Learning to Approximate Particle Smoothing Trajectories via Diffusion Generative Models

Ella Tamir
Aalto University
Espoo, Finland
ella.tamir@aalto.fi

Arno Solin
Aalto University
Espoo, Finland
arno.solin@aalto.fi

***Abstract*—Learning dynamical systems from sparse observations is critical in numerous fields, including biology, finance, and physics. Even if tackling such problems is standard in general information fusion, it remains challenging for contemporary machine learning models, such as diffusion models. We introduce a method that integrates conditional particle filtering with ancestral sampling and diffusion models, enabling the generation of realistic trajectories that align with observed data. Our approach uses a smoother based on iterating a conditional particle filter with ancestral sampling to first generate plausible trajectories matching observed marginals, and learns the corresponding diffusion model. This approach provides both a generative method for high-quality, smoothed trajectories under complex constraints, and an efficient approximation of the particle smoothing distribution for classical tracking problems. We demonstrate the approach in time-series generation and interpolation tasks, including vehicle tracking and single-cell RNA sequencing data.**

I. Introduction

Learning a time-series model based on sparse observations is a problem arising in various practical applications, in fields such as biology, finance, and physics. Past approaches using ordinary differential equations or stochastic differential equations with a neural network as the driving force or drift include for instance [1], [2] and [3]. Trajectory learning approaches for systems of sparse observations have been explored in [4], while a Markov Chain Monte Carlo (MCMC) based particle smoothing approach was studied in [5].

On the other hand, score-based diffusion models [6] for generating complex data have demonstrated state-of-the-art performance for various data modalities, such as images, graphs, and video. Diffusion models pose a generative task as sampling from a simple distribution π_0 (often $\mathcal{N}(\mathbf{0}, \mathbf{I})$), and transforming the samples to an arbitrary data distribution through applying a learned Stochastic Differential Equation (SDE) model until time T—thus obtaining samples from a distribution π_T which can be sampled, but not evaluated. The behaviour of the intermediate SDE system is often not of interest, but merely used as a vehicle for obtaining high-quality samples from π_T. For a setting where even the initial distribution π_0 is allowed to be a complex data distribution, iterative Schrödinger bridge methods such as the Iterative Proportional Fitting Procedure (IPFP) [7] allow for learning the intermediate SDE system, with additional regularization through setting a reference process whose marginals the Schrödinger bridge should match as closely as possible, while adhering to the marginal constraints at π_0 and π_T.

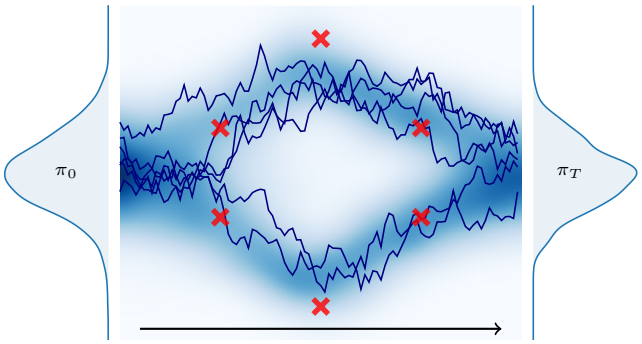


Fig. 1: Trajectories from an SDE with the neural network drift model learned from smoothing trajectories over a zero drift model with the constraints $\pi_0 = \pi_T = \mathcal{N}(0, 1)$, with observations ✖ forcing the trajectories to split into two modes.

We seek to combine the two problem settings, allowing for both sparse observations and complex constraints π_0 and π_T. To that end, we unite diffusion models with particle smoothing methods suited for complex data. The Conditional Particle Filter with Ancestor Sampling (CPF-AS, [8, 9]) combined with MCMC sampling provides an effective scheme for generating trajectories from the smoothing distribution, with theoretical guarantees of convergence. In [10], an MCMC baseline was presented for parameter inference, as a comparison to more scalable variational inference based methods. The MCMC baseline applied CPF-AS in order to effectively evaluate the joint log likelihood over observations and parameters. Inspired by the baseline method, we adopt it to a modern deep learning setting, where in addition to performing inference under complex constraints (both sparse observations and terminal distributions), we learn a model which emulates the behaviour of the inference trajectories, while providing an SDE with an explicitly defined drift function to be used in Euler–Maruyama sampling (see, e.g., [11]).

Recent work by [12] combines a differentiable particle filter with iterative bridge methods, in order to account for both the terminal constraints and sparse observations from the latent marginals. We position our method as a non-iterative version of [12], where the MCMC particle smoother allows us to incorporate data without the need for backwards trajectory

simulation. The key benefit of our approach lies in enabling faster sampling from the particle smoothing distribution after the MCMC chain for CPF-AS has converged via using the learned diffusion model, while providing a neural SDE model.

Our contributions are summarized as follows.

(i) **Inference:** We propose a tailored approach combining Conditional Particle Filtering with Ancestral Sampling to account for multiple observations and terminal constraints.
(ii) **Learning:** We apply learning strategies used for iterative Schrödinger Bridge methods to learn a model matching smoother trajectories, providing a scalable neural approximation of an MCMC particle smoother.
(iii) **Experiments:** We provide strong experimental results for simulated and empirical data including vehicle tracking and single-cell RNA data, both in time-series generation and interpolation tasks.

II. Related work

Conditional Particle Filtering with Ancestral Sampling: The CPF-AS smoother was formulated originally in [9], where it was presented for the purpose of accurate particle smoothing together with derivations of convergence results for the smoother. Obtaining good performance in the Conditional Particle Filter method for trajectory generation is a non-trivial task. It has been extensively studied in recent work [13, 14] to extend the original CPF-AS to various settings such as cases where the initial latent state distribution is a diffusion instead of a single point, or in using bridge backward sampling instead of ancestral sampling. The approach in [15] describes resampling schemes for weak observations and explores dense time slices. In recent applications of CPF-AS, [16] uses modification of CPF-AS with Rao–Blackwellized particle dynamics for a SLAM problem, while [5] explores particle Gibbs with Ancestral Sampling, a conditional Sequential Monte Carlo (SMC) approach similar to CPF-AS for parameter estimation. These sequantial Monte Carlo approaches provide an alternative to variational inference and non-linear filtering (see [17]) approaches often favoured in real-time approaches.

