likelihood estimation of discretely sampled diffusions: a closed-form approximation approach. *Econometrica*, 70(1):223–262, 2002.

David Applebaum. *Lévy processes and stochastic calculus*. Cambridge university press, 2009.

Philipp Batz, Andreas Ruttor, and Manfred Opper. Variational estimation of the drift for stochastic differential equations from the empirical density. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(8):083404, 2016.

RICHARD Bellman. Dynamic programming, princeton univ. *Press Princeton, New Jersey*, 39, 1957.

Lorenzo Boninsegna, Feliks Nüske, and Cecilia Clementi. Sparse learning of stochastic dynamical equations. *The Journal of chemical physics*, 148(24), 2018.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

George Chacko and Luis M Viceira. Spectral gmm estimation of continuous-time processes. *Journal of Econometrics*, 116(1-2):259–292, 2003.

William Davis and Bruce Buffett. Estimation of drift and diffusion functions from unevenly sampled time-series data. *Physical Review E*, 106(1):014140, 2022.

Ola Elerian, Siddhartha Chib, and Neil Shephard. Likelihood inference for discretely observed nonlinear diffusions. *Econometrica*, 69(4):959–993, 2001.

Bjørn Eraker. Mcmc analysis of diffusion models with application to finance. *Journal of Business & Economic Statistics*, 19(2):177–191, 2001.

Jing Feng, Xiaolong Wang, Qi Liu, Yongge Li, and Yong Xu. Deep learning-based parameter estimation of stochastic differential equations driven by fractional brownian motions with measurement noise. *Communications in Nonlinear Science and Numerical Simulation*, 127:107589, 2023.

Lars P Hansen and José A Scheinkman. Back to the future: Generating moment implications for continuous-time markov processes, 1993.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Zheyuan Hu, Zekun Shi, George Em Karniadakis, and Kenji Kawaguchi. Hutchinson trace estimation for high-dimensional and high-order physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 424:116883, 2024a.

Zheyuan Hu, Khemraj Shukla, George Em Karniadakis, and Kenji Kawaguchi. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*, 176:106369, 2024b.

Zheyuan Hu, Zhongqiang Zhang, George E Karniadakis, and Kenji Kawaguchi. Score-based physics-informed neural networks for high-dimensional fokker–planck equations. *SIAM Journal on Scientific Computing*, 47(3):C680–C705, 2025.

A Stan Hurn, JI Jeisman, and Kenneth A Lindsay. Seeing the wood for the trees: A critical evaluation of methods to estimate the parameters of stochastic differential equations. *Journal of Financial Econometrics*, 5(3):390–455, 2007.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Bjarke Jensen and Rolf Poulsen. Transition densities of diffusion processes: numerical comparison of approximation techniques. *Journal of Derivatives*, 9(4):18, 2002.

Samuel Karlin and Howard E Taylor. *A second course in stochastic processes*. Elsevier, 1981.

Chieh-Hsin Lai, Yuhta Takida, Naoki Murata, Toshimitsu Uesaka, Yuki Mitsufuji, and Stefano Ermon. Regularizing score-based models with score fokker-planck equations. In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. *Advances in neural information processing systems*, 30, 2017.

Alexandre Mauroy, Y Susuki, and Igor Mezic. *Koopman operator in systems and control*, volume 7. Springer, 2020.

Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

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

Asger Roer Pedersen. A new approach to maximum likelihood estimation for stochastic differential equations based on discrete observations. *Scandinavian journal of statistics*, pp. 55–71, 1995.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pp. 5301–5310. PMLR, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

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

Yinuo Ren, Yiping Lu, Lexing Ying, and Grant M Rotskoff. Statistical spatially inhomogeneous diffusion inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14820–14828, 2024.

Asbjørn N Riseth and Jake P Taylor-King. Operator fitting for parameter estimation of stochastic differential equations. *arXiv preprint arXiv:1709.05153*, 2017.

Andreas Ruttor, Philipp Batz, and Manfred Opper. Approximate gaussian process inference for the drift function in stochastic differential equations. *Advances in Neural Information Processing Systems*, 26, 2013.

Patrick Seifner, Kostadin Cvejoski, David Berghaus, Cesar Ojeda, and Ramses J Sanchez. In-context learning of stochastic differential equations with foundation inference models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Zekun Shi, Zheyuan Hu, Min Lin, and Kenji Kawaguchi. Stochastic taylor derivative estimator: Efficient amortization for arbitrary differential operators. *Advances in Neural Information Processing Systems*, 37:122316–122353, 2024.

Isao Shoji and Tohru Ozaki. Comparative study of estimation methods for continuous time stochastic processes. *Journal of time series analysis*, 18(5):485–506, 1997.

Kenneth J Singleton. Estimation of affine asset pricing models using the empirical characteristic function. *Journal of Econometrics*, 102(1):111–141, 2001.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Daniel W Stroock and SR Srinivasa Varadhan. *Multidimensional diffusion processes*. Springer, 2007.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xiaolong Wang, Jing Feng, Qi Liu, Yongge Li, and Yong Xu. Neural network-based parameter estimation of stochastic differential equations driven by lévy noise. *Physica A: Statistical Mechanics and its Applications*, 606:128146, 2022.

Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

Aiqing Zhu and Qianxiao Li. Dyngma: A robust approach for learning stochastic differential equations from data. *Journal of Computational Physics*, 513:113200, 2024.

A SUPPLEMENTARY EXPERIMENTS

A.1 KOOPSDE ALGORITHM

The pseudocode of training algorithm of KoopSDE are presented in Algorithm 1.

Algorithm 1 KoopSDE: Learning the Drift and Diffusion of High-Dimensional SDEs

Require: Discrete trajectory $(\boldsymbol{x}_l)_{l=0}^n$ with time step τ, number of eigenfunctions N, orthogonality weight α, sample counts m for HTE, learning rates η_1, η_2.
Ensure: Learned drift function $\boldsymbol{b}(\boldsymbol{x})$ and diffusion function $\mathbf{D}(\boldsymbol{x})$.

Stage 1: Learn Eigenfunctions and Eigenvalues
Initialize MLP $\widetilde{\Phi}(\boldsymbol{x}; \boldsymbol{\theta}_\Phi)$ for eigenfunctions Φ.
while *Not Converge* **do**
  Compute design matrices $\mathbf{Y}_0$ and $\mathbf{Y}_1$ via Eq. (7).
  Estimate $\widehat{\mathbf{\Lambda}} = \mathrm{diag}(\widehat{\nu}(\widetilde{\phi}_1), \ldots, \widehat{\nu}(\widetilde{\phi}_N))$ using Eq. (8).
  Compute dynamic loss $\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = \frac{1}{n}\|\mathbf{Y}_1 - \widehat{\mathbf{\Lambda}}\mathbf{Y}_0\|_F^2$.
  Compute orthogonality loss $\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi) = \|\frac{1}{n}\mathbf{Y}_0\mathbf{Y}_0^\top - \mathbf{I}\|_F^2$.
  Update $\boldsymbol{\theta}_\Phi \leftarrow \boldsymbol{\theta}_\Phi - \eta_1 \nabla_{\boldsymbol{\theta}_\Phi}(\mathcal{L}_{\mathrm{dyn}} + \alpha\mathcal{L}_{\mathrm{orth}})$.
end while
Obtain learned eigenfunctions $\widetilde{\Phi} = [\widetilde{\phi}_1, \ldots, \widetilde{\phi}_N]^\top$ and eigenvalues $\widehat{\lambda}_k = \tau^{-1} \ln \widehat{\nu}(\widetilde{\phi}_k)$.
Stage 2: Learn Drift and Diffusion Functions
Initialize MLPs $\widetilde{\boldsymbol{b}}(\boldsymbol{x}; \boldsymbol{\theta}_b)$ and $\widetilde{\mathbf{D}}(\boldsymbol{x}; \boldsymbol{\theta}_\sigma)$.
Freeze parameters of $\widetilde{\Phi}(\boldsymbol{x}; \boldsymbol{\theta}_\Phi)$.
while *Not Converge* **do**
  for $k = 1$ to N **do**
    Sample m Rademacher vectors $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}$.
    Compute $\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ using Diff-Hess trace estimator via Eq. (11).
    Compute $\widetilde{\mathcal{L}}\widetilde{\phi}_k = \widetilde{\boldsymbol{b}} \cdot \nabla\widetilde{\phi}_k + \frac{1}{2}\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$.
  end for
  Construct generator matrix $\mathbf{L}$ and $\widehat{\mathbf{\Lambda}}_{\mathcal{L}}$ via Eq. (9).
  Compute generator loss $\mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma) = \frac{1}{n}\|\mathbf{L} - \widehat{\mathbf{\Lambda}}_{\mathcal{L}}\mathbf{Y}_0\|_F^2$.
  Update $\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma \leftarrow \boldsymbol{\theta}_b - \eta_2\nabla_{\boldsymbol{\theta}_b}\mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma),\ \boldsymbol{\theta}_\sigma - \eta_2\nabla_{\boldsymbol{\theta}_\sigma}\mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma)$.
end while
return $\widetilde{\boldsymbol{b}}(\boldsymbol{x}; \boldsymbol{\theta}_b)$, $\widetilde{\mathbf{D}}(\boldsymbol{x}; \boldsymbol{\theta}_\sigma)$.

A.2 BENCHMARK SDES

We consider four autonomous SDEs with parameters fixed across dimensions. The matrix $\mathbf{I}_d$ is the $d \times d$ identity, and $\mathrm{Diag}(\boldsymbol{x})$ is the diagonal matrix with entries x_i. Nearest-neighbor indices are periodic, so $x_{-1} = x_{d-1}$ and $x_d = x_0$; for the swing network, the same convention gives $\theta_{-1} = \theta_{q-1}$ and $\theta_q = \theta_0$. All parameters below define the data-generating processes and are hidden from the learners. In the noise factors below, r is the dimension of the standard Brownian motion in the Euler–Maruyama scheme; the displayed factors are used only to generate the data and always satisfy $\boldsymbol{\sigma}\boldsymbol{\sigma}^\top = \mathbf{D}$.

Noise modes. For the OU and Lotka–Volterra systems, the correlated noise is built from four vectors. For coordinate indices $i = 0, \ldots, d-1$ and mode indices $k = 0, \ldots, 3$, define

$$u_{ik} = \sin\!\left(\frac{2\pi(k+1)(i+1/2)}{d} + 0.37k\right) + \frac{1}{2}\cos\!\left(\frac{2\pi(k+2)(i+1/4)}{d}\right),$$

$$\big[\mathbf{F}_d(\boldsymbol{c})\big]_{ik} = c_{k+1}u_{ik} \Big/ \left(\sum_{j=0}^{d-1} u_{jk}^2\right)^{1/2}, \qquad i = 0, \ldots, d-1, \quad k = 0, \ldots, 3.$$

where $\boldsymbol{c} = (c_1, c_2, c_3, c_4)$ contains the four amplitudes and j is the coordinate index used for column normalization. Each column is normalized before its amplitude is applied, and $\mathbf{F}_d(\boldsymbol{c})$ is used only to

generate data. For any q, $\mathbf{F}_q(\boldsymbol{c})$ denotes the same construction with the dimension parameter replaced by q, and therefore belongs to $\mathbb{R}^{q\times 4}$.

OU. The OU state has d real coordinates. Its drift and diffusion are

$$\boldsymbol{b}(\boldsymbol{x}) = -0.5\boldsymbol{x}, \qquad \mathbf{D}(\boldsymbol{x}) = \mathbf{F}\mathbf{F}^\top + 0.36\mathbf{I}_d, \qquad \mathbf{F} = \mathbf{F}_d(0.9, 0.6, 0.45, 0.3).$$

For trajectory generation we use the noise factor

$$\boldsymbol{\sigma}_{\mathrm{OU}} = [\mathbf{F} \quad 0.6\mathbf{I}_d], \qquad r = d + 4,$$

so that $\boldsymbol{\sigma}_{\mathrm{OU}}\boldsymbol{\sigma}_{\mathrm{OU}}^\top = \mathbf{D}$. The ith drift component is $b_i(\boldsymbol{x}) = -0.5x_i$. The first term in $\mathbf{D}$ has rank at most four and gives correlated noise, while the second term adds independent noise to every coordinate. The linear drift strictly contracts every direction and $\mathbf{D}$ is positive definite, so the system has a unique global strong solution and a nondegenerate Gaussian invariant distribution.

Stochastic Lotka–Volterra. Here $x_i > 0$ is the population state of species i. For $i = 0, \ldots, d-1$,

$$b_i(\boldsymbol{x}) = x_i\big[1.1 - x_i - 0.05(x_{i-1} + x_{i+1})\big], \qquad \mathbf{D}(\boldsymbol{x}) = \mathrm{Diag}(\boldsymbol{x})\mathbf{Q}\,\mathrm{Diag}(\boldsymbol{x}),$$

where $\mathbf{Q} = \mathbf{F}\mathbf{F}^\top + 0.04^2\mathbf{I}_d$ and $\mathbf{F} = \mathbf{F}_d(0.12, 0.09, 0.07, 0.05)$. The data-generating noise factor is

$$\boldsymbol{\sigma}_{\mathrm{LV}}(\boldsymbol{x}) = [\mathrm{Diag}(\boldsymbol{x})\mathbf{F} \quad 0.04\,\mathrm{Diag}(\boldsymbol{x})], \qquad r = d + 4,$$

which satisfies $\boldsymbol{\sigma}_{\mathrm{LV}}(\boldsymbol{x})\boldsymbol{\sigma}_{\mathrm{LV}}(\boldsymbol{x})^\top = \mathbf{D}(\boldsymbol{x})$. The term $1.1x_i$ gives linear growth, while $-x_i^2$ and the two neighbor terms provide self-limitation and competition. The matrix $\mathbf{Q}$ is the constant covariance used before multiplication by the population coordinates, which produces the state-dependent matrix $\mathbf{D}(\boldsymbol{x})$.

The trajectories start in the positive orthant and are simulated with $y_i = \log x_i$. Let

$$g_i(\boldsymbol{x}) = 1.1 - x_i - 0.05(x_{i-1} + x_{i+1}), \qquad Q_{ii} = \sum_{k=1}^{4} F_{ik}^2 + 0.04^2.$$

For the internal step size $h = \tau/M_{\mathrm{EM}}$, write $\boldsymbol{y}_{l,j} = \log \boldsymbol{x}_{l,j}$ and $\boldsymbol{x}_{l,j} = \exp(\boldsymbol{y}_{l,j})$. With independent standard Gaussian vectors $\boldsymbol{\xi}_{l,j} \in \mathbb{R}^4$ and $\boldsymbol{\eta}_{l,j} \in \mathbb{R}^d$, the log-coordinate Euler–Maruyama update is

$$\begin{aligned} y_{l,j+1,i} &= y_{l,j,i} + \left[g_i(\boldsymbol{x}_{l,j}) - \frac{1}{2}Q_{ii}\right] h + \sqrt{h}\left[\mathbf{F}\boldsymbol{\xi}_{l,j} + 0.04\boldsymbol{\eta}_{l,j}\right]_i, \\ x_{l,j+1,i} &= \exp(y_{l,j+1,i}). \end{aligned}$$

The correction $-Q_{ii}/2$ is the Itô correction, and the exponential recovery keeps every finite-time trajectory strictly positive. The coefficients are locally Lipschitz on the positive orthant, and the quadratic competition prevents finite-time explosion. For $V(\boldsymbol{x}) = 1 + \sum_{i=0}^{d-1} x_i$, the infinitesimal generator $\mathcal{L}$ satisfies the dissipative bound $\mathcal{L}V \le 1.1\sum_i x_i - \sum_i x_i^2$. This gives tight time-averaged laws and hence an invariant probability measure on the positive orthant $(0,\infty)^d$.

