NEURAL LAGRANGIAN SCHRÖDINGER BRIDGE: DIFFUSION MODELING FOR POPULATION DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Population dynamics is the study of temporal and spatial variation in the size of populations of organisms and is a major part of population ecology. One of the main difficulties in analyzing population dynamics is that we can only obtain observation data with coarse time intervals from fixed-point observations due to experimental costs or measurement constraints. Recently, modeling population dynamics by using continuous normalizing flows (CNFs) and dynamic optimal transport has been proposed to infer the sample trajectories from a fixed-point observed population. While the sample behavior in CNFs is deterministic, the actual sample in biological systems moves in an essentially random yet directional manner. Moreover, when a sample moves from point A to point B in dynamical systems, its trajectory typically follows *the principle of least action* in which the corresponding action has the smallest possible value. To satisfy these requirements of the sample trajectories, we formulate the Lagrangian Schrödinger bridge (LSB) problem and propose to solve it approximately by modeling the advection-diffusion process with regularized neural SDE. We also develop a model architecture that enables faster computation of the loss function. Experimental results show that the proposed method can efficiently approximate the population-level dynamics even for high-dimensional data and that using the prior knowledge introduced by the Lagrangian enables us to estimate the sample-level dynamics with stochastic behavior.

1 INTRODUCTION

The population dynamics of time-evolving individuals appears in various scientific fields, such as cell population in biology (Schiebinger et al., 2019; Yang & Uhler, 2018), air in meteorology (Fisher et al., 2009), and healthcare statistics (Manton et al., 2008) in medicine. However, tracking individuals over a long period is often difficult due to experimental costs. Furthermore, it can sometimes be impossible to track the time evolution. For example, since single-cell RNA sequencing (scRNA-seq) destroys all measured cells, we cannot analyze the behavior of individual cells over time in cell transcriptome measurements. Instead, we only obtain individual samples from cross-sectional populations without alignment across time steps at a few distinct time points. Under these constraints on data measurements, our goal is to better understand the time evolution of samples in the populations.

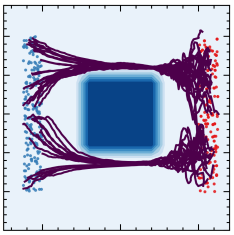


Figure 1: Example of trajectories by NLSB.

Existing methods attempt to estimate population-level dynamics following the Wasserstein gradient flow using a recurrent neural network (RNN) (Hashimoto et al., 2016) or the Jordan-Kinderlehrer-Otto (JKO) flow (Bunne et al., 2021). Recent studies have attempted to interpolate the trajectories of individual samples between cross-sectional populations at multiple time points by using optimal transport (OT) (Schiebinger et al., 2019; Yang & Uhler, 2018), or CNF (Tong et al., 2020). Using a CNF generates continuous-time non-linear sample trajectories from multiple time points. In addition, Tong et al. (2020) proposed a regularization for CNF that encourages a straight trajectory on the basis of the OT theory. Since the probability distribution transformation based on ordinary differential equations (ODEs) is used in CNF, the behavior of each sample is described by its initial condition in a completely deterministic manner. However, samples in population are known to move stochastically and diffuse in nature, *e.g.*, biological system (Kolomgorov et al., 1937).

To handle the stochastic and complex behavior of individual samples, we propose to model the advection-diffusion processes by using SDEs to describe the time evolution of the sample. Furthermore, on the basis of *the principle of least action*, we estimate the sample trajectories that minimize *action*, defined by the time integral of the Lagrangian determined from the prior knowledge. We formulate this problem as the Lagrangian Schrödinger bridge (LSB) problem, which is a special case of the stochastic optimal transport (SOT) problem, and propose an approximate solution method neural Lagrangian Schrödinger bridge (NLSB). In NLSB, we train regularized neural SDE (Li et al., 2020; Tzen & Raginsky, 2019a;b) by minimizing the Wasserstein loss between the ground-truth and the predicted population. The Lagrangian design defining regularization allows the sample-level dynamics to reflect various prior knowledge such as OT, manifold geometry, and local velocity arrows proposed by Tong et al. (2020). In addition, we parameterize a potential function instead of the drift function. Adopting the model architecture of the potential function from OT-Flow (Onken et al., 2021) will speed up the computation of the potential function's gradient and the regularization term. As a result, we capture the population-level dynamics as well as or better than conventional methods, and can more accurately predict the sample trajectories.

In short, our contributions are summarized as follows.

1. We formulate the LSB problem to estimate the stochastic sample trajectory according to *the principle of least action*.
2. We propose NLSB to approximate the LSB problem practically by modeling the advection-diffusion process with regularized neural SDE on the basis of the prior knowledge introduced by the Lagrangian.
3. Our model architecture of the potential function enables its gradient and diagonal components of the Hessian to be computed explicitly, resulting in fast computation.

2 BACKGROUND

In Section 2.1, we first introduce the method of combining CNF and the OT theory, the basis of the proposed method. Next, we explain our newly introduced theories and techniques for modeling population dynamics: the diffusion modeling using neural SDE (Section 2.2) and the SOT theory (Section 2.3).

2.1 FLOWS REGULARIZED BY OPTIMAL TRANSPORT

CNFs (Chen et al., 2018) are a method for learning the continuous transformation between two distributions p and q by modeling the ordinary differential equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} = \mathbf{v}_\theta(\mathbf{x}, t), \qquad \text{subject to} \quad \mathbf{x}(t_0) \sim p,\ \mathbf{x}(t_1) \sim q,$$

where $\mathbf{v}_\theta$ is the velocity model with learnable parameters θ.

Several regularizations of CNFs leading to straight trajectories have been proposed on the basis of the OT theory. The likelihood maximization problem of regularized CNF is derived by replacing the terminal constraint of the Brenier-Benamou formulation (Benamou & Brenier, 2000) with Kullback–Leibler (KL) divergence. RNODE (Finlay et al., 2020), OT-Flow (Onken et al., 2021), and TrajectoryNet (Tong et al., 2020) introduced a regularization $\tilde{\mathcal{R}}_e$ in Eq. (1). Potential Flow (Yang & Karniadakis, 2020) and OT-Flow modeled the potential function Φ that satisfies $\mathbf{v} = -\nabla\Phi$ instead of modeling the velocity function $\mathbf{v}$. They also proposed an additional OT-based regularization $\tilde{\mathcal{R}}_h$ derived from the Hamilton–Jacobi–Bellman (HJB) equation (Evans, 1983) satisfied by the potential function shown in Eq. (2). These OT-based regularizations have resulted in faster CNFs (Finlay et al., 2020; Onken et al., 2021) and improved the modeling of cellular dynamics (Tong et al., 2020).

$$\tilde{\mathcal{R}}_e = \int_{t_0}^{t_1} \int_\Omega \frac{1}{2} ||\mathbf{v}_\theta(\mathbf{x}, t)||^2 \, \mathrm{d}\rho_t(\mathbf{x}) \, \mathrm{d}t, \tag{1}$$

$$\tilde{\mathcal{R}}_h = \int_{t_0}^{t_1} \int_\Omega \left| \partial_t \Phi_\theta(\mathbf{x}, t) - \frac{1}{2} ||\nabla_{\mathbf{x}} \Phi_\theta(\mathbf{x}, t)||^2 \right| \, \mathrm{d}\rho_t(\mathbf{x}) \, \mathrm{d}t, \tag{2}$$

where Ω is the bounded domain and ρ_t is the law of the sample $\mathbf{x}$ at the time t.

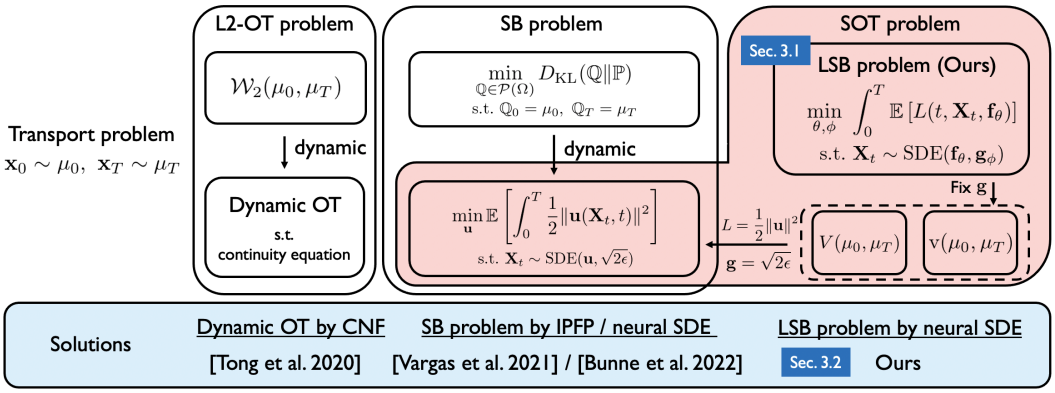


Figure 2: Overview

2.2 Neural SDE and Applications to Diffusion Modeling

SDE has been used to model real-world random phenomena in a wide range of areas, such as chemistry, biology, mechanics, economics and finance (Higham, 2001). As an extension of neural ODEs, neural SDE has been proposed to model drift and diffusion functions with neural networks (NNs), as follows.

$$\mathrm{d}\mathbf{X}_t = \mathbf{f}_\theta(\mathbf{X}_t, t)\,\mathrm{d}t + \mathbf{g}_\phi(\mathbf{X}_t, t)\,\mathrm{d}\mathbf{W}_t, \tag{3}$$

where $\{\mathbf{X}_t\}_{t\in[t_0,t_{K-1}]}$ is a continuous $\mathbb{R}^d$-valued stochastic process, $\mathbf{f}_\theta : \mathbb{R}^d \times [t_0, t_{K-1}] \mapsto \mathbb{R}^d$ is a drift function, $\mathbf{g}_\phi : \mathbb{R}^d \times [t_0, t_{K-1}] \mapsto \mathbb{R}^{d\times m}$ is a diffusion function and $\{\mathbf{W}_t\}_{t\in[t_0,t_{K-1}]}$ is an m-dimensional Wiener process. Score-based generative models (SGM) (Song et al., 2020), which use score matching to learn the reverse diffusion process of generating images from noise as SDE, have demonstrated the ability to produce high-quality data. Other recent studies on diffusion modeling using SDE (De Bortoli et al., 2021; Wang et al., 2021; Vargas et al., 2021; Chen et al., 2021a) have proposed learning SDE solutions to the SB problem. Vargas et al. (2021) and De Bortoli et al. (2021) solved the SB problem by combining iterative proportional fitting (IPF) with mean-matching regression of the SDE drift function using Gaussian process (GP) and NN, respectively.

2.3 Stochastic Optimal Transport

Mikami (2008) generalized the OT problem and defined the SOT problem as a random mechanics problem determined by *the principle of least action*. The SOT problem with the endpoint marginals fixed to μ_0 and μ_1 is represented as

$$V(\mu_0, \mu_1) := \inf_{\mathbf{X}} \left\{ \mathbb{E}\left[\int_{t_0}^{t_1} L\left(t, \mathbf{X}_t; \boldsymbol{\beta}_{\mathbf{X}}(\mathbf{X}_t, t)\right)\,\mathrm{d}t\right] \middle| \mathbf{X}_t \in \mathcal{A},\ \mathbf{X}_{t_0} \sim \mu_0,\ \mathbf{X}_{t_1} \sim \mu_1 \right\}, \tag{4}$$

where $L(t, \mathbf{x}, \mathbf{u})$ is continuous and convex in $\mathbf{u}$ and $\mathcal{A}$ is the set of all $\mathbb{R}^d$-valued, continuous semimartingales $\{\mathbf{X}_t\}_{t_0\le t\le t_1}$ on a complete filtered probability space such that there exists a Borel measurable drift function $\boldsymbol{\beta}_{\mathbf{X}}(\mathbf{X}_t, t)$ for which satisfies several conditions (see (Mikami, 2021)). The function L, called the Lagrangian, is the transport cost defined on the space-time of the system and allows us to describe phenomena consistently regardless of the choice of coordinate system.

Mikami (2008) also introduced another variational version of the SOT problem for a flow of marginal distributions which satisfies the Fokker–Planck (FP) equation, which is a advection-diffusion equation describing the time evolution of the probability density function.

$$\mathrm{v}(\mu_0, \mu_1) := \inf_{\mathbf{f},\rho_t} \left\{ \int_{t_0}^{t_1} \int_{\mathbb{R}^d} L\left(t, \mathbf{x}; \mathbf{f}(\mathbf{x}, t)\right)\,\mathrm{d}\rho_t(\mathbf{x})\,\mathrm{d}t \middle| \rho_{t_k} = \mu_k,\ (\mathbf{f}, p_t) \text{ satisfies the FP eq} \right\}, \tag{5}$$

where ρ_t and p_t are the law and density of the random variable $\mathbf{X}_t$, respectively.

Proposition 1 from (Mikami, 2021) states the optimal values of Eqs. (4) and (5) match, *i.e.* $V(\mu_0, \mu_1) = \mathrm{v}(\mu_0, \mu_1)$. When $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2$, the problem 4 and 5 are regarded as the special case of Schrödinger bridge (SB) (Jamison, 1975) problem.