Diffusion Models and Schrödinger Bridges: Diffusion models [6, 18] have been successfully applied to a multitude of generative problem settings. Conditioning diffusion models based on class label was proposed already in [6], but more complex data conditioning has been studied for instance in [19], where 3D view synthesis is conditioned on 2D images. Other conditional settings such as [20] and [21] consider graph generation for molecules, where the motif-scaffolding problem is solved through using an SMC algorithm for conditional generation and guidance of the diffusion.

For an extension of diffusion model approaches to cases where both the initial and terminal distribution are unknown, iterative approaches such as IPFP for solving Schrödinger bridges were proposed in [7] and [22]. IPFP is based on solving half-bridge problems, and has been applied to optimal transport problems for images and biological data.

Diffusion Models, Control, and Data Assimilation: The concept of applying ideas from stochastic control to diffusion model has been studied for instance in [20, 23, 24], while in [25] a deterministic particle filter is used to generate samples from nonlinear dynamical systems under observations. In [12], a particle filter with differential resampling is applied to a constrained generative task, combining the iterative half-bridge approach from [7] with particle filter based data assimilation from [25]. Further connections between Schrödinger bridges and optimal control have been explored in [26] using the forward-backward SDE formulation of Schrödinger bridges, and in [27] and [28] for additional constraints on the controlled system with nonlinear drift. Incorporating information from observations to a known dynamical systems has been explored in the field of data assimilation [29], for instance combining Schrödinger bridges with data assimilation in [30].

III. Methods

Our proposed method learns to approximate the particle smoothing distribution via utilizing samples generated through MCMC over Conditional Particle Filtering with Ancestral Sampling (CPF-AS, [9]). The learning step consists of applying trajectory reversion techniques from the diffusion model and Schrödinger Bridge literature. First, we present our particle smoothing notation in Section III-A, and briefly introduce diffusion models in Section III-B, and to MCMC smoothing using CPF-AS in Section III-C. In Section III-D, we explain how we condition the CPF-AS on multiple observations and a terminal constraint to adapt it to our setting, and in Section III-E combine trajectory learning with smoothing trajectories.

A. Particle Smoothing and Notation

We consider the state-space model

$$\begin{aligned} &\mathrm{d}\mathbf{x}_t = f(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)^2\,\mathrm{d}\boldsymbol{\beta}_t, \\ &\mathbf{x}_0 \sim \pi_0, \\ &\mathbf{y}_{t_k} \sim \mathcal{N}(\mathbf{H}\mathbf{x}_{t_k}, \boldsymbol{\Sigma}), \quad \forall k \in \{1, \dots, K\}, \end{aligned}$$

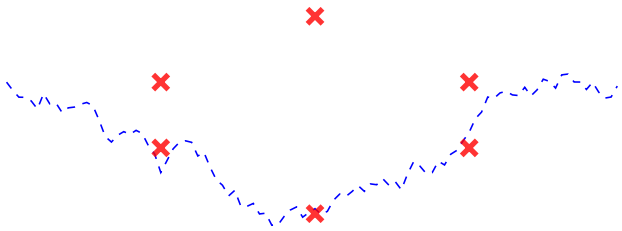
where $t \in [0, T]$ is the time horizon, π_0 is a bounded initial distribution, $\{t_k\}_{k=1}^K$ are observation times, and for each k, $\mathbf{y}_{t_k} \in \mathbb{R}^{d_y}$ and $\forall t \in [0, T]$, $\mathbf{x}_t \in \mathbb{R}^{d_x}$, and $\mathbf{H} \in \mathbb{R}^{d_x \times d_y}$. In other words, we have restricted to SDEs with nonlinear dynamics and linear Gaussian observation models, and are interested in the continuous-discrete setting. In practice, we will resort to Euler–Maruyama simulations of the SDE.

We find the conditional state distribution $p(\mathbf{x}_t \mid \{\mathbf{y}_{t_k}\}_{k=1}^K)$ through simulating particle trajectories through a particle filter. We denote by w_t^j the particle weight of the j^{th} particle at time t and by $\mathbf{x}_t^j$ its corresponding particle, and generate N particles.

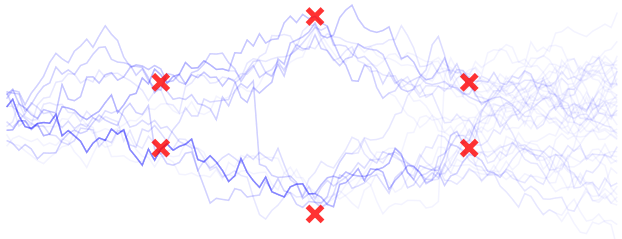
B. Diffusion Models and Schrödinger Bridges

Given two distributions, π_0 and π_T we seek to find the drift functions $f_\theta, b_\phi : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ such that the measure $\mathbb{P}$ defined by the marginals of the forward diffusion

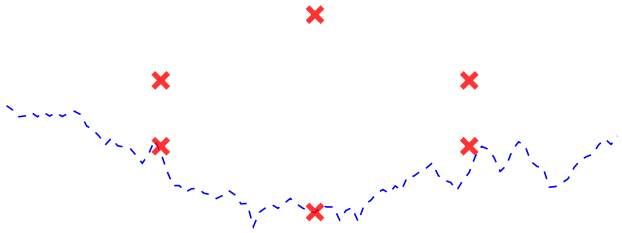
$$\mathrm{d}\mathbf{x}_t = f_\theta(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\beta}_t \quad \mathbf{x}_0 \sim \pi_0, \tag{1}$$



(a) 10th reference trajectory



(b) Particle trajectories from the CPF-AS using the 10th reference trajectory



(c) 11th reference trajectory

Fig. 2: In an iteration of the CPF-AS smoother, the reference trajectory (top) is used to create conditional particle filtering trajectories (middle), from which a new reference trajectory is sampled based on weights on the last time step (bottom). The final weights are computed based on a Kernel Density Estimate (KDE) over the distribution $\pi_T = \mathcal{N}(0, 1)$. The trajectories are sampled based on intermediate observations and the reference trajectory, resulting in multiple trajectories in the middle plot following roughly the previous reference.