Galerkin fluid. This system is a finite-dimensional periodic fluid surrogate. We require an even dimension d and represent the real state by periodic grid values x_j, $j = 0, \ldots, d-1$. The forward-normalized discrete Fourier coefficients are

$$\widehat{x}_k = \frac{1}{d}\sum_{j=0}^{d-1} x_j e^{-2\pi \mathrm{i} kj/d}, \qquad k = 0, 1, \ldots, d/2-1, -d/2, \ldots, -1,$$

with the inverse transform given by the corresponding unnormalized Fourier sum; the inverse transform is real up to numerical roundoff. Define $\mathbf{A}_d$ as the linear operator whose Fourier multiplier is $1 + (2k/d)^2$, and let P_d retain only coefficients with $|k| \le \lfloor d/3 \rfloor$. The discrete derivative ∂_d has Fourier multiplier $\mathrm{i}\,2k/d$, where $\mathrm{i}^2 = -1$. Define the projected quadratic transport term by

$$\mathcal{B}_d(\boldsymbol{x}, \boldsymbol{x}) = P_d[(P_d\boldsymbol{x})\,\partial_d(P_d\boldsymbol{x})].$$

More explicitly, P_d sets Fourier coefficients outside the stated cutoff to zero. The product is evaluated with $3/2$ zero padding: the retained coefficients are copied to a Fourier grid of size $M = 3d/2$

at indices $k \bmod M$, transformed to the enlarged grid, multiplied pointwise with the similarly transformed derivative, transformed back, and then restricted to the retained modes before the inverse transform. This is the dealiased implementation used in the experiments. The drift and diffusion are

$$\boldsymbol{b}(\boldsymbol{x}) = -0.5\mathbf{A}_d\boldsymbol{x} - \mathcal{B}_d(\boldsymbol{x},\boldsymbol{x}) + \boldsymbol{f}, \qquad \mathbf{D}(\boldsymbol{x}) = 0.09\mathbf{I}_d, \qquad f_i = 0.25\sin\left(\frac{2\pi(i+1/2)}{d}\right),$$

for $i = 0, \ldots, d-1$. For this system the simulation factor is

$$\boldsymbol{\sigma}_{\mathrm{Gal}} = 0.3\mathbf{I}_d, \qquad r = d.$$

The quadratic term is energy preserving and the linear operator is dissipative:

$$\boldsymbol{x}^\top \mathcal{B}_d(\boldsymbol{x},\boldsymbol{x}) = 0, \qquad \boldsymbol{x}^\top \mathbf{A}_d\boldsymbol{x} \geq \|\boldsymbol{x}\|_2^2.$$

For the SDE generator $\mathcal{L}$, meaning the operator that gives the instantaneous evolution of a smooth function of the state, the squared Euclidean norm satisfies

$$\mathcal{L}\|\boldsymbol{x}\|_2^2 \leq -\frac{1}{2}\|\boldsymbol{x}\|_2^2 + 2\|\boldsymbol{f}\|_2^2 + 0.09d.$$

Here $\|\cdot\|_2$ denotes the Euclidean norm. The finite-dimensional drift is locally Lipschitz, and this dissipative bound gives a unique global strong solution and an invariant probability measure.

Damped swing network. Let $d = 2q$ and write the state as $\boldsymbol{x} = (\boldsymbol{\theta}, \boldsymbol{\omega})$, where $\boldsymbol{\theta} \in \mathbb{T}^q$ contains q angles and $\boldsymbol{\omega} \in \mathbb{R}^q$ contains angular velocities. Here $\mathbb{T}^q$ means that each angle is identified modulo 2π. For $i = 0, \ldots, q-1$, the drift is

$$\begin{aligned} b_{\theta,i}(\boldsymbol{x}) &= \omega_i, \\ b_{\omega,i}(\boldsymbol{x}) &= -0.5\omega_i - \sin\theta_i + 0.2\big[\sin(\theta_{i-1} - \theta_i) + \sin(\theta_{i+1} - \theta_i)\big]. \end{aligned}$$

The diffusion matrix is

$$\mathbf{D}(\boldsymbol{x}) = \mathbf{F}\mathbf{F}^\top + \mathrm{Diag}\big(0.05^2\mathbf{1}_q, 0.25^2\mathbf{1}_q\big), \qquad \mathbf{F} = \begin{bmatrix} \mathbf{0}_{q\times 4} \\ \mathbf{F}_q(0.22, 0.16, 0.12, 0.08) \end{bmatrix}.$$

The corresponding data-generating factor is

$$\boldsymbol{\sigma}_{\mathrm{sw}} = [\mathbf{F} \;\; \mathrm{Diag}(0.05\mathbf{1}_q, 0.25\mathbf{1}_q)], \qquad r = 2q + 4,$$

and it satisfies $\boldsymbol{\sigma}_{\mathrm{sw}}\boldsymbol{\sigma}_{\mathrm{sw}}^\top = \mathbf{D}$. Here $\mathbf{0}_{q\times 4}$ is the $q \times 4$ zero matrix and $\mathbf{1}_q$ is the length-q vector of ones; in the diagonal term, the two vectors are concatenated in the order $(\boldsymbol{\theta}, \boldsymbol{\omega})$. The low-rank noise and the 0.25^2 diagonal term act on angular velocities, while the 0.05^2 term adds independent noise to the angles. After each simulation step, every angle is mapped back to $[-\pi, \pi)$. The drift is globally Lipschitz on the lifted space, and positive damping gives $\mathcal{L}(1 + \|\boldsymbol{\omega}\|_2^2) \leq -c\|\boldsymbol{\omega}\|_2^2 + C_d$ for constants $c > 0$ and $C_d > 0$ independent of the state. Because the angle space $\mathbb{T}^q$ is compact, this bound yields an invariant probability measure and, together with global Lipschitz continuity, a unique global strong solution.

Initial states. Let $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ denote independent standard Gaussian vectors. For OU, Galerkin fluid, and Lotka–Volterra, $\boldsymbol{\xi} \in \mathbb{R}^d$; for the damped swing network, $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbb{R}^q$. The initial states are $0.1\boldsymbol{\xi}$ for OU and the Galerkin fluid system, $\exp(0.05\boldsymbol{\xi})$ for Lotka–Volterra, and $(0.2\boldsymbol{\xi}, 0.1\boldsymbol{\eta})$ for the damped swing network.

A.3 Baseline Implementations

All baselines use observations only. Let $\boldsymbol{x}_l, \boldsymbol{x}_{l+1} \in \mathbb{R}^d$ be two consecutive observed states, where d is the state dimension and τ is the observation interval. We define the drift-corrected residual by

$$\boldsymbol{r}_l := \boldsymbol{x}_{l+1} - \boldsymbol{x}_l - \tau\widetilde{\boldsymbol{b}}(\boldsymbol{x}_l).$$

The neural baselines use the network architecture described in Appendix A.4.

SparseNN. Following the two-stage procedure of Ren et al. (2024), we first fit the drift by regressing the scaled increment $(\boldsymbol{x}_{l+1} - \boldsymbol{x}_l)/\tau$ on $\boldsymbol{x}_l$. We then keep the fitted drift fixed and fit the diffusion by regressing $\boldsymbol{r}_l \boldsymbol{r}_l^\top/\tau$ on $\boldsymbol{x}_l$. See Algorithm 1 in Ren et al. (2024) for details. The reported diffusion errors are evaluated for $\mathbf{D}$ itself, rather than $\mathbf{D}/2$ as used in Ren et al. (2024).

EM-qMLE. We approximate one transition by a Gaussian distribution with mean $\boldsymbol{x} + \tau\widetilde{\boldsymbol{b}}(\boldsymbol{x})$ and covariance $\tau\widetilde{\mathbf{D}}(\boldsymbol{x})$, as implied by the Euler discretization. The drift and diffusion are fitted jointly by minimizing the corresponding Gaussian negative log-likelihood, divided by the state dimension d.

DynGMA. We use the single-step midpoint Gaussian version of DynGMA (Zhu & Li, 2024). For a current state $\boldsymbol{x}$, define the midpoint

$$\boldsymbol{z} = \boldsymbol{x} + \frac{\tau}{2}\widetilde{\boldsymbol{b}}(\boldsymbol{x}),$$

and set the transition mean and covariance to

$$\boldsymbol{\mu} = \boldsymbol{x} + \tau\widetilde{\boldsymbol{b}}(\boldsymbol{z}), \qquad \boldsymbol{\Sigma} = \tau\widetilde{\mathbf{D}}(\boldsymbol{z}).$$

We minimize the negative log-likelihood of this Gaussian transition. This is the single-step adaptation used in our experiments, rather than the full multistep Gaussian-mixture construction.

EDMD-Polynomial and EDMD-RFF. Let $\boldsymbol{\psi}(\boldsymbol{x})$ be a row vector of prescribed dictionary functions. The rows of $\boldsymbol{\Psi}_0$ and $\boldsymbol{\Psi}_1$ are respectively $\boldsymbol{\psi}(\boldsymbol{x}_l)$ and $\boldsymbol{\psi}(\boldsymbol{x}_{l+1})$ for all observed transitions. We compute the regularized discrete Koopman matrix

$$\widehat{\mathbf{K}} = (\boldsymbol{\Psi}_0^\top \boldsymbol{\Psi}_0 + 10^{-4}\mathbf{I})^{-1}\boldsymbol{\Psi}_0^\top \boldsymbol{\Psi}_1, \qquad \widehat{\mathbf{G}} = (\widehat{\mathbf{K}} - \mathbf{I})/\tau.$$

Here $\mathbf{I}$ is the identity matrix of the dictionary dimension. The matrix $\widehat{\mathbf{G}}$ is a first-order approximation of the generator, the operator that describes the infinitesimal evolution of the dictionary functions. We then fit the $\widetilde{\boldsymbol{b}}$ and $\widetilde{\mathbf{D}}$ so that applying $\widetilde{\mathcal{L}}$ to $\boldsymbol{\psi}(\boldsymbol{x})$ matches $\boldsymbol{\psi}(\boldsymbol{x})\widehat{\mathbf{G}}$. EDMD-Polynomial uses all monomials of total degree at most two, giving $\binom{d+2}{2}$ dictionary functions. EDMD-RFF uses 512 sine and cosine features plus a constant function. Its frequencies are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I}_d/d)$, and its phases are sampled uniformly. Neither EDMD-Polynomial nor EDMD-RFF use the matrix logarithm $\widehat{\mathbf{G}}_{\log} = \frac{1}{\tau}\log(\widehat{\mathbf{K}})$. Instead, both methods use the first-order generator approximation $\widehat{\mathbf{G}} = \frac{\widehat{\mathbf{K}}-\mathbf{I}}{\tau}$, which avoids the numerical and computational issues associated with matrix-logarithm evaluation.

SINDy. SINDy (Boninsegna et al., 2018) uses all monomials of total degree at most three as candidate functions. We perform five rounds of least-squares fitting, removing coefficients whose magnitudes fall below a threshold after each round. The coefficient threshold and the ridge penalty are both 10^{-3}. The drift is fitted from scaled increments, after which the diffusion is estimated as the constant matrix

$$\widetilde{\mathbf{D}} = \frac{1}{n\tau}\sum_{l=1}^{n} \boldsymbol{r}_l \boldsymbol{r}_l^\top,$$

where n is the number of observed transitions. This adaptation cannot represent state-dependent diffusion and does not add system-specific terms such as the sine terms of the swing network.

KoopSDE (SDGD). The specific implementation of KoopSDE (SDGD) is based on the following expression:

$$\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k) = \sum_{i,j=1}^{d} \widetilde{\mathbf{D}}_{ij}\,\frac{\partial^2 \widetilde{\phi}_k}{\partial x_i \partial x_j}, \tag{12}$$

where $\widetilde{\mathbf{D}}_{ij}$ denotes the (i, j) component of the matrix $\widetilde{\mathbf{D}}$. Eq. (12) transforms the computation of $\mathrm{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ into a sum over d^2 second-order partial derivatives. Computing it fully incurs a time and space complexity of $\mathcal{O}(d^2 \times |\boldsymbol{\theta}|)$. KoopSDE (SDGD) keeps the KoopSDE networks, initialization, and loss unchanged, but replaces the exact Hessian trace calculation by a coordinate-block estimator.

Specifically, at each update, we sample a coordinate set $S \subset \{1, \dots, d\}$ without replacement and set $s = |S| = \mathrm{round}(\sqrt{0.15}\, d)$, where round denotes rounding to the nearest integer. The unbiased coordinate-block estimate is

$$\widehat{\mathrm{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)_{\mathrm{SDGD}} = \frac{d}{s} \sum_{i \in S} \widetilde{\mathbf{D}}_{ii} (\mathbf{H}_k)_{ii} + \frac{d(d-1)}{s(s-1)} \sum_{\substack{i,j \in S \\ i \neq j}} \widetilde{D}_{ij} (\mathbf{H}_k)_{ji}.$$

The first term corrects for sampling diagonal entries and the second corrects for sampling off-diagonal entries, so the estimator is unbiased (Hu et al., 2024b). The selected block contains approximately 15% of the d^2 Hessian entries. We evaluate the required second derivatives through Hessian–vector products, which evaluate $\mathbf{H}_k$ applied to a vector without explicitly forming the full Hessian matrix. Only the trace estimator changes relative to KoopSDE. The empirical comparison is reported in Table 2 and Appendix A.8. Unlike the specialized trace estimators considered above, SDGD is a broadly applicable stochastic approximation method that can be used for various differential operators and higher-order derivative calculations, including the solution of partial differential equations with physics-informed neural networks (PINNs) (Hu et al., 2024b; Shi et al., 2024).

A.4 Experiment Setup

Reproducibility Statement To facilitate reproducibility, we provide an anonymous code repository at `https://anonymous.4open.science/r/KoopSDE-Code-1185/`. The repository contains the implementation of KoopSDE and the baselines, the experimental configurations, and scripts for reproducing the reported results and generating the main figures and tables.

Parameterization. We employ Fourier feature embedding to map the input $\boldsymbol{x}$ before feeding it into the MLP, enhancing its ability to learn high-frequency eigenfunctions. This is a standard preprocessing step for MLPs (Appendix A.11). Inspired by the efficient fine-tuning technique LoRA (Low-Rank Adaptation, Hu et al. (2022)) used in large language models (LLMs), we parameterize the diffusion function $\widetilde{\mathbf{D}}(\boldsymbol{x})$ using the method described in Appendix A.12.

Data and simulation. For each SDE, dimension, observation interval, and random seed, we generate a trajectory with 40,000 observations using the Euler–Maruyama scheme described in Appendix A.10. The trajectory is split into training and validation sets with an 80%/20% ratio. Each SDE is evaluated with three independent seeds, $0, 1, 2$, and all methods share the same training set, validation set, and an independent test set containing 10,000 states. For $\tau = (0.05, 0.1, 0.25)$, the Euler–Maruyama substep counts are $(4, 8, 32)$ for OU, $(1, 2, 4)$ for Lotka–Volterra, $(4, 16, 32)$ for the Galerkin fluid system, and $(8, 16, 32)$ for the damped swing network. Lotka–Volterra is simulated in log coordinates to preserve positivity of the population states.

Architectures. For $d < 100$, $100 \leq d \leq 1000$, and $d > 1000$, the hidden-layer depth and width are $(4, 128)$, $(4, 256)$, and $(5, 256)$, respectively, with SiLU activations. The number of eigenfunctions is $N = 6, 9, 12$ for $d < 100$, $100 \leq d < 1000$, and $d \geq 1000$, respectively. Thus, at $d = 1000$ the network uses the middle architecture group $(4, 256)$, while the number of eigenfunctions is $N = 12$. The eigenfunction network uses 32 fixed Fourier frequencies with scale $0.3/\sqrt{d}$.

Optimization. We use the FP32 AdamW optimizer (Loshchilov & Hutter, 2017) with zero weight decay and clip the gradient norm at 10. The first-stage learning rate starts at 3×10^{-4} and follows a cosine decay. In the second stage, the learning rates for the drift and diffusion networks are 10^{-3} and 5×10^{-5}, respectively. In practice, the orthogonality loss typically converges to zero, so no dedicated tuning of α is needed; we set $\alpha = 1.0$ throughout.

For the update-budget groups $d \leq 100$, $100 < d \leq 5000$, and $d > 5000$, the first-stage update budgets are 3,000, 1,500, and 600, respectively, while the corresponding second-stage budgets are 1,000, 600, and 300. Every 50 parameter updates, we evaluate the validation loss using at most 1,024 validation pairs. The validation set is not used for parameter updates; it is used only to monitor improvement and select the best checkpoint. Second-stage training uses early stopping, but runs for at least 200 updates. Thereafter, training stops if the validation loss does not improve for 300 consecutive updates.

Within the same dimension groups, the wall-time limits for each stage or baseline are 2, 3, and 5 minutes, respectively. Neural baselines use a learning rate of 10^{-3} and the same dimension-dependent second-stage update budgets.

A.5 Observation-Interval Ablation

This section presents the main-experiment error distributions at $\tau = 0.1$, followed by GPU memory and relative-error results at $\tau = 0.05$ and 0.25. All comparisons use the same four SDEs, three random seeds, model families, and dimension-dependent training budgets. KoopSDE (HTE) uses $m = 10$ throughout. The number of observations is fixed, so changing τ also changes the observed physical time horizon. The comparison therefore assesses performance at different sampling intervals.

Evaluation metrics. For test states $\{\boldsymbol{x}_l\}_{l=1}^{n_{\text{test}}}$ shared by all methods, we measure the relative errors of the learned drift and diffusion functions as

$$\mathrm{RelErr}_b = \left(\frac{\sum_{l=1}^{n_{\text{test}}} \|\widetilde{\boldsymbol{b}}(\boldsymbol{x}_l) - \boldsymbol{b}(\boldsymbol{x}_l)\|_2^2}{\sum_{l=1}^{n_{\text{test}}} \|\boldsymbol{b}(\boldsymbol{x}_l)\|_2^2} \right)^{1/2}, \quad \mathrm{RelErr}_D = \left(\frac{\sum_{l=1}^{n_{\text{test}}} \|\widetilde{\mathbf{D}}(\boldsymbol{x}_l) - \mathbf{D}(\boldsymbol{x}_l)\|_F^2}{\sum_{l=1}^{n_{\text{test}}} \|\mathbf{D}(\boldsymbol{x}_l)\|_F^2} \right)^{1/2}.$$

Here, $\boldsymbol{b}$ and $\mathbf{D}$ are the true functions, and tildes denote their learned estimates. The norms $\|\cdot\|_2$ and $\|\cdot\|_F$ are the Euclidean and Frobenius norms, respectively. Both metrics normalize the aggregate squared estimation error by the corresponding aggregate squared true-function norm. Lower values indicate better accuracy, and a value of 0.1 corresponds to a 10% relative error.

