Supplementary Information for Learning Stochastic Dynamics via Evolving Weight of Neural Network

Contents

I Proof of Theorem

Let μ_0 and μ_1 be the probability measures on $\mathbb{R}^d$, and let (p^*, v^*) be the optimal path of the energy functional:

$$(p^*, v^*) = \arg\min_{(p,v)\in\mathcal{F}} \mathcal{E}(p,v), \tag{1}$$

subject to the continuity equation:

$$\partial_t p + \nabla(pv) = 0, \tag{2}$$

with $p(t=0) = \mu_0$ and $p(t=1) = \mu_1$, and the energy functional [1]:

$$\mathcal{E}(p,v) := \int_0^1 \int \left[\frac{1}{2}||v(x,t)^2|| + \frac{\sigma^4}{8}||\nabla \log p(x,t)||^2\right] p(x,t)\mathrm{d}x\mathrm{d}t. \tag{3}$$

Let $v_g(x,t)$ be a parameterized neural network defining an ODE:

$$\frac{\mathbf{d}\theta}{\mathbf{d}t} = g(\theta, t), \tag{4}$$

where $\frac{d\theta}{dt}$ and $\frac{dp_\theta}{dt}$ satisfies a linear transform [2], and subject to the following continuity equation:

$$\partial_t p_\theta + \nabla(p_\theta g) = 0. \tag{5}$$

A Lemma 1

If there exists a reference parameter θ_0 such that $p_{g_0} \approx p^*$ and $v_{g_0} \approx v^*$, and $\mathcal{E}(p,v)$ is continuous, then we have

$$|\mathcal{E}(p_{g_0}, v_{g_0}) - \mathcal{E}(p^*, v^*)| \leq C\varepsilon. \tag{6}$$

Proof: Let $|\mathcal{E}(p_{g_0}, v_{g_0}) - \mathcal{E}(p^*, v^*)| = \Delta_1 + \Delta_2$, where

$$\Delta_1 = \left|\frac{1}{2}\int \left[||v_g(x,t)^2||p_{g_0} - ||v(x,t)^2||p^*\right]\mathrm{d}x\mathrm{d}t\right| \tag{7}$$

and

$$\Delta_2 = \left| \frac{\sigma^4}{8} ||\nabla \log p_\theta(x,t)||^2 - \frac{\sigma^4}{8} ||\nabla \log p(x,t)||^2 \right] \mathrm{d}x\mathrm{d}t \Big|. \tag{8}$$

Based on the triangle inequality, we have

$$\Delta_1 \leq \left| \frac{1}{2} \int \left(||v_g(x,t)^2|| - ||v(x,t)^2|| \right) p^* \mathrm{d}x\mathrm{d}t \right| + \left| \frac{1}{2} \int ||v_g(x,t)^2|| \left(p_{g_0} - p^* \right) \mathrm{d}x\mathrm{d}t \right|. \tag{9}$$

Consinder the first term of the RHS in (9), we have

$$||v_g(x,t)^2|| - ||v(x,t)^2|| \leq ||v_g(x,t) - v(x,t)|| \cdot ||v_g(x,t) + v(x,t)||. \tag{10}$$

Because $v_g(x,t) \approx v(x,t)$, the integral of the first term of the RHS in (9) satisfies

$$\left| \int \left(||v_g(x,t)^2|| - ||v(x,t)^2|| \right) p^* \mathrm{d}x\mathrm{d}t \right| \leq C\varepsilon. \tag{11}$$

Moreover, due to $p_{g_0} \approx p^*$, we obtain $|p_{g_0} - p^*| < \varepsilon$. Then the second term of the RHS in (9) satisfies

$$\left| \int ||v_g(x,t)^2|| \left(p_{g_0} - p^* \right) \mathrm{d}x\mathrm{d}t \right| \leq C\varepsilon. \tag{12}$$

Therefore, we prove that $\delta_1 \leq C\varepsilon$. Similar to Δ_1, for Δ_2 we also have

$$\Delta_2 \leq \left(||p_{g_0} - p^*|| + ||\nabla \log p_{g_0} - \nabla \log p^*|| \right) \leq C\varepsilon. \tag{13}$$

In sum, we have

$$|\mathcal{E}(p_{g_0}, v_{g_0}) - \mathcal{E}(p^*, v^*)| = \Delta_1 + \Delta_2 \leq C\varepsilon \tag{14}$$

Q.E.D.

B Lemma 2

If the trained minimizer θ^* satisfies $p_{\theta^*}(1) \approx \mu_1$, then

$$\mathcal{E}(p_{\theta^*}, v_{g^*}) \geq \mathcal{E}(p^*, v^*) - o(1). \tag{15}$$

Proof: Let $(\tilde{p}(t),\tilde{v}(t))$ defines a new path in the functional space, where $\tilde{p}(t=0)=\mu_0$ and $\tilde{p}(t=1)=\mu_1$. For $t\in[0,1-\delta]$, $\tilde{p}(t):=p_{\theta^*}(t)$ and $\tilde{v}(t):=v_{g^*}(t)$. Hence, we have $||\tilde{p}(1)-\mu_1||<\varepsilon$, where $\varepsilon\ll 1$. For $t\in[1-\delta,1]$, we introduce a correction segment of the path $(\tilde{p}(t),\tilde{v}(t))=(p_a(t),v_a(t))$. Therefore,

$$\begin{aligned}\mathcal{E}(\tilde{p},\tilde{v}) &= I_1+I_2\\ &= \int_0^{1-\delta}\int\left[\frac{1}{2}||v_{g^*}(x,t)^2||+\frac{\sigma^4}{8}||\nabla\log p_{\theta^*}(x,t)||^2\right]p_{\theta^*}(x,t)\mathrm{d}x\mathrm{d}t\\ &+\int_{1-\delta}^{1}\int\left[\frac{1}{2}||v_{corr}(x,t)^2||+\frac{\sigma^4}{8}||\nabla\log p_{corr}(x,t)||^2\right]p_{corr}(x,t)\mathrm{d}x\mathrm{d}t,\end{aligned}\tag{16}$$

where $I_1\approx\mathcal{E}(p_{\theta^*},v_{g^*})$. Based on [3], $I_2\le C\cdot\varepsilon$. Thus, we obtain

$$\mathcal{E}(p^*,v^*)\le\mathcal{E}(\tilde{p},\tilde{v})\le\mathcal{E}(p_{\theta^*},v_{g^*})+C\varepsilon.\tag{17}$$

Q.E.D.