3 Proposed Method

In this section, we first introduce Lagrangian-Schrödinger bridge (LSB) problem. We next present an approximate solution for the LSB problem: neural Lagrangian Schrödinger bridge (NLSB). Finally, we give specific examples of our method for specific use cases. Figure 2 illustrates the position of the LSB problem in the transport problem and the relationship between the NLSB and existing methods.

3.1 Lagrangian Schrödinger Bridge

We consider the problem of estimating dynamics at both the population and individual sample levels by using probability densities at two known end points. To solve this problem, we make two assumptions about the target system model.

1. The stochastic behavior of individual samples yields the population diffusion phenomenon.
2. Individual samples are encouraged to move according to *the principle of least action*.

The principle of least action states that when a sample moves from point A to point B, its trajectory is the one that have least action, which is known as a fundamental principle in dynamical systems.

We propose to explicitly model diffusion phenomena as well as drift by using SDEs for the time evolution of the sample and to explore the sample paths that minimize *action* defined by the time integral of the Lagrangian. Later in Section 3.3, we will give some examples of prior information described by the Lagrangian. We formulate this problem as a special case of the SOT problem and call it *the Lagrangian-Schrödinger Bridge (LSB) problem*.

Definition 1 (LSB problem)**.**

$$\begin{aligned} \underset{\theta,\phi}{\text{minimize}} \quad & \int_{t_0}^{t_1} \int_{\Omega} L(t, \mathbf{x}, \mathbf{f}_\theta(\mathbf{x}, t))\, \mathrm{d}\rho_t(\mathbf{x}; \theta, \phi)\, \mathrm{d}t, \\ \text{subject to} \quad & \mathrm{d}\mathbf{X}_t = \mathbf{f}_\theta(\mathbf{X}_t, t)\, \mathrm{d}t + \mathbf{g}_\phi(\mathbf{X}_t, t)\, \mathrm{d}\mathbf{W}_t, \\ & \mathbf{X}_0 \sim \rho_{t_0} = \mu_0,\ \mathbf{X}_1 \sim \rho_{t_1} = \mu_1, \end{aligned} \tag{6}$$

where ρ_t is the law of the random variable $\mathbf{X}_t$, depending on the functions $\mathbf{f}_\theta$ and $\mathbf{g}_\phi$.

The connection between the LSB problem and the SB problem formulated as stochastic optimal control (SOC) problem (Chen et al., 2021b; Dai Pra, 1991; Léonard, 2013; Mikami, 1990) is discussed in Appendix C.

3.2 Neural Lagrangian Schrödinger Bridge

In the setting of our paper, it is difficult to solve the LSB problem because the exact endpoint constraints μ_0 and μ_1 are unknown and only samples from them are available. Therefore, we propose a practical solution method for the LSB problem using neural SDEs (Eq. (3)) with regularization. We propose to approximate the LSB problem by learning neural SDEs with the gradients of the loss (See Appendix A for details.):

$$\underset{\theta,\phi}{\text{minimize}} \quad \mathcal{W}_2(\mu_1,\ \rho_{t_1}) + \mathcal{R}_e(\theta; t_0, t_1) + \mathcal{R}_h(\theta, \phi; t_0, t_1), \tag{7}$$

$$\mathcal{R}_e(\theta; t_0, t_1) = \int_{t_0}^{t_1} \int_{\Omega} L(t, \mathbf{x}, \mathbf{f}_\theta(\mathbf{x}, t))\, \mathrm{d}\rho_t(\mathbf{x})\, \mathrm{d}t, \tag{8}$$

$$\begin{aligned} \mathcal{R}_h(\theta, \phi; t_0, t_1) = \int_{t_0}^{t_1} \int_{\Omega} \left| \partial_t \Phi_\theta(\mathbf{x}, t) + \sum_{i,j=1}^{d} D_{i,j}(\mathbf{x}, t; \phi) \left[\nabla^2_{\mathbf{x}} \Phi_\theta(\mathbf{x}, t)\right]_{i,j} + h(\mathbf{x}, t) \right| \mathrm{d}\rho_t(\mathbf{x})\, \mathrm{d}t, \\ h(\mathbf{x}, t) = L(t, \mathbf{x}, \mathbf{f}_\theta(\mathbf{x}, t)) + \nabla_{\mathbf{x}} \Phi_\theta(\mathbf{x}, t)^\top \mathbf{f}_\theta(\mathbf{x}, t), \end{aligned} \tag{9}$$

where $D_{i,j}(\cdot;\phi)$ is the entry in the i-th row and j-th column of the diffusion coefficient matrix $\mathbf{D}_\phi = \frac{1}{2}\mathbf{g}_\phi\mathbf{g}_\phi^\top$ and Φ_θ is the potential function satisfying $\mathbf{f}_\theta = [\nabla_\mathbf{u} L]^{-1}(-\nabla_\mathbf{x}\Phi_\theta)$.

We briefly explain the action cost $\mathcal{R}_e$ and the HJB regularization $\mathcal{R}_h$. Equation 7 is actually computed by Eq. (11). First, similar to the OT-based regularization (Finlay et al., 2020; Onken et al., 2021; Tong et al., 2020), we relax the constraint at time t_1 with the Wasserstein-2 cost, *i.e.* $\mathcal{R}_e + \mathcal{W}_2(\mu_1, \rho_{t_1})$. The use of Wasserstein-2 cost instead of KL-divergence is based on existing studies on population dynamics (Bunne et al., 2021; Hashimoto et al., 2016). This objective function can be reinterpreted with $\mathcal{W}_2$ as the main cost and the action cost $\mathcal{R}_e$ as the regularization. Second, we exploit SOT theory by incorporating further structure into the modeling. Similar to OT-Flow (Onken et al., 2021), we model the drift functions by using the potential function $\mathbf{f}_\theta = [\nabla_\mathbf{u} L]^{-1}(-\nabla_\mathbf{x}\Phi_\theta)$ and encourage the potential function Φ_θ to satisfy the (stochastic) HJB equation (Yong & Zhou, 1999) by minimizing the PDE loss $\mathcal{R}_h$. The relation between the drift and the potential function is an analogue of Hamilton's equations of motion. The HJB equation represents Bellman's principle of optimality in continuous-time optimization.

3.3 Examples of Lagrangian in Neural Lagrangian Schrödinger Bridge

In this section, we provide the three Lagrangian examples of the NLSB and their use cases. The examples demonstrate that the Lagrangian design allows a variety of prior knowledge to be reflected in the sample trajectories.

Potential-free system. Without a specific external force on the individual sample, *the principle of least action* typically indicates that when a sample moves from point A to point B, it tries to minimize energy by reducing the travel distance. This means that the drift is encouraged to be straight and the stochastic movement to be small, *i.e.*, the Lagrangian is formulated by $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2$, and the drift function is given as $\mathbf{f}_\theta = -\nabla_\mathbf{x}\Phi_\theta(\mathbf{x}, t)$.

Cellular system. Tong et al. (2020) proposed to introduce the prior knowledge of manifold geometry and local velocity arrows such as RNA-velocity for modeling cellular systems. In the NLSB, these prior knowledge can be handled consistently by designing the Lagrangian. First, we introduce a density-based penalty to constrain the sample trajectories on the data manifold. We estimate the density function $U(\mathbf{x}, t)$ from the data, *e.g.*, by using Gaussian mixture models (GMM) and add it to the Lagrangian. Next, we redefine the velocity regularization, which is formulated as cosine similarity maximization by Tong et al. (2020), as a squared error minimization and add it to the Lagrangian as well. Therefore, the Lagrangian for the cellular system is defined by

$$L(t, \mathbf{x}, \mathbf{u}) = \underbrace{\frac{1}{2}||\mathbf{u}(\mathbf{x}, t)||^2}_{\text{Energy}} - \underbrace{U(\mathbf{x}, t)}_{\text{Density}} + \underbrace{\frac{1}{2}||\mathbf{u}(\mathbf{x}, t) - \mathbf{v}(\mathbf{x}, t)||^2}_{\text{Velocity}}, \tag{10}$$

where $\mathbf{v}(\mathbf{x}, t)$ is the reference velocity of the cell at the position $\mathbf{x}$ and the time t. The drift function is obtained by $\mathbf{f}_\theta = \frac{1}{2}(\mathbf{v}(\mathbf{x}, t) - \nabla_\mathbf{x}\Phi_\theta(\mathbf{x}, t))$.

Random dynamical system. Let m be the mass of the sample, $U(\mathbf{x})$ be the potential energy of the system, and $L(t, \mathbf{x}, \mathbf{u}) = \frac{m}{2}\mathbf{u}(\mathbf{x}, t)^\top\mathbf{u}(\mathbf{x}, t) - U(\mathbf{x})$ be the Lagrangian in the Newtonian dynamical system. Then, the individual samples are encouraged to follow a stochastic analogue of *the equations of motion* of the Newtonian mechanics. In practice, the potential function can be used to roughly incorporate information such as obstacles and regions where samples cannot or are likely to exist. Practical examples are shown in Figure 1 and Appendix E.

4 Implementation Details of Neural Lagrangian Schrödinger Bridge

4.1 Training for Neural Lagrangian Schrödinger Bridge

In this section, we describe a practical learning method for neural SDEs by minimizing loss 7 on the training data, as described in Algorithm 1. First, for the data, only individual samples from a cross-sectional population with no alignment across time steps at K separate time points are available. Let $T = \{t_0, \ldots, t_{K-1}\}$ be a set of time points and denote the data set at time t_i as $\mathcal{X}_{t_i}$. Next, we

describe how we calculate the loss 7 using the training data $\{\mathcal{X}_{t_i}\}_{t_i \in T}$. The first term, the Wasserstein cost, is computed using the L2-Sinkhorn divergence $\overline{\mathcal{W}}_\epsilon$ between the observed data and the predicted sample at all observation points in T except the initial point at time t_0. The Sinkhorn divergence can efficiently approximate the Wasserstein distance and solve the entropic bias problem when using the Sinkhorn algorithm, *i.e.* $\mu = \nu \Leftrightarrow \overline{\mathcal{W}}_\epsilon(\mu, \nu) = 0$. The prediction sample is obtained by numerically simulating neural SDE from the training data at one previous time point by a standard SDE solver such as the Euler-Maruyama method. The second and third terms, $\mathcal{R}_e$ and $\mathcal{R}_h$, are also approximated on the sample paths obtained from the numerical simulations of SDE. Therefore, the computational cost of empirical $\mathcal{R}_e$ and $\mathcal{R}_h$ is not high and can be further accelerated by the model architecture described in the next section. To summarize, we simultaneously solve $K-1$ approximated LSB problems by minimizing the subsequent loss for the model parameters $\theta,\ \phi$.

$$\ell(\theta, \phi) = \sum_{t_k \in T \setminus t_0} \overline{\mathcal{W}}_\epsilon(\mu_k,\ \rho_{t_k}^{\theta,\phi}) + \lambda_e(t_{k-1}, t_k)\hat{\mathcal{R}}_e^\theta(t_{k-1}, t_k) + \lambda_h(t_{k-1}, t_k)\hat{\mathcal{R}}_h^{\theta,\phi}(t_{k-1}, t_k), \quad (11)$$

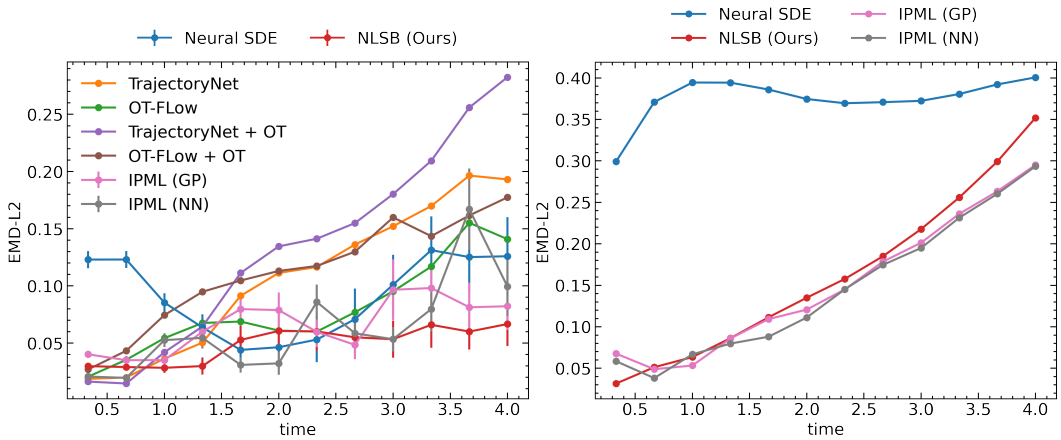
where $\mu_k,\ \rho_{t_k}^{\theta,\phi}$ are the ground-truth and predicted probability measures at time t_k expressed by data samples, respectively. $\hat{\mathcal{R}}_e^\theta(t_{k-1}, t_k)$ and $\hat{\mathcal{R}}_h^{\theta,\phi}(t_{k-1}, t_k)$ are empirical quantities computed on the simulated sample paths from t_{k-1} to t_k with data $\forall \mathbf{x}(t_{k-1}) \in \mathcal{X}_{t_{k-1}}$ as the initial value. The weight coefficients $\lambda_e(t_{k-1}, t_k)$ and $\lambda_h(t_{k-1}, t_k)$ are tuned for each interval $[t_{k-1}, t_k]$ respectively.