Plotting and reporting conventions. All three error figures use the same format. Each figure contains an upper drift-error block and a lower diffusion-error block, each with eight dimension panels. Colors identify the nine methods and marker shapes identify the four SDEs.

For each method and system, relative errors are averaged over available seeds. Boxes summarize these system-level means, with a median line, quartile boundaries, and whiskers extending to the most extreme values within 1.5 interquartile ranges. System means are overlaid as points. The annotation $\uparrow k/4$ counts the systems whose mean error for the displayed metric exceeds 1. These points are omitted, but all finite values, including relative errors greater than 1, are included in the boxplot statistics before the plot is clipped to the displayed range. When results are available for only one system, only the point is shown.

All panels use linear axes starting at zero. Each panel has an independently chosen upper limit capped at 1 to prevent extreme errors from compressing the boxes of other methods and obscuring their differences. Dotted lines mark 10% error.

The memory tables follow Table 2 and average peak allocated GPU memory over completed runs across systems and seeds. Bold marks the lowest value at the displayed precision. In these tables, OOM denotes GPU memory exhaustion under the 80 GiB limit, NaN denotes numerical failure. Reported mean errors average seeds within each system and then weight available systems equally.

$\tau = 0.1$. Figure 2 shows the system-level relative drift and diffusion error distributions. Its boxes show medians and quartiles of the system-level seed averages. KoopSDE (HTE) maintains relative drift and diffusion errors around 10% across the tested dimensions, with accuracy comparable to its vanilla and SDGD variants where these are available. At $d = 500$ and 1000, its diffusion errors are lower than those of SparseNN, EM-qMLE, and DynGMA, while their drift errors remain similar.

The EDMD baselines exhibit larger errors and substantial variation across systems, with several system-level means exceeding 1. At $d = 100{,}000$, the mean relative drift and diffusion errors of KoopSDE (HTE) over available results are 11.40% and 8.68%, respectively, indicating that its accuracy is maintained as the dimension increases.

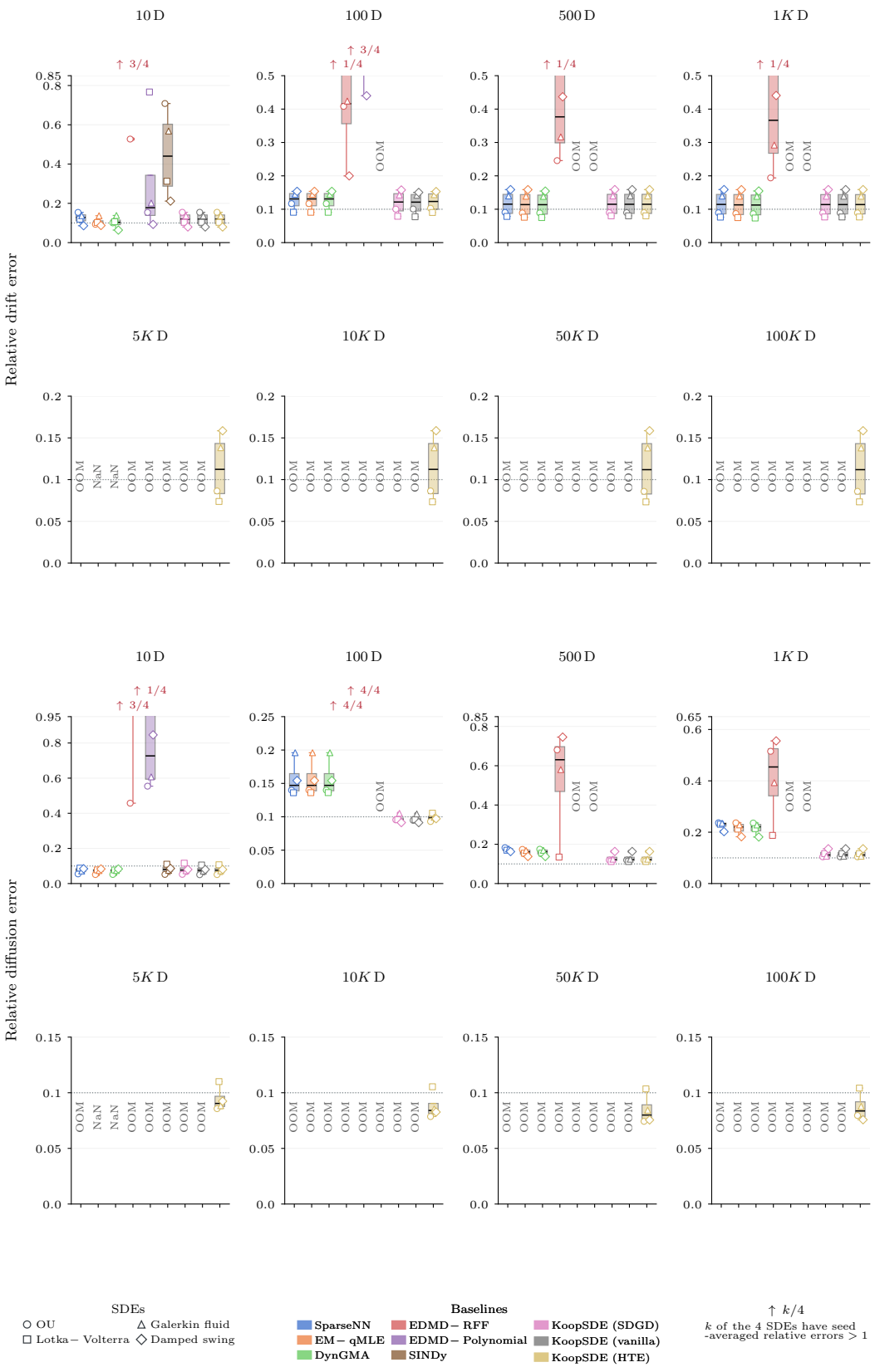

Figure 2: Relative drift and diffusion errors at $\tau = 0.1$ across eight dimensions. The upper block shows drift errors and the lower block shows diffusion errors. Plotting conventions are defined at the start of Appendix A.5.

$\tau = 0.05$. Table 3 and Figure 3 summarize the GPU memory and relative-error results at the shorter observation interval. KoopSDE (HTE) retains a stable drift error between 12.8% and 14.2% over all eight dimensions. Its diffusion error is lowest at 10 D (4.8%), increases to 12.9% at 500 D, and then remains between 7.2% and 10.9% at larger dimensions. Thus, reducing the observation interval does not produce a systematic deterioration in either learned function. KoopSDE likewise supports the learning task at 100,000 D, with the HTE memory footprint increasing from 0.14 GiB to 12.99 GiB. In contrast, most external baselines become OOM or unavailable before the largest dimensions, and the isolated EM-qMLE result at 5K D is based on one completed run rather than a four-system comparison. Several EDMD and SINDy system means exceed one at low dimensions, as indicated by the overflow annotations in Figure 3.

Table 3: Mean peak allocated GPU memory at $\tau = 0.05$. Lower is better. Aggregation and status conventions are defined at the start of Appendix A.5.

Method	GPU Memory (GiB) (↓)							
	10 D	100 D	500 D	1K D	5K D	10K D	50K D	100K D
SparseNN	**0.02**	0.17	1.17	4.59	OOM	OOM	OOM	OOM
EM-qMLE	**0.02**	**0.13**	0.93	3.63	63.08	OOM	OOM	OOM
DynGMA	0.03	0.16	1.07	4.17	NaN	OOM	OOM	OOM
EDMD-RFF	0.05	0.30	1.06	4.11	OOM	OOM	OOM	OOM
EDMD-Polynomial	**0.02**	0.52	OOM	OOM	OOM	OOM	OOM	OOM
SINDy	0.08	OOM	OOM	OOM	OOM	OOM	OOM	OOM
KoopSDE (SDGD)	0.14	1.09	1.47	7.15	OOM	OOM	OOM	OOM
KoopSDE (vanilla)	0.14	2.66	3.55	17.81	OOM	OOM	OOM	OOM
KoopSDE (HTE)	0.14	0.38	**0.19**	**0.40**	**1.07**	**1.33**	**6.51**	**12.99**

$\tau = 0.25$. Table 4 and Figure 4 report the corresponding results at the longer observation interval. The drift error of KoopSDE (HTE) remains tightly concentrated between 12.3% and 13.4%. The diffusion error is higher than at $\tau = 0.05$, ranging from 15.0% to 19.0%, with the largest values occurring at the lower dimensions and a gradual reduction at higher dimensions. The HTE memory profile is unchanged across the tested dimensions, from 0.14 GiB at 10 D to 12.99 GiB at 100,000 D, while the other methods are increasingly affected by OOM or unavailable runs. These results indicate that the longer sampling interval mainly affects diffusion identification, consistent with lower temporal resolution and the finite-lag approximations in the training objective, rather than introducing a substantial memory overhead.

Table 4: Mean peak allocated GPU memory at $\tau = 0.25$. Lower is better. Aggregation and status conventions are defined at the start of Appendix A.5.

Method	GPU Memory (GiB) (↓)							
	10 D	100 D	500 D	1K D	5K D	10K D	50K D	100K D
SparseNN	**0.02**	0.17	1.17	4.59	OOM	OOM	OOM	OOM
EM-qMLE	**0.02**	**0.13**	0.93	3.63	63.08	OOM	OOM	OOM
DynGMA	0.03	0.16	1.07	4.17	NaN	OOM	OOM	OOM
EDMD-RFF	0.05	0.30	1.06	4.11	OOM	OOM	OOM	OOM
EDMD-Polynomial	**0.02**	0.52	OOM	OOM	OOM	OOM	OOM	OOM
SINDy	0.08	OOM	OOM	OOM	OOM	OOM	OOM	OOM
KoopSDE (SDGD)	0.14	1.09	1.47	7.15	OOM	OOM	OOM	OOM
KoopSDE (vanilla)	0.14	2.66	3.55	17.81	OOM	OOM	OOM	OOM
KoopSDE (HTE)	0.14	0.38	**0.19**	**0.40**	**1.07**	**1.33**	**6.51**	**12.99**

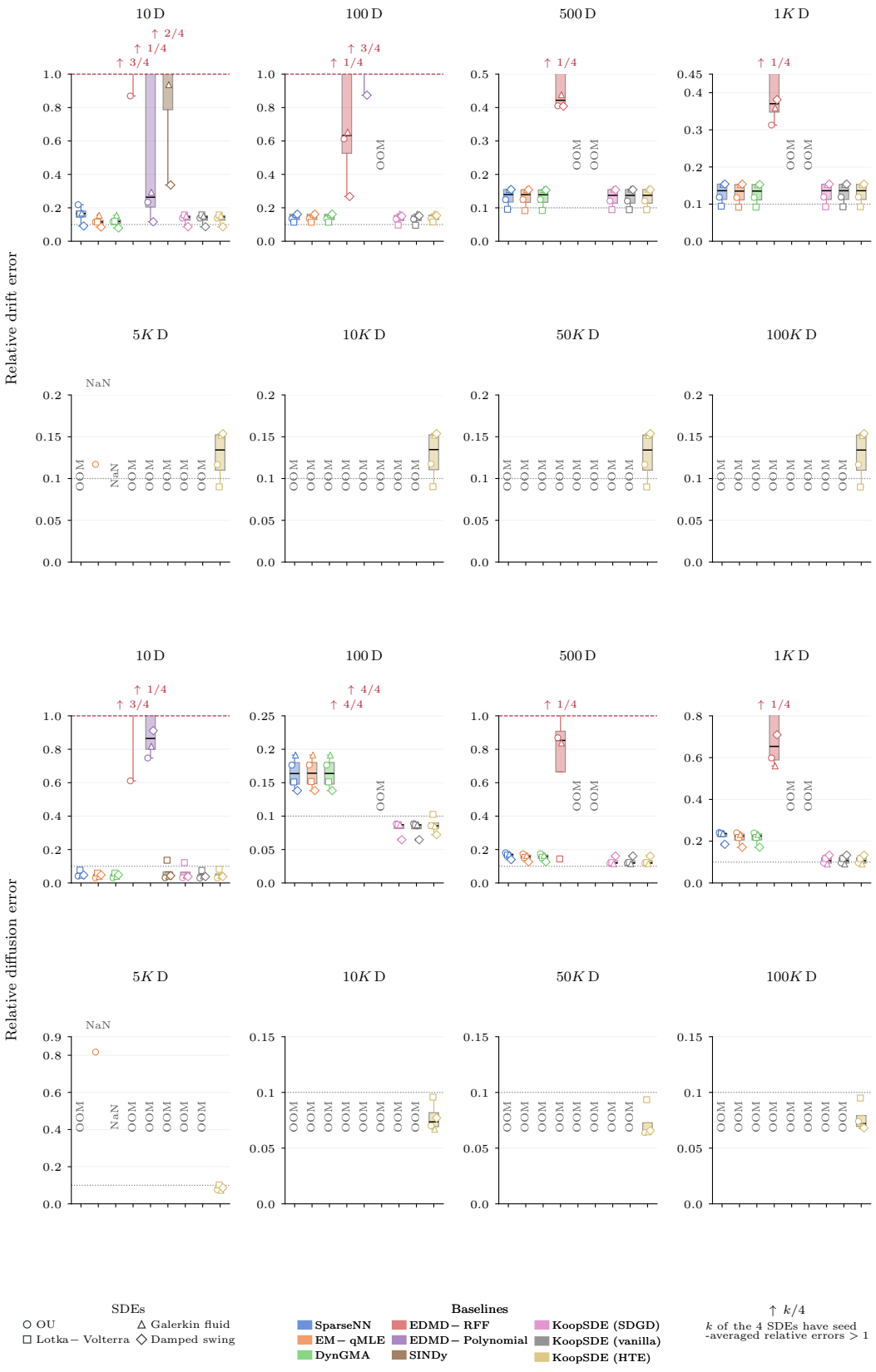

Figure 3: Relative drift and diffusion errors at $\tau = 0.05$ across eight dimensions. The upper block shows drift errors and the lower block shows diffusion errors. Plotting conventions are defined at the start of Appendix A.5.

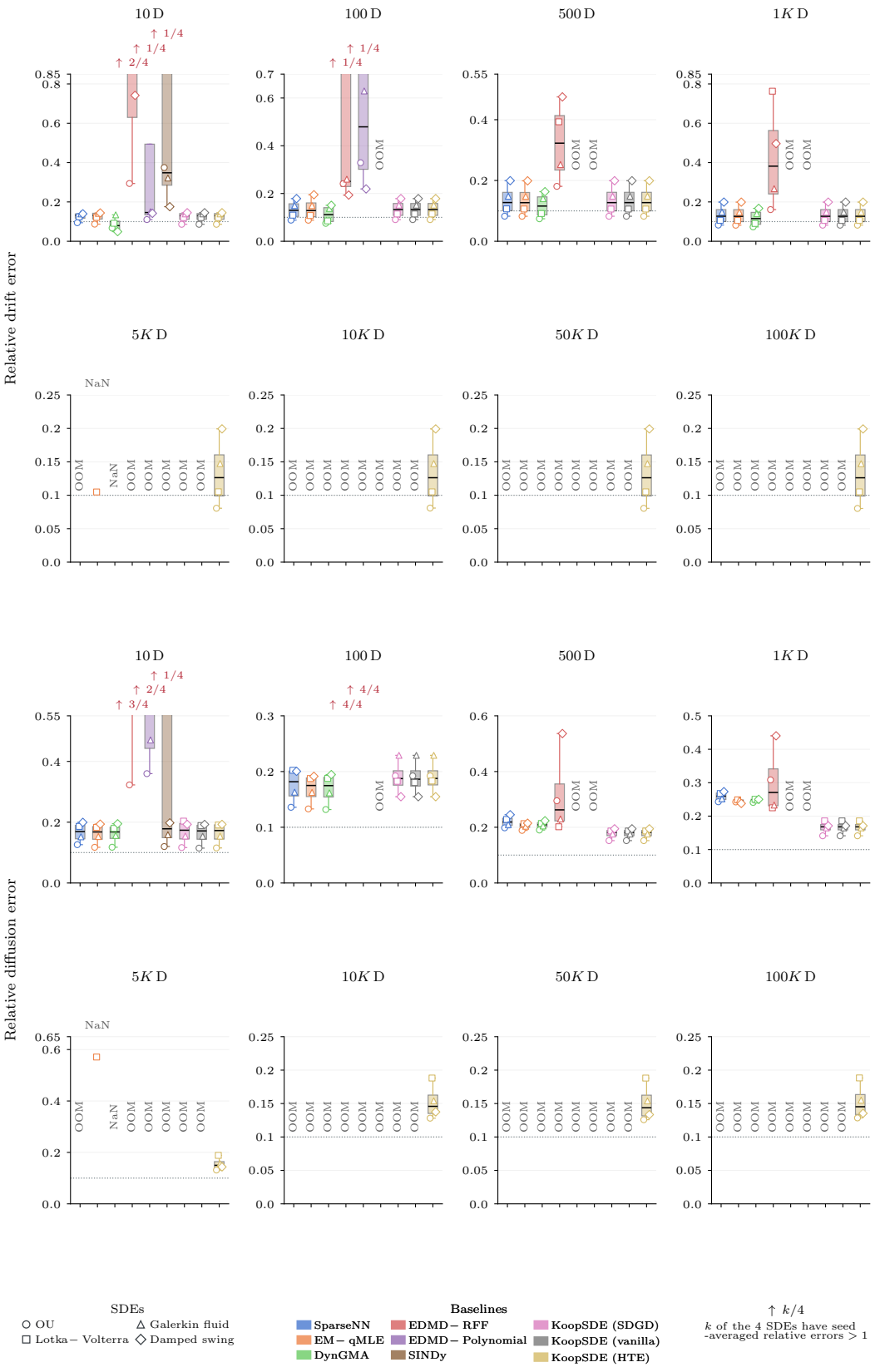

Figure 4: Relative drift and diffusion errors at $\tau = 0.25$ across eight dimensions. The upper block shows drift errors and the lower block shows diffusion errors. Plotting conventions are defined at the start of Appendix A.5.

