Score matching for bridges without learning time-reversals

Elizabeth L. Baker
Dept. of Computer Science
University of Copenhagen

Moritz Schauer
Dept. of Mathematical Sciences
Chalmers University of Technology
and University of Gothenburg

Stefan Sommer
Dept. of Computer Science
University of Copenhagen

Abstract

We propose a new algorithm for learning bridged diffusion processes using score-matching methods. Our method relies on reversing the dynamics of the forward process and using this to learn a score function, which, via Doob's h-transform, yields a bridged diffusion process; that is, a process conditioned on an endpoint. In contrast to prior methods, we learn the score term $\nabla_x \log p(t, x; T, y)$ directly, for given t, y, completely avoiding first learning a time-reversal. We compare the performance of our algorithm with existing methods and see that it outperforms using the (learned) time-reversals to learn the score term. The code can be found at `https://github.com/libbylbaker/forward_bridge`.

1 INTRODUCTION

Given a diffusion process, we consider conditioning its endpoint to take a specific value or distribution. In general, it can be shown by Doob's h-transform that a conditioned diffusion process can also be written as a stochastic differential equation (SDE), containing the score term $\nabla_x \log p(t, x; T, y)$, where p is the transition density of the unconditioned SDE. The problem is, this term is in general intractable, since there is no closed form for the transition density. Finding alternative methods to sample from the bridge process has received considerable attention in the past (Delyon and Hu, 2006; Papaspiliopoulos and Roberts, 2012; Bladt and Sørensen, 2014; Schauer et al., 2017; Heng et al., 2022). Diffusion processes have a multitude of applications across numerous areas, where observations need to be included, leading to bridge processes (see Papaspiliopoulos and Roberts (2012) for some examples). Bridges are also strongly linked with generative modelling. There, the aim is effectively to learn a bridge between two given distributions, for example two data distributions, or a data distribution and a normal distribution (De Bortoli et al., 2021; Zhou et al., 2024). These methods rely on time-reversals of a linear SDE, where, due to the linearity, the transition densities have a closed form. Bridges also are used within shape analysis. For example, in medical imaging, to model how shapes of organs change in the face of disease (Arnaudon et al., 2017), or in evolutionary biology to model how shapes of organisms change over time (Baker et al., 2024). See Figure 1 for an example of a conditioned shape process.

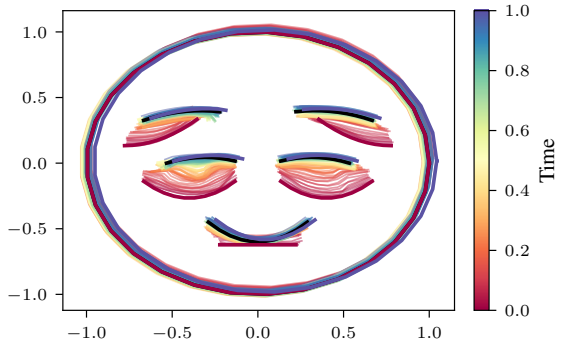

Figure 1: We learn the SDE in Equation (29) conditioned to hit the shape in black at time $t = 1$. We see that, starting the trajectory at the shape in red at time $t = 0$, we get a path between the two emojis, where each time point in the path represents a shape.

Contributions We propose a new algorithm to learn bridged diffusion processes by learning the score term occurring in the conditioned process. We learn this term by combining the score matching methods proposed in Heng et al. (2022), with adjoint diffusions introduced in Milstein et al. (2004). Importantly, these

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

adjoint diffusions can be simulated directly, and replace the need for learning a time-reversal. This differs from Heng et al. (2022), where in order to learn the score term, they first learn the time-reversed diffusion and then train a second time on the learned trajectories to get the forward-in-time bridge. Our work negates the need for learning the time-reversal, meaning we only need train once, effectively halving the necessary training time. We compare our method with other existing methods and evaluate it on a range of linear and non-linear SDEs.

2 BACKGROUND

We are interested in conditioning an SDE to take a certain value at its end time. The conditioned process can again be written as an SDE via Doob's h-transform which we describe in Section 2.1. However, this SDE has an intractable term, the score $\nabla_x \log p(t, x; T, y)$. In Section 2.2 we discuss the adjoint of the SDE which, up to a correction term ϖ, has the transition density $q(t', y; t, \cdot) = \varpi(t', y; t)^{-1} p(t, \cdot; t', y)$ when started in y at time t'. In the proposed method, we use the adjoint processes to directly learn the term $\nabla_x \log p(t, x; T, y)$, used for forward bridges. Score learning provides a method for learning a similar but different term $\nabla_x \log p(0, x_0; t, x)$ and is discussed in Section 2.3.