4.2 Model Architecture Selection for Speedup

We adopt the model of the potential function Φ_θ proposed by OT-Flow (Onken et al., 2021) because it has two advantages in our framework. First, it can compute the gradient $\nabla_{\mathbf{x}}\Phi$ explicitly, which enables us to calculate the drift function easily. Second, the model is designed for the fast and exact computation of the diagonal component of the potential function's Hessian. Moreover, when we assume that the diffusion model's output is a diagonal matrix, *i.e.* $\mathbf{g}_\phi(\mathbf{x}, t) \in \mathbb{R}^{d\times d}$ and $\mathbf{g}_{i,j} = 0\ (i \neq j)$, the function $\mathbf{D}_\phi$ is also a diagonal matrix, and the $\sum_{i,j}$ in the second term of $\mathcal{R}_h$ shown in Eq. (9) turns into the sum of the diagonal components only. Combining these two tricks enables speeding up the computation of $\mathcal{R}_h$, which requires the expensive computation of the Hessian of the potential function Φ. While this model architecture trick was originally used for speeding up the computation of the Jacobian term in neural ODE maximum likelihood training, we propose to use it as a technique to speed up the computation of R_h. We use a two-layer fully connected neural network (FCNN) for the diffusion function. We also adopt the device used by Kidger et al. (2021) in which the activation function $\tanh$ is used after the final layer to prevent the output of the diffusion function from becoming excessively large. For details of the model architecture, see Appendix B.

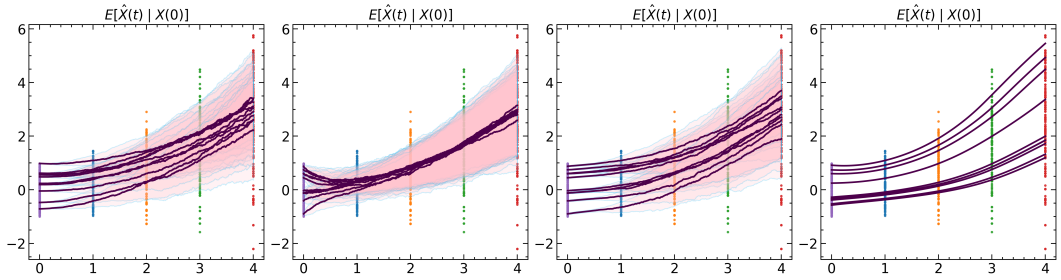
5 Experiments

We evaluated our methods on two datasets. First, we used artificial synthetic data generated from one-dimensional SDEs, where the predicted trajectory and uncertainty can be compared with the ground-truth and easily evaluated by visualization. We set the Lagrangian for the potential-free system, *i.e.* $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2$. Second, we used the evolution of single-cell populations obtained from a developing human embryo system. We used the Lagrangian for the cellular system and compared several combinations of the regularization terms. In Tables 1 and 2, "E" is the energy term, "D" is the density term, and "V" is the velocity term. The density term $U(\mathbf{x}, t)$ is the log-likelihood function of the data estimated by GMM. We compared our methods against standard neural SDE, OT-Flow (Onken et al., 2021), TrajectoryNet (Tong et al., 2020) and an iterative proportional maximum likelihood (IPML) algorithm (Vargas et al., 2021). The drift and diffusion models of neural SDE are the same as NLSB. The velocity model of TrajectoryNet includes the concatsquash layers used in Grathwohl et al. (2018). The base models of OT-Flow and TrajectoryNet were trained with the standard neural ODEs scheme. +OT represents a model trained with the OT-based regularization defined by Eqs. (1) and (2). We used only $\tilde{\mathcal{R}}_e$ for TrajectoryNet; we used both $\tilde{\mathcal{R}}_e$ and $\tilde{\mathcal{R}}_h$ for OT-Flow. We set the interval-dependent coefficients $\tilde{\lambda}_e,\ \tilde{\lambda}_h$ for the OT-based regularization as well as Eq. (11). The drift models of IPML (GP) and IPML (NN) are sparse GP and the same networks as NLSB, respectively. The diffusion coefficients of IPML were tuned as hyperparameters. See Appendix D for more details on hyperparameters, learning rates, and optimizers.



(a) MDD at 12 time points on synthetic OU process data.(b) CDD at 12 time points on synthetic OU process data.

Figure 3: Numerical evaluation on synthetic OU process data. All MDD and CDD values were computed between the ground-truth and the estimated samples within generated trajectories all-step ahead from initial samples $\mathbf{x}(t_0)$.



(a) ground-truth SDE (b) Neural SDE (c) NLSB (Ours) (d) TrajectoryNet + OT

Figure 4: 1D OU process data and predictions. The five colored point clouds in the background are the ground-truth data given at each time point. The pink area and the light blue line are the one-sigma empirical confidence intervals and their boundaries for each trajectory, respectively. All trajectories were generated by all-step prediction from the initial samples at the time $t = 0$.

5.1 Synthetic Population Dynamics: Ornstein–Uhlenbeck Process

Data. We used a time-dependent one-dimensional Ornstein–Uhlenbeck (OU) process defined by the following SDE:

$$\mathrm{d}X_t = (\mu t - \theta X_t)\,\mathrm{d}t + \left(\frac{2t\sigma}{t_{K-1}}\right)\,\mathrm{d}W_t,$$

where $\mu = 0.4,\ \theta = 0.1,\ \sigma = 0.8$, and $t_{K-1} = 4$. First, we simulated the several trajectories from $t = 0$ to 4 and then extracted only the data at the time of $T = \{0, 1, 2, 3, 4\}$ as snapshots for training. We generated 2048 and 512 samples for each time point as training and validation data, respectively.

Performance metrics. We evaluated the estimation performance of the dynamics in the time interval $[t_0, t_{K-1}]$ by using two metrics on test data: marginal distribution discrepancy (MDD) and conditional distribution discrepancy (CDD).

A smaller MDD between μ_t and ρ_t calculated on the basis of the Wasserstein-2 distance $\mathcal{W}(\mu_t,\ \rho_t)$ indicates better prediction of population-level dynamics at time t. We calculated MDD at 12 equally spaced time points and the earth mover's distance with L2 cost (EMD-L2) between 1000 samples generated from the ground-truth SDE and predicted by the model. When evaluating the SDE-based method, we ran 100 simulations from the same initial values, computing the MDD value each time and computing their mean and variance.

A smaller CDD between $\mu_{\mathbf{x}(t)|\mathbf{x}(t_0)}$ and $\rho_{\mathbf{x}(t)|\mathbf{x}(t_0)}$ using the Wasserstein-2 distance indicates better prediction of the time evolution of the initial sample $\mathbf{x}(t_0)$. In short, it is a metric for evaluating the time evolution of at the individual sample level. In the actual CDD calculation, we first prepared 1000 samples $\mathbf{x}(t_0)$ at the initial time point. We then generated 100 trajectories from each initial sample by the trained model and the ground-truth SDE, and calculated the EMD-L2 for the samples from $\mu_{\mathbf{x}(t)|\mathbf{x}(t_0)}$ and $\rho_{\mathbf{x}(t)|\mathbf{x}(t_0)}$ at 12 equally spaced time points.

Results. The evaluation results are shown in Figure 3, and the visualization of trajectories is shown in Figure 4. Figures 3 show that NLSB outperforms neural SDE and is comparable to other ODE-based methods in estimating populations with small variance. In contrast, the SDE-based methods outperform ODE-based methods when estimating populations with a large variance. That indicates that NLSB can estimate population-level dynamics even when the population variance is large or small. Furthermore, NLSB has a smaller CDD value than neural SDE and is comparable to IPML. Figure 4b shows that the average behavior of samples $E[\hat{X}(t)|X(0)]$ estimated by neural SDE is different from that of the ground-truth SDE (see Figure 4a), especially in the interval $[0, 1]$. In contrast, the prediction by NLSB in Figure 4c is much closer to the ground-truth. These results show that the prior knowledge introduced by the Lagrangian helps to estimate the sample-level dynamics.

5.2 Single-Cell Population Dynamics

Data. We evaluated on embryoid body scRNA-seq data (Moon et al., 2019). This data shows the differentiation of human embryonic stem cells from embryoid bodies into diverse cell lineages, including mesoderm, endoderm, neuroectoderm, and neural crest, over 27 days. During this period, cells were collected at five different snapshots (t_0: day 0 to 3, t_1: day 6 to 9, t_2: day 12 to 15, t_3: day 18 to 21, t_4: day 24 to 27). The collected cells were then measured by scRNAseq, filtered at the quality control stage, and mapped to a low-dimensional feature space using a principal component analysis (PCA). For details, see Appendix E.2 in (Tong et al., 2020). We split the dataset into train, validation($\sim 8.5\%$) and test data ($\sim 15\%$).

Performance metrics. Unlike the experiment described in Section 5.1, there are no ground-truth trajectories in the real data. Thus, MDD can be calculated only at the time of observation, and CDD between the ground truth and predicted trajectories cannot be calculated. To evaluate the sample-level dynamics, the model was trained on the full data and the data without only one intermediate snapshot, respectively. We then calculated CDD between the predicted trajectories by those models. Let $\mathcal{D}_{-t_k}$ be the training data without a snapshot at time t_k. Larger CDD between them indicates the prediction of the sample-level dynamics is not robust to the exclusion of intermediate snapshots, representing poorer performance in interpolating the sample-level dynamics. When evaluating the SDE-based methods, we calculated the mean and standard deviation of 100 MDD scores. All performance metrics are calculated on the test data.

Results. Table 1 and Figure 5 show that the NLSB can predict population-level dynamics with better performance and can be trained in a shorter time against all existing ODE-based methods as the data become higher dimensional. In particular, the SDE-based methods significantly outperform the ODE-based ones in predicting the transitions where the samples from t_1 to t_2 and from t_3 to t_4 are highly diffuse, indicating that the explicit modeling of diffusion is effective. (see Figure 8 in Appendix E). Overall, the standard deviation of the MDD values is smaller for NLSB than for neural SDE, demonstrating less variation in the approximation accuracy of the marginal distribution. Table 2 shows that NLSB estimates the sample-level dynamics robustly with and without population at the intermediate time point than neural SDE with some exceptions. Especially, energy regularization is the most stable and effective. This result suggests that the LSB-based regularization helps interpolate the sample-level dynamics. See the Appendix E for a visualization of trajectories with uncertainty.

6 Discussion and Conclusion

In this work, we proposed a novel framework for estimating population dynamics that reflect prior knowledge about the target system. Unlike existing methods with OT (Schiebinger et al., 2019; Yang & Uhler, 2018), or CNF (Tong et al., 2020) for a biological system, we explicitly modeled the diffusion phenomena by using SDEs for the samples with stochastic behavior. This allowed us to handle the uncertainty of the trajectory (Fig. 7) and to successfully capture the diffuse transitions

Table 1: Evaluation results for population-level dynamics on five-dimensional (5D) PCA space at time of observation for scRNA-seq data. The MDD value at t_k is computed between the ground-truth and the samples predicted from the previous ground-truth samples at t_{k-1} for each $k = 1, 2, 3$ and 4.

MDD ↓	t_1	t_2	t_3	t_4
NLSB (E)	0.63 ± 0.015	0.75 ± 0.017	0.75 ± 0.014	$\mathbf{0.72} \pm 0.010$
NLSB (D)	$\mathbf{0.60} \pm 0.012$	0.77 ± 0.022	0.77 ± 0.017	$\mathbf{0.72} \pm 0.014$
NLSB (V)	0.63 ± 0.015	0.77 ± 0.019	0.75 ± 0.017	0.75 ± 0.014
NLSB (E+D+V)	0.61 ± 0.014	$\mathbf{0.74} \pm 0.021$	$\mathbf{0.73} \pm 0.015$	$\mathbf{0.72} \pm 0.015$
Neural SDE	0.62 ± 0.016	0.78 ± 0.021	0.77 ± 0.017	0.75 ± 0.017
OT-Flow	0.74	0.96	0.93	0.95
OT-Flow + OT	0.75	0.93	0.93	0.88
TrajectoryNet	0.64	0.87	0.78	0.89
TrajectoryNet + OT	0.66	0.87	0.78	0.97
IPML (GP)	0.63 ± 0.015	0.88 ± 0.026	0.82 ± 0.017	0.87 ± 0.024
IPML (NN)	0.65 ± 0.016	0.78 ± 0.021	0.76 ± 0.018	0.76 ± 0.017

Table 2: Mean value of CDD on 5D PCA space evaluated at 7 equally spaced time points within time period $[t_{k-1}, t_{k+1}]$ around excluded time point t_k for each $k = 1, 2$ and 3. The CDD value at the time $t \in [t_{k-1}, t_{k+1}]$ was computed between the two groups of predicted samples generated from the previous samples at t_{k-1} using the model trained on all data and the data $\mathcal{D}_{-t_k}$.