Effect of the observation interval. Across the three settings, changing τ has a limited effect on drift identification but a clearer effect on diffusion identification. The drift error of KoopSDE (HTE) remains within a narrow range of approximately 12%–14%, whereas the diffusion error is lowest for the shorter intervals and increases at $\tau = 0.25$. This trend is consistent with the loss of temporal resolution at larger observation intervals, which makes the finite-lag approximation of second-order effects less informative. The HTE memory profile is essentially unchanged across τ, and the method remains able to complete the $100{,}000$-dimensional learning task. Thus, within the tested range, τ primarily controls the accuracy of diffusion recovery, while the memory scaling and high-dimensional feasibility remain stable.

A.6 Influence Mechanism of N

We investigate how the number of eigenfunctions learned in the first stage affects the relative drift and diffusion errors in this section.

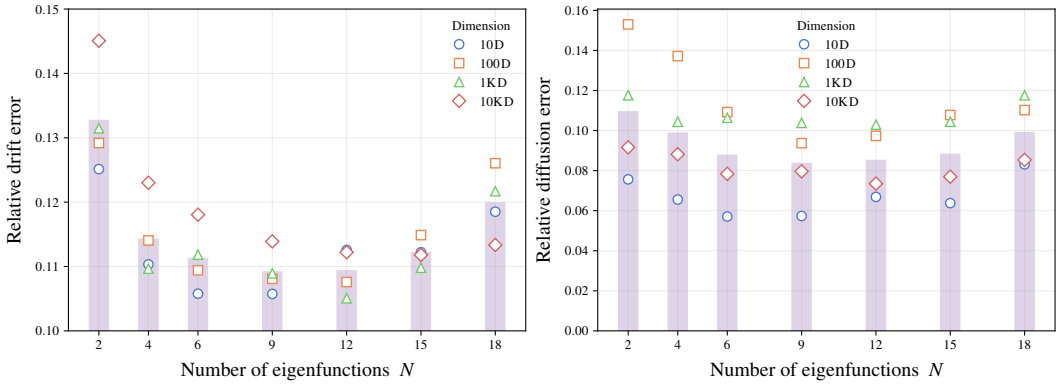

Figure 5: Relative drift and diffusion errors as functions of the number of learned eigenfunctions N. The left and right panels report the relative drift and diffusion errors, respectively, across four representative dimensions. Each relative error is computed by averaging the results across the four SDE systems. The bars show the arithmetic mean of the relative errors over the four displayed dimensions at each N.

As shown in Figure 5, both the relative drift and diffusion errors initially decrease and then increase as the number of eigenfunctions N learned in the first stage grows. This behavior can be explained as follows. When the parameter capacity of the eigenfunction network $\widetilde{\Phi}$ is fixed and only its output dimension N is changed, increasing N requires the network to learn more eigenfunction–eigenvalue pairs $(\widetilde{\phi}, \widehat{\lambda})$. Each accurately learned pair provides an effective constraint for the second-stage learning. Therefore, increasing N within a suitable range can improve the accuracy of the learned $\boldsymbol{b}(\boldsymbol{x})$ and $\mathbf{D}(\boldsymbol{x})$.

However, when N becomes too large relative to the capacity of $\widetilde{\Phi}$, the network may fail to accurately represent all N eigenfunctions. The resulting eigenfunction–eigenvalue pairs become less accurate, which weakens the constraints imposed on the second-stage learning and causes the relative drift and diffusion errors to increase. Thus, using a larger N generally requires increasing the parameter count of $\widetilde{\Phi}$ accordingly to improve its expressive capacity.

At the same time, N should not be chosen excessively large. According to the oscillation theorem in Sturm–Liouville theory (Karlin & Taylor, 1981), the eigenvalue spectrum of most stationary SDEs is discrete and tends to negative infinity, while the corresponding eigenfunctions exhibit increasingly strong oscillations. Combined with modal decay, $e^{\lambda_k \tau}$ is typically small for the higher-order modes added as N increases. These modes decay rapidly over the interval τ and contribute little to the dynamics at that time scale. Even if they are learned accurately, they provide limited additional constraints for the second-stage learning while introducing unnecessary computational overhead.

Spectral guidance for choosing N. Lemma 5.3 and Corollary 5.4 provide a direct criterion for selecting the number of eigenfunctions. Under exact orthogonality, the dynamic loss satisfies

$$\mathcal{L}_{\mathrm{dyn}} = N - \sum_{k=1}^{N} e^{2\widehat{\lambda}_k \tau},$$

so minimizing this loss favors the N largest Koopman eigenvalues. Accordingly, N should be large to include slowly decaying modes with non-negligible contributions, but need not be increased further once the remaining contributions are negligible or a clear spectral gap appears.

This spectral benefit must be balanced against the estimation error. In particular, the empirical eigenfunction-learning bound in Theorem B.1 contains terms that grow with N, including terms of order $N\varepsilon_\Phi^2$ and N^2/n, where ε_Φ denotes the approximation error of the eigenfunction network and n is the trajectory length. Consequently, increasing N improves the second-stage constraints only while the additional modes remain accurately representable and statistically resolvable. A practical choice is therefore the smallest N for which the next-mode contribution falls below a prescribed threshold,

$$N^\star = \min\left\{N : \; e^{2\widehat{\lambda}_{N+1}\tau} \le \epsilon_{\mathrm{mode}}\right\},$$

with ϵ_{mode} chosen, for example, as 0.01 or 0.05. When the spectrum is not sharply separated, the same principle can be implemented by selecting the smallest N at which the validation errors reach a stable plateau. This provides a theoretically motivated interpretation of the candidate range used in our ablation study and of the default choice $N \in \{6, 9, 12\}$ as a compromise between spectral coverage and estimation complexity.

A.7 PEMS-BAY Real-data Evaluation

PEMS-BAY is a public traffic-speed dataset collected from 325 sensors in the San Francisco Bay Area. It contains 52,128 time points sampled every five minutes from January through June 2017. At each time point, the observation is a vector $\boldsymbol{y}_t \in \mathbb{R}^d$ with $d = 325$, whose entries are average vehicle speeds in miles per hour (mph). We use two time origins fixed before testing. Each origin defines a one-hour forecast window containing 12 successive five-minute steps. The two windows cover different traffic conditions, reducing the dependence on any single period and providing a simple check of predictive stability. For each window, the model is fitted only with observations before the origin and is then evaluated on the following 12 time points.

We first evaluate one-step probabilistic prediction. In a transition from time t to time $t+1$, $t+1$ denotes the next five-minute time point, and $\boldsymbol{y}_{t+1}$ is its true speed vector. The set $\mathcal{O}_{t+1}$ contains exactly those sensors with a recorded speed at that target time. Given the state at time t, the model produces a next-step predictive mean $\widehat{\boldsymbol{\mu}}_t$ and covariance $\widehat{\boldsymbol{\Sigma}}_t$, both expressed in physical speed units. The subscript t identifies the prediction origin, whereas the subscript $t+1$ identifies the target observations. For the observed sensors, the prediction residual is

$$\boldsymbol{r}_{t,\mathcal{O}_{t+1}} = \boldsymbol{y}_{t+1,\mathcal{O}_{t+1}} - \widehat{\boldsymbol{\mu}}_{t,\mathcal{O}_{t+1}}.$$

Here, $\widehat{\boldsymbol{\Sigma}}_{t,\mathcal{O}_{t+1}}$ is the covariance submatrix for the same observed sensors. If $|\mathcal{O}_{t+1}|$ is their number, the joint Gaussian negative log-likelihood for this transition is included in the following average:

$$\mathrm{NLL} = \frac{1}{\sum_t |\mathcal{O}_{t+1}|} \sum_t \frac{1}{2}\left[\boldsymbol{r}_{t,\mathcal{O}_{t+1}}^{\mathsf{T}} \widehat{\boldsymbol{\Sigma}}_{t,\mathcal{O}_{t+1}}^{-1} \boldsymbol{r}_{t,\mathcal{O}_{t+1}} + \log\det \widehat{\boldsymbol{\Sigma}}_{t,\mathcal{O}_{t+1}} + |\mathcal{O}_{t+1}|\log(2\pi)\right].$$

In this expression, the quadratic term measures the residual relative to the predicted covariance, while the log-determinant term penalizes an unnecessarily diffuse predictive distribution. The sum over t runs over the one-step transitions in the test window, and the denominator counts all sensor–time targets with an observed speed. Sensors without a target observation are omitted from both the covariance submatrix and the count. Before fitting, each sensor is standardized using its mean and sample standard deviation from the fitting prefix only. At evaluation time, the predictive mean and covariance are transformed back to mph, with the corresponding fixed scale correction included in the log-likelihood; no test-window statistics are used. Lower NLL indicates better probabilistic prediction of the next five-minute state.

We then evaluate one-hour trajectory prediction. For each test window, all paths start from the last observed speed vector before the forecast origin, denoted by $\boldsymbol{y}_t$. We generate $S = 100$ stochastic paths. Here, $s \in \{1, \ldots, S\}$ indexes a path, and $h \in \{1, \ldots, 12\}$ indexes a five-minute forecast step. The model advances each path recursively with newly sampled diffusion noise and never receives future observations. Let $\widehat{\boldsymbol{y}}_{t+h}^{(s)}$ be the speed vector produced by path s at step h. The path-mean prediction is

$$\overline{\boldsymbol{y}}_{t+h} = \frac{1}{S} \sum_{s=1}^{S} \widehat{\boldsymbol{y}}_{t+h}^{(s)}.$$

For each step h, let $\mathcal{O}_{t,h}$ be the set of sensors with a recorded speed at time $t + h$, and let i index a sensor in this set. The one-hour rollout mean absolute error is

$$\mathrm{MAE}_{1\mathrm{h}} = \frac{1}{\sum_{h=1}^{12} |\mathcal{O}_{t,h}|} \sum_{h=1}^{12} \sum_{i \in \mathcal{O}_{t,h}} \left| \overline{y}_{t+h,i} - y_{t+h,i} \right|,$$

where $\overline{y}_{t+h,i}$ and $y_{t+h,i}$ are the predicted and recorded speeds of sensor i, respectively. Thus, the numerator sums the absolute speed errors over all valid sensors and all 12 forecast steps, while the denominator is the number of valid sensor–time targets. The resulting MAE is reported in mph; a lower value means that the predicted one-hour trajectory is closer to the observed traffic speeds.

A.8 Ablation Study of m

Table 5: This table compares the second-stage training speed and peak GPU memory of KoopSDE variants across different numbers of Hutchinson probes m, with vanilla and SDGD as references. Experiments were conducted on an NVIDIA A800 GPU (80 GiB). OOM indicates that the memory requirement exceeded 80 GiB. Each configuration was warmed up for 20 updates, followed by 100 timed updates. Speed denotes the number of updates completed per second during the measurement window.

Metric	Method	10 D	100 D	500 D	1K D	5K D	10K D	50K D	100K D
Memory (GiB) (↓)	KoopSDE (HTE, $m = 1$)	**0.052**	**0.125**	**0.099**	**0.178**	**0.525**	**0.796**	**3.832**	**7.628**
	KoopSDE (HTE, $m = 10$)	0.140	0.379	0.185	0.397	1.064	1.332	6.511	12.991
	KoopSDE (HTE, $m = 25$)	0.286	0.801	0.331	0.761	1.964	2.229	10.986	21.936
	KoopSDE (HTE, $m = 50$)	0.524	1.507	0.571	1.374	3.463	3.724	18.440	36.842
	KoopSDE (vanilla)	0.140	2.884	4.662	23.627	OOM	OOM	OOM	OOM
	KoopSDE (SDGD)	0.081	1.171	1.808	8.898	OOM	OOM	OOM	OOM
Second-stage Speed (it/s) (↑)	KoopSDE (HTE, $m = 1$)	32.899	32.578	**30.741**	**30.908**	**23.799**	**18.640**	**11.415**	**7.156**
	KoopSDE (HTE, $m = 10$)	32.399	**32.707**	30.021	30.482	23.325	17.270	8.993	5.257
	KoopSDE (HTE, $m = 25$)	32.375	30.009	30.209	29.455	21.230	16.047	6.667	3.865
	KoopSDE (HTE, $m = 50$)	33.551	23.681	30.332	26.584	18.259	13.902	4.963	2.638
	KoopSDE (vanilla)	33.424	17.206	11.142	3.845	OOM	OOM	OOM	OOM
	KoopSDE (SDGD)	**33.680**	26.655	20.126	10.140	OOM	OOM	OOM	OOM
First-stage Speed (it/s) (↑)	KoopSDE	**244.38**	**237.15**	**231.68**	**223.11**	**190.54**	**169.93**	**99.35**	**71.24**

Increasing m draws more Hutchinson probes for the second-stage trace estimate. This can reduce the sampling variance of the Diff-Hess trace estimator, but each additional probe requires further Hessian-vector computations and associated intermediate storage. Consequently, the memory cost rises with m, while the measured update throughput generally falls as dimension increases. At 100K D, increasing m from 1 to 50 raises peak memory from 7.628 to 36.842 GiB and reduces speed from 7.156 to 2.638 it/s. Vanilla and SDGD encounter OOM at 5K D and above in this benchmark.

Increasing m generally reduces the variance of Diff-Hess trace estimator and improves its accuracy, but incurs greater computational cost. Moreover, the first stage trains substantially faster than the second, indicating that the computational bottleneck of KoopSDE lies primarily in the second stage.

A.9 One-Dimensional Nonlinear Ornstein-Uhlenbeck (OU) Process

We first validate the learning performance of KoopSDE using a one-dimensional SDE example. In the one-dimensional case, the drift and diffusion functions can be visualized intuitively, facilitating result analysis. The OU process $dY_t = -Y_t dt + dB_t$ is chosen as the test case, primarily for two

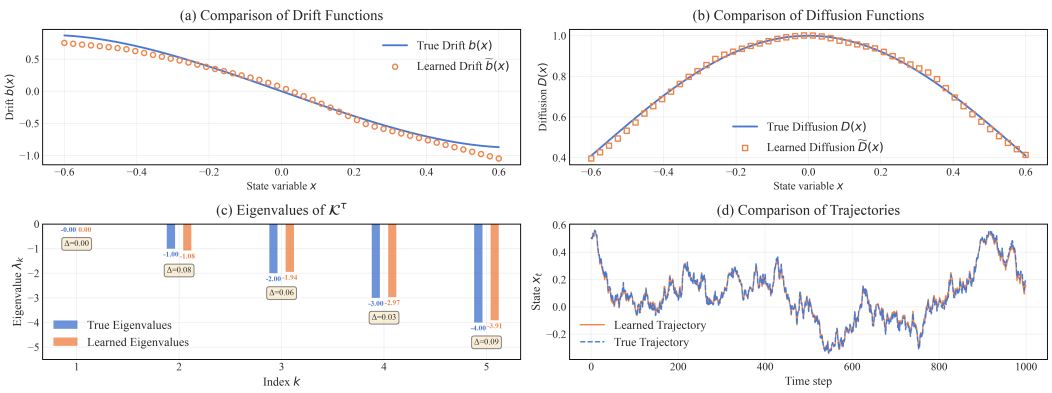

Figure 6: KoopSDE Algorithm Performance on Nonlinear OU Process

reasons: **(a)** the OU process is widely used in stochastic modeling; **(b)** its eigenvalues can be derived analytically as $0, -1, -2, \ldots$ (see Appendix B.8), which facilitates verifying whether KoopSDE can accurately learn the first N largest eigenvalues. Considering that the drift and diffusion functions of the linear OU process are both linear and overly simplistic, we apply a nonlinear transformation $X_t = \tanh(Y_t)$. Using Itô's formula, the resulting nonlinear OU process is:

$$dX_t = -(1 - X_t^2)\left[\operatorname{arctanh}(X_t) + X_t\right] dt + (1 - X_t^2) dB_t, \quad X_t \in (-1, 1), \tag{13}$$

where $\boldsymbol{b}(x) = -(1 - x^2)\left[\operatorname{arctanh}(x) + x\right]$ and $\mathbf{D}(x) = (1 - x^2)^2$. Since $X_t = \tanh(Y_t)$ is an invertible transformation, the eigenvalues of this process remain unchanged.