satisfies $\mathbb{P}_T = \pi_T$, and the backward diffusion

$$\mathrm{d}\mathbf{x}_t = b_\phi(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\beta}_t \quad \mathbf{x}_0 \sim \pi_T \tag{2}$$

satisfies $\mathbb{P}_0 = \pi_0$. Denote the density function of the marginal distribution of $\mathbb{P}$ at time t by $p_t(\mathbf{x})$. In diffusion models, the initial distribution π_T is set to $\mathcal{N}(0, I)$ and the model b_ϕ is fixed to a linear function, noising the data distribution π_T. The reverse diffusion f_θ from noise to data is equal to

$$f_\theta(\mathbf{x}_t, t) = b_\phi(\mathbf{x}_t, t) - g(t)^2 \nabla \ln p_t(\mathbf{x}_t, t), \tag{3}$$

that is, we can learn to generate data from noise (or π_0 from π_T) through reversing the dynamics via learning to approximate the score function $\nabla \ln p_t(\mathbf{x}_t, t)$ based on the trajectories. More generally, for Schrödinger bridges [31, 32], we define a reference measure $\mathbb{Q}$ on $\mathbb{R}^d \times [0, T]$, and seek to minimize the

Algorithm 1 Generating the first reference trajectory

Require: Initial distribution π_0, observations $\mathcal{D} = \{(t_k, \mathbf{y}_k)\}_{k=1}^k$, number of particles N, discretization $\{t_j\}_{j=0}^{N_T}$
Ensure: Reference trajectory $\mathbf{z}_{0:T}$
  Sample initial particles $\{\mathbf{x}_0^i\}_{i=1}^N \sim \pi_0$ with weights $w_0^i = \frac{1}{N}$
  for $j = 0$ **to** N_T **do**
    for $i = 1$ **to** N **do**
      $a_i \sim$ **Categ**$(\{i\}_{i=1}^N, \mathbf{w}_j)$ ▷ Draw ancestor
      $\mathbf{x}_{t_j}^i = \mathbf{x}_{t_{j-1}}^{a_i} + f(\mathbf{x}_{t_{j-1}}^{a_i}, t)\,\mathrm{d}t + \sigma\epsilon$ ▷ $\epsilon \sim \mathcal{N}(0,\, \mathrm{d}t)$
      $\mathbf{x}_{0:t_j}^i \leftarrow \{\mathbf{x}_{0:t_{j-1}}^{a_i}, \mathbf{x}_{t_j}^i\}$ ▷ Append trajectories
      $w_{t_j}^i \leftarrow p(\mathbf{y}_{t_j} \,|\, \mathbf{x}_{0:t_{j-1}}^i, \mathbf{y}_{0:t_j})$ ▷ Compute weights
    end for
  end for
  $\mathbf{z}_{0:T} \sim$ **Categ**$(\{\mathbf{x}_{0:T}^i\}_{i=1}^N, \mathbf{w}_T)$ ▷ Sample reference

Kullback–Leibler divergence $\mathrm{KL}(\mathbb{P}, \mathbb{Q})$. The reference measure may be used to encode prior information, for instance it may be defined through simulation of an SDE with drift f_0, as in [7]. In order to solve the Kullback–Leibler minimization problem while adhering to the terminal constraints of matching the distributions π_0 and π_T, [7] applies an iterative half-bridge method [33], alternating between the minimization targets

$$\min_{\mathbb{P}_{2i} \in \mathbb{P}(\cdot, \pi_T)} \mathrm{KL}(\mathbb{P}_{2i} \,|\, \mathbb{P}_{2i-1}) \quad \text{and} \quad \min_{\mathbb{P}_{2i+1} \in \mathbb{P}(\pi_0, \cdot)} \mathrm{KL}(\mathbb{P}_{2i+1} \,|\, \mathbb{P}_{2i}), \tag{4}$$

where we denote by $\mathbb{P}(\cdot, \pi_T)$ measures which match π_T at time T, and by $\mathbb{P}(\pi_0, \cdot)$ measures which match π_0 at time 0. For each of the minimization targets in Eq. (4), the learning step consists of reversing trajectory dynamics via a mean matching objective. Suppose that the backwards drift b_ϕ is fixed, and we learn the forward drift f_θ. Then the loss function is set to

$$\mathcal{L}(\phi) = \sum_{i=1}^{N} \sum_{j=1}^{N_T} \|b_\phi(\mathbf{x}_t, t)\Delta_j - (\mathbf{x}_{t_{j+1}}^i + f_\theta(\mathbf{x}_{t_{j+1}}^i, t_j)\Delta_j - \mathbf{x}_{t_j}^i - f_\theta(\mathbf{x}_{t_j}^i, t_j)\Delta_j)\|^2, \tag{5}$$

where $\{\mathbf{x}_t^i\}_{i=1}^N$ are the simulated particles from the SDE in Eq. (2) and Δ_j denotes the time step length at t_j, and N_T is the number of steps in the discretization. In other words, the forward drift should match the change in mean set by the backward model. Instead of directly comparing subsequent particles $\mathbf{x}_{t_j}^i$ to $\mathbf{x}_{t_{j-1}}^i$, the loss function considers the Gaussian distribution $\mathcal{N}(\mathbf{x}_{t_{j-1}}^i + f_\theta(\mathbf{x}_{t_{j-1}}^i, t_{j-1}), \Delta_k g(t_j)^2 \mathbf{I})$ from which $\mathbf{x}_{t_j}^i$ is sampled.