Mean CDD ↓	$[t_0, t_2]$	$[t_1, t_3]$	$[t_2, t_4]$
NLSB (E)	**0.83**	**0.66**	0.74
NLSB (D)	1.56	0.69	**0.71**
NLSB (V)	1.07	0.76	0.80
NLSB (E+D+V)	0.90	0.76	0.78
Neural SDE	1.30	0.78	0.82
IPML (GP)	0.88	0.94	0.97
IPML (NN)	0.84	0.88	0.91

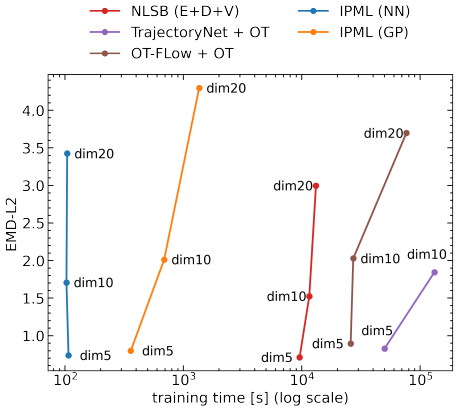


Figure 5: Relationship between data dimension, performance, and training time. The x-axis is the learning time until sufficient convergence. The y-axis is the mean value of MDD over time points t_1 to t_4, representing population-level performance.

(Tab. 1 and Fig. 8). In contrast, Vargas et al. (2021) and Bunne et al. (2022) estimated the sample trajectories of biological systems using the SDE solution to the SB problem. Vargas et al. (2021) proposed IPML algorithm to solve the SB problem. Bunne et al. (2022) proposed GSB-Flow, in which two SB problems are solved sequentially. The first SB problem with Gaussian marginals was solved in closed form, and the next SB problem using the closed-form solution as the reference path was solved by maximum likelihood training. Compared with these methods, our method handled a wider class of SDEs, and the diffusion function of the SDE was learned from the data using a backpropagation. Moreover, designing the Lagrangian enables us to flexibly incorporate prior knowledge about the target system into the model. Our Lagrangian-based regularization also treated the biological constraints proposed by Tong et al. (2020) and Maoutsa & Opper (2021) in a unified manner. We demonstrated that NLSB can efficiently estimate the population-level dynamics with better performance than existing methods even for high-dimensional data and that the prior knowledge introduced by the Lagrangian is useful to estimate the sample-level dynamics. Our method is limited in that it cannot model reaction phenomena such as cell birth and death, and restriction to the diagonal diffusion matrix (Section 4.2) may cause the model to be less expressive. Future work includes developing methods that can handle reaction phenomena with advection and diffusion.

ACKNOWLEDGMENTS

We thank the lab members for their discussions and inspiration for our research.

REFERENCES

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Jean-David Benamou and Yann Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Charlotte Bunne, Laetitia Meng-Papaxanthos, Andreas Krause, and Marco Cuturi. Jkonet: Proximal optimal transport modeling of population dynamics. *arXiv preprint arXiv:2106.06345*, 2021.

Charlotte Bunne, Ya-Ping Hsieh, Marco Cuturi, and Andreas Krause. Recovering stochastic dynamics via gaussian schrödinger bridges. *arXiv preprint arXiv:2202.05722*, 2022.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Tianrong Chen, Guan-Horng Liu, and Evangelos Theodorou. Likelihood training of schrödinger bridge using forward-backward sdes theory. In *International Conference on Learning Representations*, 2021a.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. Stochastic control liaisons: Richard sinkhorn meets gaspard monge on a schrodinger bridge. *SIAM Review*, 63(2):249–313, 2021b.

Paolo Dai Pra. A stochastic control approach to reciprocal diffusion processes. *Applied mathematics and Optimization*, 23(1):313–329, 1991.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet. Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709, 2021.

Lawrence C Evans. An introduction to mathematical optimal control theory version 0.2. *Lecture notes available at http://math. berkeley. edu/˜ evans/control. course. pdf*, 1983.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pp. 3154–3164. PMLR, 2020.

Mike Fisher, Jorge Nocedal, Yannick Trémolet, and Stephen J Wright. Data assimilation in weather forecasting: a case study in pde-constrained optimization. *Optimization and Engineering*, 10(3): 409–426, 2009.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

Tatsunori Hashimoto, David Gifford, and Tommi Jaakkola. Learning population-level diffusions with generative rnns. In *International Conference on Machine Learning*, pp. 2417–2426. PMLR, 2016.

Desmond J Higham. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM review*, 43(3):525–546, 2001.

Benton Jamison. The markov processes of schrödinger. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 32(4):323–331, 1975.

Patrick Kidger, James Foster, Xuechen Li, and Terry J Lyons. Neural sdes as infinite-dimensional gans. In *International Conference on Machine Learning*, pp. 5453–5463. PMLR, 2021.

AN Kolomgorov, IG Petrovskii, and NS Piskunov. Study of a diffusion equation that is related to the growth of a quality of matter, and its application to a biological problem, byul. *Mosk. Gos. Univ. Ser. A Mat. Mekh*, 1(1):26, 1937.

Christian Léonard. A survey of the schr\" odinger problem and some of its connections with optimal transport. *arXiv preprint arXiv:1308.0215*, 2013.

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*, pp. 3870–3882. PMLR, 2020.

Kenneth G Manton, XiLiang Gu, and Gene R Lowrimore. Cohort changes in active life expectancy in the us elderly population: Experience from the 1982–2004 national long-term care survey. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, 63(5):S269–S281, 2008.

Dimitra Maoutsa and Manfred Opper. Deterministic particle flows for constraining stochastic nonlinear systems. *arXiv preprint arXiv:2112.05735*, 2021.

Toshio Mikami. Variational processes from the weak forward equation. *Communications in mathematical physics*, 135(1):19–40, 1990.

Toshio Mikami. Optimal transportation problem as stochastic mechanics. *Selected Papers on Probability and Statistics, Amer. Math. Soc. Transl. Ser*, 2(227):75–94, 2008.

Toshio Mikami. Stochastic optimal transport revisited. *SN Partial Differential Equations and Applications*, 2(1):1–26, 2021.

Kevin R Moon, David van Dijk, Zheng Wang, Scott Gigante, Daniel B Burkhardt, William S Chen, Kristina Yim, Antonia van den Elzen, Matthew J Hirn, Ronald R Coifman, et al. Visualizing structure and transitions in high-dimensional biological data. *Nature biotechnology*, 37(12): 1482–1492, 2019.

Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

Geoffrey Schiebinger, Jian Shu, Marcin Tabaka, Brian Cleary, Vidya Subramanian, Aryeh Solomon, Joshua Gould, Siyan Liu, Stacie Lin, Peter Berube, et al. Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming. *Cell*, 176(4):928–943, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

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*, pp. 9526–9536. PMLR, 2020.

Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *arXiv preprint arXiv:1905.09883*, 2019a.

Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pp. 3084–3114. PMLR, 2019b.

Francisco Vargas, Pierre Thodoroff, Austen Lamacraft, and Neil Lawrence. Solving schrödinger bridges via maximum likelihood. *Entropy*, 23(9):1134, 2021.

Gefei Wang, Yuling Jiao, Qian Xu, Yang Wang, and Can Yang. Deep generative learning via schrödinger bridge. In *International Conference on Machine Learning*, pp. 10794–10804. PMLR, 2021.

Karren D Yang and Caroline Uhler. Scalable unbalanced optimal transport using generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Liu Yang and George Em Karniadakis. Potential flow generator with l_2 optimal transport regularity for generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 1999.

Algorithm 1 Training of NLSB

Input: Dataset $\{\mathcal{X}_t\}_{t\in T}$, potential function $\mathbf{\Phi}_\theta$, diffusion function $\mathbf{g}_\phi$, Lagrangian L, regularization coefficients $\lambda_e,\ \lambda_h$.

while $\theta,\ \phi$ have not converged **do**

Set loss $\mathcal{L}=0$.

for $k \leftarrow 0$ to $K-2$ **do**

Sample mini-batch $\mathbf{x}^{(k)}$ from $\mathcal{X}_{t_k}$. Set $\mathbf{z}_{t_k}=\mathbf{x}^{(k)}$.

Numerically solve augmented SDE $\mathrm{d}\mathbf{s}_t=\tilde{\mathbf{f}}_\theta(\mathbf{s}_t,t)\,\mathrm{d}t+\tilde{\mathbf{g}}_\phi(\mathbf{s}_t,t)\,\mathrm{d}\mathbf{W}_t$ from t_k to t_{k+1}.

$$\tilde{\mathbf{f}}_\theta(\mathbf{s}_t,t)=\begin{bmatrix}\left[\nabla_{\mathbf{u}}L\right]^{-1}\left(-\nabla\Phi_\theta(\mathbf{z}_t,t)\right)\\ L(t,\mathbf{z}_t,\left[\nabla_{\mathbf{u}}L\right]^{-1}\left(-\nabla\Phi_\theta(\mathbf{z}_t)\right))\\ \left|\partial_t\Phi_\theta(\mathbf{z}_t,t)+\sum_{i,j}D_{i,j}(\mathbf{z}_t,t)\nabla^2\Phi_\theta(\mathbf{z}_t,t)+h(\mathbf{z}_t,t)\right|\end{bmatrix},$$

$$\tilde{\mathbf{g}}_\phi(\mathbf{s}_t,t)=\begin{bmatrix}\mathbf{g}_\phi(\mathbf{z}_t,t) & & \\ & 0 & \\ & & 0\end{bmatrix},\ \mathbf{s}_t=\begin{bmatrix}\mathbf{z}_t\\ \partial_t\mathcal{R}_e\\ \partial_t\mathcal{R}_h\end{bmatrix},\ \mathbf{s}_{t_k}=\begin{bmatrix}\mathbf{z}_{t_k}\\ 0\\ 0\end{bmatrix}.$$

Sample mini-batch $\mathbf{x}^{(k+1)}$ from $\mathcal{X}_{t_{k+1}}$ and calculate the loss function $\ell(t_k,t_{k+1})$.

$$\ell(t_k,t_{k+1})=\overline{\mathcal{W}}_\epsilon(\mathbf{z}_{t_{k+1}},\mathbf{x}^{(k+1)})+\lambda_e(t_k,t_{k+1})\hat{\mathcal{R}}_e(t_k,t_{k+1})+\lambda_h(t_k,t_{k+1})\hat{\mathcal{R}}_h(t_k,t_{k+1}),$$
$$\mathbf{s}_{t_{k+1}}=\begin{bmatrix}\mathbf{z}_{t_{k+1}}\\ \hat{\mathcal{R}}_e^\theta(t_k,t_{k+1})\\ \hat{\mathcal{R}}_h^{\theta,\phi}(t_k,t_{k+1})\end{bmatrix}.$$

Accumulate loss $\mathcal{L}\leftarrow\mathcal{L}+\ell(t_k,t_{k+1})$.

Update $\theta,\ \phi$ with gradient $\nabla_\theta\mathcal{L}$ and $\nabla_\phi\mathcal{L}$, respectively.

Output: $\Phi_\theta,\mathbf{g}_\phi$

A Derivation of Neural Lagrangian Schrödinger Bridge

We consider the LSB problem constrained by the Fokker-Planck (FP) equation corresponding to Ito SDE (Eq. (3)) over the bounded domain $\Omega\subset\mathbb{R}^d$.

$$\begin{aligned}\underset{\theta,\phi}{\text{minimize}}\quad & \int_{t_0}^{t_1}\int_\Omega L(t,\mathbf{x},\mathbf{f}_\theta(\mathbf{x},t))\,\mathrm{d}\rho_t(\mathbf{x};\theta,\phi)\,\mathrm{d}t,\\ \text{subject to}\quad & \frac{\partial p_t}{\partial t}=-\operatorname{div}(p_t\mathbf{f}_\theta)+\sum_{i,j=1}^d\frac{\partial^2}{\partial x_i\partial x_j}\left[D_{i,j}(\mathbf{x},t;\phi)p_t(\mathbf{x})\right],\\ & \rho_{t_0}=\mu_0,\ \rho_{t_1}=\mu_1,\end{aligned}\tag{12}$$

where p_t is the probability density of the probability measure ρ_t.