C Lemma 3

If $\theta^*=\arg\min_\theta(\lambda\mathcal{E}(p_\theta,v_g)+\mathcal{L}(g))$ with $\lambda>0$, then we have

$$\mathcal{E}(p_{\theta^*},v_{g^*})\le\mathcal{E}(p_{\theta_0},v_{g_0})+o(1).\tag{18}$$

Proof: By the definition of the training process, $\mathcal{L}(g^*)+\lambda\mathcal{E}(p_{\theta^*},v_{g^*})\le\mathcal{L}(g_0)+\lambda\mathcal{E}(p_{\theta_0},v_{g_0})$. This leads to $\mathcal{E}(p_{\theta^*},v_{g^*})\le\frac{1}{\lambda}(\mathcal{L}(g_0)-\mathcal{L}(g^*))+\mathcal{E}(p_{\theta_0},v_{g_0})$. Given the Lemma 1 and 2, the loss of the training process satisfies $\mathcal{L}(g^*)\le\mathcal{L}(\theta_0)+o(1)$. Thus, we obtain the following:

$$\mathcal{E}(p_{\theta^*},v_{g^*})\le\mathcal{E}(p_{\theta_0},v_{g_0})+o(1).\tag{19}$$

Q.E.D.

D Theorem

Assuming following conditions are satisfied:

(**C1**) There exists a reference parameter θ_0 such that $p_{g_0}\approx p^*$ and $v_{g_0}\approx v^*$;

(**C2**) The trained minimizer θ^* satisfies $p_{\theta^*}(1)\approx\mu_1$;

(**C3**) $\theta^* = \arg\min_\theta(\lambda\mathcal{E}(p_\theta, v_g) + \mathcal{L}(g))$ with $\lambda > 0$, where

$$\mathcal{E}(p_\theta, v_g) \coloneqq \int_0^1 \int \left[\frac{1}{2}||v_g(x,t)^2|| + \frac{\sigma^4}{8}||\nabla \log p_\theta(x,t)||^2\right] p_\theta(x,t)\mathrm{d}x\mathrm{d}t, \tag{20}$$

and $\mathcal{L}(g) = -\mathbb{E}_{x\sim\mu_0}[\log p_{\theta_0}(x)] - \mathbb{E}_{x\sim\mu_1}[\log p_{\theta_1}(x)]$ is the approximation error for target measures μ_0 and μ_1. Then, under **C1**-**C3**, we have

$$|\mathcal{E}(p_{\theta^*}, v_{g^*}) - \mathcal{E}(p^*, v^*)| \leq \delta \tag{21}$$

Proof: When Lemma 1 suggests

$$\mathcal{E}(p_{g_0}, v_{g_0}) = \mathcal{E}(p_{\theta_0}, v_{g_0}) \leq \mathcal{E}(p^*, v^*) + C\varepsilon. \tag{22}$$

Incorporating Lemma 1 with 3 leads to

$$\mathcal{E}(p_{\theta^*}, v_{g^*}) \leq \mathcal{E}(p^*, v^*) + C\varepsilon + o(1). \tag{23}$$

Moreover, Lemma 2 indicates that

$$\mathcal{E}(p^*, v^*) \leq \mathcal{E}(p_{\theta^*}, v_{g^*}) + C'\varepsilon. \tag{24}$$

Thus, we finally obtain

$$|\mathcal{E}(p_{\theta^*}, v_{g^*}) - \mathcal{E}(p^*, v^*)| \leq \delta, \tag{25}$$

where $\delta = C''\varepsilon + o(1)$

II Implementation of WeightFlow

A Transformer-based Backbone

As stated in the main text, WeightFlow supports any autoregressive architecture as its backbone. Here we detail the implementation of a Transformer-based backbone. The Transformer's self-attention mechanism is well-suited for capturing complex dependencies among the dimensions of the state vector. To model the conditional probability $p(x_i|\mathbf{x}_{<i},t)$, the Transformer takes the sequence of preceding dimensions, $\mathbf{x}_{<i} = (x_1,...,x_{i-1})$, as input tokens. A crucial component is the use of a causal (or look-ahead) mask in the self-attention layers. This mask ensures that the prediction for the probability distribution of dimension x_i only depends on the outputs from the preceding dimensions x_j where $j < i$, thereby strictly maintaining the autoregressive property. The final output representation for each position is then passed through a linear layer to produce the parameters of the conditional probability distribution for the next dimension.

B Autoregressive Network

B.1 Discrete State Space

For a d-dimensional discrete-state system, we assume that each dimension has a fixed number of L candidate states. In this setup, WeightFlow's autoregressive backbone models the conditional probability distribution $p(x_i|\mathbf{x}_{<i})$ by outputting a categorical distribution over the L candidate states for each dimension i.

B.2 Continuous State Space

For a d-dimensional continuous-state system, the state of each dimension can be any real value. To model this continuous distribution, WeightFlow's backbone employs a Mixture Density Network (MDN). Specifically, we approximate the conditional probability distribution of each dimension with a Gaussian Mixture Model (GMM). For each dimension i, the autoregressive backbone predicts the parameters of the GMM-specifically, the mixture weights (π), means (μ), and standard deviations (σ) for its L components.

C Experimental Configurations

Here, we provide the hyperparameter settings for WeightFlow used in all system experiments reported in the main text.

Our framework consists of two main components: a **Backbone** network and a **Hypernetwork**. The backbone is a GRU-based autoregressive network that parameterizes the state distribution at each snapshot. For discrete systems, it outputs a categorical distribution over the candidate states, while for continuous

systems, it employs a Mixture Density Network (MDN) head to output the parameters of a Gaussian Mixture Model.

The hypernetwork is a Graph Neural Differential Equation (GNDE) that learns the evolution dynamics of the backbone's weights. In our experiments, this is implemented as a Controlled Differential Equation (CDE), which is guided by a one-dimensional latent path. This path is obtained by first projecting the pre-trained backbone anchor weights using a self-supervised Autoencoder (AE) and then performing cubic spline interpolation.

The training process is divided into two stages: a warm-up pre-training stage to fit the backbone to each snapshot and a main training stage to learn the dynamics with the GNDE. The specific configurations are summarized in Table 1.