C. Conditional Particle Filter with Ancestral Sampling and Smoothing

We present the details of an MCMC smoother using Conditional Particle Filter with Ancestral Sampling (CPF-AS), as defined in [9]. Instead of a forward–backward particle smoother where the particle filtering weights from the forward pass between $t = 0$ to T are used when accruing information from T to 0, we will generate particle smoothing trajectories via MCMC steps consisting of the CPF-AS procedure. In a

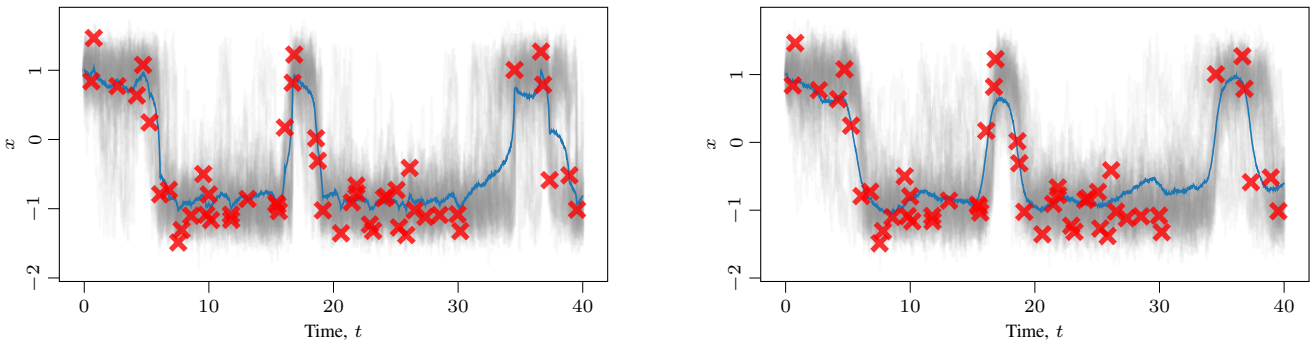


Fig. 3: (a) Samples from the particle smoothing distribution using the known drift of the double-well system. (b) Trajectories generated from the learned diffusion process. The mean (in blue) and the trajectory samples (in gray) follow similar patterns, and the learned SDE roughly follows the observations without access to them at sampling time.

single run of the CPF-AS (see Algorithm 2), the trajectories are conditioned on both the observations and a reference trajectory $\{\mathbf{z}_{t_j}\}_{j=0}^{N_T}$. At each time step, we draw an ancestor for each trajectory. For particle indices in $i \in [1, N-1]$, the probability of assigning ancestor a^i are proportional to the trajectory weights $w_{t_j}^i$. The N^{th} particle retains information from the reference trajectory, and the probability of assigning the ancestor a^i to the reference trajectory is proportional to $w_{t_j}^{a^i} f(\mathbf{z}_{t_j} \,|\, \mathbf{x}_{t_{j-1}}^{a^i})$, that is, the probability that the reference trajectory sample $\mathbf{z}_t$ was generated from ancestor a^i. After generating N particles via CPF-AS, a new reference trajectory $\{\mathbf{z}_{t_j}\}_{j=0}^{N_T}$ is sampled from the weights at the terminal time, proportional to $\{w_T^i\}_{i=1}^N$. The MCMC chain is initialized as in Algorithm 1, with a particle filtering algorithm with ancestral sampling, but no reference trajectory.

The CPF-AS [8, 9] combined with MCMC provides an algorithm for generating trajectories from the smoothing distribution. The iterative algorithm consists of creating CPF-AS trajectories, each time conditioning on the previous reference trajectory to obtain a new reference, repeated for M iterations. In [9], an analysis of the convergence rate of the MCMC to a stationary distribution is provided, while [8] introduces the Conditional Particle Filtering step of the algorithm.

Algorithm 2 Conditional Particle Filter with Ancestral Sampling (CPF-AS)

Require: Initial distribution π_0, observations $\mathcal{D} = \{(t_k, \mathbf{y}_k)\}_{k=1}^k$, number of filtering particles N, initial trajectory $\mathbf{z}_{0:T}$, discretization $\{t_j\}_{j=0}^{N_T}$.
Ensure: New reference trajectory $\hat{\mathbf{z}}_{0:T}$
  Sample initial particles $\{\mathbf{x}_0^i\}_{i=1}^N \sim \pi_0$ with weights $w_0^i = \frac{1}{N}$
  for $j = 0$ **to** N_T **do**
    $a_N \sim$ **Categ**$(\{i\}_{i=1}^N, \{w_{t_j}^i p(\mathbf{z}_{t_j} \,|\, \mathbf{x}_{t_{j-1}}^i)\}_{i=1}^N)$ ▷ Ancestor
    for $i = 1$ **to** $N-1$ **do**
      $a_i \sim$ **Categ**$(\{i\}_{i=1}^N, \mathbf{w}_{t_j})$
      $\mathbf{x}_{t_j}^i = \mathbf{x}_{t_{j-1}}^{a_i} + f(\mathbf{x}_{t_{j-1}}^{a_i}, t)\,\mathrm{d}t + \sigma\epsilon$ ▷ $\epsilon \sim \mathcal{N}(0, \mathrm{d}t)$
    end for
    for $i = 1$ to N **do**
      $\mathbf{x}_{0:t_j}^i \leftarrow \{\mathbf{x}_{0:t_{j-1}}^{a_i}, \mathbf{x}_{t_j}^i\}$ ▷ Append trajectories
      $w_{t_j}^i \leftarrow p(\mathbf{y}_{t_j} \,|\, \mathbf{x}_{0:t_{j-1}}^i, \mathbf{y}_{0:t_j})$ ▷ Compute weights
    end for
  end for
  $j \sim$ **Categ**$(\{i\}_{i=1}^N, \mathbf{w}_T)$ ▷ Sample reference
  $\hat{\mathbf{z}}_{0:T} = \mathbf{x}_{0:T}^j$ ▷ Set reference trajectory

D. Efficient CPF-AS Smoothing Over a Diffusion Model

In this section, we explain how we perform MCMC sampling over CPF-AS when multiple observations are available, and how we utilize parallelization under a setting with multi-modal observations. The extension of CPF-AS to multiple observations enables the use of the smoother for generating trajectories such that the marginal distribution matches a full observed data distribution, such as single-cell data in Section IV-C, instead of a single realization of a stochastic process. As explained in Section III-C, the CPF-AS particle smoother generates only a single trajectory sample at each iteration. In order to reduce the heavy computational load of the particle smoother, we account for the possibly multi-modal marginals by splitting the MCMC computation to multiple, parallelizable chains.