In the experiment, we set $N = 5$. The results are shown in Figure 6, leading to the following conclusions:

- **Function Learning (a)(b)** After the second-stage learning, we evaluate the learned $\boldsymbol{b}(x)$ and $\mathbf{D}(x)$ on 50 test points. The results show that KoopSDE can reconstruct the true function values reasonably well. It is worth noting that the diffusion functions of many SDEs are simple (e.g., constant or linear); the invertible transformation here increases the learning difficulty.
- **Eigenvalue Learning (c)** KoopSDE accurately learns the first five largest eigenvalues of Eq. (13), with the maximum deviation $\Delta_{\max}$ being only 0.09. This verifies the convergence result stated in Theorem 5.2 and aligns with the conclusion in Corollary 5.4 regarding the preferential learning of the first N largest eigenvalues.
- **Trajectory Validation (d)** Using the learned $\widetilde{\boldsymbol{b}}(x)$ and $\widetilde{\mathbf{D}}(x)$, we generate a 1000-step trajectory $\{X_t\}$ following the method in Appendix A.10 and compare it with a trajectory generated using the true $\boldsymbol{b}(x)$ and $\mathbf{D}(x)$. The two trajectories largely coincide, further validating the correctness of $\widetilde{\boldsymbol{b}}(x)$ and $\widetilde{\mathbf{D}}(x)$.

A.10 SDE Discretization and the Euler–Maruyama Scheme

To generate a discrete trajectory $(\boldsymbol{x}_l)_{l=0}^n$ with observation interval τ, we numerically simulate the SDE in Eq. (1) using the Euler–Maruyama method. Each observation interval is subdivided into M_{EM} subintervals. The four-system experiments use the calibrated, system-dependent values of M_{EM} reported in Appendix A.4, while the separate earlier experiments use the fixed value $M_{\mathrm{EM}} = 1000$. The internal step size is

$$h = \frac{\tau}{M_{\mathrm{EM}}}.$$

Let $t_{l,j} = l\tau + jh$, $j = 0, \ldots, M_{\mathrm{EM}}$, so that $t_{l,0} = l\tau$ and $t_{l,M_{\mathrm{EM}}} = (l+1)\tau$. Starting from $X_{t_{l,0}} = \boldsymbol{x}_l$, the Euler–Maruyama iteration is

$$X_{t_{l,j+1}} = X_{t_{l,j}} + \boldsymbol{b}\big(X_{t_{l,j}}\big)h + \boldsymbol{\sigma}\big(X_{t_{l,j}}\big)\Delta B_{l,j}, \tag{14}$$

where

$$\Delta B_{l,j} = B_{t_{l,j+1}} - B_{t_{l,j}} \sim \mathcal{N}(0, h\mathbf{I}_r)$$

are independent Brownian increments. Equivalently, writing $\Delta B_{l,j} = \sqrt{h}\,\boldsymbol{\xi}_{l,j}$ with $\boldsymbol{\xi}_{l,j} \sim \mathcal{N}(0, \mathbf{I}_r)$, the update can be expressed as

$$X_{t_{l,j+1}} = X_{t_{l,j}} + \boldsymbol{b}\big(X_{t_{l,j}}\big)h + \boldsymbol{\sigma}\big(X_{t_{l,j}}\big)\sqrt{h}\,\boldsymbol{\xi}_{l,j}.$$

After M_{EM} refinement steps, we retain only

$$\boldsymbol{x}_{l+1} = X_{t_{l,M_{\mathrm{EM}}}}$$

as the next observation and discard the intermediate states. The resulting observation sequence satisfies $\boldsymbol{x}_l \approx X_{l\tau}$.

Under standard regularity conditions, the conditional moments of the increment satisfy

$$\begin{aligned}\mathbb{E}\left[X_{(l+1)\tau} - X_{l\tau} \,\middle|\, X_{l\tau} = \boldsymbol{x}\right] &= \boldsymbol{b}(\boldsymbol{x})\tau + \mathcal{O}(\tau^2),\\ \mathrm{Cov}\left[X_{(l+1)\tau} - X_{l\tau} \,\middle|\, X_{l\tau} = \boldsymbol{x}\right] &= \mathbf{D}(\boldsymbol{x})\tau + \mathcal{O}(\tau^2).\end{aligned}$$

These relations motivate the first-order local Gaussian approximation

$$X_{(l+1)\tau} - X_{l\tau} \mid X_{l\tau} \approx \mathcal{N}\left(\boldsymbol{b}(X_{l\tau})\tau, \mathbf{D}(X_{l\tau})\tau\right),$$

which underlies the SparseNN baseline (Ren et al., 2024). We do not use this approximation to generate the benchmark trajectories. Instead, we use the refined Euler–Maruyama scheme in Eq. (14) to reduce the numerical error in the observed data. The complete trajectory-generation procedure is summarized in Algorithm 2.

Algorithm 2 Refined Euler–Maruyama Simulation of SDE Trajectories

Require: Initial state $\boldsymbol{x}_0 \in \mathbb{R}^d$, number of observations $n+1$, observation interval τ, refinement count M_{EM}, drift function $\boldsymbol{b}$, and noise factor $\boldsymbol{\sigma}$ satisfying $\boldsymbol{\sigma}\boldsymbol{\sigma}^\top = \mathbf{D}$.
Ensure: Observation sequence $(\boldsymbol{x}_l)_{l=0}^n$ satisfying $\boldsymbol{x}_l \approx X_{l\tau}$.

```
Set h = τ / M_EM.
Initialize X ← x_0 and record x_0 ← X.
for l = 0 to n − 1 do
    for j = 0 to M_EM − 1 do
        Sample ξ ~ N(0, I_r).
        Update
                X ← X + b(X)h + σ(X)√h ξ.
    end for
    Record x_{l+1} ← X.
end for
return (x_l)_{l=0}^n.
```

When generating trajectories from the learned functions, we replace $\boldsymbol{b}$ by $\widetilde{\boldsymbol{b}}$ and construct a matrix factor $\widetilde{\boldsymbol{\sigma}}$ satisfying

$$\widetilde{\boldsymbol{\sigma}}(\boldsymbol{x})\widetilde{\boldsymbol{\sigma}}(\boldsymbol{x})^\top = \widetilde{\mathbf{D}}(\boldsymbol{x}).$$

For the one-dimensional nonlinear OU experiment, we use the nonnegative square root

$$\widetilde{\boldsymbol{\sigma}}(x) = \sqrt{\widetilde{\mathbf{D}}(x)}.$$

Thus, $\widetilde{\mathbf{D}}$ remains the learned diffusion function, whereas its matrix factor $\widetilde{\boldsymbol{\sigma}}$ is used only for numerical trajectory simulation.

A.11 Fourier Feature Embedding

In the KoopSDE algorithm, we employ an MLP to parameterize the eigenfunction vector $\Phi(\boldsymbol{x})$. According to the oscillation theorem in Sturm–Liouville theory, for most stationary SDEs, the eigenvalue spectrum is discrete and tends to negative infinity, with the corresponding eigenfunctions exhibiting increasingly stronger oscillations. For a detailed discussion on this topic, we refer readers to Chapter 15 of Karlin & Taylor (1981).

However, standard MLPs suffer from a pronounced spectral bias: they tend to preferentially learn low-frequency components while struggling to fit high-frequency details (Rahaman et al., 2019). A series of works proposed using Fourier feature embedding to mitigate this issue (Rahaman et al., 2019; Sitzmann et al., 2020). The core idea is to map the low-dimensional input into a high-dimensional space via a set of sine and cosine functions with different frequencies before feeding it into the neural network. By introducing multi-frequency oscillatory patterns, this method provides the model with a richer set of basis functions, enabling it to more effectively capture high-frequency variations in the data. Such works have led to various practical tools and have made Fourier feature embedding a standard preprocessing step for coordinate-based MLPs (Vaswani et al., 2017; Li et al., 2020; Mildenhall et al., 2021).

Therefore, to enable $\widetilde{\Phi}(\boldsymbol{x};\boldsymbol{\theta}_\Phi)$ to learn high-frequency eigenfunctions, we apply Fourier feature embedding to the input coordinate $\boldsymbol{x}\in\mathbb{R}^d$. Specifically, we construct a frequency matrix $\mathbf{B}\in\mathbb{R}^{q\times d}$ whose elements follow a Gaussian distribution, i.e., $\mathbf{B}_{ij}\sim\mathcal{N}(0,\sigma^2)$, where q is the chosen number of frequencies and σ is a scale parameter controlling the spread of the frequencies. Denoting the q row vectors of $\mathbf{B}$ as $\mathbf{b}_1,\dots,\mathbf{b}_q\in\mathbb{R}^{1\times d}$, we define the Fourier feature embedding as

$$\gamma(\boldsymbol{x}) = \left[\cos(2\pi\mathbf{b}_1\boldsymbol{x}),\ \sin(2\pi\mathbf{b}_1\boldsymbol{x}),\ \dots,\ \cos(2\pi\mathbf{b}_q\boldsymbol{x}),\ \sin(2\pi\mathbf{b}_q\boldsymbol{x})\right]^\top \in \mathbb{R}^{2q}.$$

The Fourier embedding supplies the nonlinear branch of the eigenfunction network. In the four-system experiments, $q=32$ and $\sigma=0.3/\sqrt{d}$. The input is standardized using fitting-set statistics, a learned linear skip is added to the MLP output, and affine-free batch normalization fixes the output scale. Running normalization statistics are frozen during the second stage, so input derivatives act on a fixed eigenfunction network.

A.12 Low-Rank-Plus-Diagonal Parameterization of the Diffusion Function

By definition, $\mathbf{D}(\boldsymbol{x})$ should be a symmetric positive semidefinite matrix. If a neural network directly predicts all entries of $\mathbf{D}(\boldsymbol{x})$, its output dimension will reach $\mathcal{O}(d^2)$, and an unconstrained network output cannot automatically guarantee the symmetry and positive semidefiniteness of the matrix. On the other hand, using only a low-rank matrix is also overly restrictive, because even when the individual coordinates are driven by mutually independent noise, the corresponding diffusion matrix may still be a full-rank diagonal matrix. Therefore, we adopt a low-rank-plus-diagonal structure to parameterize $\mathbf{D}(\boldsymbol{x})$.

Let $p\le d$ be a prespecified correlation rank. A neural network with two output heads produces

$$\widetilde{\mathbf{F}}(\boldsymbol{x})\in\mathbb{R}^{d\times p},\qquad \widetilde{\boldsymbol{q}}(\boldsymbol{x})\in\mathbb{R}^d.$$

To ensure that the diagonal entries are positive, we define

$$\widetilde{\delta}_i(\boldsymbol{x}) = \mathrm{SoftPlus}\big(\widetilde{q}_i(\boldsymbol{x})\big)+\varepsilon_D,\qquad i=1,\dots,d,$$

where $\varepsilon_D>0$ is a small constant introduced for numerical stability. The diffusion function is finally parameterized as

$$\widetilde{\mathbf{D}}(\boldsymbol{x}) = \widetilde{\mathbf{F}}(\boldsymbol{x})\widetilde{\mathbf{F}}(\boldsymbol{x})^\top + \mathrm{Diag}\big(\widetilde{\boldsymbol{\delta}}(\boldsymbol{x})\big),$$

where $\widetilde{\mathbf{F}}(\boldsymbol{x})$ is a state-dependent low-rank factor. This structure automatically satisfies the constraints imposed on a diffusion matrix. For any vector $\boldsymbol{v}$, we have

$$\boldsymbol{v}^\top\widetilde{\mathbf{D}}(\boldsymbol{x})\boldsymbol{v} = \left\|\widetilde{\mathbf{F}}(\boldsymbol{x})^\top\boldsymbol{v}\right\|_2^2 + \sum_{i=1}^{d}\widetilde{\delta}_i(\boldsymbol{x})v_i^2 \ge \varepsilon_D\|\boldsymbol{v}\|_2^2.$$

Therefore, when $\varepsilon_D > 0$, $\widetilde{\mathbf{D}}$ is a symmetric positive definite matrix; when $\varepsilon_D = 0$ and all diagonal coefficients are nonnegative, it is a symmetric positive semidefinite matrix.

Under this parameterization, the diagonal term represents coordinate-wise independent noise and allows the entire diffusion matrix to remain full rank, whereas the low-rank term represents state-dependent noise correlations between coordinates. Therefore, KoopSDE does not assume that the entire diffusion matrix is low rank. When $p = d$, this form can, in principle, represent any positive semidefinite diffusion matrix through a matrix square-root factorization. Another important advantage of this parameterization is that it does not require the explicit construction of the $d \times d$ diffusion matrix. For any vector $\boldsymbol{z}$, we have

$$\widetilde{\mathbf{D}}(\boldsymbol{x})\boldsymbol{z} = \widetilde{\mathbf{F}}(\boldsymbol{x}) \left[\widetilde{\mathbf{F}}(\boldsymbol{x})^\top \boldsymbol{z}\right] + \widetilde{\boldsymbol{\delta}}(\boldsymbol{x}) \odot \boldsymbol{z}.$$

Therefore, storing the implicit representation of $\widetilde{\mathbf{D}}$ requires only $\mathcal{O}(dp + d)$ memory, and the matrix–vector product also requires only $\mathcal{O}(dp + d)$ operations. When p is fixed, both costs grow linearly with the state dimension d.

This structure can be directly combined with the Diff-Hess trace estimator. Let

$$\boldsymbol{h}_k^{(l)} = \nabla \left(\boldsymbol{z}^{(l)\top} \nabla \widetilde{\phi}_k\right) = \mathbf{H}_k \boldsymbol{z}^{(l)}.$$

The quadratic form required by the trace estimator can be written as

$$\boldsymbol{z}^{(l)\top} \widetilde{\mathbf{D}} \boldsymbol{h}_k^{(l)} = \left(\widetilde{\mathbf{F}}^\top \boldsymbol{z}^{(l)}\right)^\top \left(\widetilde{\mathbf{F}}^\top \boldsymbol{h}_k^{(l)}\right) + \left(\widetilde{\boldsymbol{\delta}} \odot \boldsymbol{z}^{(l)}\right)^\top \boldsymbol{h}_k^{(l)}.$$

Therefore, neither the Hessian matrix $\mathbf{H}_k$ nor the diffusion matrix $\widetilde{\mathbf{D}}$ needs to be explicitly constructed in the actual computation.

If the width of the shared feature layer of the neural network is h, the output layers of the low-rank factor and the diagonal vector contain $\mathcal{O}\big(hd(p+1)\big)$ parameters, whereas directly outputting the full diffusion matrix requires $\mathcal{O}(hd^2)$ parameters. After fixing p, the former grows linearly with the dimension d. The use of such compact matrix factors is conceptually similar to LoRA (Hu et al., 2022). LoRA parameterizes the update of a high-dimensional matrix as the product of two low-rank factors. In this work, we apply such a decomposition to a state-dependent diffusion function and add a diagonal component to guarantee the positive semidefiniteness of the diffusion matrix while retaining the ability to represent full-rank coordinate-wise noise.

If the learned diffusion function subsequently needs to be used to generate trajectories, an auxiliary noise factor can be constructed after training:

$$\widetilde{\boldsymbol{\sigma}}_{\mathrm{aux}}(\boldsymbol{x}) = \left[\widetilde{\mathbf{F}}(\boldsymbol{x}) \quad \mathrm{Diag}\left(\sqrt{\widetilde{\boldsymbol{\delta}}(\boldsymbol{x})}\right)\right].$$

It satisfies $\widetilde{\boldsymbol{\sigma}}_{\mathrm{aux}} \widetilde{\boldsymbol{\sigma}}_{\mathrm{aux}}^\top = \widetilde{\mathbf{D}}$. This auxiliary factor is used only for numerical simulation, is not the learning target of KoopSDE, and does not need to be identical to the $\boldsymbol{\sigma}$ used to generate the true data. The noise factor itself is not unique, but all equivalent noise factors produce the same diffusion function $\mathbf{D}$. Therefore, this non-uniqueness does not affect the generator loss or experimental evaluation of KoopSDE.