D Training Strategy

The energy functional $\mathcal{E}(p_\theta, v_g)$ in our objective is defined on the data space. For computational tractability, we derive an equivalent and simplified formulation in the weight space Θ. The kinetic energy term $\mathcal{L}_{KE} = \int \frac{1}{2}\|v_g\|^2 p_\theta d\mathbf{x}$ induces a Riemannian metric on the manifold of probability distributions. Our model defines a map $G : \boldsymbol{\theta} \mapsto p_{\boldsymbol{\theta}}$, which parameterizes this manifold. This map induces a pullback metric on the weight space Θ, allowing the kinetic energy to be expressed in terms of the weight dynamics $\dot{\boldsymbol{\theta}} = d\boldsymbol{\theta}/dt$. The energy is given by the quadratic form:

$$\mathcal{L}_{KE} = \frac{1}{2}\dot{\boldsymbol{\theta}}^T \mathbf{M}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}, \tag{26}$$

where $\mathbf{M}(\boldsymbol{\theta})$ is the Fisher Information Matrix, serving as the pullback metric tensor. Minimizing the data-space kinetic energy is thus equivalent to minimizing this energy in the weight space. However, for a high-dimensional neural network, the Fisher Information Matrix $\mathbf{M}(\boldsymbol{\theta})$ is computationally intractable to compute at each time step. We therefore introduce a principled simplification by approximating the complex Riemannian metric with a standard Euclidean metric, i.e., letting $\mathbf{M}(\boldsymbol{\theta}) \approx \mathbf{I}$ (the identity matrix). This approximation yields a computationally efficient energy loss that regularizes the path in the Euclidean geometry of the weight space. The objective encourages the shortest path for the parameters $\boldsymbol{\theta}$, penalizing complex changes and promoting smoother learned dynamics. Thus, the energy loss term implemented in our training procedure becomes the integrated kinetic energy of the weight path itself:

$$\mathcal{L}_{\text{energy}} = \mathcal{E}(\boldsymbol{\theta}) = \int \frac{1}{2}\|\dot{\boldsymbol{\theta}}\|_2^2 dt = \int \frac{1}{2}\|g_\phi(\boldsymbol{\theta}_t, t)\|_2^2 dt. \tag{27}$$

This integral is computed efficiently alongside the CDE solver via the adjoint method. Whohle training process is discribed in Algorithm 1.

All experiments were completed on a single workstation with the following configuration: Intel Core i5-14600KF CPU, NVIDIA RTX 4090 GPU, and 64GB RAM, running on the Windows 11 operating system.

Appendix Table 1. Hyperparameter settings for WeightFlow.

Component	Hyperparameter	Value
	Discrete Systems	
Backbone	Architecture	GRU
	Hidden Dimension	8
	Number of Layers	1
	Mode	Discrete
Hypernetwork	Dynamics Type	NeuralCDE
	Latent Path Dimension (*z_dim*)	1
	Path Projection Method (*z_method*)	Autoencoder (AE)
	Hidden Dimension	64
	Number of Layers	2
	Attention Heads	4
	Dropout	0.1
	Continuous Systems	
Backbone	Architecture	GRU
	Hidden Dimension	32
	Number of Layers	1
	Mode	Continuous (MDN)
	Number of Mixtures	10
Hypernetwork	Dynamics Type	NeuralCDE
	Latent Path Dimension (*z_dim*)	1
	Path Projection Method (*z_method*)	Autoencoder (AE)
	Hidden Dimension	64
	Number of Layers	2
	Attention Heads	4
	Dropout	0.1
	Training	
Pre-training	Batch Size	64
	Learning Rate	0.01
	Fitting Steps	25
Main Training	Batch Size	64
	Learning Rate	0.001
	Max Epochs	100
	λ	0.000001

Algorithm 1 WeightFlow Training Procedure

Require: Observed data snapshots $\{\hat{\mu}_{t_i}\}_{i=1}^{N}$
Ensure: Trained dynamics model (hypernetwork) g_ϕ

// Stage 1: Pre-train anchor weights (Warm-up)
Initialize backbone network G with parameters $\boldsymbol{\theta}$.
for $i = 1$ to N **do**
    if $i > 1$ **then**
        Initialize current backbone with weights $\boldsymbol{\theta}_{t_{i-1}}$ from previous step (sequential-aligning).
    end if
    Pre-train to find anchor weights $\boldsymbol{\theta}_{t_i} \leftarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{NLL}}(\hat{\mu}_{t_i}; \boldsymbol{\theta})$.
end for
// Construct the latent control path
Obtain latent vectors $\{z_{t_i}\}_{i=1}^{N}$ by passing anchor weights through an encoder E.
Construct continuous control path Z_t via cubic spline interpolation of $\{z_{t_i}\}_{i=1}^{N}$.
// Stage 2: Train dynamics model g_ϕ
Initialize hypernetwork g_ϕ.
for each training epoch **do**
    $\mathcal{L}_{\text{total}} \leftarrow 0$.
    for $i = 1$ to N **do**
        Predict weight trajectory endpoint by solving the CDE:
        $\boldsymbol{\theta}'_{t_i} \leftarrow \boldsymbol{\theta}_0 + \int_0^{t_i} g_\phi(\boldsymbol{\theta}_t, t) \frac{dZ_t}{dt} dt$.
        Calculate reconstruction loss: $\mathcal{L}_{\text{recon}} \leftarrow ||\boldsymbol{\theta}'_{t_i} - \boldsymbol{\theta}_{t_i}||_2^2$.
        Calculate path energy loss $\mathcal{L}_{\text{energy}} \leftarrow \mathcal{E}(\boldsymbol{\theta}'_{t_i})$.
        $\mathcal{L}_{\text{total}} \leftarrow \mathcal{L}_{\text{total}} + \mathcal{L}_{\text{recon}} + \lambda \mathcal{L}_{\text{energy}}$.
    end for
    Update parameters ϕ of g_ϕ using gradient of $\mathcal{L}_{\text{total}}$.
end for
return g_ϕ

III Datasets

A Epidemic

This is a time-dependent Susceptible-Infected-Recovered (SIR) model that describes the transmission of an infectious disease in a periodic environment. The contact rate c_1 is time-dependent, modeled as a periodic function $c_1(t) = c_0(1 + \varepsilon \sin(\omega t))$, while other rates are constant. The reactions are [4]:

$$S + I \xrightarrow{c_1} 2I, \quad I \xrightarrow{c_2} R, \quad R \xrightarrow{c_3} S, \quad S \xrightarrow{c_4} \emptyset, \quad I \xrightarrow{c_5} \emptyset, \quad R \xrightarrow{c_6} \emptyset. \tag{28}$$

The simulation parameters are set to: $c_0 = 0.003, \varepsilon = 0.2, \omega = \pi/3, c_2 = 0.02, c_3 = 0.007, c_4 = 0.002, c_5 = 0.05, c_6 = 0.002$.