Adapted Observation Model: In order to incorporate data from a partially observed marginal, we compute the weights of each trajectory according to a bootstrap filter proposal combined with local kernel density metric over the H nearest observations to each trajectory. We may compute the log-weights as

$$\log w_{t_j}^i = -\frac{1}{2\sigma_{\text{obs}}^2} \sum_{h=1}^{H} \|(\mathbf{y}_{t_j,h} - \mathbf{x}_{t_j}^i)\|^2, \tag{6}$$

where $\mathbf{y}_{t_j,h}$ is the h^{th} nearest observation to the particle $\mathbf{x}_{t_j}^i$. In practice, we apply the observation model $p(\mathbf{y}_{t_j} \,|\, \mathbf{x}_{t_j}) = \mathcal{N}(\mathbf{x}_{t_j}, \sigma_{\text{obs}}^2)$. In addition, we interpret the terminal distribution π_T to consist of observations, so that when generating particle filtering trajectories, we observe a set of samples of size S, $\{\mathbf{y}_s\}_{s=1}^S$ where $\mathbf{y}_s \sim \pi_T$. We may further adjust the importance of adhering to the desired terminal distribution by letting the observation model depend on time, for instance via setting σ_{obs}^2 to be lower at $t = T$, or by adjusting the local kernel density estimate through modifying H.

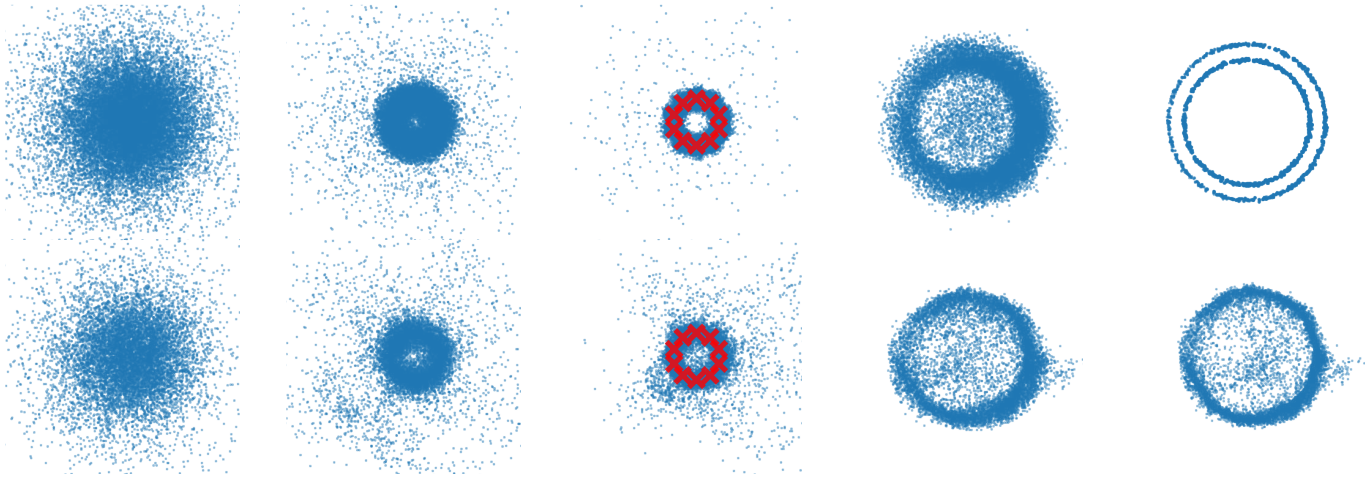


Fig. 4: Particle smoother marginals (top row) and marginals of the neural SDE model (bottom row), generated from a constrained system where the final distribution is the scikit-learn two-circles data set, and intermediate data consists of 10 points lying on a circle with a smaller radius (in red). The marginals above are observed at times $t = \{0, 1.4, 1.5, 2.9, T\}$, where $T = 3$. Without access to samples from the terminal distribution, the learned model accomplishes modeling a circle with the correct radius, but struggles to separate the two circles from each other.

Parallelization: While the CPF-AS based smoother provides accurate trajectories matching the observations, it is computationally inefficient. In order to better utilize GPU computational resources, we generate multiple MCMC chains using the CPF-AS by running multiple parallel chains for problem settings where the chains converge quickly, and the need for a large sample of trajectories arises from building a training set for the learning step detailed in Section III-E that is sufficiently representative. Sampling from multiple chains is further motivated by our setting, since we may observe a multimodal marginal distribution and exploration of all the modes through a single MCMC chain could prove inefficient.

E. Learning Over CPF-AS Trajectories and Sampling from the Neural SDE

Our learning scheme consists of using the smooth trajectories obtained in the previous step to learn a dynamical system. While generating the CPF-AS smoother trajectories, we store further information to be used in the trajectory learning step. That is, while sampling from the bootstrap particle filtering proposal, we store the change in mean for each trajectory at time t_j,

$$\mathbf{x}_{t_j}^{i,\text{diff}} = (\mathbf{x}_{t_{j+1}}^{i} + f(\mathbf{x}_{t_{j+1}}^{i}, t_j)\Delta_j - \mathbf{x}_{t_j}^{i} - f(\mathbf{x}_{t_j}^{i}, t_j)\Delta_j, \quad (7)$$

where $\Delta_j = t_{j+1} - t_j$. After each ancestor sampling step, we update the mean-matching trajectory history to align with the ancestors, overwriting the mean-matching history of trajectory i with its ancestor's a^i history. At the last step, when a new reference trajectory is sampled, we choose the mean-matching target trajectory accordingly, generating $\{\mathbf{z}_{t_j}^{\text{diff}}\}_{j=1}^{N_T}$. We find the neural network drift f_θ such that the following loss is minimized for a particle i at time t_j,

$$\mathcal{L}(\theta, i, j) = (f_\theta(\mathbf{x}_{t_j}^{i}, t_j)\Delta_j - \mathbf{z}_{t_j}^{\text{diff}})^2, \quad (8)$$

to learn to reverse the mean at each time-step. In practice, we apply stochastic gradient descent to a neural network f_θ over batches consisting of samples from the pool of particle values $\{\mathbf{x}_{t_j}^{i}\}_{i=1,j=1}^{N,N_T}$. After the learning step, we have access to a neural network drift f_θ such that the trajectories generated through Euler–Maruyama sampling of the SDE