The domain Ω satisfies the no-flux condition:

$$\mathbf{f}_t\cdot\mathbf{n}=0\quad\text{on}\quad\partial\Omega\times[t_0,t_1],\tag{13}$$

where $\mathbf{n}$ is the unit outward normal vector to the boundary of Ω. We will solve the LSB problem as shown in Eq. (6) by reformulating it as a saddle point problem for $(p_t,\mathbf{m}_t):=(p_t,p_t\mathbf{f}_t)$. Let $\mathcal{L}$ be the the Lagrangian with the time-space-dependent Lagrange multiplier $\Phi(\mathbf{x},t)$.

$$\mathcal{L}(\Phi, p, \mathbf{m}) := \int_{t_0}^{t_1} \int_\Omega L\left(t, \mathbf{x}, \frac{\mathbf{m}_t}{p_t}\right) p_t(\mathbf{x}) \,\mathrm{d}\mathbf{x}\,\mathrm{d}t$$
$$- \int_{t_0}^{t_1} \int_\Omega \Phi(\mathbf{x}, t) \left(\underbrace{\partial_t p_t}_{(a)} + \underbrace{\mathrm{div}(\mathbf{m}_t)}_{(b)} - \underbrace{\sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left[D_{i,j}(\mathbf{x}, t) p_t(\mathbf{x})\right]}_{(c)} \right) \mathrm{d}\mathbf{x}\,\mathrm{d}t.$$

The term (a) is transformed by performing a partial integral over t:

$$\int_{t_0}^{t_1} \int_\Omega \Phi(\mathbf{x}, t) \partial_t p_t(\mathbf{x}) \,\mathrm{d}\mathbf{x}\,\mathrm{d}t = \int_\Omega \Phi(\mathbf{x}, t_1) p(\mathbf{x}, t_1) \,\mathrm{d}\mathbf{x} - \int_\Omega \Phi(\mathbf{x}, t_0) p(\mathbf{x}, t_0) \,\mathrm{d}\mathbf{x}$$
$$- \int_{t_0}^{t_1} \int_\Omega \partial_t \Phi(\mathbf{x}, t) p_t(\mathbf{x}) \,\mathrm{d}\mathbf{x}\,\mathrm{d}t.$$

The term (b) is simplified by the Stokes' theorem and the no-flux condition, as shown in Eq. (13):

$$\begin{aligned}
\int_\Omega \Phi(\mathbf{x}, t) \,\mathrm{div}(\mathbf{m}_t) \,\mathrm{d}\mathbf{x} &= \int_\Omega \mathrm{div}(\Phi \mathbf{m}_t) - \mathbf{m}_t(\mathbf{x})^\top \nabla_\mathbf{x} \Phi(\mathbf{x}, t) \,\mathrm{d}\mathbf{x} \\
&= \oint_{\partial\Omega} \Phi(\mathbf{x}, t)(\mathbf{m}_t(\mathbf{x}) \cdot \mathbf{n}) \,\mathrm{d}\mathbf{x} - \int_\Omega \mathbf{m}_t(\mathbf{x})^\top \nabla_\mathbf{x} \Phi(\mathbf{x}, t) \,\mathrm{d}\mathbf{x} \\
&= - \int_\Omega \mathbf{m}_t(\mathbf{x})^\top \nabla_\mathbf{x} \Phi(\mathbf{x}, t) \,\mathrm{d}\mathbf{x}.
\end{aligned}$$

The term (c) is transformed by performing a partial integral over $\mathbf{x}$:

$$\begin{aligned}
\int_\Omega \sum_{i,j=1}^{d} \frac{\partial^2 D_{i,j}(\mathbf{x}, t) p_t(\mathbf{x})}{\partial x_i \partial x_j} \Phi(\mathbf{x}, t) \,\mathrm{d}\mathbf{x} &= - \sum_{i,j=1}^{d} \int_\Omega \frac{\partial \left(D_{i,j}(\mathbf{x}, t) p_t(\mathbf{x})\right)}{\partial x_j} \frac{\partial \Phi(\mathbf{x}, t)}{\partial x_i} \,\mathrm{d}\mathbf{x} \\
&= \sum_{i,j=1}^{d} \int_\Omega D_{i,j}(\mathbf{x}, t) p_t(\mathbf{x}) \frac{\partial^2 \Phi(\mathbf{x}, t)}{\partial x_i \partial x_j} \,\mathrm{d}\mathbf{x} \\
&= \int_\Omega \left(\sum_{i,j=1}^{d} D_{i,j}(\mathbf{x}, t) \left[\nabla_\mathbf{x}^2 \Phi(\mathbf{x}, t)\right]_{i,j} \right) p_t(\mathbf{x}) \,\mathrm{d}\mathbf{x}.
\end{aligned}$$

Then, we can rewrite the LSB problem as

$$\inf_{p, \mathbf{m}} \sup_{\Phi} \mathcal{L}(p, \mathbf{m}, \Phi), \tag{14}$$

$$\begin{aligned}
\mathcal{L}(p, \mathbf{m}, \Phi) = &\int_{t_0}^{t_1} \int_\Omega \left(L\left(t, \mathbf{x}, \frac{\mathbf{m}_t}{p_t}\right) + \partial_t \Phi(\mathbf{x}, t) + \sum_{i,j=1}^{d} D_{i,j} \left[\nabla_\mathbf{x}^2 \Phi(\mathbf{x}, t)\right]_{i,j} \right) p_t(\mathbf{x}) \,\mathrm{d}\mathbf{x}\,\mathrm{d}t \\
&+ \int_{t_0}^{t_1} \int_\Omega \mathbf{m}_t(\mathbf{x})^\top \nabla_\mathbf{x} \Phi(\mathbf{x}, t) \,\mathrm{d}\mathbf{x}\,\mathrm{d}t \\
&- \int_\Omega \Phi(\mathbf{x}, t_1) p(\mathbf{x}, t_1) \,\mathrm{d}\mathbf{x} + \int_\Omega \Phi(\mathbf{x}, t_0) p(\mathbf{x}, t_0) \,\mathrm{d}\mathbf{x}.
\end{aligned} \tag{15}$$

The saddle point $(p^*, \mathbf{m}^*, \Phi^*)$ of the problem in Eq. (14) satisfies the following conditions:

$$\partial_\Phi \mathcal{L}|_{(p^*, \mathbf{m}^*, \Phi^*)} = 0 \Leftrightarrow \begin{cases} \partial_t p_t^* + \sum_{i=1}^{d} \frac{\partial}{\partial x_i} m_i^*(\mathbf{x}, t) - \sum_{i,j=1}^{d} \frac{\partial^2}{\partial x_i \partial x_j} \left[D_{i,j}(\mathbf{x}, t) p_t^*\right] = 0 \\ p^*(\cdot, t_0) = \mu_0,\ p^*(\cdot, t_1) = \mu_1 \end{cases},$$

$$\partial_p \mathcal{L}|_{(p^*, \mathbf{m}^*, \Phi^*)} = 0 \Leftrightarrow \begin{array}{l} \partial_t \Phi^* + \sum_{i,j=1}^{d} D_{i,j}(\mathbf{x}, t) \left[\nabla_\mathbf{x}^2 \Phi(\mathbf{x}, t)\right]_{i,j} + h(\mathbf{x}, t) = 0 \\ h(\mathbf{x}, t) = L\left(t, \mathbf{x}, \frac{\mathbf{m}^*}{p^*}\right) - \nabla_\mathbf{u} L(t, \mathbf{x}, \mathbf{u})|_{\mathbf{u} = \frac{\mathbf{m}^*}{p^*}}^\top \left(\frac{\mathbf{m}^*}{p^*}\right) \end{array},$$

$$\partial_\mathbf{m} \mathcal{L}|_{(p^*, \mathbf{m}^*, \Phi^*)} = 0 \Leftrightarrow \nabla_\mathbf{u} L(t, \mathbf{x}, \mathbf{u})|_{\mathbf{u} = \frac{\mathbf{m}^*}{p^*}} = -\nabla_\mathbf{x} \Phi^*.$$

Therefore, the optimal drift function is given by $\mathbf{f}^* = \frac{\mathbf{m}^*}{p^*} = [\nabla_{\mathbf{u}} L]^{-1} (-\nabla_{\mathbf{x}} \Phi)$ and the potential function Φ is the solution of the HJB equation:

$$\partial_t \Phi(\mathbf{x}, t) + \sum_{i,j=1}^{d} D_{i,j}(\mathbf{x}, t) \left[\nabla_{\mathbf{x}}^2 \Phi(\mathbf{x}, t)\right]_{i,j} + L\left(t, \mathbf{x}, \mathbf{f}^*\right) + \nabla_{\mathbf{x}} \Phi(\mathbf{x}, t)^\top \mathbf{f}^*(\mathbf{x}, t) = 0.$$

B Model Architecture

The model structure of the potential function proposed in OT-Flow (Onken et al., 2021) is shown below.

$$\Phi(\mathbf{s}) = \mathbf{w}^\top N\left(\mathbf{s}; \{\mathbf{K}_i, \mathbf{b}_i\}_{0 \le i \le M}\right) + \frac{1}{2}\mathbf{s}^\top (\mathbf{A}^\top \mathbf{A})\mathbf{s} + \mathbf{b}^\top \mathbf{s} + c,$$
$$N\left(\mathbf{s}; \{\mathbf{K}_i, \mathbf{b}_i\}_{0 \le i \le M}\right) = \mathbf{u}_M,$$
$$\mathbf{u}_i = \mathbf{u}_{i-1} + h\sigma(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i) \quad (1 \le i \le M),\ \mathbf{u}_0 = \sigma(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0),$$

where $\mathbf{s} = (\mathbf{x}, t) \in \mathbb{R}^{d+1}$ is a input vector, $\mathbf{w}, \mathbf{K}_0 \in \mathbb{R}^{m \times (d+1)}, \mathbf{K}_i\ (1 \le i \le M) \in \mathbb{R}^{m \times m},\ \mathbf{b}_i\ (0 \le i \le M) \in \mathbb{R}^{m \times m},\ \mathbf{A},\ \mathbf{b}$, and c are learnable parameters, h is a fixed step size, and the activation function is defined by $\sigma(\mathbf{x}) = \log(\exp(\mathbf{x}) + \exp(-\mathbf{x}))$.

The gradient of the potential function is described .

$$\nabla_{\mathbf{s}} \Phi(\mathbf{s}) = \nabla_{\mathbf{s}} N\left(\mathbf{s}; \{\mathbf{K}_i, \mathbf{b}_i\}_{0 \le i \le M}\right) \mathbf{w} + (\mathbf{A}^\top \mathbf{A})\mathbf{s} + \mathbf{b},$$
$$\nabla_{\mathbf{s}} N\left(\mathbf{s}; \{\mathbf{K}_i, \mathbf{b}_i\}_{0 \le i \le M}\right) = \mathbf{K}_0^\top \operatorname{diag}(\sigma'(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0))\mathbf{z}_1,$$
$$\mathbf{z}_i = \mathbf{z}_{i+1} + h\mathbf{K}_i^\top \operatorname{diag}(\sigma'(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i))\mathbf{z}_{i+1} \quad (1 \le i \le M),\ \mathbf{z}_{M+1} = \mathbf{1}.$$

We can write σ' down as $\tanh$ since σ is defined as above.

Finally, the diagonal components of the potential function's Hessian is shown below.

$$\begin{aligned}
\nabla_{\mathbf{s}}^2 \Phi(\mathbf{s}) &= \nabla_{\mathbf{s}}(\mathbf{K}_0^\top \operatorname{diag}(\sigma'(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0))\mathbf{z}_1) \\
&\quad + h \sum_{i=1}^{M} \nabla_{\mathbf{s}} \mathbf{u}_{i-1} \nabla_{\mathbf{s}}(\mathbf{K}_i^\top \operatorname{diag}(\sigma'(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i))\mathbf{z}_{i+1}) \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top \\
&= \mathbf{K}_0^\top \operatorname{diag}(\sigma''(\mathbf{K}_0 \mathbf{s} + \mathbf{b}_0) \odot \mathbf{z}_1)\mathbf{K}_0 \\
&\quad + \nabla_{\mathbf{s}} \mathbf{u}_{i-1} \mathbf{K}_i^\top \operatorname{diag}(\sigma''(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i) \odot \mathbf{z}_{i+1})\mathbf{K}_i \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top,
\end{aligned}$$

$$\begin{aligned}
\left[\nabla_{\mathbf{s}}^2 \Phi(\mathbf{s})\right]_{i,i} &= \left[(\sigma''(\mathbf{K}_0 \mathbf{s} + \mathbf{K}_0) \odot \mathbf{z}_1)^\top (\mathbf{K}_0 \odot \mathbf{K}_0)\right]_i \\
&\quad + h \sum_{i=1}^{M} \left[(\sigma''(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i) \odot \mathbf{z}_{i+1})^\top (\mathbf{K}_i \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top \odot \mathbf{K}_i \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top)\right]_i,
\end{aligned}$$

where $\odot$ is the element-wise product, $\nabla_{\mathbf{s}} \mathbf{u}_i$ can be obtained by using the following update equation:

$$\nabla_{\mathbf{s}} \mathbf{u}_i^\top \leftarrow \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top + \operatorname{diag}(h\sigma'(\mathbf{K}_i \mathbf{u}_{i-1} + \mathbf{b}_i))\mathbf{K}_i \nabla_{\mathbf{s}} \mathbf{u}_{i-1}^\top.$$

The diagonal component of the potential function Hessian can be computed at a computation cost of $O(m^2 d)$.

C Schrödinger Bridge and Lagrangian Schrödinger Bridge

We briefly explain a connection between the SB problem and the LSB problem.

Definition 2 (SB problem; (Jamison, 1975))**.** Let $\Omega = C([0, T], \mathbb{R}^d)$ be the space of $\mathbb{R}^d$-valued continuous functions on time interval $[0, T]$. Denote by $\mathcal{P}(\Omega)$ the probability measures space on the path pace Ω. The SB problem is defined by

$$\min_{\mathbb{Q} \in \mathcal{P}(\Omega)} D_{\mathrm{KL}}(\mathbb{Q} \| \mathbb{P}), \qquad \text{subject to} \quad \mathbb{Q}_0 \sim \mu_0,\ \mathbb{Q}_T \sim \mu_T, \tag{16}$$

where $\mu, \nu \in \mathcal{P}(\mathbb{R}^d)$ and the relative entropy $D_{\mathrm{KL}} = \int \log\left(\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}\right) \mathrm{d}\mathbb{Q}$ if $\mathbb{Q} \ll \mathbb{P}$, and $D_{\mathrm{KL}} = \infty$ otherwise.

The case of no prior dynamics is considered classically, *i.e.*, the reference path measure $\mathbb{P}$ is the Brownian diffusion SDE $\mathrm{d}\mathbf{X}_t = \sqrt{2\epsilon}\,\mathrm{d}\mathbf{W}_t$, where $\mathbf{W}_t$ is the standard Wiener process. Chen et al. (2021b) derived the optimality condition for problem 16, which is characterized by the forward and backward time-harmonic equations.