B Toggle Switch

This system models a genetic toggle switch where two proteins, P_x and P_y, mutually inhibit each other's gene expression (G_x and G_y, respectively), leading to a multimodal distribution with four stable states. The eight reactions governing the system are [4]:

$$G_x \xrightarrow{s_x} G_x + P_x, \quad G_y \xrightarrow{s_y} G_y + P_y, \quad P_x \xrightarrow{d_x} \emptyset, \quad P_y \xrightarrow{d_y} \emptyset \tag{29}$$

$$2P_x + G_y \xrightarrow{b_y} \bar{G}y, \quad 2Py + G_x \xrightarrow{b_x} \bar{G}x, \quad \bar{G}y \xrightarrow{u_y} 2P_x + G_y, \quad \bar{G}x \xrightarrow{u_x} 2Py + G_x. \tag{30}$$

The simulation parameters are set to: $s_x = s_y = 50.0$, $d_x = d_y = 1.0$, $b_x = b_y = 10^{-4}$, and $u_x = u_y = 0.1$.

C Signalling Cascade1

This is a linear intracellular signaling cascade, a common biological motif where a series of reactions occurs sequentially, with one species catalyzing the production of the next. The reactions for a system with M species are [4]:

$$\emptyset \xrightarrow{\beta_0} X_1, \quad X_i \xrightarrow{k_c} X_i + X_{i+1} \quad (i = 1, ..., M-1), \quad X_i \xrightarrow{\gamma} \emptyset \quad (i = 1, ..., M). \tag{31}$$

The simulation parameters are set to: $\beta_0 = 10.0$, $k_c = 5.0$, and $\gamma = 1.0$.

D Signalling Cascade2

This system is a nonlinear version of the signaling cascade, where the constant catalytic rate k_c is replaced by a nonlinear Hill activation function, $H(x_i)$, to model more complex biological interactions. The Hill

function is given by [4]:

$$H(x_i) = b + \frac{k_m x_i^h}{k_0 + x_i^h} \tag{32}$$

where b is the basal rate, k_m controls the activation strength, k_0 is the substrate affinity, and h is the Hill coefficient. The simulation parameters are set to: $\beta_0 \approx 166.7, \gamma = 2.5, b \approx 16.7, k_m \approx 1666.7, k_0 \approx 6.67$, and $h = 1.0$.

E Ecological Evolution

This system models the adaptive evolution of a population under a "strong selection, weak mutation" regime, a common framework in eco-evolutionary dynamics [5]. The core dynamic is the fixation of new mutations, where the probability of a new mutation j replacing the current state i is governed by the Kimura formula:

$$p_{i \to j} = \frac{1 - e^{-2s_i(j)}}{1 - e^{-2Ns_i(j)}}, \tag{33}$$

where N is the population size and $s_i(j) = \frac{f_j}{f_i} - 1$ is the selection coefficient, determined by the fitness values f_i and f_j of the respective genotypes. A key property of this system is that the logarithmic fitness aligns with the energy of a thermodynamic system. We uses a two-locus setting where each locus has 100 possible mutation types.

F *In Vitro* Pancreatic β-cell Differentiation

This dataset tracks the differentiation of human pluripotent stem cells towards pancreatic β-like cells in a 3D suspension culture [6]. The data contains 51,274 cells collected across eight distinct time points. For modeling purposes, the high-dimensional gene expression space was projected down to 30 dimensions using Principal Component Analysis (PCA).

G Human Embryoid Body Differentiation

This dataset tracks the dynamic time course of human embryoid body differentiation over a 27-day period, with single-cell RNA-sequencing (scRNA-seq) measurements collected at several time windows (days 0-3, 6-9, 12-15, 18-21, and 24-27) [7]. The biological process begins with a single population of embryonic stem cells which then differentiate into approximately four distinct cell precursor types, creating a complex branching structure. For modeling, the high-dimensional gene expression data was first reduced to 100 dimensions using PCA.

Appendix Table 2. Statistic information of benchmark systems.

	Epidemic	Toggle Switch	Signalling Cascade1	Signalling Cascade2	Ecological Evolution	β-cell	Embryoid
Time Range	[0, 50]	[0, 5]	[0, 2]	[0, 3]	[0, 100]	$[0,1,2,3,4,5,6,7]$	$[0,1,2,3,4]$
State Space	52^3	85^4	13^{10}	12^{10}	100^2	30	100
RNN Size	852	1,119	453	444	1,308	4,350	4,350

IV Baselines

A LatentSDE

LatentSDE [8] models stochastic dynamics by assuming the observed data is generated from a latent trajectory governed by a prior Stochastic Differential Equation (SDE). Within a variational inference framework, it uses a second SDE as an approximate posterior to infer this latent path from the data. The model is trained by optimizing the Evidence Lower Bound (ELBO), with scalable gradients computed via the stochastic adjoint method. Future dynamics are predicted by simulating the learned prior SDE forward in time.

B NeuralMJP

NeuralMJP [9] models discrete-state stochastic dynamics by inferring a hidden Markov Jump Process (MJP) from observed data. It employs a variational framework where a neural ODE-based encoder processes the time series to parameterize a posterior MJP. The master equation of this posterior process is then solved to find the latent path. Future dynamics are predicted by simulating a learned prior MJP forward in time.

C T-IB

T-IB [10] models stochastic dynamics by learning a latent representation optimized for a specific, large time lag, τ. Its core objective trains an encoder to preserve the slow dynamics predictive of the future at this time lag, while simultaneously using an information bottleneck to filter out faster, irrelevant processes. Future dynamics are then predicted efficiently via latent simulation, where a transition model simulates the system's evolution directly in this simplified latent space using large time steps.

D NLSB

NLSB [11] (Neural Lagrangian Schrödinger Bridge) models stochastic dynamics by framing the problem of interpolating between population snapshots as a Lagrangian Schrödinger Bridge (LSB) problem. It explicitly models the evolution of individual samples with stochastic behavior and diffusion using a neural SDE. The key to its approach is that the SDE is regularized by an action cost derived from a user-defined

Lagrangian, which enforces the principle of least action and allows for the flexible incorporation of prior knowledge into the dynamics.