2.1 Doob's h-transform

Consider an SDE with $X(0) = x_0 \in \mathbb{R}^d$ defined as

$$\mathrm{d}X(t) = f(t, X(t))\mathrm{d}t + \sigma(t, X(t))\mathrm{d}W(t), \tag{1}$$

where $f\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^d, \sigma\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times m}$ and $W(t)$ is a m-dimensional Wiener process. We assume that f and σ satisfy a global Lipschitz condition so that Equation (1) has a Markov solution (Schilling and Partzsch, 2014, Theorem 19.23). Assume further that $X(t)$ has a transition density $p(t, x; t', x')$ with $t' > t$, satisfying

$$\mathbb{E}\left[X(t') \in A \mid X(t) = x\right] = \int_A p(t, x; t', x')\mathrm{d}x'. \tag{2}$$

It is well known that $X(t)$ conditioned on $X(T) = y$ for a fixed y, satisfies another SDE, namely

$$\begin{aligned}\mathrm{d}\bar{X}(t) &= f(t, \bar{X}(t))\mathrm{d}t \\ &\quad + (\sigma\sigma^\top)(t, \bar{X}(t))\nabla \log p(t, \bar{X}(t); T, y)\mathrm{d}t \\ &\quad + \sigma(t, \bar{X}(t))\mathrm{d}W(t),\end{aligned} \tag{3}$$

with $\bar{X}(0) = x_0$ and where ∇ is the gradient with respect to the argument $\bar{X}(t)$ (Särkkä and Solin, 2019, Chapter 7.5). However, the score $\nabla_x \log p(t, x; T, y)$ is intractable for non-linear SDEs.

2.2 Adjoint processes with reversed dynamics

Here we motivate the adjoint process $\{Y(t), \mathcal{Y}(t)\}$ of the unconditioned process $X(t)$ which receives its name from its relation to the adjoint operator of the infinitesimal generator of X seen as a Markov process. This process is properly defined in Equation (6). Intuitively, the adjoint process has the reversed dynamics of the unconditioned process: for example, the adjoint $Y(t)$ of a Wiener process with drift μ, i.e. $W(t) + t\mu$, is a Wiener process with drift $-\mu$. In this sense, it is a reversal, but not a time-reversal as in Haussmann and Pardoux (1986), since it does not satisfy $Y(T - t) = X(t)$.

The theory presented in this section is based on Milstein et al. (2004); Bayer and Schoenmakers (2013) and is also developed in Van der Meulen and Schauer (2020, Section 11). The significance of the adjoint process is that for fixed T, y, we can use it to access $p(t, x; T, y)$ and therefore the score $\nabla \log p(t, x; T, y)$. To compare to the forward process, fix x_0, t_0, y, T and let g be a real-valued function. Then

$$\mathbb{E}[g(X(t))] = \int g(x)p(t_0, x_0; t, x)\mathrm{d}x \tag{4}$$

$$\mathbb{E}[g(Y_{t,y}(T))\mathcal{Y}_{t,y}(T)] = \int g(x)p(t, x; T, y)\mathrm{d}x. \tag{5}$$

Here, $\mathcal{Y}$ is a weight process accounting for the fact that adjoint dynamics in general are not Markovian necessitating a Feynman-Kac representation. The processes arise by using the Fokker-Planck equation for the transition density p, and then multiplying this by $g(x)$ and integrating with respect to x. Doing so leads to a Feynman-Kac representation for $\int g(x)p(0, x_0; t, x)\mathrm{d}x$, with SDE Y and weight function $\mathcal{Y}$. For details we refer to Milstein et al. (2004).

For fixed y, t_0, T, with $t_0 \le t' \le T$, and $T_0 = T + t_0$, the adjoint process $\{Y_{t_0,y}(t), \mathcal{Y}_{t_0,y}(t)\}$ is defined as follows:

$$\mathrm{d}Y = \alpha(t, Y)\mathrm{d}t + \tilde{\sigma}(t, Y)\mathrm{d}\tilde{W}(t), \quad Y(t_0) = y, \tag{6a}$$

$$\mathrm{d}\mathcal{Y} = c(t, Y)\mathcal{Y}\mathrm{d}t, \quad \mathcal{Y}(t_0) = 1, \tag{6b}$$

where

$$\begin{aligned}\alpha^i(t', x) \coloneqq &\sum_{j=1}^d \frac{\partial}{\partial x^j}(\sigma\sigma^\top)^{ij}(T_0 - t', x) \\ &- f^i(T_0 - t', x),\end{aligned} \tag{7a}$$

$$\tilde{\sigma}(t', x) \coloneqq \sigma(T_0 - t', x), \tag{7b}$$

$$\begin{aligned}c(t', x) \coloneqq &\frac{1}{2}\sum_{i,j=1}^d \frac{\partial^2(\sigma\sigma^\top)^{ij}}{\partial x^i \partial x^j}(T_0 - t', x) \\ &- \sum_{i=1}^d \frac{\partial f^i}{\partial x^i}(T_0 - t', x).\end{aligned} \tag{7c}$$