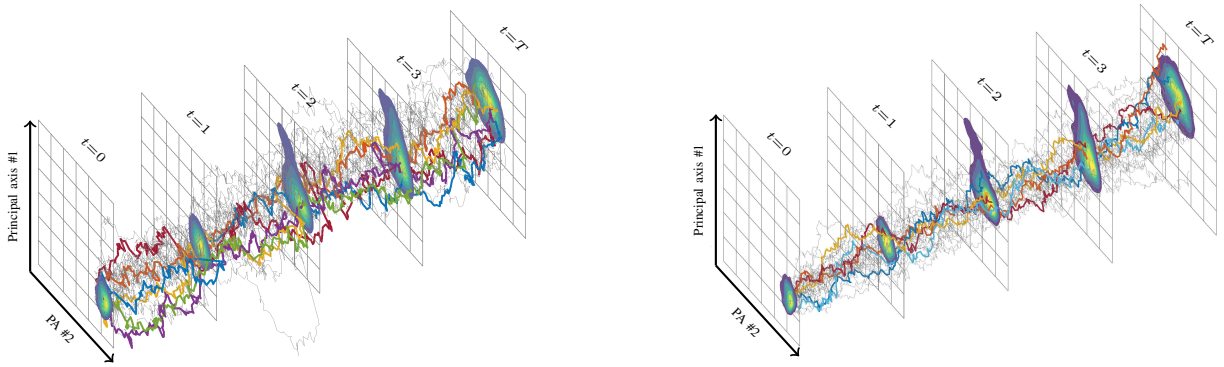
$$\mathrm{d}\mathbf{x}_t = f_\theta(\mathbf{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\boldsymbol{\beta}_t, \quad \mathbf{x}_0 \sim \pi_0, \quad (9)$$

closely match the smoothing trajectories, and provide an approximation to its marginals without access to the observations $\{\mathbf{y}_{t_k}\}_{k=1}^{K}$ or samples from π_T. As such, the learned diffusion provides a scalable proxy to the CPF-AS smoother trajectories.

IV. Experiments

We apply the CPF-AS smoother to various settings, consisting of scenarios where the observations are from a single underlying realization of a stochastic process (Section IV-A Section IV-D), or a full marginal distribution (Section IV-B, Section IV-C) in the middle of a transport problem. After generating the smoother trajectories, we learn the drift of a dynamical system as a neural network and compare the behaviour of the raw smoothed trajectories to the learned dynamical system. In addition to learning a constrained generative model for time-series data, we demonstrate the capability of our method to interpolate in a setting where the sparsity of observations is artificial: we let our model utilize only every S^{th} data point, and compare the learned trajectories to the full set of observations in the Section IV-D interpolation experiments.

In all our experiments, we fix the observation model to $p(\mathbf{y}_t \,|\, \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t, \sigma_{\text{obs}}^2\mathbf{I})$, allowing for scaling the observation noise at $t = T$ as explained in Section III-D. We implement both the particle smoothing and trajectory reversal code in PyTorch, and utilize GPU computation to parallelize the computation over trajectories. The neural network design is as in [7, 12], with four hidden layers, largest of which is 128 in width, and using a sinusoidal embedding for the time dimension.



(a) Our CPF-AS based model

(b) Iterative Smoothing Bridge solution

Fig. 5: The CPF-AS MCMC smoother trajectories cover the known marginals of the single-cell process (projected onto the first two principal axes for visualization on 2D planes), compared to the Iterative Smoothing Bridge which only explores high-density regions of the marginals.

A. Double-well

We perform particle smoothing over a double-well system with 50 observations. The true drift f to be used within the smoothing algorithm is $f(x_t, t) = 4x_t(1 - x_t^2)$, the diffusion is $g(t) = 1$, and we set $\Delta_t = 0.01$ and $T = 40$. We iterate the CPF-AS particle filtering algorithm for 1000 steps with 100 particles used in the filter, and evaluate the performance of our method by subsequently learning the neural drift f_θ. For the learning step, we optimize over 200 epochs with a learning rate of 10^{-4} and a batch size of 2048, and use the same process noise of $g(t) = 1$ during sampling. We use only the last 500 reference trajectories from the smoother. The learned process manages to oscillate between the two modes of the double-well distribution matching the oscillation patterns of the particle smoother trajectories, see Fig. 3 for a comparison.

B. 2D Scikit-learn Two-circles

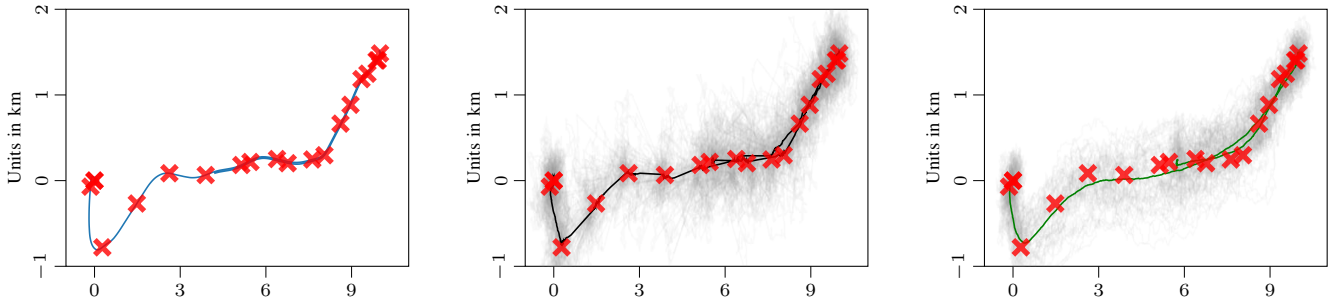
We perform particle smoothing and trajectory learning over a constructed data set, where the final data distribution consists of samples from the scikit-learn two circles data set, and at the middle of the process, 10 points on a single circle are observed, and the initial distribution is a Gaussian. The purpose of this experiment is to demonstrate how CPF-AS can be used to condition on complex observation sets, and how to learn to match even such restricted generative processes. We set $\Delta_t = 0.01$ and let $T = 3$, and simulate from 10 MCMC chains over 2000 iterations of CPF-AS, each using 1000 particles. For the training step, we set the learning rate to 10^{-4} and the batch size to 1024 over 200 epochs. Instead of a time-invariant $g(t)$, we use a noise schedule which allows both for exploration of the state-space and is suitable for generating a sharp approximation of the scikit-learn two circles. To that end, we use a noise schedule where $g(t) = 5$ at $t \in [0, T/2]$, and $g(t)$ decreases linearly to 0.01 at $t = T$. We set the observation noise $\sigma_{\text{obs}} = 0.5$ in the middle observations, scaled by a factor of 0.01 at the scikit-learn two circles observations, and consider the $H = 5$ nearest observations when computing the particle filtering weights at the terminal time, and $H = 3$ at the middle. The particle smoother is able to generate trajectories which smoothly transform a Gaussian distribution first to 10 points on a circle, and then to the scikit-learn data set. See Fig. 4 for selected marginals from the smoothing process and the learned diffusion model.