E DeepRUOT

DeepRUOT [1] models unbalanced stochastic dynamics by solving the Regularized Unbalanced Optimal Transport (RUOT) problem. It uses a Fisher information regularization to transform the underlying SDE dynamics into a more tractable probability flow ODE. Neural networks are then trained to learn the ODE's drift and a separate growth/death rate. These are integrated forward in time to simultaneously predict sample trajectories and changes in population mass.

We adopt the official open-source implementations for all models and follow the default parameter settings. For the discrete systems, a specific adaptation was made for NLSB and DeepRUOT, as they are designed for continuous-state systems. We used the principal components from a PCA that capture 99% of the variance of the dynamics samples as their modeling target. The final prediction results were then mapped back to the original space for evaluation.

V Metrics

To quantitatively evaluate the performance of all methods, we employ several standard metrics to measure the discrepancy between the predicted probability distribution, $q(x)$, and the true distribution, $p(x)$.

A Wasserstein Distance

The Wasserstein-1 distance [1], also known as the Earth Mover's Distance, intuitively measures the minimum "cost" required to transform one distribution into another, where the cost is the product of the amount of mass moved and the distance it is moved. It is particularly well-suited for comparing distributions that may not have overlapping support. In practice, we use its dual formulation, which is more amenable to computation from samples:

$$W_1(p,q) = \sup_{|f|L\leq 1} \mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{x\sim q}[f(x)] \tag{34}$$

where the supremum is taken over all 1-Lipschitz functions $f : \mathbb{R}^d \to \mathbb{R}$.

B Maximum Mean Discrepancy

The Maximum Mean Discrepancy (MMD) [12] is a metric defined in a Reproducing Kernel Hilbert Space (RKHS). It measures the distance between the mean embeddings of the distributions p and q in this space. A zero MMD value indicates that the two distributions are identical. We use the common squared MMD with a Gaussian kernel, $k(x,x') = exp(-||x-x'||^2/2\sigma^2)$, which can be computed from samples as:

$$\text{MMD}^2(p,q) = \mathbb{E}_{x,x'\sim p}[k(x,x')] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)] + \mathbb{E}_{y,y'\sim q}[k(y,y')]. \tag{35}$$

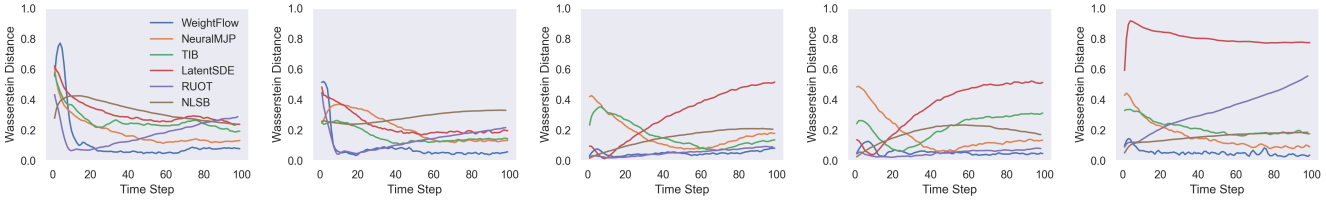
C Jensen-Shannon Divergence

The Jensen-Shannon Divergence (JSD) [5] is a symmetrized and smoothed version of the Kullback-Leibler (KL) divergence, and it measures the similarity between two probability distributions. Unlike the KL divergence, JSD is symmetric and always has a finite value. It is defined based on the KL divergence as:

$$\text{JSD}(p,q) = \frac{1}{2}D_{KL}(p|m) + \frac{1}{2}D_{KL}(q|m) \tag{36}$$

where $m = \frac{1}{2}(p+q)$ is the mixture distribution, and $D_{KL}(p||m) = \int_{\mathbb{R}^d} p(x)\log(\frac{p(x)}{m(x)})dx$.

VI Additional Results

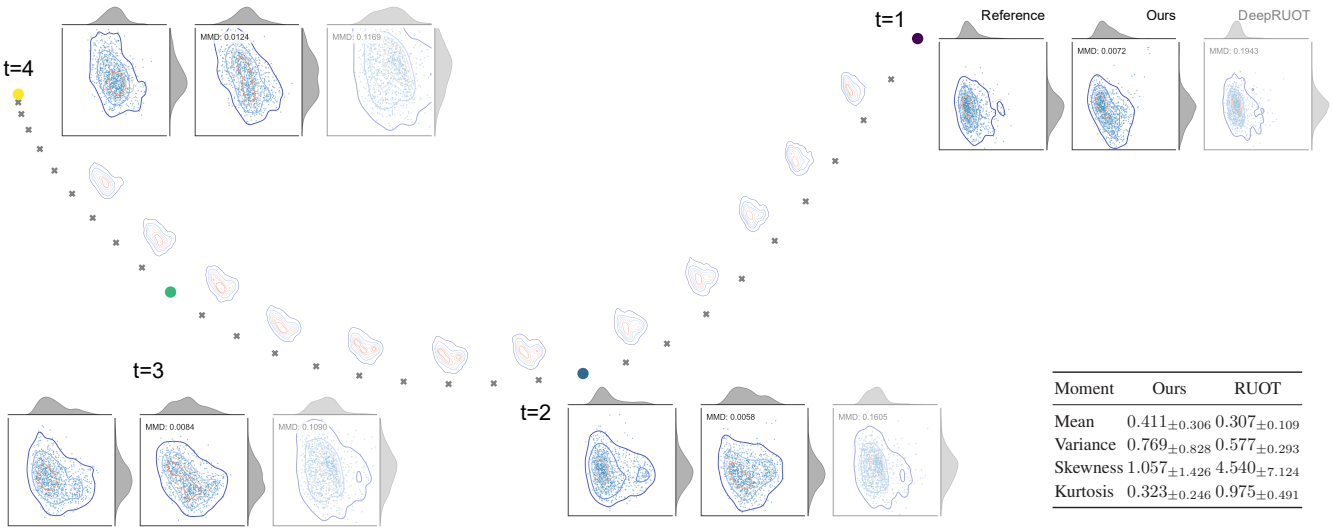


(a) Epidemic **(b)** Toggle Switch **(c)** Signaling Cascade1 **(d)** Signaling Cascade2 **(e)** Ecological Evolution

Appendix Figure 1. Average Wasserstein distance as a function of time step for different methods across five dynamical systems.