From now on, we denote $Y_y \coloneqq Y_{0,y}$ and $\mathcal{Y}_y \coloneqq \mathcal{Y}_{0,y}$, but the following also holds for $t_0 \neq 0$. Crucially, Y_y and $\mathcal{Y}_y$ can be computed explicitly, either by hand or using automatic differentiation. Therefore, we can sample from the adjoint process using, for example, the Euler-Maruyama scheme. Moreover, following Bayer and Schoenmakers (2013, Theorem 3.3), we can introduce more time points. For a time grid $\mathcal{T} = \{0 = t_0 < t_1 < \cdots < t_L = T\}$, and setting $\hat{t}_i = T - t_{L-i}$ and $y_{L+1} = y$, it holds

$$\begin{aligned}&\mathbb{E}[f(Y_y(\hat{t}_L), \cdots, Y_y(\hat{t}_1))\mathcal{Y}_y(T)] \\ &= \int_{\mathbb{R}^{d\times L}} f(y_1, \cdots, y_L) \prod_{i=1}^{L} p(t_{i-1}, y_i, t_i, y_{i+1})\mathrm{d}y_i.\end{aligned} \tag{8}$$

In particular, we can now represent the score $s(t,x)$ as the gradient of the log density $q(T-t,\cdot)$, where $q(T-t,\cdot) = p(t,x;T,y)$ is the marginal distribution of $Y_y(T-t)$ weighted by $\mathcal{Y}_y$, and $\hat{t} \in [0,T]$ by

$$\int_A f(x) q(\hat{t}, x)\mathrm{d}x = \mathbb{E}[f(Y_y(\hat{t}))\mathcal{Y}_y(\hat{t}_L)]. \tag{9}$$

2.3 Score matching bridge sampling

In score matching bridge sampling, the task is to learn a neural approximation $s_\theta(t,x)$ to the score $s(t,x) = \nabla_x \log p(t,x;T,y)$. In our case, samples from $Y_y(T-t)$ weighted with $\mathcal{Y}_y(T-t)$ can be used in training, as they represent the unnormalised density $q(T-t,\cdot) = p(t,\cdot;T,v)$. This precisely mimics the use of samples of the noising process in generative diffusion models with $Y_y(\hat{t}), \mathcal{Y}_y(\hat{t})$ in the role of the noising process.

To wit, vanilla score matching as introduced in Hyvärinen (2005) takes the form, for $\hat{t} = T - t$

$$\min_\theta \int_0^T \int q(\hat{t}, x) \sum_{i=1}^{N} S_i \mathrm{d}x\mathrm{d}t, \tag{10}$$

$$S_i \coloneqq \frac{\partial}{\partial x_i} s_{\theta,i}(t,x) + \frac{1}{2} s_{\theta,i}(t,x)^2, \tag{11}$$

where we introduce S_i for notational brevity. Note in particular that knowledge of the transition density is not required, however, computing the divergence term is expensive. To avoid computing the divergence term when learning $\nabla_x \log p(0,x_0;t,x)$, instead one can use a similar loss objective to that proposed by Vincent (2011)

$$\mathcal{L}(\theta) = \int_0^T \mathbb{E}[\|s_\theta(t,X_t) - s(t,X_t)\|^2]\mathrm{d}t, \tag{12}$$

where $s(t,x) = \nabla_x \log p(0,x_0;t,x)$. We instead will take the expectation with respect to conditioned trajectories, and will let $s(t,x) \coloneqq \nabla \log p(t,x;T,y)$. In and of itself, this optimisation problem is still intractable. However, Heng et al. (2022) show that for C independent of θ

$$\mathcal{L}(\theta) = C + \sum_{l=0}^{L-1} \int_{t_l}^{t_{l+1}} \mathbb{E}[\|s_\theta(t,X_t) - s_l\|^2_{\sigma\sigma^\top}]\mathrm{d}t, \tag{13a}$$

$$s_l = \nabla \log p(t_l, X_{t_l}; t, X_t), \tag{13b}$$

where s_l is used to simplify the notation, and $\|x\|_A = \|A^{1/2}x\|$ is the weighted norm for A a positive definite matrix. For small time steps, the transition density $p(t_l, X_{t_l}; t, X_t)$ can be estimated using the Euler-Maruyama scheme.

3 RELATED WORK

In this section, we survey the existing methods for computing bridge processes. We concentrate specifically on methods that can be applied to any diffusion process to find the forward bridge. For this reason, we deem usual score matching methods as out of scope, since the main motivation of these methods is to find a time-reversal of a specific diffusion, usually an Ornstein-Uhlenbeck, where the score function can be written in closed form.