C. Single-cell RNA Sequencing Data

We apply the CPF-AS smoother to a single-cell RNA data set, which we preprocessed and projected into 5 dimensions using Principal Component Analysis (PCA) as in [34]. Each observation represents active RNA sequence counts from an embryonic cell, which is destroyed by the measurement process, resulting in a time-series data set without any trajectory information. Instead, the observed data consists of a large number of samples from an intermediate marginal distribution at 5 points in time. We evaluate only the performance of the particle smoother in generating the marginal observations, as it accomplishes a clear improvement compared to earlier methods. As in earlier work [22, 34], we set the process noise to $g(t) = 1$ and discretize in time to $T = 4$, $\Delta_t = 0.01$. For the smoother hyperparameters, we run 8 chains of length 500, discarding the first half of each chain. The observation noise is set to a constant 0.3, and the nearest 5 points are used to compute the log-weights. The CPF-AS smoother trajectories replicate the marginal distributions faithfully, see Table I for a comparison to earlier work using methods based on optimal transport such as [34] or iterative Schrödinger bridge approaches based on a reference process defined by data [22] or using a particle filter over a diffusion model [12]. Fig. 5 shows how the CPF-AS MCMC smoother trajectories cover the known marginals of the single-cell process, compared to the Iterative Smoothing Bridge which only explores high-density regions of the marginals.

D. Vehicle Tracking

We model a 2D vehicle tracking system (a segment of data from [35]), where the observations give the location of a single



(a) True vehicle tracking trajectory

(b) Particle smoother trajectories

(c) Euler–Maruyama trajectories from the learned SDE model

Fig. 6: A comparison of the vehicle trajectories from the data, and generated trajectories using CPF-AS particle smoothing, and learned diffusion model trajectories. The trajectory mean (bolded) follows the observations (marked by ✖) in the smoother and simulated trajectories from the neural SDE, and the sampled trajectories (light gray) present similar behaviour in both plots.

TABLE I: Conditional smoothing trajectories generate the lowest Earth Mover's Distance in the intermediate stages.

	Earth mover's distance				
METHOD	$t=0$	$t=1$	$t=2$	$t=3$	$t=T$
TrajectoryNet	0.62	1.15	1.49	1.26	0.99
IPML	**0.34**	1.13	1.35	1.01	**0.49**
IPFP (no obs)	0.57	1.53	1.86	1.32	0.85
ISB (single-cell obs)	0.57	1.04	1.24	0.94	0.83
CPF-AS Smoothing	–	**0.85**	**0.93**	**0.66**	0.94

vehicle at 1000 points in time. We sparsify the observations by allowing our model to use only every 50$^{\text{th}}$ data point, with the goal of generating plausible and smooth trajectories between the observations, while comparing to the real tracking data. The observation noise of this state-space system is set to $\sigma^2_{\text{obs}} = 0.01$, the drift f to zero, and the process noise to $g(t) = 0.1$, with the process starting from a single point, $\mathbf{y} = (0, 0)$. We set the time step to $\Delta_t = 0.01$. In order to obtain particle smoothing trajectories using CPF-AS, we ran a single MCMC chain of length 400, and discarded the first half as burn-in. In each iteration, we ran the CPF-AS filter using 1000 particles. When optimizing the neural drift in the learning step, we applied a learning rate of 10^{-4} over 300 epochs of the training data set consisting of samples from the particle smoothing trajectories, with a batch size of 2048. The mean squared error (MSE) of the mean smoothing trajectory compared to the true observations was 0.215, while the MSE of the mean over learned SDE trajectories was 0.1625. In Fig. 6 we compare the behaviour of the full trajectories to the means of particle smoothing trajectories and learned SDE trajectories. The latter remains a plausible approximation to the observations, without access to the sparse observations at sampling time.

V. Discussion and Conclusions

The MCMC smoother iteratively applying the Conditional Particle Filter with Ancestral Sampling (CPF-AS) provides an accurate particle smoothing distribution, but is slow to generate new trajectories even after convergence. We have presented a new approach assimilating information from sparse observations or full marginals to the CPF-AS particle filter, including a learning step to find a neural network drift to approximate the behaviour of the smoother. The learned neural diffusion can be used to simulate trajectories under complex constraints on both the intermediate marginal distributions and the terminal distribution π_T, and it allows for an efficient approximation of the MCMC based CPF-AS smoothing trajectories. Further, our method allows for studying the system behaviour due to the analytic expression provided via the learned drift function, allowing for instance to restart the system from an intermediate point in time or to discard the observational data yet still generate novel trajectories from an approximation to the particle smoothing trajectories.

For future work, we propose additional guidance to the MCMC smoother via learned diffusion, where a guided drift could aid the exploration of the CPF-AS trajectories, speeding up convergence. In order to assure that the full terminal distribution is explored efficiently, in the experiments in Section IV we often set the process noise $g(t)$ to a high level, especially when conditioning on observations from a complex, high-variance marginal compared to more learning-based approaches such as [12]. An adaptive drift function used within CPF-AS could plausibly be used to replace the high noise level, and to create easier to learn trajectories due to their less noisy nature. As demonstrated by our experiments, the learned SDE manages to mimic the behaviour of the particle smoothing trajectories well in settings related to learning time series, but does not generate a sharp enough replica of difficult terminal constraints, such as the scikit-learn two circles data set. Combining the trajectory learning step over smoothing trajectories with information from samples of the terminal distribution at training time could further improve the quality of the generated data under such challenging constraints.

Acknowledgements and Disclosure of Funding

Authors acknowledge funding from the Academy of Finland (grant 339730), and the computational resources provided by the CSC – IT Center for Science, Finland. We wish to thank Prakhar Verma for his helpful advice on the code of [10].

References

[1] Y. Rubanova, R. T. Q. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates, Inc., 2019, pp. 5320–5330.