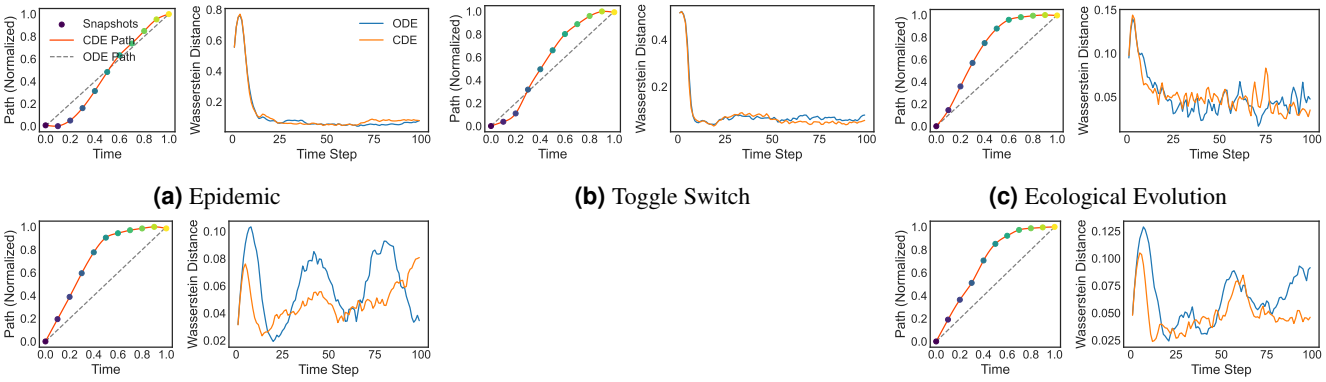
	Epidemic	Switch	Cascade1	Cascade2	Evolution
PCA	0.127	0.093	0.065	0.070	0.061
MDS	0.121	0.105	0.066	0.053	0.060
Isomap	0.132	0.094	0.068	0.054	0.064
AE	0.110	0.082	0.048	0.049	0.051

Appendix Table 3. Average Wasserstein distance for WeightFlow under different path projection methods across five systems.



Moment	Ours	RUOT
Mean	$0.411_{\pm 0.306}$	$0.307_{\pm 0.109}$
Variance	$0.769_{\pm 0.828}$	$0.577_{\pm 0.293}$
Skewness	$1.057_{\pm 1.426}$	$4.540_{\pm 7.124}$
Kurtosis	$0.323_{\pm 0.246}$	$0.975_{\pm 0.491}$

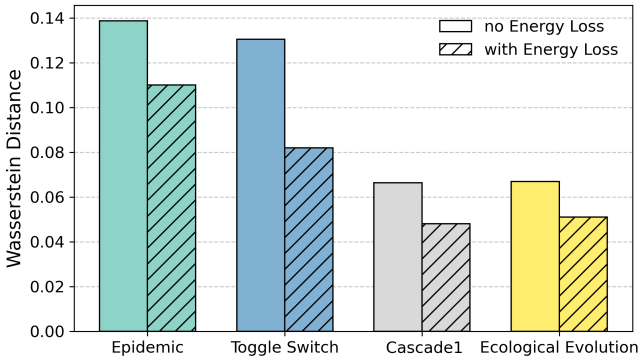
Appendix Figure 2. Weight prediction for embryoid body differentiation. The trajectory shows the continuous evolution of weights and corresponding ensemble distributions (PCA). Highlights compare the results at observed snapshots: reference (left), WeightFlow's prediction (center), and DeepRUOT's prediction (right). The table reports the average relative error of the first four statistical moments.



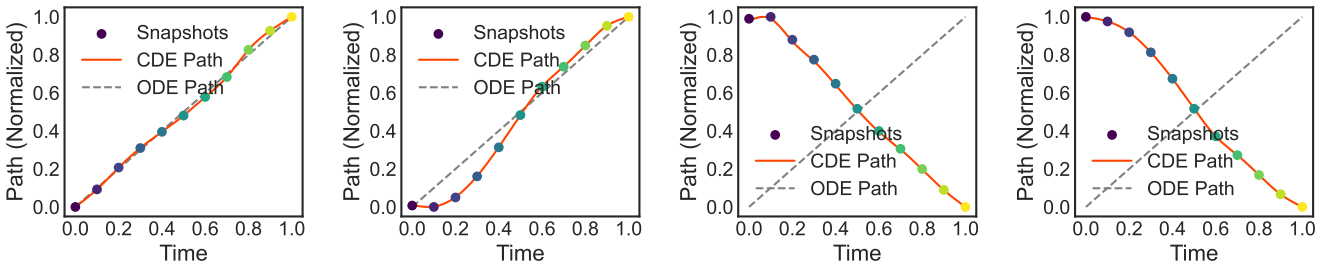
(a) Epidemic **(b)** Toggle Switch **(c)** Ecological Evolution

(d) Signalling Cascade1 **(e)** Signalling Cascade2

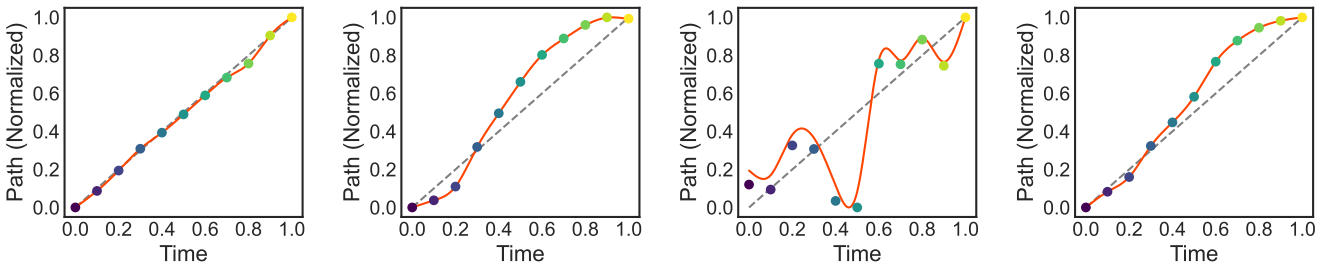
Appendix Figure 3. Comparing path interpolation (left) and average Wasserstein distance (right) for ODE and CDE models across five benchmarking systems.