3.1 Score matching methods

Most related to our work is Heng et al. (2022)'s, who first learn the time-reversal of the bridge process. They observe that this is equivalent to learning the term $\nabla_x \log p(0,x_0;t,x)$. Repeating the process a second time – now using the learned time-reversals in place of unconditioned forward processes – gives a forward-in-time bridge. Moreover, they propose a loss function that works with non-linear SDEs, where the transition density is not tractable. We build on Heng et al. (2022)'s work by using their loss function for bridging diffusion bridges. However, our implementation requires only half the compute as follows: Instead of training on time-reversals (which must be learned), we train using adjoint processes. The adjoint processes can be written in closed form and therefore we may directly simulate from them. Using these, we learn the term $\nabla \log p(t,x;T,y)$ (and thereby the bridged process) directly, without needing to train twice.

Recently, Singhal et al. (2024) consider using non-linear SDEs instead of linear ones within generative modelling, meaning the transition densities for the SDE are also unknown. However, instead of using Euler-Maruyama approximations for the transition kernels as in Heng et al. (2022), they consider other higher-order approximations. Due to the non-linear SDEs this is closer to our problem, however they find time-reversals and not forward bridges, which is the focus of this work.

3.2 MCMC methods

One method of sampling from the bridge processes is to sample from a second approximate bridge, where the ratio between the true bridge and the approximate bridge is known. For an unconditional process as in Equation (1), the approximate bridges have form

$$\begin{aligned} \mathrm{d}X(t) = f(t, X(t))\mathrm{d}t + s(t, X(t))\mathrm{d}t \\ + \sigma(t, X(t))\mathrm{d}B(t), \end{aligned} \tag{14}$$

where different authors have considered different options for the function s. In Pedersen (1995) $s = 0$, that is, the samples are taken from the original unconditioned process. Delyon and Hu (2006) instead take the Brownian bridge drift term. More recently, Schauer et al. (2017) set s to be the term arising from conditioning an auxiliary process with a known transition density. Then, one can derive a ratio between the path probability of the true bridge and the bridge approximation and can therefore use Markov chain Monte Carlo methods to sample from the true distribution.

3.3 Adjoint methods

Milstein et al. (2004) estimate the transition density of an SDE by introducing adjoint processes. Simulation happens both forward from the start state and backwards from the end state. When connected, this is called a forward-reverse representation, and it gives rise to a Monte-Carlo estimator. Bayer and Schoenmakers (2013) expand this to estimating expectations of functions of multiple time-steps of conditioned SDEs, again with Monte-Carlo methods. We use the probabilistic representations with adjoints they introduce, however we use these to construct a loss function to learn $\nabla_x \log p(t, x; T, y)$ for varying t, x, y. Thereby, we can sample directly from the conditioned trajectories.

4 METHOD

This section presents our algorithm to find the score function $\nabla_x \log p(t, x; T, y)$ that will be used to simulate the bridge process. In Section 4.1, we derive the main loss function for learning the score function and show it minimises the Kullback-Leibler divergence between the conditioned path and the path measure induced by the learnt conditioned process. In Section 4.2, we provide details on how we minimise the loss function. In Section 4.3, we consider varying values of y, and the case that the end point y is forced to follow a given distribution, instead of being a fixed point. Proofs can be found in Appendix A.

4.1 A loss function

In order to sample the bridged SDE, we learn the score term $\nabla_x \log p(t, x; T, y)$ for fixed T, y. Note that, although similar, this is different from the term $\nabla_x \log p(0, x_0; t, x)$ that is learnt for time-reversed processes, as in score-matching for diffusion samplers. Indeed, score-matching methods rely on the SDE X_t having $p(0, x_0; t, x)$ as a transition density, and therefore by sampling from the process X_t, we are sampling from the distribution of the process at time t. This is not true for $p(t, x; T, y)$. Our proposal is instead to sample from the adjoint stochastic process, where we can recover the transitions $p(t, x; T, y)$.

Given a function $s\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$, we define the SDE X^s with path distribution $\mathbb{Q}^s$ as

$$\begin{aligned} \mathrm{d}X^s(t) = f(t, X^s(t))\mathrm{d}t \\ + (\sigma\sigma^\top)(t, X^s(t))s(t, X^s(t))\mathrm{d}t \\ + \sigma(t, X^s(t))\mathrm{d}W(t). \end{aligned} \tag{15}$$