[2] C. Yildiz, M. Heinonen, and H. Lahdesmaki, "ODE2VAE: Deep generative second order ODEs with Bayesian neural networks," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates, Inc., 2019, pp. 13 412–13 421.

[3] X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 108. PMLR, 2020, pp. 3870–3882.

[4] H. Lavenant, S. Zhang, Y.-H. Kim, and G. Schiebinger, "Towards a mathematical theory of trajectory inference," *The Annals of Applied Probability*, vol. 34, no. 1A, pp. 428 – 500, 2024.

[5] A. Wigren, R. S. Risuleo, L. Murray, and F. Lindsten, "Parameter elimination in particle Gibbs sampling," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates, Inc., 2019, pp. 8918–8929.

[6] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations (ICLR)*, 2021.

[7] V. De Bortoli, J. Thornton, J. Heng, and A. Doucet, "Diffusion Schrödinger bridge with applications to score-based generative modeling," in *Advances in Neural Information Processing Systems 34 (NeurIPS)*. Curran Associates, Inc., 2021, pp. 17 695–17 709.

[8] F. Lindsten, M. I. Jordan, and T. B. Schön, "Particle Gibbs with ancestor sampling," *Journal of Machine Learning Research*, vol. 15, no. 63, pp. 2145–2184, 2014.

[9] A. Svensson, T. B. Schön, and M. Kok, "Nonlinear state space smoothing using the conditional particle filter," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 975–980, 2015, 17th IFAC Symposium on System Identification (SYSID).

[10] P. Verma, V. Adam, and A. Solin, "Variational Gaussian process diffusion processes," in *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS), to appear*, ser. Proceedings of Machine Learning Research. PMLR, 2024.

[11] S. Särkkä and A. Solin, *Applied stochastic differential equations*. Cambridge University Press, 2019.

[12] E. Tamir, M. Trapp, and A. Solin, "Transport with support: Data-conditional diffusion bridges," *Transactions on Machine Learning Research*, 2023.

[13] S. Karppinen and M. Vihola, "Conditional particle filters with diffuse initial distributions," *Statistics and Computing*, vol. 31, 05 2021.

[14] S. Karppinen, S. S. Singh, and M. Vihola, "Conditional particle filters with bridge backward sampling," *Journal of Computational and Graphical Statistics*, pp. 1–15, 2023.

[15] N. Chopin, S. S. Singh, T. Soto, and M. Vihola, "On resampling schemes for particle filters with weakly informative observations," *The Annals of Statistics*, vol. 50, no. 6, pp. 3197–3222, 2022.

[16] M. Kok, A. Solin, and T. B. Schön, "Rao-Blackwellized particle smoothing for simultaneous localization and mapping," *arXiv preprint arXiv:2306.03953*, 2023.

[17] A. Solin, E. Tamir, and P. Verma, "Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation," in *Advances in Neural Information Processing Systems 34 (NeurIPS)*. Curran Associates, Inc., 2021, pp. 417–429.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Curran Associates, Inc., 2020, pp. 6840–6851.

[19] D. Watson, W. Chan, R. M. Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi, "Novel view synthesis with diffusion models," in *International Conference on Learning Representations (ICLR)*, 2023.

[20] G. Cardoso, Y. J. E. Idrissi, S. L. Corff, and E. Moulines, "Monte Carlo guided diffusion for Bayesian linear inverse problems," in *International Conference on Learning Representations (ICLR), to appear*, 2024.

[21] B. L. Trippe, J. Yim, D. Tischer, D. Baker, T. Broderick, R. Barzilay, and T. S. Jaakkola, "Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem," in *International Conference on Learning Representations (ICLR)*, 2023.

[22] F. Vargas, P. Thodoroff, A. Lamacraft, and N. D. Lawrence, "Solving Schrödinger bridges via maximum likelihood," *Entropy*, vol. 23, no. 9, p. 1134, 2021.

[23] J. Berner, L. Richter, and K. Ullrich, "An optimal control perspective on diffusion-based generative modeling," in *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

[24] T. Chen, J. Gu, L. Dinh, E. A. Theodorou, J. Susskind, and S. Zhai, "Generative modeling with phase stochastic bridges," in *International Conference on Learning Representations (ICLR), to appear*, 2024.

[25] D. Maoutsa and M. Opper, "Deterministic particle flows for constraining stochastic nonlinear systems," *Physical Review Research*, vol. 4, p. 043035, Oct 2022.

[26] T. Chen, G.-H. Liu, and E. Theodorou, "Likelihood training of Schrödinger bridge using forward-backward SDEs theory," in *International Conference on Learning Representations (ICLR)*, 2022.

[27] K. F. Caluya and A. Halder, "Reflected Schrödinger bridge: Density control with path constraints," *2021 American Control Conference (ACC)*, pp. 1137–1142, 2020.

[28] ——, "Wasserstein proximal algorithms for the Schrödinger bridge problem: Density control with nonlinear drift," *IEEE Transactions on Automatic Control*, vol. 67, pp. 1163–1178, 2019.

[29] M. Asch, M. Bocquet, and M. Nodet, *Data Assimilation*. Society for Industrial and Applied Mathematics, 2016.

[30] S. Reich, "Data assimilation: The Schrödinger perspective," *Acta Numerica*, vol. 28, p. 635–711, 2019.

[31] E. Schrödinger, "Sur la théorie relativiste de l'électron et l'interprétation de la mécanique quantique," *Annales de l'institut Henri Poincaré*, vol. 2, no. 4, pp. 269–310, 1932.

[32] C. Léonard, "A survey of the Schrödinger problem and some of its connections with optimal transport," *Discrete & Continuous Dynamical Systems*, vol. 34, no. 4, pp. 1533–1574, 2014.

[33] L. Ruschendorf, "Convergence of the iterative proportional fitting procedure," *The Annals of Statistics*, vol. 23, no. 4, pp. 1160–1174, 1995.

[34] A. Tong, J. Huang, G. Wolf, D. Van Dijk, and S. Krishnaswamy, "TrajectoryNet: A dynamic optimal transport network for modeling cellular dynamics," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 9526–9536.

[35] A. Solin and S. Särkkä, "State space methods for efficient inference in Student-t process regression," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 38. PMLR, 2015, pp. 885–893.