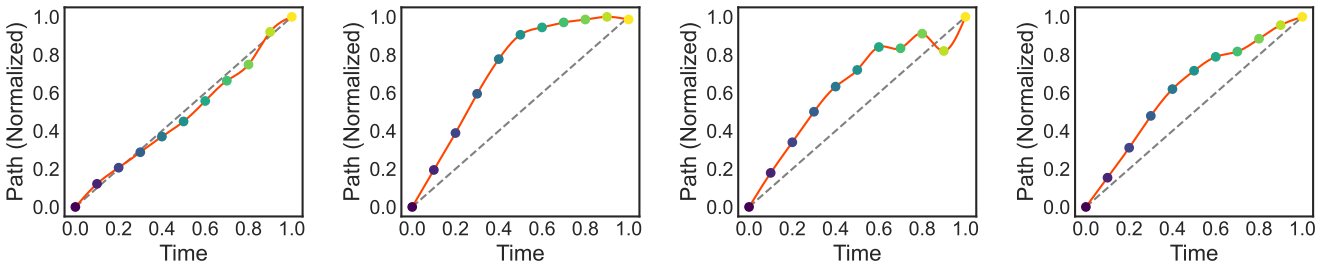
Appendix Figure 4. Impact of the energy loss term. Test Wasserstein distance on four systems, comparing models trained with and without the energy regularization.



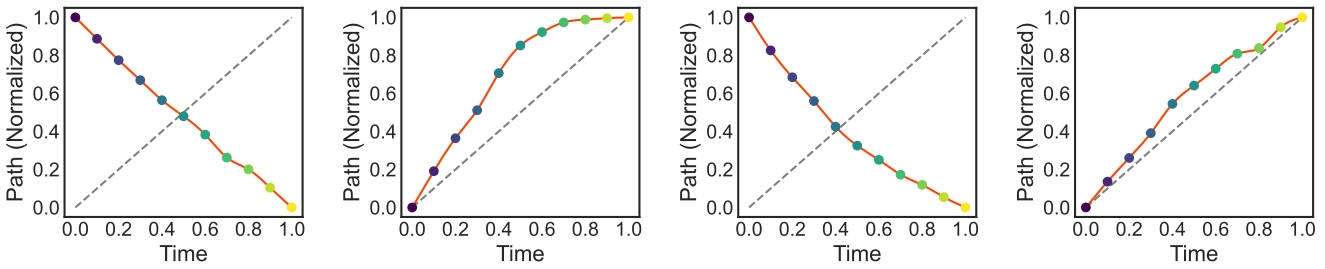
(a) Epidemic (AE) **(b)** Epidemic (PCA) **(c)** Epidemic (MDS) **(d)** Epidemic (Isomap)



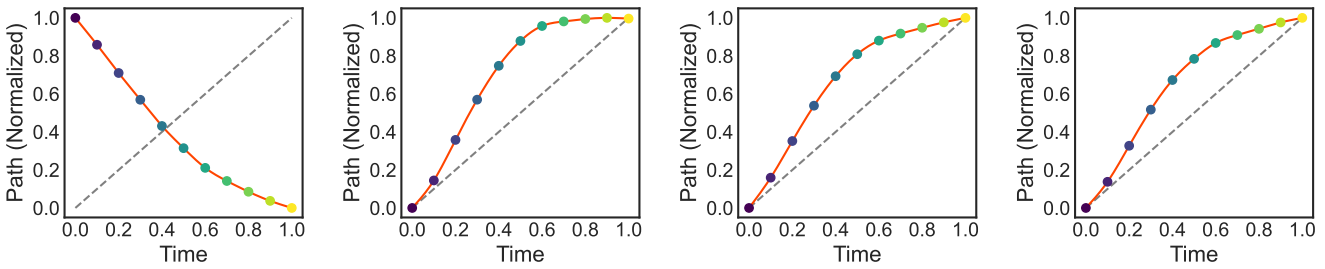
(e) Toggle Switch (AE) **(f)** Toggle Switch (PCA) **(g)** Toggle Switch (MDS) **(h)** Toggle Switch (Isomap)



(i) Signalling Cascade1 (AE) **(j)** Signalling Cascade1 (PCA) **(k)** Signalling Cascade1 (MDS) **(l)** Signalling Cascade1 (Isomap)



(m) Signalling Cascade2 (AE) **(n)** Signalling Cascade2 (PCA) **(o)** Signalling Cascade2 (MDS) **(p)** Signalling Cascade2 (Isomap)



(q) Ecological Evolution (AE) **(r)** Ecological Evolution (PCA) **(s)** Ecological Evolution (MDS) **(t)** Ecological Evolution (Isomap)

Appendix Figure 5. Comparing projected path of different methods for CDE models across five benchmarking systems.

VII Limitations & future work

The principle of least action implied by WeightFlow may fail for open systems. A promising future direction is to embed physical priors, such as conservation laws, as regularizers into the weight evolution process to handle such complex dynamics.

VIII Discussion of Related Work Suggested During Review

During the peer review process, reviewers provided valuable feedback and suggested discussing several related research works. These works provide a broader academic context for our WeightFlow framework. We discuss these references in this appendix to further clarify the positioning and contribution of our work.

A Inferring SDE Dynamics from Trajectories

Reviewers highlighted a body of literature, particularly from the physics community, focused on inferring stochastic differential equations or Langevin dynamics from observed trajectory data. For example, Yildiz et al. proposed using Gaussian Processes to infer SDEs without gradient matching. Frishman and Ronceray utilized linear regression on basis functions to learn force fields from stochastic trajectories. More recent work has begun to integrate neural networks more closely with physical models. For instance, Gao, Barzel, and Yan employs graph neural networks to infer Langevin dynamics from experimental data. Similarly, Bae, Ha, and Jeong uses Bayesian neural networks to infer SDEs while also providing uncertainty quantification.

Relationship to WeightFlow. The core objective of these methods is to **parameterize and estimate the drift and diffusion terms** of a stochastic differential equation. They attempt to discover the underlying functional form of the SDE governing the system's evolution from data.

In contrast, our WeightFlow framework does not directly estimate these SDE parameters. Our goal is to **directly model the continuous evolution of the probability density function itself**. We learn the continuous evolution of weights over time through a projection of the probability distribution, which is described by the Fokker-Planck equation, into the neural network's weight space. Therefore, WeightFlow aims to capture the holistic morphological evolution of the distribution rather than fitting a specific underlying SDE form.

B Interpretability in Dynamics Modeling

Reviewers suggested focusing on the interpretability of dynamics modeling. Yu et al. introduced OnsagerNet, a model that learns stable and physically interpretable dynamical models based on the generalized Onsager principle. It can explicitly decompose complex dynamics into physically meaningful components such as free energy, dissipation, and conservative motion.