Theorem 3. *Let $\Psi(\mathbf{x}, t)$ and $\hat{\Psi}(\mathbf{x}, t)$ be the solutions to the following PDEs:*

$$\begin{cases} \frac{\partial \Psi}{\partial t} = -\epsilon \Delta \Psi \\ \frac{\partial \widehat{\Psi}}{\partial t} = \epsilon \Delta \widehat{\Psi} \end{cases} \quad s.t.\ \Psi(\cdot, 0)\widehat{\Psi}(\cdot, 0) = p_0, \Psi(\cdot, T)\widehat{\Psi}(\cdot, T) = p_T\ , \tag{17}$$

where p_0 and p_T are the probability density of μ_0 and μ_T, respectively. Then, the solution to the SB problem (16) can be described by the following forward or backward SDE:

$$\begin{aligned} \mathrm{d}\mathbf{X}_t &= 2\epsilon \nabla_{\boldsymbol{x}} \log \Psi\left(\mathbf{X}_t, t\right)\ \mathrm{d}t + \sqrt{2\epsilon}\,\mathrm{d}\mathbf{W}_t, \quad \mathbf{X}_0 \sim \mu_0, \\ \mathrm{d}\mathbf{X}_t &= -2\epsilon \nabla_{\boldsymbol{x}} \log \widehat{\Psi}\left(\mathbf{X}_t, t\right)\ \mathrm{d}t + \sqrt{2\epsilon}\,\mathrm{d}\mathbf{W}_t, \quad \mathbf{X}_T \sim \mu_T. \end{aligned}$$

Dai Pra (1991) considered a dynamic formulation of the SB problem (16) by interpreting it as a SOC problem with the additional terminal constraint.

Theorem 4. *Let $\mathbf{u}^* = 2\epsilon \nabla_{\boldsymbol{x}} \log \Psi$, where Ψ satisfies the SB optimality (17). Then, $\mathbf{u}^*$ is the minimizer of the following optimization problem:*

$$\begin{aligned} &\min_{\mathbf{u} \in \mathcal{U}} \mathbb{E}\left[\int_0^T \frac{1}{2}\left\|\mathbf{u}\left(\mathbf{X}_t, t\right)\right\|^2\ \mathrm{d}t\right], \\ &s.t.\ \begin{cases} \mathrm{d}\mathbf{X}_t = \mathbf{u}\left(\mathbf{X}_t, t\right)\ \mathrm{d}t + \sqrt{2\epsilon}\,\mathrm{d}\mathbf{W}_t, \\ \mathbf{X}_0 \sim \mu_0, \quad \mathbf{X}_T \sim \mu_T, \end{cases} \end{aligned} \tag{18}$$

where $\mathcal{U} = \left\{\mathbf{u} : [0, T] \times \mathbb{R}^d \mapsto \mathbb{R}^d \mid \langle \mathbf{u}, \mathbf{u} \rangle < \infty\right\}$.

Léonard (2013) formulated the SB problem16 into a variational SOC problem equivalent to the above problem 18.

Theorem 5. *Let $\mathbf{v}_t^*(\mathbf{x}) = \nabla_{\boldsymbol{x}} \log \mathbb{E}\left[\Psi(\mathbf{X}_T, T) \mid \mathbf{X}_t = \mathbf{x}\right]$, where Ψ satisfies the SB optimality (17). Then, $\mathbf{v}_t^*$ is the minimizer of the following optimization problem:*

$$\begin{aligned} &\min_{\mathbf{v}_t, \rho_t} \int_0^T \mathbb{E}_{\mathbf{X}_t \sim \rho_t}\left[\frac{1}{2}\left\|\mathbf{v}_t\left(\mathbf{X}_t\right)\right\|^2\right]\ \mathrm{d}t, \\ &s.t.\ \begin{cases} \partial_t p_t = -\operatorname{div}(p_t \mathbf{v}_t) + \epsilon \Delta p_t, \\ \mathbf{X}_0 \sim \rho_0 = \mu_0, \quad \mathbf{X}_T \sim \rho_T = \mu_T, \end{cases} \end{aligned} \tag{19}$$

where p_t is the probability density of the probability measure ρ_t.

The generalizations of problems 18 and 19 correspond to problems 4 and 5, respectively. When the Lagrangian $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2$ and the diffusion function is fixed, the LSB problem 6 is also reduced to the problem 19.

D Experimental Details

In our implementation, we modified code in the TrajectoryNet [1], OT-FLow[2] and IPML[3] repositories, which were released under the MIT license. The scRNA-seq data used in Section 5.2 are licensed under Creative Commons Attribution 4.0 International license. Our experimental environment consists of an Intel Xeon Plantinum 8360Y (36-core) CPU and a single NVIDIA A100 GPU. Code is available in the supplementary material.

D.1 Training Configuration for Experiments on Synthetic Data

We describe the details of the training setup for the experiments on synthetic data conducted in Section 5.1.

[1] `https://github.com/KrishnaswamyLab/TrajectoryNet`

[2] `https://github.com/EmoryMLIP/OT-Flow`

[3] `https://github.com/AforAnonyMeta/IPML-2548`

Table 3: Weight coefficients for regularization terms in experiments with synthetic data

	$[t_0, t_1]$	$[t_1, t_2]$	$[t_2, t_3]$	$[t_3, t_4]$
λ_e, λ_h for NLSB	0.3, 0.2	0.1, 0.01	0.01, 0.0001	0.01, 0.0001
$\tilde{\lambda}_e$ for TrajectoryNet	0.1	0.1	0.001	0.001
$\tilde{\lambda}_e$, $\tilde{\lambda}_h$ for OT-Flow	$0.1, 0.01$	$0.1, 0.01$	$0.001, 0.001$	$0.001, 0.001$

COMMON SETTINGS FOR NLSB, NEURAL SDE, TRAJECTORYNET AND OT-FLOW

We generated 2048 and 512 samples for each time point as training and validation data, respectively. We trained all models with a batch size of 512 for each time point and used the early stopping method, which monitors the EMD-L2 value on validation data. We used Adam optimizer to optimize all learnable parameters with a learning rate of 0.001 and the decay rate of $\beta_1 = 0.9,\ \beta_2 = 0.999$. We searched all weight coefficients of regularization terms in $[0.0, 0.5]$ and selected those with the largest possible coefficients among those with sufficiently small EMD-L2 values on the validation data.

SETTINGS FOR NLSB AND NEURAL SDE

For the potential model $\Phi_\theta(\mathbf{s})$, we set the number of ResNet layers $M = 2$, the step size $h = 1.0$, the rank of matrix $\mathrm{rank}(\mathbf{A}) = 10$, and the dimension of the hidden vector $\mathbf{z}$ to 2. We used a two-layer FCNN of a hidden dimension 16 for the diffusion function. The activations functions were LipSwish. We trained the neural SDE in the framework of the torchsde library [4] and used the Euler-Maruyama method with the constant step size of 0.01 as an SDE solver. Backpropagation was performed without using the adjoint method. In NLSB, we used the Lagrangian for the potential-free system $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2$. We set the weight coefficients $\lambda_e,\ \lambda_h$ shown in Table 3.

SETTINGS FOR TRAJECTORYNET

We used three concatsquash layers of a hidden dimension 16 with hyperbolic tangent activations to model the velocity function $\mathbf{v}$. A concatsquash layer cs was defined in the released code of FFJORD (Grathwohl et al., 2018) as:

$$\mathrm{cs}(\mathbf{x}, t) = (\mathbf{W}_x \mathbf{x} + \mathbf{b}_x)\sigma(\mathbf{W}_t t + \mathbf{b}_t) + (\mathbf{W}_b t + \mathbf{b}_b t),$$

where σ is the sigmoid function, and $\mathbf{W}_x,\ \mathbf{W}_t,\ \mathbf{W}_b,\ \mathbf{b}_x,\ \mathbf{b}_t,\ \mathbf{b}_b$ are all learnable parameters. We trained the model with torchdiffeq library [5] and used the dopri5 solver with both absolute and relative tolerances set to 10^{-5}. In the training for TrajectoryNet with OT-based regularization, we set the weight coefficients shown in Table 3.

SETTINGS FOR OT-FLOW

We used the same potential model Φ in NLSB and neural SDE with the same hyperparameters. We trained the model with torchdiffeq library. We used the Euler method as the ODE solver with a constant step size of 0.01 and both absolute and relative tolerances set to 10^{-5}. In the training for OT-Flow with OT-based regularization, we set the weight coefficients shown in Table 3.

SETTINGS FOR IPML

We generated 2048 and 512 samples for each time point as training and validation data, respectively. The drift model of IPML (GP) is sparse GP with the exponential kernel. We selected 100 inducing points using the K-means algorithm. The drift model of IPML (NN) is the same network as for NLSB, including hyperparameters. We solved the SB problem by using IPF algorithm with 15 iterations. We

[4] https://github.com/google-research/torchsde

[5] https://github.com/rtqichen/torchdiffeq

Table 4: Weight coefficients for regularization terms in experiments with scRNA-seq data

	$[t_0, t_1]$	$[t_1, t_2]$	$[t_2, t_3]$	$[t_3, t_4]$
λ_e, λ_h for NLSB (E)	0.1, 0.01	0.01, 0.01	0.001, 0.0001	0.01, 0.001
λ_e, λ_h for NLSB (V)	0.01, 0.001			
λ_e, λ_h for NLSB (D)	0.01, 0.001			
λ_e, λ_h for NLSB (E+D+V)	0.01, 0.001	0.001, 0.001	0.001, 0.001	0.001, 0.001
$\tilde{\lambda}_e$ for TrajectoryNet	0.01	0.01	0.1	0.1
$\tilde{\lambda}_e$, $\tilde{\lambda}_h$ for OT-Flow	0.01, 0.01	0.01, 0.01	0.001, 0.01	0.001, 0.01

used the Euler-Maruyama method as the SDE solver and set the diffusion coefficients, as follows.

$$\mathbf{g}(t) = \begin{cases} 0.2 & t \in [0.0, 1.0) \\ 0.6 & t \in [1.0, 2.0) \\ 1.0 & t \in [2.0, 3.0) \\ 1.4 & t \in [3.0, 4.0] \end{cases}. \tag{20}$$

Other experimental settings have not been changed from (Vargas et al., 2021).

D.2 Training Configuration for Experiments on ScRNA-seq Data

We describe the details of the training setup for the experiments on scRNA-seq data conducted in Section 5.2.

Common settings for NLSB, neural SDE, TrajectoryNet and OT-Flow

We reused the pre-processed data available in the released repository of TrajectoryNet [1] and split the data into 200 samples ($\sim 8.5\%$) of validation data, 350 samples ($\sim 15\%$) of test data, and the rest as train data for each time point. We trained all models with a batch size of 1000 for each time point and used the early stopping method, which monitors the EMD-L2 value on the validation data. We used Adam optimizer for training all methods with learning rate 0.001 and the decay rate $\beta_1 = 0.9,\ \beta_2 = 0.999$. We searched all weight coefficients of regularization terms in $[0.0, 0.5]$ and selected those with the largest possible coefficients among those with sufficiently small EMD-L2 values on the validation data.

Settings for NLSB

All settings for the potential model, the diffusion function model, and the SDE solver were the same as in Appendix D.1. We set the noise type to diagonal. We set the weight coefficients of regularization terms shown in Table 4. For the calculation of the density regularization term, the time-dependent density function $U(\mathbf{x}, t)$ was defined by

$$U(\mathbf{x}, t) = c \log p(\mathbf{x}; \Theta_t), \quad \Theta_t = \{\mu_m^{(t)}, \Sigma_m^{(t)}\}_{m=1}^{M_t},$$

where c is a hyperparameter to change the scale, and $\mu_m^{(t)}, \Sigma_m^{(t)}$ are mean and variance parameters of the mixed Gaussian distribution, respectively. When $t \in [t_k, t_{k+1}]$, the parameters Θ_t were estimated with the data at t_k and t_{k+1} by GMM and the number of mixture components M_t was determined by the value of Bayesian information criterion (BIC). The hyperparameter c was searched among $\{0.1, 1.0, 10.0\}$ and we set $c = 10.0$ for NLSB (D) and $c = 0.1$ for NLSB (E+D+V). For the calculation of the velocity regularization term, we used the same reference velocity as those used in TrajectoryNet (Tong et al., 2020).

Settings for neural SDE, TrajectoryNet and OT-Flow

We used the velocity model of TrajectoryNet with 64 hidden dimensions. All other settings such as potential models, diffusion function models, SDE solvers, and ODE solvers were the same as in Appendix D.1. In the training for TrajectoryNet and OT-Flow with OT-based regularization, we set the weight coefficients shown in Table 4.

SETTINGS FOR IPML

We set the diffusion coefficients to $g(t) = 0.5(t \in [0.0, 1.0]),\ 1.0 \quad (t \in [1.0, 4.0])$. All other settings were the same as in Appendix D.1.

D.3 TRAINING CONFIGURATION FOR ADDITIONAL EXPERIMENTS ON SYNTHETIC DATA

We describe the details of the training setup for the addtional experiments on synthetic data conducted in Section E.3.