In what follows, let $K_\epsilon(x, y)$ be a function, such that $K_\epsilon(x, y) \to \delta_0(x - y)$ as $\epsilon \to 0$, for δ_0 the dirac delta function. For fixed y, T with $Y \coloneqq Y_{0,y}, \mathcal{Y} \coloneqq \mathcal{Y}_{0,y}$ (see Equations (6a) and (6b)) and $\hat{t} \coloneqq T - t$ we introduce

$$\mathcal{L}(s, x_0) \coloneqq \mathbb{E}\left[\frac{\lim_{\epsilon\to 0} \mathcal{Y}(T) K_\epsilon(Y(T), x_0)}{p(0, x_0; T, y)} \sum_{\ell=1}^{L} I_\ell\right], \tag{16a}$$

$$I_\ell = \int_{t_{\ell-1}}^{t_\ell} \|s(t, Y(\hat{t})) - s_\ell(t, Y(\hat{t}))\|^2_{\sigma\sigma^\top} \mathrm{d}t, \tag{16b}$$

$$s_\ell(t, Y(\hat{t})) = \nabla \log p(t, Y(\hat{t}); t_\ell, Y(\hat{t_\ell})), \tag{16c}$$

where s_ℓ, I_ℓ have been used to make the notation more compact. By minimising this, we are minimising a Kullback-Leibler divergence:

Theorem 4.1. Let X be a diffusion process as defined in Equation (1) and suppose further that f, σ are C^2 and that X admits C^2 transition densities. Let $\bar{X}$ be the conditioned diffusion satisfying Equation (3). Let $\bar{\mathbb{P}}$ be the path probability of $\bar{X}$ and $\mathbb{Q}^s$ the path probability of X^s in Equation (15) for a function s. Then for a given starting point x_0

$$\arg\inf_s \mathrm{KL}(\bar{\mathbb{P}} \parallel \mathbb{Q}^s) = \arg\inf_s \mathcal{L}(s, x_0), \tag{17}$$

where the infimum is over functions $s\colon [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$.

Proof. First, we use the adjoint SDE and Bayer and Schoenmakers (2013, Theorem 3.7) to write the expectations as integrals over the transition densities $p(t, x; T, y)$. Secondly, as proposed in Heng et al. (2022),

we use the Chapman-Kolmogorov equation to split the integration into smaller, approximable parts. A full proof is given in Appendix A.1. □

We have shown that finding s that minimises the loss function in Equation (18a), also minimises Kullback-Leibler divergence between the conditioned path probability space and the path space $\mathbb{Q}^s$. The difference is that it is possible to approximate $\mathcal{L}(s)$. For example, we can take ϵ small, and for $t' - t$ small we can approximate $\nabla_x \log p(t, x; t', x')$. Next, we remove the dependence of the loss function on the start point.

Theorem 4.2. We assume the same setup as in Theorem 4.1. Define $\bar{\mathcal{L}}(s) \coloneqq \int \mathcal{L}(s, x_0)\mathrm{d}x_0$. Then, letting I_ℓ be defined as in Equation (16b):

$$\bar{\mathcal{L}}(s) = \mathbb{E}\left[\frac{\mathcal{Y}(T)}{p(0, Y(T); T, y)} \sum_{l=1}^{L} I_\ell\right]. \tag{18a}$$

Proof. This is a consequence of the integration of $K_\epsilon(Y(T), x_0)$. See Appendix A.2 for details. □

Finally, we define a simplified loss function, which we use in the experiments. It is a weighted version of $\bar{\mathcal{L}}(s)$:

$$\mathcal{L}(s) \coloneqq \mathbb{E}\left[\sum_{\ell=1}^{L} I_\ell\right] \tag{19}$$

where I_ℓ is as in Equation (16b).

4.2 Optimising the loss function

To solve the optimisation problem, we let s be parameterised by a neural network. We simulate batches of trajectories of the adjoint process $\{Y(t), \mathcal{Y}(t)\}$ using the Euler-Maruyama algorithm. Like Heng et al. (2022), we approximate $p(t, x; t + \epsilon, y)$ as a Gaussian, so that $\nabla \log p(t, x; t + \epsilon, y) \approx g(t, x; t + \epsilon, y)$, where g is a Euler-Maruyama step, i.e. for $(\sigma\sigma^\top) \coloneqq \Sigma$

$$g(t, x, t', x') = \frac{1}{\Delta t}\Sigma(t, x)^{-1}(x' - x - \Delta t f(t, x)). \tag{20}$$

We then approximate the loss function as

$$\mathcal{L}(s_\theta) \approx \frac{\Delta t}{N} \sum_{Y \in B_N} \sum_{\ell=0}^{L-1} \|T_\ell\|^2_{\sigma\sigma^\top}, \tag{21a}$$

$$T_\ell \coloneqq s_\theta(t_\ell, Y_\ell) - g(t_\ell, Y_\ell; t_{\ell+1}, Y_{\ell+1}), \tag{21b}$$

where $Y_\ell \coloneqq Y(T - t_\ell)$ and B_N is a batch of N simulations of $Y(t)$ started at the value y on which we are conditioning. Algorithm 1 gives the exact steps.