Finally, the three variants, KoopSDE (vanilla), KoopSDE (SDGD), and KoopSDE (HTE), must use exactly the same diffusion function parameterization and the same rank p. The only difference among them should be the way in which $\mathrm{tr}\left(\widetilde{\mathbf{D}}\mathbf{H}_k\right)$ is computed.

B THEORETICAL PROOF

B.1 PROOF OF EQ. (3)

Let $f \in C^2(\mathbb{R}^d)$. Applying Itô's formula to the diffusion process X_t yields

$$df(X_t) = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i}(X_t) dX_t^i + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial x_i \partial x_j}(X_t) d\langle X^i, X^j \rangle_t, \tag{15}$$

where $\langle X^i, X^j \rangle_t$ denotes the quadratic covariation process of X^i and X^j. From Eq. (1) we have

$$dX_t^i = \boldsymbol{b}_i(X_t)dt + \sum_{k=1}^{r} \boldsymbol{\sigma}_{ik}(X_t) dB_t^k.$$

Therefore,

$$d\langle X^i, X^j \rangle_t = \sum_{k=1}^{r} \boldsymbol{\sigma}_{ik}(X_t)\boldsymbol{\sigma}_{jk}(X_t)dt = \mathbf{D}_{ij}(X_t)dt,$$

where $\mathbf{D}_{ij} = (\boldsymbol{\sigma}\boldsymbol{\sigma}^\top)_{ij} = \sum_{k=1}^{r} \boldsymbol{\sigma}_{ik}\boldsymbol{\sigma}_{jk}$. Substituting these expressions into Eq. (15) gives

$$\begin{aligned}
df(X_t) &= \sum_{i=1}^{d} \frac{\partial f}{\partial x_i}(X_t) \Big(\boldsymbol{b}_i(X_t)dt + \sum_{k=1}^{r} \boldsymbol{\sigma}_{ik}(X_t) dB_t^k \Big) \\
&\quad + \frac{1}{2} \sum_{i,j=1}^{d} \frac{\partial^2 f}{\partial x_i \partial x_j}(X_t) \mathbf{D}_{ij}(X_t) dt \\
&= \Big(\sum_{i=1}^{d} \boldsymbol{b}_i(X_t) \frac{\partial f}{\partial x_i}(X_t) + \frac{1}{2} \sum_{i,j=1}^{d} \mathbf{D}_{ij}(X_t) \frac{\partial^2 f}{\partial x_i \partial x_j}(X_t) \Big) dt \\
&\quad + \sum_{i=1}^{d} \sum_{k=1}^{r} \frac{\partial f}{\partial x_i}(X_t) \boldsymbol{\sigma}_{ik}(X_t) dB_t^k.
\end{aligned}$$

Define the operator

$$\mathcal{A}f(\boldsymbol{x}) := \sum_{i=1}^{d} \boldsymbol{b}_i(\boldsymbol{x}) \frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \frac{1}{2} \sum_{i,j=1}^{d} \mathbf{D}_{ij}(\boldsymbol{x}) \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x}),$$

so that

$$df(X_t) = \mathcal{A}f(X_t)dt + \nabla f(X_t)^\top \boldsymbol{\sigma}(X_t) dB_t.$$

Recall that the infinitesimal generator $\mathcal{L}$ of the diffusion process X_t is defined by

$$\mathcal{L}f(\boldsymbol{x}) = \lim_{t \to 0} \frac{1}{t} \big[\mathbb{E}_{\mathbb{P}} \big[f(X_t) \mid X_0 = \boldsymbol{x} \big] - f(\boldsymbol{x}) \big].$$

Starting from the integral form of Itô's formula,

$$f(X_t) = f(X_0) + \int_0^t \mathcal{A}f(X_s)ds + \int_0^t \nabla f(X_s)^\top \boldsymbol{\sigma}(X_s) dB_s,$$

and taking conditional expectation $\mathbb{E}_{\mathbb{P}}[\,\cdot \mid X_0 = \boldsymbol{x}]$ on both sides, we obtain

$$\mathbb{E}_{\mathbb{P}} \big[f(X_t) \mid X_0 = \boldsymbol{x} \big] = f(\boldsymbol{x}) + \mathbb{E}_{\mathbb{P}} \Big[\int_0^t \mathcal{A}f(X_s)ds \Big| X_0 = \boldsymbol{x} \Big].$$

Hence,

$$\frac{1}{t} \big[\mathbb{E}_{\mathbb{P}} \big[f(X_t) \mid X_0 = \boldsymbol{x} \big] - f(\boldsymbol{x}) \big] = \frac{1}{t} \int_0^t \mathbb{E}_{\mathbb{P}} \big[\mathcal{A}f(X_s) \mid X_0 = \boldsymbol{x} \big] ds.$$

Under suitable regularity conditions, the map $s \mapsto \mathbb{E}_{\mathbb{P}}[\mathcal{A}f(X_s) \mid X_0 = \boldsymbol{x}]$ is continuous at $s = 0$. Thus, letting $t \to 0$ yields

$$\mathcal{L}f(\boldsymbol{x}) = \mathcal{A}f(\boldsymbol{x}).$$

Consequently,

$$\mathcal{L}f(\boldsymbol{x}) = \sum_{i=1}^{d} \boldsymbol{b}_i(\boldsymbol{x}) \frac{\partial f}{\partial x_i}(\boldsymbol{x}) + \frac{1}{2} \sum_{i,j=1}^{d} \mathbf{D}_{ij}(\boldsymbol{x}) \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x}) \cdot \nabla f(\boldsymbol{x}) + \frac{1}{2} \operatorname{tr}\left[\mathbf{D}(\boldsymbol{x}) \mathbf{H}_f(\boldsymbol{x})\right].$$

which completes the proof of Eq. (3).

B.2 Proof of Lemma 4.1

The conditions of Lemma 4.1 give $\mathbb{E}[z_i z_j] = \delta_{ij}$. Therefore,

$$\mathbb{E}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z}\right] = \mathbb{E}_{\boldsymbol{z}}\left[\sum_{i,j} \mathbf{A}_{ij} z_i z_j\right] = \sum_i \mathbf{A}_{ii} \mathbb{E}[z_i^2] + \sum_{i \neq j} \mathbf{A}_{ij} \mathbb{E}[z_i z_j] = \sum_i \mathbf{A}_{ii} = \operatorname{tr}(\mathbf{A}),$$

which completes the proof of Lemma 4.1.

B.3 Proof of Proposition 5.1

Let $\boldsymbol{z} = (z_1, \ldots, z_d)^\top$ be a random vector whose components are independent and identically distributed (i.i.d.), satisfying

$$\mathbb{E}[z_i] = 0, \ \operatorname{Var}(z_i) = 1, \ \mathbb{E}[z_i^4] = \mu_4.$$

From Lemma 4.1, we have:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{z}}\left[\widehat{\operatorname{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)\right] &= \frac{1}{m} \mathbb{E}_{\boldsymbol{z}}\left[\sum_{l=1}^{m} \boldsymbol{z}^{(l)\top} \widetilde{\mathbf{D}} \mathbf{H}_k \boldsymbol{z}^{(l)}\right] \\
&= \frac{1}{m} \sum_{l=1}^{m} \mathbb{E}_{\boldsymbol{z}}\left[\boldsymbol{z}^{(l)\top} \widetilde{\mathbf{D}} \mathbf{H}_k \boldsymbol{z}^{(l)}\right] \\
&= \frac{1}{m} \sum_{l=1}^{m} \operatorname{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k) = \operatorname{tr}(\widetilde{\mathbf{D}}\mathbf{H}_k).
\end{aligned}$$

Denote $\mathbf{A} = \widetilde{\mathbf{D}}\mathbf{H}_k$. Consider the variance

$$\operatorname{Var}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z}\right] = \mathbb{E}_{\boldsymbol{z}}\left[(\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z})^2\right] - \left(\mathbb{E}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z}\right]\right)^2.$$

First,

$$(\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z})^2 = \Big(\sum_{i,j} \mathbf{A}_{ij} z_i z_j\Big)\Big(\sum_{r,s} \mathbf{A}_{rs} z_r z_s\Big) = \sum_{i,j,r,s} \mathbf{A}_{ij} \mathbf{A}_{rs} z_i z_j z_r z_s,$$

hence

$$\mathbb{E}_{\boldsymbol{z}}[(\boldsymbol{z}^\top \mathbf{A} \boldsymbol{z})^2] = \sum_{i,j,r,s} \mathbf{A}_{ij} \mathbf{A}_{rs} \mathbb{E}[z_i z_j z_r z_s].$$

We compute the fourth-order moment $\mathbb{E}[z_i z_j z_r z_s]$. Due to independence, the expectation is nonzero only when the indices are paired. There are four possible pairing cases:

- $i = j = r = s$: $\mathbb{E}[z_i^4] = \mu_4$, contributing $\sum_i \mathbf{A}_{ii}^2 \mu_4$.
- $i = j, \ r = s$ with $i \neq r$: $\mathbb{E}[z_i^2 z_r^2] = 1$, contributing $\sum_{i \neq r} \mathbf{A}_{ii} \mathbf{A}_{rr}$.
- $i = r, \ j = s$ with $i \neq j$: $\mathbb{E}[z_i^2 z_j^2] = 1$, contributing $\sum_{i \neq j} \mathbf{A}_{ij}^2$.
- $i = s, \ j = r$ with $i \neq j$: $\mathbb{E}[z_i^2 z_j^2] = 1$, contributing $\sum_{i \neq j} \mathbf{A}_{ij} \mathbf{A}_{ji}$.

All other pairing patterns (e.g., three indices equal or only one pair equal) yield zero because of the presence of an independent zero-mean factor.

Thus,

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}}[(\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z})^2] &= \mu_4 \sum_i \mathbf{A}_{ii}^2 + \sum_{i\neq r} \mathbf{A}_{ii}\mathbf{A}_{rr} + \sum_{i\neq j} \mathbf{A}_{ij}^2 + \sum_{i\neq j} \mathbf{A}_{ij}\mathbf{A}_{ji} \\ &= \mu_4 \sum_i \mathbf{A}_{ii}^2 + \sum_{i\neq r} \mathbf{A}_{ii}\mathbf{A}_{rr} + \sum_{i\neq j} \mathbf{A}_{ij}(\mathbf{A}_{ij} + \mathbf{A}_{ji}).\end{aligned}$$

Since $\mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}] = \operatorname{tr} \mathbf{A} = \sum_i \mathbf{A}_{ii}$, we have

$$\left(\mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}]\right)^2 = (\operatorname{tr} \mathbf{A})^2 = \sum_i \mathbf{A}_{ii}^2 + \sum_{i\neq r} \mathbf{A}_{ii}\mathbf{A}_{rr}.$$

Therefore,

$$\begin{aligned}\operatorname{Var}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}\right] &= \mathbb{E}_{\boldsymbol{z}}[(\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z})^2] - \left(\mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}]\right)^2 \\ &= \left[\mu_4 \sum_i \mathbf{A}_{ii}^2 + \sum_{i\neq r} \mathbf{A}_{ii}\mathbf{A}_{rr} + \sum_{i\neq j} \mathbf{A}_{ij}(\mathbf{A}_{ij} + \mathbf{A}_{ji})\right] - \left(\sum_i \mathbf{A}_{ii}^2 + \sum_{i\neq r} \mathbf{A}_{ii}\mathbf{A}_{rr}\right) \\ &= \sum_{i\neq j} \mathbf{A}_{ij}(\mathbf{A}_{ij} + \mathbf{A}_{ji}) + (\mu_4 - 1)\sum_i \mathbf{A}_{ii}^2.\end{aligned}$$

Note that $\sum_{i\neq j} \mathbf{A}_{ij}(\mathbf{A}_{ij} + \mathbf{A}_{ji}) = \sum_{i<j} (\mathbf{A}_{ij} + \mathbf{A}_{ji})^2$. Hence,

$$\operatorname{Var}_{\boldsymbol{z}}\left[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}\right] = \sum_{i<j} (\mathbf{A}_{ij} + \mathbf{A}_{ji})^2 + (\mu_4 - 1)\sum_i \mathbf{A}_{ii}^2. \tag{16}$$

We now show that the variance is minimized when $\boldsymbol{z}$ follows the Rademacher distribution.

Note that $\mu_4 - 1 = \mathbb{E}[z_i^4] - (\mathbb{E}[z_i^2])^2 = \operatorname{Var}(z_i^2) \geq 0$. Since the term $\sum_{i<j} (\mathbf{A}_{ij} + \mathbf{A}_{ji})^2$ in Eq. (16) does not depend on the distribution of $\boldsymbol{z}$, the variance attains its minimum $\sum_{i<j} (\mathbf{A}_{ij} + \mathbf{A}_{ji})^2$ when $\mu_4 - 1 = 0$.

The condition $\mu_4 - 1 = 0$ implies $\operatorname{Var}(z_i^2) = 0$. Together with $\mathbb{E}[z_i^2] = 1$, we obtain $z_i^2 = 1$ almost surely, i.e., $z_i = \pm 1$ almost surely.

According to $\mathbb{E}[z_i] = 0$, we have

$$\mathbb{P}(z_i = 1) - \mathbb{P}(z_i = -1) = 0, \quad \mathbb{P}(z_i = 1) + \mathbb{P}(z_i = -1) = 1,$$

which yields $\mathbb{P}(z_i = 1) = \mathbb{P}(z_i = -1) = \frac{1}{2}$. Hence, $\boldsymbol{z}$ follows the Rademacher distribution.

Finally, because the estimator $\widehat{\operatorname{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)$ is the sample mean of m i.i.d. copies of $\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}$,

$$\operatorname{Var}_{\boldsymbol{z}}\left[\widehat{\operatorname{tr}}(\widetilde{\mathbf{D}}\mathbf{H}_k)\right] = \frac{1}{m} \operatorname{Var}\left[\boldsymbol{z}^\top \mathbf{A}\boldsymbol{z}\right] = \frac{1}{m} \sum_{i<j} (\mathbf{A}_{ij} + \mathbf{A}_{ji})^2,$$

which completes the proof of Proposition 5.1.

B.4 Proof of Theorem 5.2

Let $Y_l = X_{l\tau}$, and let $P_\tau(\boldsymbol{x}, A) = \mathbb{P}(X_\tau \in A \mid X_0 = \boldsymbol{x})$ denote the transition kernel of the sampled chain. Write

$$A_{k,n} := \frac{1}{n} \sum_{l=0}^{n-1} \phi_k(Y_l)\phi_k(Y_{l+1}), \qquad S_{k,n} := \frac{1}{n} \sum_{l=0}^{n-1} \phi_k^2(Y_l),$$

so that $\widehat{\nu}(\phi_k) = A_{k,n}/S_{k,n}$. Since $\phi_k \in L^2(\mu)$, the ergodic theorem for Markov chains (see Chapter 17 of Meyn & Tweedie, 2012) gives, for μ-almost every initial state,

$$S_{k,n} \longrightarrow \int_{\mathcal{S}} \phi_k^2(\boldsymbol{x})\, \mu(d\boldsymbol{x}) = s_k^2 \quad \text{almost surely.}$$

In particular, $s_k^2 > 0$ because ϕ_k is nonzero.

Consider the augmented chain $Z_l = (Y_l, Y_{l+1})$. It is ergodic with invariant probability measure

$$\pi(d\boldsymbol{x}, d\boldsymbol{y}) = \mu(d\boldsymbol{x}) P_\tau(\boldsymbol{x}, d\boldsymbol{y}).$$

For $h(\boldsymbol{x}, \boldsymbol{y}) = \phi_k(\boldsymbol{x})\phi_k(\boldsymbol{y})$, the Cauchy–Schwarz inequality and the invariance of μ imply

$$\begin{aligned}\int_{\mathcal{S}\times\mathcal{S}} |h|\, d\pi &\le \left(\int_{\mathcal{S}} \phi_k^2(\boldsymbol{x})\, \mu(d\boldsymbol{x})\right)^{1/2} \left(\int_{\mathcal{S}}\int_{\mathcal{S}} \phi_k^2(\boldsymbol{y}) P_\tau(\boldsymbol{x}, d\boldsymbol{y})\mu(d\boldsymbol{x})\right)^{1/2} \\ &= s_k^2 < \infty.\end{aligned}$$

Thus, the ergodic theorem applied to $\{Z_l\}_{l\ge 0}$, together with the Koopman eigenfunction relation, yields

$$\begin{aligned}A_{k,n} &\longrightarrow \int_{\mathcal{S}\times\mathcal{S}} \phi_k(\boldsymbol{x})\phi_k(\boldsymbol{y})\, \pi(d\boldsymbol{x}, d\boldsymbol{y}) \\ &= \int_{\mathcal{S}} \phi_k(\boldsymbol{x})(\mathcal{K}^\tau \phi_k)(\boldsymbol{x})\, \mu(d\boldsymbol{x}) = e^{\lambda_k \tau} s_k^2 \quad \text{almost surely.}\end{aligned}$$

Since $S_{k,n} \to s_k^2 > 0$, it follows that $\widehat{\nu}(\phi_k) \to e^{\lambda_k \tau}$ almost surely.