DATA

We used two-dimensional uniform distributions $\mathcal{U}_0$ and $\mathcal{U}_1$ for the endpoints at time $t = 0$ and $t = 1$.

$$(X_0, Y_0) \sim \mathcal{U}_0 : -1.25 \leq X_0 \leq -1,\ -1 \leq Y_0 \leq 1,$$
$$(X_1, Y_1) \sim \mathcal{U}_1 : 1 \leq X_1 \leq 1.25,\ -1 \leq Y_1 \leq 1.$$

We generated 2048 and 512 samples from two endpoint distributions as training and validation data, respectively.

SETTINGS FOR NLSB

We set the Lagrangian for the random dynamical system with the mass $m = 1$, *i.e.* $L(t, \mathbf{x}, \mathbf{u}) = \frac{1}{2}||\mathbf{u}||^2 - U(\mathbf{x})$ and compared several practical potential functions U shown below.

1. Box-shaped obstacle

$$U(x, y) = \begin{cases} -100 & -0.5 \leq x, y \leq 0.5 \\ 0 & \text{otherwise} \end{cases}.$$

2. Slit-shaped obstacle

$$U(x, y) = \begin{cases} -100 & -0.1 \leq x \leq 0.1 \wedge (y \leq -0.25 \vee 0.25 \leq y) \\ 0 & \text{otherwise} \end{cases}.$$

3. Hill potential

$$U(x, y) = -2.5(x^2 + y^2).$$

4. Well potential

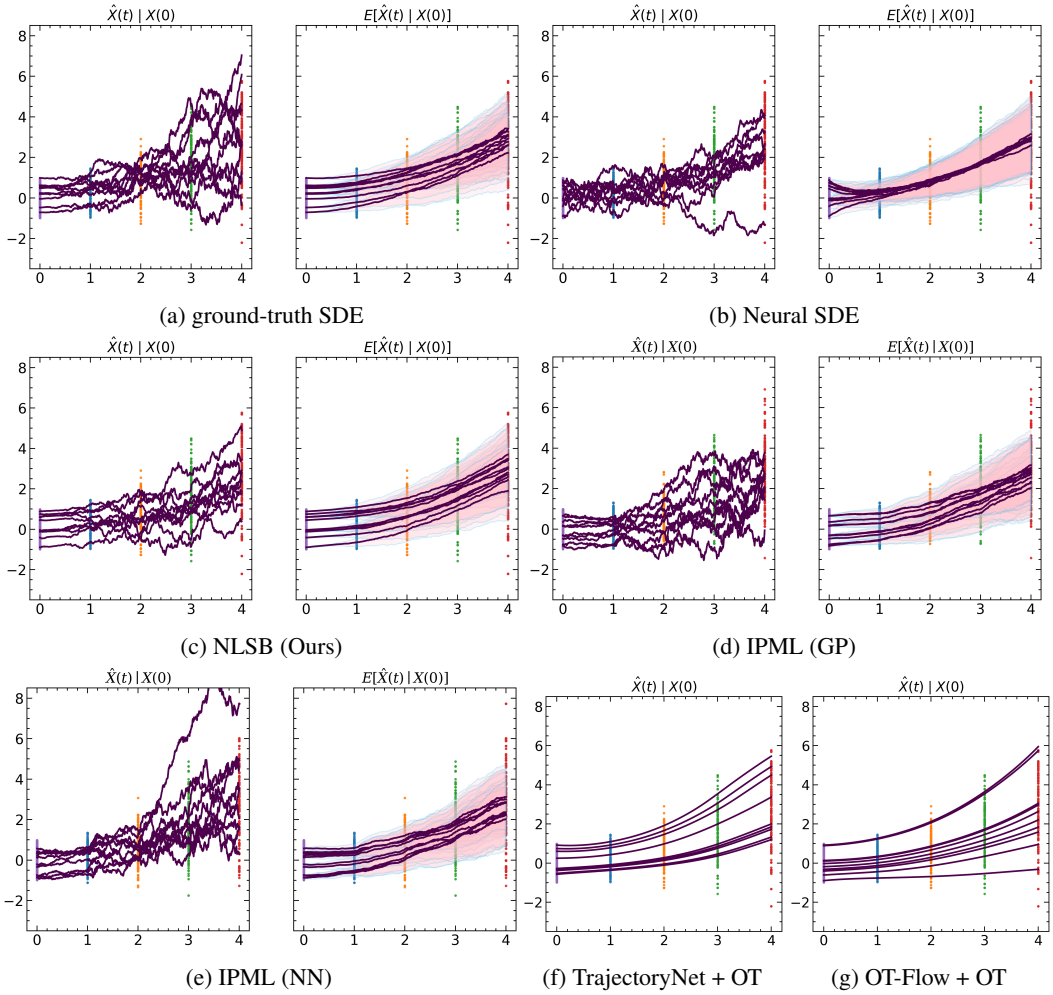
$$U(x, y) = -10 \exp(-(x^2 + y^2)).$$

We defined the box and slit-shaped obstacle potential functions as differentiable by using the sigmoid functions. We trained NLSB with a batch size of 512 and used the early stopping method, which monitors the validation loss value. We used Adam optimizer to optimize all learnable parameters with a learning rate of 0.001 and the decay rate of $\beta_1 = 0.9,\ \beta_2 = 0.999$. All weight coefficients of regularization terms were searched uniformly in $\{0.01, 0.001\}$. All settings for the potential model Φ_θ, the diffusion function model $\mathbf{g}_\phi$, and the SDE solver were the same as in Appendix D.1.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 SYNTHETIC POPULATION DYNAMICS: TIME-DEPENDENT ORNSTEIN–UHLENBECK PROCESS

The figure 6 shows the visualization of both the original trajectory and the averaged trajectory by the SDE-based methods (Neural SDE, NLSB, and IPML) and the only original trajectory by the ODE-based methods (TrajectoryNet and OT-Flow). All trajectories were generated by all-step prediction from the initial samples at the time $t = 0$. The five colored point clouds in the background are the ground-truth data given at each time point. The pink area and the light blue line are the one-sigma empirical confidence intervals and their boundaries for each trajectory, respectively.



(a) ground-truth SDE (b) Neural SDE

(c) NLSB (Ours) (d) IPML (GP)

(e) IPML (NN) (f) TrajectoryNet + OT (g) OT-Flow + OT

Figure 6: 1D OU process data and predictions.

E.2 Single-Cell Population Dynamics

The time evolution of the distribution for a single sample is visualized as a heat map in Figure 7. The sample population in Figure 8 and the trajectories are visualized in Figure 9. All figures are visualizations in the space of the first and second principal components. The x-axis represents the first principal component and the y-axis the second principal component.

Figure 7 shows that the SDE-based methods, including NLSB, can handle the uncertainty of the trajectories in contrast to ODE-based methods. Figure 8 shows that NLSB outperforms ODE-based methods in predicting the transitions with a high diffusion of samples from t_1 to t_2 and from t_3 to t_4, indicating that the explicit modeling of diffusion is effective. Figure 9 shows that the drift estimated by NLSB (E) is linear, the trajectories by NLSB (D) pass on the data manifold, and the trajectories by NLSB (E+D+V) appear to reflect all other regularization effects.

A GIF animation of the NLSB simulation in PCA space is also included in the supplemental materials.

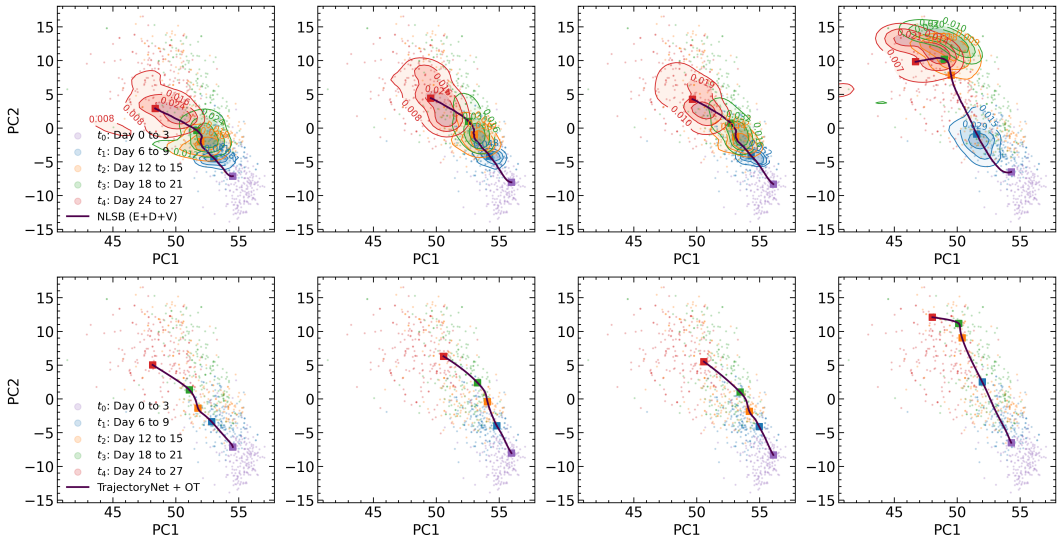
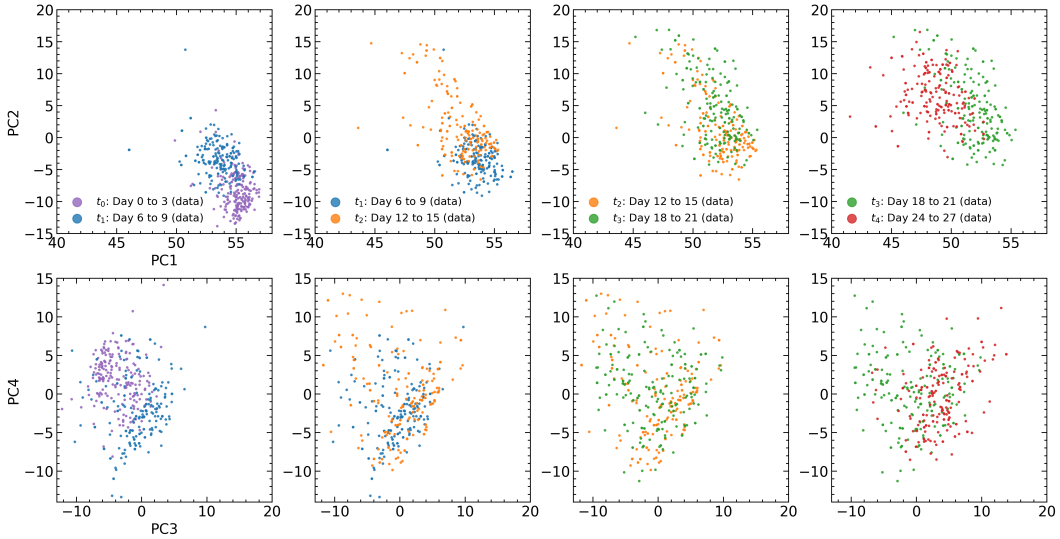
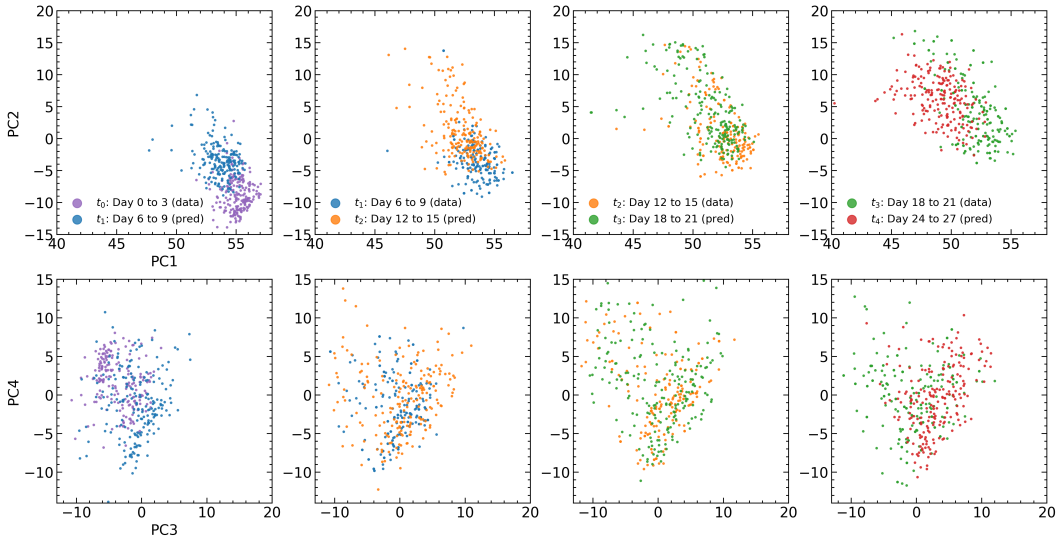


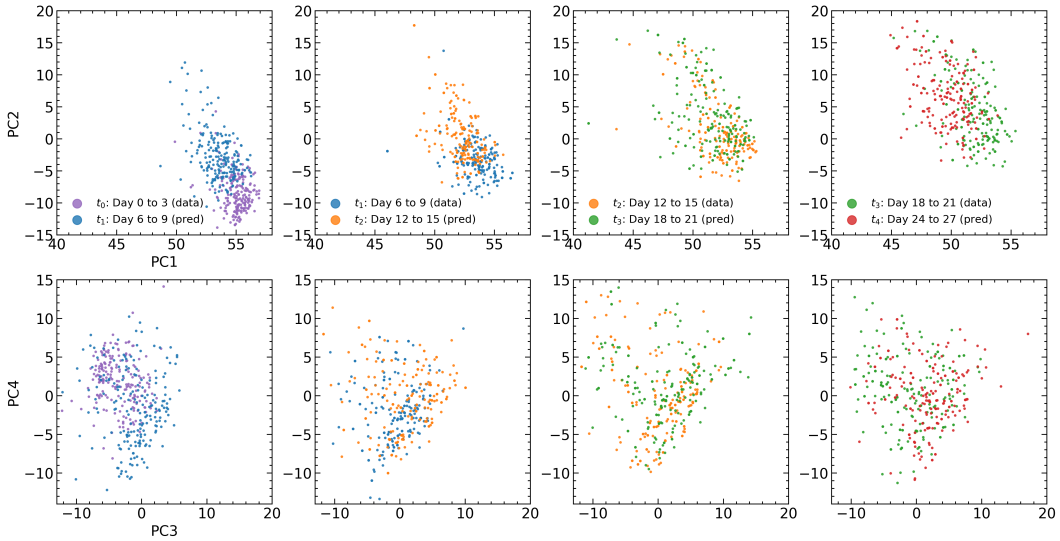
Figure 7: Visualization of the time evolution for a single sample on scRNA-seq data. The upper and lower images are predictions from the same initial sample at t_0, with the upper row predicted by NLSB and the lower row predicted by TrajectoryNet + OT. The color gradients depict the magnitude of the probability density. The probability density function is estimated by GMM with five mixture components.