After using Algorithm 1 to learn s_θ, we can then sample from the conditioned SDE

$$\begin{aligned}\mathrm{d}X(t) = (\sigma\sigma^\top)(t, X(t))s_\theta(t, X(t))\mathrm{d}t \\ + f(t, X(t))\mathrm{d}t + \sigma(t, X(t))\mathrm{d}W(t),\end{aligned} \tag{22}$$

using, for example, the Euler-Maruyama algorithm.

4.3 Training on multiple end points

We can extend to training on multiple end points in two ways. The first is where the end point y is not fixed, rather sampled from some distribution π_T. That is we are conditioning by forcing the SDE to have a given distribution at the end time as in Baudoin (2002) (see also Corstanje et al. (2023, Example 2.4)). In this case we learn the term $\nabla \log h(t, x)$, where $h(t, x) \coloneqq \int \frac{p(t,x;T,y)}{p(0,x_0;T,y)}\pi_T(\mathrm{d}y)$. The second, is where we learn fixed y for multiple y values at once; that is for fixed T and *variable* y, we learn $s_\theta(t, x; y) \approx \nabla_x \log p(t, x; T, y)$. Both of these are straightforward extensions to the method, also in Heng et al. (2022).

Distributions of endpoints To condition on a distribution π_T instead of a fixed point y, we integrate over the loss function for different values of y. In practice, all that changes is instead of simulating trajectories $Y, \mathcal{Y}$ starting from the same value y, we now also sample the start value $y \sim \pi_T$. Letting $Y_{y,\ell} \coloneqq Y_y(T - t_\ell)$, the loss in Equation (21a) function becomes

$$\mathcal{L}(s_\theta) \approx \frac{\Delta t}{N} \sum_{y \sim \pi_T} \sum_{\ell=0}^{L-1} \|T_\ell\|^2_{\sigma\sigma^\top}, \tag{23a}$$

$$T_\ell \coloneqq s_\theta(t_\ell, Y_{y,\ell}) - g(t_\ell, Y_{y,\ell}; t_{\ell+1}, Y_{y,\ell+1}). \tag{23b}$$

Multiple fixed endpoints For varied $y \in B \subseteq \mathbb{R}^d$, we integrate the loss function in Equation (18a) over y and take the loss over functions $s\colon [0, T] \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$. Then we take a batch B_N of N uniformly distributed $y \in B$. In practice, the loss function that we optimise changes from Equation (21a) to

$$\mathcal{L}(s_\theta) \approx \frac{\Delta t}{N} \sum_{y \in B_N} \sum_{\ell=0}^{L-1} \|T_\ell\|^2_{\sigma\sigma^\top}, \tag{24a}$$

$$T_\ell \coloneqq s_\theta(t_\ell, Y_{y,\ell}, y) - g(t_\ell, Y_{y,\ell}; t_{\ell+1}, Y_{y,\ell+1}), \tag{24b}$$

where we have used $Y_{y,\ell} \coloneqq Y_y(T - t_\ell)$, to mean the adjoint process Y started from y at time 0.

5 EXPERIMENTS

We compare only to methods considering the problem where we are *given* an SDE that we condition on given boundary conditions. This is different to the setup of generative modelling or denoising diffusion bridge models, where the SDE can be chosen freely. In this case, a natural choice is a linear SDE, which has explicit transition densities. Further experiment details can be found in Appendix C. Code is available at `https://github.com/libbylbaker/forward_bridge`.