Suppose now that $\phi_k \in L^4(\mu)$, and let $\mathcal{F}_l = \sigma(Y_0, \dots, Y_l)$. Define

$$\xi_{k,l+1} := \phi_k(Y_l)\left[\phi_k(Y_{l+1}) - e^{\lambda_k \tau}\phi_k(Y_l)\right].$$

By the Markov property and the Koopman eigenfunction relation,

$$\mathbb{E}[\xi_{k,l+1} \mid \mathcal{F}_l] = \phi_k(Y_l)\left[(\mathcal{K}^\tau \phi_k)(Y_l) - e^{\lambda_k \tau}\phi_k(Y_l)\right] = 0.$$

Hence $\{\xi_{k,l+1}, \mathcal{F}_{l+1}\}_{l\ge 0}$ is a martingale difference sequence. Moreover,

$$\sqrt{n}\left(\widehat{\nu}(\phi_k) - e^{\lambda_k \tau}\right) = \frac{n^{-1/2}\sum_{l=0}^{n-1} \xi_{k,l+1}}{S_{k,n}}.$$

The conditional second moment of the martingale difference is

$$\mathbb{E}[\xi_{k,l+1}^2 \mid \mathcal{F}_l] = \phi_k^2(Y_l)\left[(\mathcal{K}^\tau \phi_k^2)(Y_l) - e^{2\lambda_k \tau}\phi_k^2(Y_l)\right] = \mathcal{W}_k(Y_l),$$

where

$$\mathcal{W}_k(\boldsymbol{x}) := \phi_k^2(\boldsymbol{x})\left[(\mathcal{K}^\tau \phi_k^2)(\boldsymbol{x}) - e^{2\lambda_k \tau}\phi_k^2(\boldsymbol{x})\right].$$

Conditional Jensen's inequality gives $\mathcal{W}_k \ge 0$. Furthermore, the Cauchy–Schwarz inequality, the invariance of μ, and $\phi_k \in L^4(\mu)$ give

$$\begin{aligned}\int_{\mathcal{S}} \phi_k^2(\mathcal{K}^\tau \phi_k^2)\, d\mu &= \mathbb{E}_\mu[\phi_k^2(Y_0)\phi_k^2(Y_1)] \\ &\le \left(\mathbb{E}_\mu[\phi_k^4(Y_0)]\right)^{1/2}\left(\mathbb{E}_\mu[\phi_k^4(Y_1)]\right)^{1/2} = \int_{\mathcal{S}} \phi_k^4\, d\mu < \infty.\end{aligned}$$

Consequently, $\mathcal{W}_k \in L^1(\mu)$ and $\int_{\mathcal{S}} \mathcal{W}_k\, d\mu = q_k$. Another application of the ergodic theorem therefore yields

$$\frac{1}{n}\sum_{l=0}^{n-1} \mathbb{E}[\xi_{k,l+1}^2 \mid \mathcal{F}_l] = \frac{1}{n}\sum_{l=0}^{n-1} \mathcal{W}_k(Y_l) \longrightarrow q_k \quad \text{almost surely.}$$

It remains to verify the conditional Lindeberg condition without imposing a stationary initial distribution. Let

$$\zeta_k(\boldsymbol{x}, \boldsymbol{y}) := \phi_k(\boldsymbol{x})\left[\phi_k(\boldsymbol{y}) - e^{\lambda_k \tau}\phi_k(\boldsymbol{x})\right]$$

and, for $R > 0$, define

$$\mathcal{T}_{k,R}(\boldsymbol{x}) := \int_{\mathcal{S}} \zeta_k^2(\boldsymbol{x}, \boldsymbol{y}) \mathbf{1}_{\{|\zeta_k(\boldsymbol{x},\boldsymbol{y})| > R\}} P_\tau(\boldsymbol{x}, d\boldsymbol{y}).$$

Then $0 \le \mathcal{T}_{k,R} \le \mathcal{W}_k$, and $\mathcal{T}_{k,R}(\boldsymbol{x}) \to 0$ as $R \to \infty$ for μ-almost every $\boldsymbol{x}$. Dominated convergence gives

$$\int_{\mathcal{S}} \mathcal{T}_{k,R}\, d\mu \longrightarrow 0.$$

For any $\epsilon > 0$, the conditional Lindeberg term is

$$\begin{aligned} L_n(\epsilon) &:= \frac{1}{n} \sum_{l=0}^{n-1} \mathbb{E}\left[\xi_{k,l+1}^2 \mathbf{1}_{\{|\xi_{k,l+1}| > \epsilon\sqrt{n}\}} \,\middle|\, \mathcal{F}_l \right] \\ &= \frac{1}{n} \sum_{l=0}^{n-1} \mathcal{T}_{k,\epsilon\sqrt{n}}(Y_l). \end{aligned}$$

Fix a positive integer R. Whenever $n \ge (R/\epsilon)^2$,

$$L_n(\epsilon) \le \frac{1}{n} \sum_{l=0}^{n-1} \mathcal{T}_{k,R}(Y_l).$$

Taking the intersection of the full-probability ergodic events over $R \in \mathbb{N}$ gives

$$\limsup_{n\to\infty} L_n(\epsilon) \le \int_{\mathcal{S}} \mathcal{T}_{k,R}\, d\mu \quad \text{almost surely}$$

for every $R \in \mathbb{N}$. Letting $R \to \infty$ proves the conditional Lindeberg condition. The martingale central limit theorem now gives

$$\frac{1}{\sqrt{n}} \sum_{l=0}^{n-1} \xi_{k,l+1} \xrightarrow{d} \mathcal{N}(0, q_k).$$

Combining this limit with $S_{k,n} \to s_k^2$ and applying Slutsky's theorem yields

$$\sqrt{n}\left(\widehat{\nu}(\phi_k) - e^{\lambda_k \tau}\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{q_k}{s_k^4}\right).$$

Finally, $\widehat{\nu}(\phi_k) \to e^{\lambda_k\tau} > 0$ almost surely, so $\widehat{\nu}(\phi_k) > 0$ eventually almost surely. For the corresponding oracle estimator obtained by setting $\widetilde{\phi}_k = \phi_k$ in the definition of $\widehat{\lambda}_k$, continuity of the logarithm gives

$$\widehat{\lambda}_k = \tau^{-1} \log \widehat{\nu}(\phi_k) \longrightarrow \tau^{-1} \log(e^{\lambda_k\tau}) = \lambda_k \quad \text{almost surely}.$$

Applying the delta method to the same logarithmic transformation, whose derivative at $e^{\lambda_k\tau}$ is $1/(\tau e^{\lambda_k\tau})$, gives

$$\sqrt{n}\left(\widehat{\lambda}_k - \lambda_k\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{q_k}{\tau^2 e^{2\lambda_k\tau} s_k^4}\right),$$

which completes the proof of Theorem 5.2.

B.5 Proof of Lemma 5.3

When $\mathcal{L}_{\mathrm{orth}}(\boldsymbol{\theta}_\Phi) = 0$, the empirical Gram matrix satisfies $\frac{1}{n}\mathbf{Y}_0\mathbf{Y}_0^\top = \mathbf{I}$. On the other hand, both $\frac{1}{n}\mathbf{Y}_0\mathbf{Y}_0^\top$ and $\frac{1}{n}\mathbf{Y}_1\mathbf{Y}_1^\top$ estimate the population Gram matrix $\int_{\mathcal{S}} \widetilde{\Phi}\widetilde{\Phi}^\top d\mu$, i.e.,

$$\mathbf{Y}_1\mathbf{Y}_1^\top = n\mathbf{I}.$$

The dynamic loss can be expanded as

$$\begin{aligned} \mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) &= \frac{1}{n}\|\mathbf{Y}_1 - \widehat{\boldsymbol{\Lambda}}\mathbf{Y}_0\|_F^2 = \frac{1}{n}\operatorname{tr}\left((\mathbf{Y}_1 - \widehat{\boldsymbol{\Lambda}}\mathbf{Y}_0)(\mathbf{Y}_1 - \widehat{\boldsymbol{\Lambda}}\mathbf{Y}_0)^\top\right) \\ &= \frac{1}{n}\left[\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) - 2\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_0^\top\widehat{\boldsymbol{\Lambda}}) + \operatorname{tr}(\widehat{\boldsymbol{\Lambda}}\mathbf{Y}_0\mathbf{Y}_0^\top\widehat{\boldsymbol{\Lambda}})\right]. \end{aligned} \tag{17}$$

Using the orthogonality condition $\mathbf{Y}_0\mathbf{Y}_0^\top = n\mathbf{I}$ and the diagonality of $\widehat{\mathbf{\Lambda}}$, we have

$$\operatorname{tr}(\widehat{\mathbf{\Lambda}}\mathbf{Y}_0\mathbf{Y}_0^\top\widehat{\mathbf{\Lambda}}) = \operatorname{tr}(\widehat{\mathbf{\Lambda}}(n\mathbf{I})\widehat{\mathbf{\Lambda}}) = n\operatorname{tr}(\widehat{\mathbf{\Lambda}}^2) = n\sum_{k=1}^{N}\widehat{\nu}(\widetilde{\phi}_k)^2.$$

Simultaneously, since $\widehat{\mathbf{\Lambda}} = \operatorname{diag}\big(\widehat{\nu}(\widetilde{\phi}_1),\dots,\widehat{\nu}(\widetilde{\phi}_N)\big)$ is a diagonal matrix,

$$\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_0^\top\widehat{\mathbf{\Lambda}}) = \sum_{k=1}^{N}(\mathbf{Y}_1\mathbf{Y}_0^\top)_{kk}\widehat{\nu}(\widetilde{\phi}_k) = \sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)\widehat{\nu}(\widetilde{\phi}_k).$$

Substituting this into Eq. (17) yields:

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = \frac{1}{n}\left[\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) - 2\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)\widehat{\nu}(\widetilde{\phi}_k) + n\sum_{k=1}^{N}\widehat{\nu}(\widetilde{\phi}_k)^2\right]. \tag{18}$$

Define the row vectors

$$\mathbf{Y}_{0k} = \big[\widetilde{\phi}_k(\boldsymbol{x}_0;\boldsymbol{\theta}_\Phi),\cdots,\widetilde{\phi}_k(\boldsymbol{x}_{n-1};\boldsymbol{\theta}_\Phi)\big] \in \mathbb{R}^{1\times n},$$
$$\mathbf{Y}_{1k} = \big[\widetilde{\phi}_k(\boldsymbol{x}_1;\boldsymbol{\theta}_\Phi),\cdots,\widetilde{\phi}_k(\boldsymbol{x}_n;\boldsymbol{\theta}_\Phi)\big] \in \mathbb{R}^{1\times n},$$

for $k = 1,\dots,N$. It is clear that $(\mathbf{Y}_0\mathbf{Y}_1^\top)_{kk} = \mathbf{Y}_{0k}\mathbf{Y}_{1k}^\top$ and $(\mathbf{Y}_0\mathbf{Y}_0^\top)_{kk} = \mathbf{Y}_{0k}\mathbf{Y}_{0k}^\top$.

From $\mathbf{Y}_0\mathbf{Y}_0^\top = n\mathbf{I}$, we have $\|\mathbf{Y}_{0k}\|^2 = \mathbf{Y}_{0k}\mathbf{Y}_{0k}^\top = n$. Thus, Eq. (8) gives

$$\widehat{\nu}(\widetilde{\phi}_k) = \frac{(\mathbf{Y}_0\mathbf{Y}_1^\top)_{kk}}{(\mathbf{Y}_0\mathbf{Y}_0^\top)_{kk}} = \frac{\mathbf{Y}_{0k}\mathbf{Y}_{1k}^\top}{\|\mathbf{Y}_{0k}\|^2} = \frac{\mathbf{Y}_{0k}\mathbf{Y}_{1k}^\top}{n} = \frac{\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top}{n}, \quad k = 1,\dots,N.$$

Substituting this into Eq. (18):

$$\begin{aligned}
\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) &= \frac{1}{n}\left[\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) - 2\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)\left(\frac{\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top}{n}\right) + n\sum_{k=1}^{N}\left(\frac{\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top}{n}\right)^2\right] \\
&= \frac{1}{n}\left[\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) - \frac{2}{n}\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)^2 + \frac{1}{n}\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)^2\right] \\
&= \frac{1}{n}\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) - \frac{1}{n^2}\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)^2.
\end{aligned}$$

Since the eigenfunctions are orthonormal on $(\boldsymbol{x}_l)_{l=0}^n$,

$$\frac{1}{n}\operatorname{tr}(\mathbf{Y}_1\mathbf{Y}_1^\top) = \frac{1}{n}\operatorname{tr}(n\mathbf{I}) = N.$$

Furthermore, from $\widehat{\lambda}_k = \tau^{-1}\ln\widehat{\nu}(\widetilde{\phi}_k)$, we obtain

$$\frac{1}{n^2}\sum_{k=1}^{N}(\mathbf{Y}_{1k}\mathbf{Y}_{0k}^\top)^2 = \frac{1}{n^2}\sum_{k=1}^{N}[n\widehat{\nu}(\widetilde{\phi}_k)]^2 = \sum_{k=1}^{N}e^{2\widehat{\lambda}_k\tau}.$$

Therefore,

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}_\Phi) = N - \sum_{k=1}^{N}e^{2\widehat{\lambda}_k\tau},$$

which completes the proof of Lemma 5.3.

B.6 Empirical Eigenfunction-Loss Bound

Theorem B.1. *Let $\mathcal{F}(P,L)$ be a neural network class with parameter space Θ_Φ, and let $\phi_1,\dots,\phi_N$ be orthonormal eigenfunctions of $\mathcal{K}^\tau$ with eigenvalues $\lambda_1,\dots,\lambda_N$, where $\lambda_k \le 0$. Suppose that $\{X_{l\tau}\}_{l\ge 0}$ is stationary with invariant measure μ and is β-mixing with $\beta(q) \le C_\beta\rho^q$ for some*

$C_\beta > 0$ and $\rho \in (0,1)$. Assume also that $|\widetilde{\phi}_k(\boldsymbol{x};\boldsymbol{\theta}_\Phi)| \le B$ uniformly over $\boldsymbol{x} \in \mathcal{S}$, $\boldsymbol{\theta}_\Phi \in \Theta_\Phi$, and $k = 1,\dots,N$, and that Θ_Φ is restricted to parameters for which the denominators in Eq. (8) *are positive almost surely.*

Define the approximation error and the transition-noise level by

$$\varepsilon_\Phi(P,L) := \inf_{\boldsymbol{\theta}_\Phi \in \Theta_\Phi} \max_{1\le k\le N} \|\widetilde{\phi}_k(\cdot;\boldsymbol{\theta}_\Phi) - \phi_k\|_{L^2(\mu)},$$

$$\mathcal{V}_{\mathrm{dyn}} := \sum_{k=1}^{N} \left(1 - e^{2\lambda_k \tau}\right), \qquad C_{\mathrm{mix}} := 1 + \frac{8C_\beta \rho}{1-\rho}.$$

Then

$$\mathbb{E}\left[\inf_{\boldsymbol{\theta}_\Phi \in \Theta_\Phi} \mathcal{L}_{\mathrm{eigen}}(\boldsymbol{\theta}_\Phi)\right] \le \mathcal{V}_{\mathrm{dyn}} + 4\sqrt{N\mathcal{V}_{\mathrm{dyn}}}\,\varepsilon_\Phi(P,L) + 4N\varepsilon_\Phi^2(P,L) \tag{19}$$

$$+ \alpha N^2 \left(2\varepsilon_\Phi(P,L) + \varepsilon_\Phi^2(P,L)\right)^2 + \frac{\alpha N^2 B^4 C_{\mathrm{mix}}}{n}, \tag{20}$$

where the expectation is with respect to the stationary trajectory.

Proof. Write $\nu_k = e^{\lambda_k \tau}$ and fix any $\varepsilon > \varepsilon_\Phi(P,L)$. By the definition of $\varepsilon_\Phi(P,L)$, there exists $\boldsymbol{\theta}_\Phi^\varepsilon \in \Theta_\Phi$ such that, with

$$\overline{\phi}_k := \widetilde{\phi}_k(\cdot;\boldsymbol{\theta}_\Phi^\varepsilon), \qquad h_k := \overline{\phi}_k - \phi_k,$$

we have $\|h_k\|_{L^2(\mu)} \le \varepsilon$ for every k. Let $\overline{\Phi} = (\overline{\phi}_1,\dots,\overline{\phi}_N)^\top$.