Relationship to WeightFlow. OnsagerNet provides a powerful approach for constructing structured and interpretable dynamical models. Our WeightFlow also offers theoretical interpretability. As we demonstrate in Section III of the main paper, we connect the evolution in weight space to an energy

functional in the Dynamic Optimal Transport problem.

The primary difference lies in the pathway to achieving interpretability. OnsagerNet achieves this by **imposing a specific physical structure, namely the Onsager principle**, on the model. In contrast, WeightFlow's interpretability stems from the theoretical derivation that **equivalently maps the evolution in probability space to weight space**. Exploring how to integrate physical priors from OnsagerNet, such as symmetry or dissipative structures, into our weight space evolution model is a valuable future research direction.

C Theory of Neural Differential Equations

Finally, reviewers suggested we consider recent advances in the theory of neural differential equations. This is relevant because our Hypernetwork adopts graph neural differential equations, specifically Graph CDEs or ODEs, to model the continuous evolution of weights. Specifically, Marion et al. explored how deep residual networks, or ResNets, approach Neural ODEs via implicit regularization. Gao et al. delved into the influence of factors such as activation functions on the global convergence of Neural ODEs.

Relationship to WeightFlow. These theoretical works on the stability and convergence of Neural ODEs are crucial for understanding and improving the training dynamics of WeightFlow. The performance of the WeightFlow framework depends not only on the theoretical foundations of optimal transport but also **directly on the numerical stability of its core component, the graph neural differential equation solver**.

Although our research focuses on the dynamical modeling framework itself, the solver's stability will directly impact the model's training efficiency and final performance. This is especially true when handling long time series or complex weight graphs. These theoretical insights provide important support for future optimizations of our model, for instance by improving activation functions, network architectures, or regularization strategies.

D Concurrent Work in Weight-Space Learning

We also note the concurrent work, WARP [20], which explores weight-space learning for sequence modeling. Their approach differs from ours: WARP parameterizes an RNN's hidden state as the weights of an auxiliary network for adaptation. In contrast, WeightFlow models the continuous evolution of the weights themselves as a projection of the system's probability distribution. Both works, however, validate the potential of weight-space as a domain for modeling complex dynamics.

References

[1] Zhenyi Zhang, Tiejun Li, and Peijie Zhou. "Learning stochastic dynamics from snapshots through regularized unbalanced optimal transport". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[2] Lucas Böttcher, Nino Antulov-Fantulin, and Thomas Asikis. "AI Pontryagin or how artificial neural networks learn to control dynamical systems". In: *Nature communications* 13.1 (2022), p. 333.

[3] Cédric Villani et al. *Optimal transport: old and new*. Vol. 338. Springer, 2008.

[4] Ying Tang, Jiayu Weng, and Pan Zhang. "Neural-network solutions to stochastic reaction networks". In: *Nature Machine Intelligence* 5.4 (2023), pp. 376–385.

[5] Ruikun Li, Huandong Wang, Qingmin Liao, and Yong Li. "Predicting the Energy Landscape of Stochastic Dynamical System via Physics-informed Self-supervised Learning". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[6] Adrian Veres, Aubrey L Faust, Henry L Bushnell, Elise N Engquist, Jennifer Hyoje-Ryu Kenty, George Harb, Yeh-Chuin Poh, Elad Sintov, Mads Gürtler, Felicia W Pagliuca, et al. "Charting cellular identity during human in vitro β-cell differentiation". In: *Nature* 569.7756 (2019), pp. 368–373.

[7] Alexander Tong, Jessie Huang, Guy Wolf, David Van Dijk, and Smita Krishnaswamy. "Trajectorynet: A dynamic optimal transport network for modeling cellular dynamics". In: *International conference on machine learning*. PMLR. 2020, pp. 9526–9536.

[8] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. "Scalable gradients for stochastic differential equations". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3870–3882.

[9] Patrick Seifner and Ramsés J Sánchez. "Neural markov jump processes". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 30523–30552.

[10] Marco Federici, Patrick Forré, Ryota Tomioka, and Bastiaan S Veeling. "Latent Representation and Simulation of Markov Processes via Time-Lagged Information Bottleneck". In: *The Twelfth International Conference on Learning Representations*. 2024.

[11] Takeshi Koshizuka and Issei Sato. "Neural Lagrangian Schrödinger Bridge: Diffusion Modeling for Population Dynamics". In: *The Eleventh International Conference on Learning Representations.* 2023.

[12] Kacper Kapusniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Joey Bose, and Francesco Di Giovanni. "Metric flow matching for smooth interpolations on the data manifold". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 135011–135042.

[13] Cagatay Yildiz, Markus Heinonen, Jukka Intosalmi, Henrik Mannerstrom, and Harri Lahdesmaki. “Learning stochastic differential equations with gaussian processes without gradient matching”. In: *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2018, pp. 1–6.

[14] Anna Frishman and Pierre Ronceray. “Learning force fields from stochastic trajectories”. In: *Physical Review X* 10.2 (2020), p. 021009.

[15] Ting-Ting Gao, Baruch Barzel, and Gang Yan. “Learning interpretable dynamics of stochastic complex systems from experimental data”. In: *Nature communications* 15.1 (2024), p. 6029.

[16] Youngkyoung Bae, Seungwoong Ha, and Hawoong Jeong. “Inferring the Langevin equation with uncertainty via Bayesian neural networks”. In: *Chaos, Solitons & Fractals* 197 (2025), p. 116440.

[17] Haijun Yu, Xinyuan Tian, Weinan E, and Qianxiao Li. “OnsagerNet: Learning stable and interpretable dynamics using a generalized Onsager principle”. In: *Physical Review Fluids* 6.11 (2021), p. 114402.

[18] Pierre Marion, Yu-Han Wu, Michael E Sander, and Gérard Biau. “Implicit regularization of deep residual networks towards neural ODEs”. In: *arXiv preprint arXiv:2309.01213* (2023).

[19] Tianxiang Gao, Siyuan Sun, Hailiang Liu, and Hongyang Gao. “Global convergence in neural odes: Impact of activation functions”. In: *The Thirteenth International Conference on Learning Representations*. 2025.

[20] Roussel Desmond Nzoyem, Nawid Keshtmand, Enrique Crespo Fernandez, Idriss Tsayem, Raul Santos-Rodriguez, David AW Barton, and Tom Deakin. “Weight-Space Linear Recurrent Neural Networks”. In: *arXiv preprint arXiv:2506.01153* (2025).