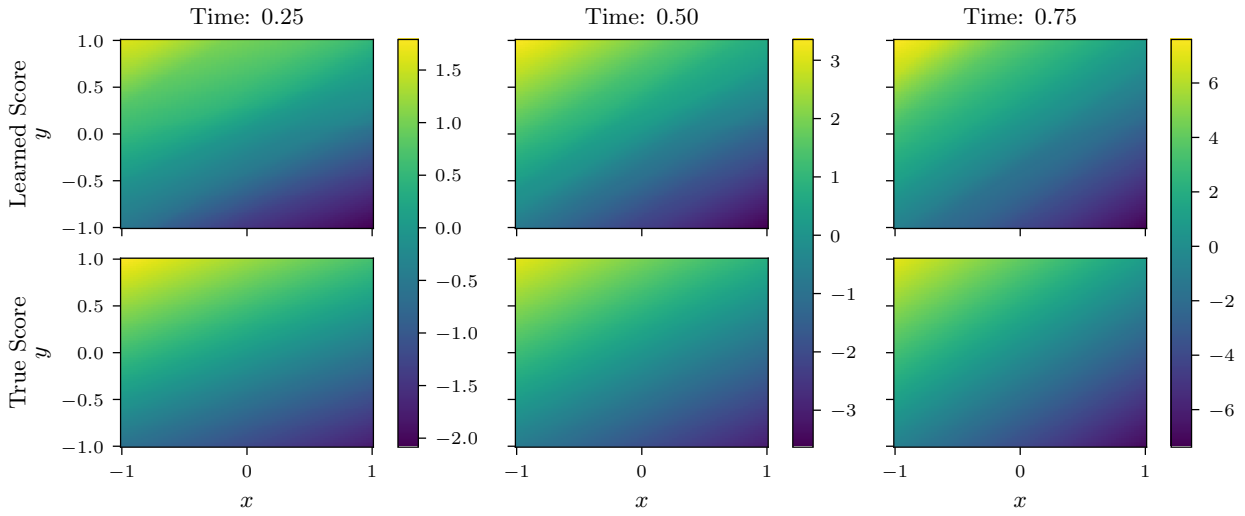

Figure 2: For varying x, y and different times $t \in [0.25, 0.5, 0.75]$, we plot the true score $\nabla_x \log p(t, x; 1, y)$ and the learned score $s_\theta(t, x; y)$, as described in Section 5.1. The learned score takes a fixed end time $T = 1$, but takes both x and y as input values. The absolute error between the true and learned score is reported in Figure 7.

5.1 Ornstein–Uhlenbeck

First, we condition an Ornstein-Uhlenbeck process. This is a good experiment to evaluate the method, since the true score is known, so we can compute the error. Explicitly, we consider

$$\mathrm{d}X(t) = -X(t)\mathrm{d}t + \mathrm{d}W(t). \tag{25}$$

The formulas for the adjoint process and the score function can be found in Appendix C.1.

In our first experiment we learn $s_\theta(t, x; y) \approx \nabla \log p(t, x; T, y)$, for $y \in [-1.0, 1.0]$, as in Section 4.3. We plot the true and learned score in Figure 2. In Figures 6 and 7 we also plot the absolute error and the generated trajectories. We see that at earlier times the generated trajectories are less dispersed, thereby leading to higher errors in the score when the difference between x and y is larger.

Next, we compare our results to Heng et al. (2022) (using the authors' code [1]). We want to show the errors that occur when using the adjoint processes to learn the bridge, are similar to the errors using the forward processes to learn the time-reversal, and either similar or better than using the time-reversals to learn the forward bridge.

In Figure 3, we plot the squared error averaged over time points from $t = 0$ to $t = 1$ for various dimensions d with $x_0 = 1.0$, and various end times T, with $d = 1$. We compare this to the forward and time-reversed (or backwards) diffusion bridges from Heng et al. (2022). The time-reversed, DB (bw), learns $\nabla_x \log p(0, x_0; t, x)$ instead of $\nabla_x \log p(t, x; T, y)$. The forward, DB (fw), leverages the time-reversal to learn $\nabla_x \log p(t, x; T, y)$.

As shown in Figure 3, the time-reversed bridges (DB (bw)) demonstrate a similar error to the proposed approach. However, in learning the forward bridge, the proposed method is not only twice as efficient as DB (fw) – requiring only half the number of trajectories due to being trained once – but also exhibits greater stability in higher dimensions.

Finally, to see how it scales to slightly higher dimensions we train our proposed method under the same setup, also for 5 different random seeds. We report the mean squared error and standard deviation results in Table 1.

Table 1: Ornstein-Uhlenbeck Errors

Dim.	MSE	SD
32	0.553	0.008
50	0.548	0.019
100	0.567	0.003
200	0.566	0.003

5.2 Wiener process on distributed endpoints

We may also wish to condition on a distribution, rather than fixed endpoints. To demonstrate the method in this situation, we condition a Wiener process on hitting a circle. That is, we take the SDE $\mathrm{d}X(t) = \mathrm{d}W(t)$, and we sample the end points y_0 uniformly from a circle of radius 3.0. In Figure 4 (left), we plot 20 trajectories, each starting from the origin, and conditioned to hit the circle at time $T = 1.0$. In Figure 4 (right) we plot

[1] `https://github.com/jeremyhengjm/DiffusionBridge`.