We first control the dynamic loss. Since $\widehat{\nu}(\overline{\phi}_k)$ in Eq. (8) minimizes the empirical least-squares loss over the scalar coefficient, it follows that

$$\mathcal{L}_{\mathrm{dyn}}(\overline{\Phi}) \le \frac{1}{n}\sum_{l=0}^{n-1}\sum_{k=1}^{N} \left|\overline{\phi}_k(\boldsymbol{x}_{l+1}) - \nu_k \overline{\phi}_k(\boldsymbol{x}_l)\right|^2. \tag{21}$$

For a stationary pair (X_0, X_τ), the eigenfunction relation and orthonormality give

$$\begin{aligned}\mathbb{E}\left|\phi_k(X_\tau) - \nu_k\phi_k(X_0)\right|^2 &= \mathbb{E}[\phi_k^2(X_\tau)] - 2\nu_k\mathbb{E}[\phi_k(X_\tau)\phi_k(X_0)] + \nu_k^2\mathbb{E}[\phi_k^2(X_0)] \\ &= 1 - \nu_k^2.\end{aligned}$$

Stationarity also yields

$$\|h_k(X_\tau) - \nu_k h_k(X_0)\|_{L^2} \le \|h_k(X_\tau)\|_{L^2} + \nu_k\|h_k(X_0)\|_{L^2} \le 2\varepsilon,$$

where $0 < \nu_k \le 1$ because $\lambda_k \le 0$. Hence, by the Minkowski inequality,

$$\left\|\overline{\phi}_k(X_\tau) - \nu_k\overline{\phi}_k(X_0)\right\|_{L^2} \le \sqrt{1-\nu_k^2} + 2\varepsilon.$$

Taking expectations in Eq. (21), using stationarity, and applying the Cauchy–Schwarz inequality gives

$$\begin{aligned}\mathbb{E}\mathcal{L}_{\mathrm{dyn}}(\overline{\Phi}) &\le \sum_{k=1}^{N}\left(\sqrt{1-\nu_k^2} + 2\varepsilon\right)^2 \\ &\le \mathcal{V}_{\mathrm{dyn}} + 4\sqrt{N\mathcal{V}_{\mathrm{dyn}}}\,\varepsilon + 4N\varepsilon^2.\end{aligned} \tag{22}$$

We next control the orthogonality loss. Define

$$\widehat{\mathbf{G}} := \frac{1}{n}\sum_{l=0}^{n-1}\overline{\Phi}(\boldsymbol{x}_l)\overline{\Phi}(\boldsymbol{x}_l)^\top, \qquad \mathbf{G} := \mathbb{E}_\mu[\overline{\Phi}(\boldsymbol{x})\overline{\Phi}(\boldsymbol{x})^\top].$$

For every i, j,

$$\begin{aligned}|\mathbf{G}_{ij} - \delta_{ij}| &= |\langle\phi_i + h_i, \phi_j + h_j\rangle_\mu - \langle\phi_i,\phi_j\rangle_\mu| \\ &\le \|h_i\|_{L^2(\mu)} + \|h_j\|_{L^2(\mu)} + \|h_i\|_{L^2(\mu)}\|h_j\|_{L^2(\mu)} \le 2\varepsilon + \varepsilon^2.\end{aligned}$$

Consequently,

$$\|\mathbf{G} - \mathbf{I}\|_F^2 \le N^2(2\varepsilon + \varepsilon^2)^2.$$

It remains to bound the fluctuation of $\widehat{\mathbf{G}}$. Fix i, j and let $Z_l = \overline{\phi}_i(X_{l\tau})\overline{\phi}_j(X_{l\tau})$. Then $|Z_l| \le B^2$. The standard covariance inequality for β-mixing variables gives

$$|\operatorname{Cov}(Z_0, Z_q)| \le 4B^4\beta(q) \le 4B^4 C_\beta \rho^q.$$

Therefore,

$$\begin{aligned}\mathbb{E}|\widehat{\mathbf{G}}_{ij} - \mathbf{G}_{ij}|^2 &= \operatorname{Var}\left(\frac{1}{n}\sum_{l=0}^{n-1} Z_l\right) \\ &\le \frac{B^4}{n} + \frac{2}{n^2}\sum_{q=1}^{n-1}(n-q)|\operatorname{Cov}(Z_0, Z_q)| \\ &\le \frac{B^4}{n}\left(1 + \frac{8C_\beta\rho}{1-\rho}\right) = \frac{B^4 C_{\mathrm{mix}}}{n}.\end{aligned}$$

Since $\mathbb{E}\widehat{\mathbf{G}} = \mathbf{G}$, the cross term in the following expansion vanishes:

$$\begin{aligned}\mathbb{E}\mathcal{L}_{\mathrm{orth}}(\overline{\Phi}) &= \mathbb{E}\|\widehat{\mathbf{G}} - \mathbf{I}\|_F^2 \\ &= \|\mathbf{G} - \mathbf{I}\|_F^2 + \mathbb{E}\|\widehat{\mathbf{G}} - \mathbf{G}\|_F^2 \\ &\le N^2(2\varepsilon + \varepsilon^2)^2 + \frac{N^2 B^4 C_{\mathrm{mix}}}{n}.\end{aligned} \tag{23}$$

Finally,

$$\mathbb{E}\left[\inf_{\boldsymbol{\theta}_\Phi \in \Theta_\Phi} \mathcal{L}_{\mathrm{eigen}}(\boldsymbol{\theta}_\Phi)\right] \le \mathbb{E}\mathcal{L}_{\mathrm{eigen}}(\boldsymbol{\theta}_\Phi^\varepsilon).$$

Combining Eqs. (22) and (23), and then letting $\varepsilon \downarrow \varepsilon_\Phi(P, L)$, proves Eq. (19).

B.7 Empirical Generator-Loss Bound

Theorem B.2. *Fix the trajectory $(\boldsymbol{x}_l)_{l=0}^n$, the learned eigenfunctions $\widetilde{\Phi}$, and $\widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}$. Let Θ_b and Θ_σ denote the parameter spaces of $\widetilde{\boldsymbol{b}}$ and $\widetilde{\mathbf{D}}$, respectively. Suppose that each $\widetilde{\phi}_k$ belongs to the domain of $\mathcal{L}$ and*

$$\|\nabla\widetilde{\phi}_k(\boldsymbol{x})\| \le B_1, \qquad \|\mathbf{H}_k(\boldsymbol{x})\|_F \le B_2, \qquad k = 1, \dots, N,$$

for all $\boldsymbol{x} \in \mathcal{S}$. Define the empirical eigenpair defect

$$\mathcal{E}_\Phi := \frac{1}{n}\sum_{l=0}^{n-1}\sum_{k=1}^{N}\left|\mathcal{L}\widetilde{\phi}_k(\boldsymbol{x}_l) - \widehat{\lambda}_k\widetilde{\phi}_k(\boldsymbol{x}_l)\right|^2.$$

For $\boldsymbol{\theta}_\sigma \in \Theta_\sigma$, let

$$\mathcal{V}_{\mathrm{HTE}}(\boldsymbol{\theta}_\sigma) := \frac{1}{n}\sum_{l=0}^{n-1}\sum_{k=1}^{N}\sum_{i<j}\left[\left(\widetilde{\mathbf{D}}(\boldsymbol{x}_l;\boldsymbol{\theta}_\sigma)\mathbf{H}_k(\boldsymbol{x}_l)\right)_{ij} + \left(\widetilde{\mathbf{D}}(\boldsymbol{x}_l;\boldsymbol{\theta}_\sigma)\mathbf{H}_k(\boldsymbol{x}_l)\right)_{ji}\right]^2.$$

If every trace in $\mathbf{L}$ is evaluated with m independent Rademacher vectors as in Eq. (11)*, then*

$$\mathbb{E}_{\boldsymbol{z}}\left[\inf_{\substack{\boldsymbol{\theta}_b \in \Theta_b \\ \boldsymbol{\theta}_\sigma \in \Theta_\sigma}} \mathcal{L}_{\mathrm{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma) \,\middle|\, (\boldsymbol{x}_l)_{l=0}^n, \widetilde{\Phi}, \widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}\right] \tag{24}$$

$$\le 3\mathcal{E}_\Phi + \inf_{\substack{\boldsymbol{\theta}_b \in \Theta_b \\ \boldsymbol{\theta}_\sigma \in \Theta_\sigma}}\left\{\frac{3NB_1^2}{n}\sum_{l=0}^{n-1}\|\widetilde{\boldsymbol{b}}(\boldsymbol{x}_l;\boldsymbol{\theta}_b) - \boldsymbol{b}(\boldsymbol{x}_l)\|^2\right. \tag{25}$$

$$\left. + \frac{3NB_2^2}{4n}\sum_{l=0}^{n-1}\|\widetilde{\mathbf{D}}(\boldsymbol{x}_l;\boldsymbol{\theta}_\sigma) - \mathbf{D}(\boldsymbol{x}_l)\|_F^2 + \frac{\mathcal{V}_{\mathrm{HTE}}(\boldsymbol{\theta}_\sigma)}{4m}\right\}. \tag{26}$$

Proof. Condition on $(\boldsymbol{x}_l)_{l=0}^n$, $\widetilde{\Phi}$, and $\widehat{\boldsymbol{\Lambda}}_{\mathcal{L}}$. Fix $(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma) \in \Theta_b \times \Theta_\sigma$. For $l = 0, \ldots, n-1$ and $k = 1, \ldots, N$, define the exact-trace residual

$$R_{kl} := \widetilde{\boldsymbol{b}}(\boldsymbol{x}_l; \boldsymbol{\theta}_b) \cdot \nabla \widetilde{\phi}_k(\boldsymbol{x}_l) + \frac{1}{2} \operatorname{tr}\left(\widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) \mathbf{H}_k(\boldsymbol{x}_l)\right) - \widehat{\lambda}_k \widetilde{\phi}_k(\boldsymbol{x}_l).$$

Let

$$\xi_{kl} := \widehat{\operatorname{tr}}\left(\widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) \mathbf{H}_k(\boldsymbol{x}_l)\right) - \operatorname{tr}\left(\widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) \mathbf{H}_k(\boldsymbol{x}_l)\right)$$

be the HTE error. The residual used in $\mathcal{L}_{\text{gen}}$ is $R_{kl} + \xi_{kl}/2$. By Proposition 5.1,

$$\mathbb{E}_{\boldsymbol{z}}[\xi_{kl}] = 0, \qquad \mathbb{E}_{\boldsymbol{z}}[\xi_{kl}^2] = \frac{1}{m} \sum_{i<j} \left[(\mathbf{A}_{kl})_{ij} + (\mathbf{A}_{kl})_{ji}\right]^2,$$

where

$$\mathbf{A}_{kl} := \widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) \mathbf{H}_k(\boldsymbol{x}_l).$$

It follows that

$$\mathbb{E}_{\boldsymbol{z}} \mathcal{L}_{\text{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma) = \frac{1}{n} \sum_{l=0}^{n-1} \sum_{k=1}^{N} |R_{kl}|^2 + \frac{\mathcal{V}_{\text{HTE}}(\boldsymbol{\theta}_\sigma)}{4m}. \tag{27}$$

To bound the exact-trace residual, add and subtract the true generator:

$$\begin{aligned} R_{kl} = {} & \left[\widetilde{\boldsymbol{b}}(\boldsymbol{x}_l; \boldsymbol{\theta}_b) - \boldsymbol{b}(\boldsymbol{x}_l)\right] \cdot \nabla \widetilde{\phi}_k(\boldsymbol{x}_l) \\ & + \frac{1}{2} \operatorname{tr}\left(\left[\widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) - \mathbf{D}(\boldsymbol{x}_l)\right] \mathbf{H}_k(\boldsymbol{x}_l)\right) \\ & + \mathcal{L}\widetilde{\phi}_k(\boldsymbol{x}_l) - \widehat{\lambda}_k \widetilde{\phi}_k(\boldsymbol{x}_l). \end{aligned}$$

Using $|a+b+c|^2 \le 3(|a|^2 + |b|^2 + |c|^2)$, the derivative bounds in the theorem, and $|\operatorname{tr}(\mathbf{AH})| \le \|\mathbf{A}\|_F \|\mathbf{H}\|_F$, we obtain

$$\begin{aligned} \frac{1}{n} \sum_{l=0}^{n-1} \sum_{k=1}^{N} |R_{kl}|^2 \le 3\mathcal{E}_\Phi + \frac{3NB_1^2}{n} \sum_{l=0}^{n-1} \|\widetilde{\boldsymbol{b}}(\boldsymbol{x}_l; \boldsymbol{\theta}_b) - \boldsymbol{b}(\boldsymbol{x}_l)\|^2 \\ + \frac{3NB_2^2}{4n} \sum_{l=0}^{n-1} \|\widetilde{\mathbf{D}}(\boldsymbol{x}_l; \boldsymbol{\theta}_\sigma) - \mathbf{D}(\boldsymbol{x}_l)\|_F^2. \end{aligned} \tag{28}$$

For every fixed pair $(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma)$,

$$\mathbb{E}_{\boldsymbol{z}}\left[\inf_{\substack{\boldsymbol{\vartheta}_b \in \Theta_b \\ \boldsymbol{\vartheta}_\sigma \in \Theta_\sigma}} \mathcal{L}_{\text{gen}}(\boldsymbol{\vartheta}_b, \boldsymbol{\vartheta}_\sigma)\right] \le \mathbb{E}_{\boldsymbol{z}} \mathcal{L}_{\text{gen}}(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma).$$

Combining Eqs. (27) and (28), and then taking the infimum over $(\boldsymbol{\theta}_b, \boldsymbol{\theta}_\sigma)$, proves Eq. (24).

B.8 Derivation of Orthogonal Eigenfunctions and Eigenvalues for the One-Dimensional OU Process (Appendix A.9)

Consider the one-dimensional OU process X_t governed by the SDE:

$$dX_t = \theta(\mu - X_t)dt + \sigma dB_t, \quad \theta > 0, \sigma > 0, \tag{29}$$

where B_t is a standard Brownian motion, μ is the long-term mean, θ is the mean-reversion rate, and σ is the volatility.

According to Eq. (3), the generator for Eq. (29) is:

$$\mathcal{L}f(x) = \theta(\mu - x)f'(x) + \frac{\sigma^2}{2} f''(x).$$

Next, we solve the eigen-equation $\mathcal{L}\phi = \lambda\phi$:

$$\frac{\sigma^2}{2}\phi''(x) + \theta(\mu - x)\phi'(x) = \lambda\phi(x). \tag{30}$$

Introduce the dimensionless variable $y = \sqrt{\frac{\theta}{\sigma^2}}(x - \mu) = \frac{x-\mu}{\sigma/\sqrt{\theta}}$, and let $\phi(x) = \psi(y)$.

Compute the derivatives:

$$\frac{d\phi}{dx} = \frac{d\psi}{dy}\frac{dy}{dx} = \sqrt{\frac{\theta}{\sigma^2}}\psi'(y), \quad \frac{d^2\phi}{dx^2} = \frac{\theta}{\sigma^2}\psi''(y).$$

Substituting into the eigenvalue equation Eq. (30) yields:

$$-\sigma\sqrt{\theta}y \cdot \sqrt{\frac{\theta}{\sigma^2}}\psi'(y) + \frac{\sigma^2}{2} \cdot \frac{\theta}{\sigma^2}\psi''(y) = \lambda\psi(y).$$

Simplifying, we obtain:

$$\psi''(y) - 2y\psi'(y) - \frac{2\lambda}{\theta}\psi(y) = 0.$$

Comparing this with the Hermite equation of physicist:

$$H_n''(y) - 2yH_n'(y) + 2nH_n(y) = 0,$$

we identify $\psi(y) = H_n(y)$, with eigenvalues satisfying $\lambda/\theta = -n$, i.e.:

$$\lambda_n = -n\theta, \quad n = 0, 1, 2, \ldots,$$

when $\theta = 1$, $\lambda_n = -n$.

The corresponding unnormalized eigenfunctions are:

$$\phi_n(x) = H_n\left(\sqrt{\theta}\,\frac{x-\mu}{\sigma}\right).$$

The stationary distribution of Eq. (29) is $\mathcal{N}\left(\mu, \frac{\sigma^2}{2\theta}\right)$, with density function

$$p(x) = \sqrt{\frac{\theta}{\pi\sigma^2}}\exp\left(-\frac{\theta(x-\mu)^2}{\sigma^2}\right).$$

We then normalize $\phi_n(x)$ under the weighted inner product $\langle f, g\rangle_\mu = \int f(x)g(x)p(x)dx$. Using the orthogonality of Hermite polynomials:

$$\int_{-\infty}^{\infty} H_m(z)H_n(z)e^{-z^2}dz = \sqrt{\pi}\,2^n n!\,\delta_{mn},$$

we compute:

$$\|\phi_n\|^2_{L^2(\mu)} = \int_{-\infty}^{\infty} H_n^2\left(\sqrt{\theta}\frac{x-\mu}{\sigma}\right)p(x)dx = 2^n n!.$$

Therefore, the normalized eigenfunctions are:

$$\phi_n(x) = \frac{1}{\sqrt{2^n n!}}\,H_n\left(\sqrt{\theta}\,\frac{x-\mu}{\sigma}\right),$$

which form an orthonormal basis for $L^2(\mu)$: $\langle\phi_m, \phi_n\rangle_\mu = \delta_{mn}$.