(a) Ground-truth data

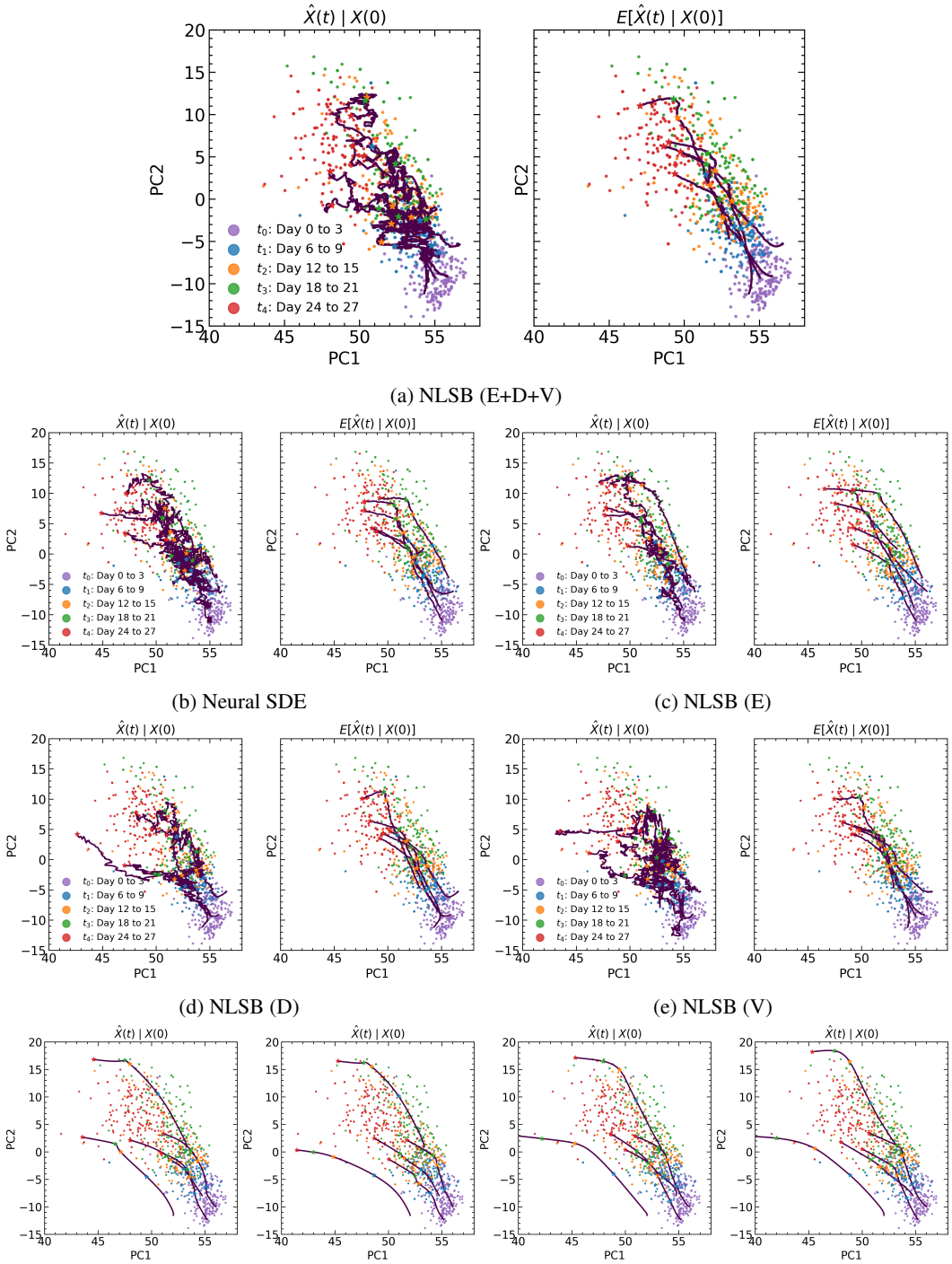


(b) NLSB (E+D+V)



(c) TrajectoryNet + OT

Figure 8: The ground-truth data and one-step ahead sample prediction on scRNA-seq data.



(a) NLSB (E+D+V)

(b) Neural SDE

(c) NLSB (E)

(d) NLSB (D)

(e) NLSB (V)

(f) TrajectoryNet

(g) TrajectoryNet + OT

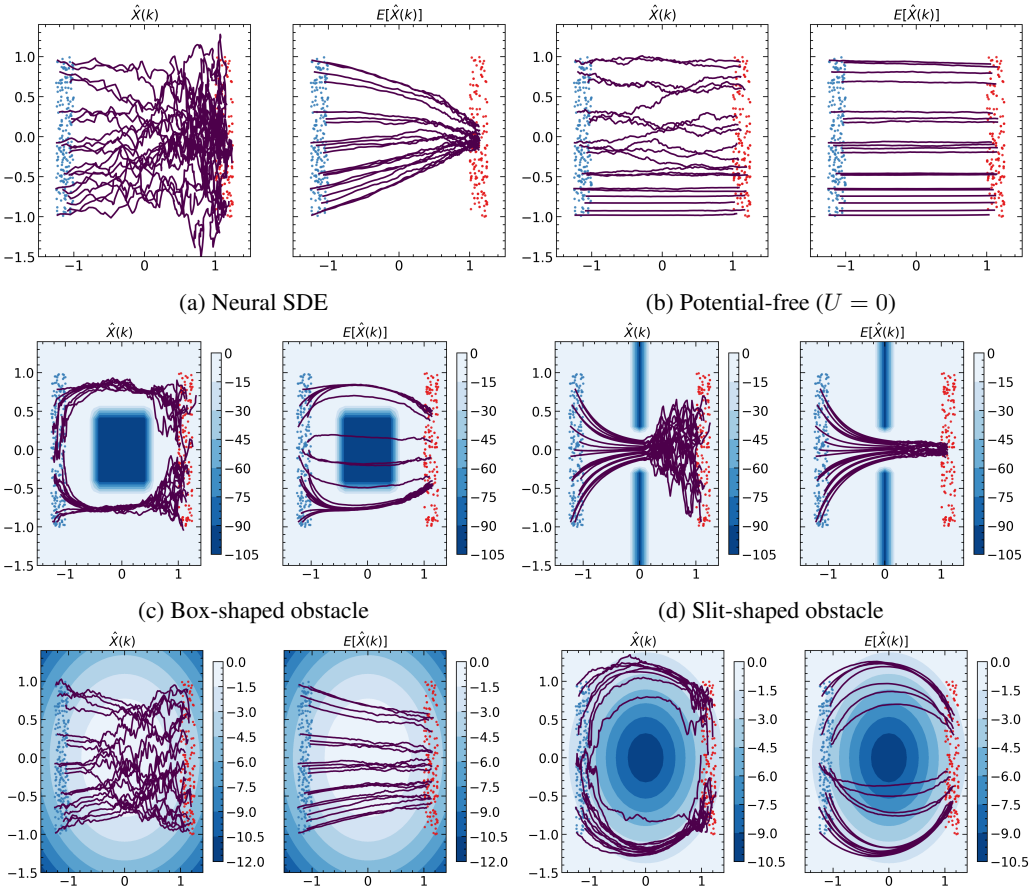
(h) OT-Flow

(i) OT-Flow + OT

Figure 9: scRNA-seq data and predictions. The x- and y-axes denote the first and second principal components, respectively. The five colored point clouds in the background are the ground-truth data given at each time point. All five trajectories are generated by all-step prediction from the initial samples at t_0.

E.3 Synthetic Population Dynamics: Trajectories Reflecting the Potential Function

This section shows practical examples of NLSB with the Lagrangian for the random dynamical system using artificial synthetic data. The detailed experimental setup, including the data and potential functions, is described in Section D.3. The visualization of trajectories is shown in Figure 10. Figure 10 shows that NLSB can estimate various trajectories that reflect information about obstacles or regions where samples cannot or are likely to exist represented by the potential function.



(a) Neural SDE (b) Potential-free ($U = 0$)

(c) Box-shaped obstacle (d) Slit-shaped obstacle

(e) Hill potential (f) Well potential

Figure 10: Visualization of trajectories reflecting the potential function. The color gradients depict the magnitude of the potential function.

E.4 Population Dynamics Simulation by Reverse-time SDE

In this section, we describe a method for estimating the trajectories backwards through time and show experimental results using artificial synthetic data. According to the result from (Anderson, 1982), the reverse-time SDE of the original SDE (Eq. (3)) is given by

$$\mathrm{d}\mathbf{X}_t = \{\mathbf{f}_\theta(\mathbf{X}_t, t) - \mathbf{u}_{\theta,\phi}(\mathbf{X}_t, t)\}\ \mathrm{d}t + \mathbf{g}_\phi(\mathbf{X}_t, t)\ \mathrm{d}\tilde{\mathbf{W}}_t,$$
$$\mathbf{u}_{\theta,\phi}(\mathbf{X}_t, t) = \mathrm{div}\left(\mathbf{g}_\phi(\mathbf{X}_t, t)\mathbf{g}_\phi(\mathbf{X}_t, t)^\top\right) - \mathbf{g}_\phi(\mathbf{X}_t, t)\mathbf{g}_\phi(\mathbf{X}_t, t)^\top \nabla_{\mathbf{x}} \log p_t(\mathbf{X}_t),$$

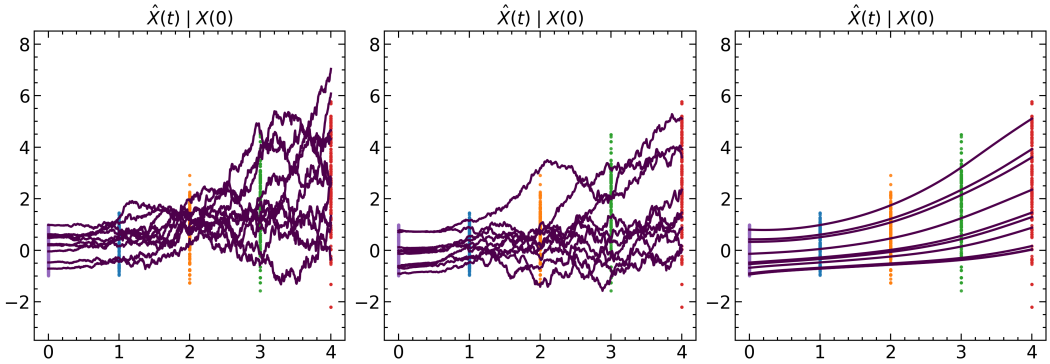
where $\tilde{\mathbf{W}}_t$ is a Wiener process that flows backwards in time.

However, the computation of the modification term $\mathbf{u}$ of the drift function requires an estimation of the score function and high computational cost. Therefore, we newly parameterize the modification term $\mathbf{u}_\xi$ by NN with parameters ξ. Then, we train only the term $\mathbf{u}_\xi$ using Sinkhorn divergence loss in

Eq. (21) by numerically simulating the reverse-time SDE. For simplicity of implementation, we used the same model architecture for modification term $\mathbf{u}_\xi$ as for the drift $\mathbf{f}_\theta$.

$$\tilde{\ell}(\xi) = \sum_{t_k \in T \setminus t_{K-1}} \overline{\mathcal{W}}_\epsilon(\mu_k,\ \rho^{\xi}_{t_k}) \tag{21}$$

Figure 11 shows the visualization of trajectory estimation by reverse-time SDE using artificial data. All trajectories from NLSB and TrajectoryNet were generated by all-step prediction from the initial samples at the time $t = 4$ to $t = 0$.



(a) ground-truth forward SDE (b) NLSB (Ours) (c) TrajectoryNet + OT

Figure 11: 1D OU process data and predictions backwards through time.