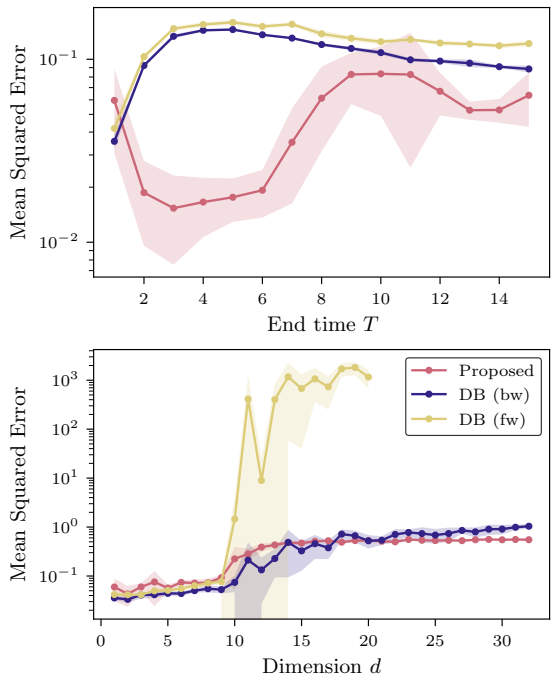

Figure 3: We plot the square error between the true and learned score averaged over times from $t = 0$ to $t = 1$. For the proposed method and DB (fw) we learn $\nabla_x \log(t, x; T, y)$, and for DB (bw) we learn $\nabla_x \log(0, x_0; t, x)$. *Top:* Trained for $x_0 = 1$, $y = 1$, with 1-dimension and varying end times. *Bottom:* Trained for $x_0 = 1.0$, $y = 1.0$, with $T = 1.0$. See 5.1 for details.

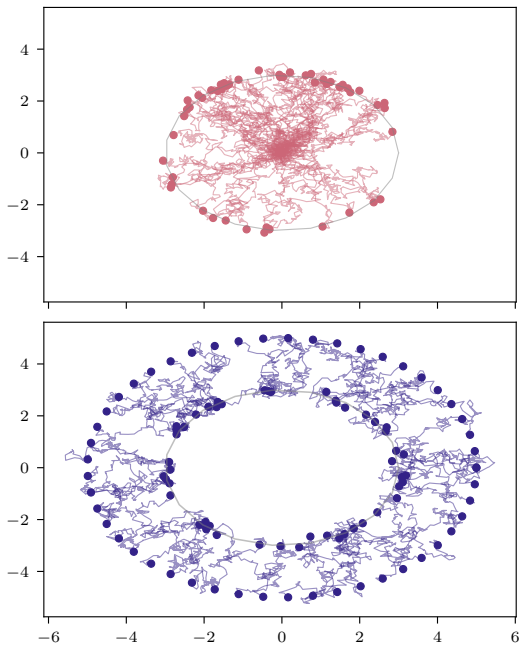

Figure 4: We train a 2D Wiener process process to hit a circle of radius 3 at time $T = 1.0$. On the top we plot 20 trajectories started from the centre, and on the bottom we plot 20 trajectories started from evenly spaced points on a circle of radius 5.

20 trajectories, this time starting from evenly sampled initial values on a circle of radius 5.0. Note that both figures use the same trained score function. We only need to train once, since the training is independent of the initial value x_0, and only depends on the final value or distribution on which we are conditioning.

5.3 Cell model

So far we have only considered linear models. To test on non-linear models, we next consider the cell differentiation and development model from Wang et al. (2011), also considered by Heng et al. (2022). The SDE is defined as

$$\mathrm{d}X_1(t) = f(X_1(t), X_2(t))\mathrm{d}t + \sigma \mathrm{d}W(t) \tag{26}$$
$$\mathrm{d}X_2(t) = f(X_2(t), X_1(t))\mathrm{d}t + \sigma \mathrm{d}W(t), \tag{27}$$
$$f(x, y) = \frac{x(t)^4}{2^{-4} + x(t)^4} + \frac{2^{-4}}{2^{-4} + y(t)^4} - x(t) \tag{28}$$

and describes the process of cell differentiation into three cell types. In Figure 8 we plot some trajectories of the unconditioned process for illustration. We compare to the time-reversed bridge and the forward bridge of Heng et al. (2022) and the guided bridge method from Schauer et al. (2017). We choose to compare with these methods, since Heng et al. (2022) have already shown favourable results compared to Pedersen (1995); Durham and Gallant (2002); Delyon and Hu (2006).

The true bridge is unknown, so instead we compare against trajectories from the unconditioned process with similar the boundary values. In Figure 5, we plot in grey trajectories from the unconditioned process satisfying $x_1(T) \in (1.3, 1.7), x_2(T) \in (0.0, 0.4)$ at time $T = 2.0$. We then plot trajectories starting at $(0.1, 0.1)$, conditioned on $(1.5, 0.2)$ at time $T = 2.0$, via the proposed method, the forward bridge (DB (fw)) of Heng et al. (2022) and with MCMC using guided proposals as in Schauer et al. (2017). For the time-reversed bridge, the trajectories are simulated from the end point $(1.5, 2.0)$, and run backwards in time, being conditioned to hit the point $(0.1, 0.1)$. The trajectories from the time-reversed SDE are then used to train the forward bridge (Heng et al., 2022, Section 2.3).

We see in Figure 5 that the proposed method, the time-reversed bridge and the forward bridge produce similar trajectories that follow the correct data distribution. However, again the proposed method need only be trained once, whereas the forward bridge was trained on the time-reversed trajectories thereby needing double the amount of training.

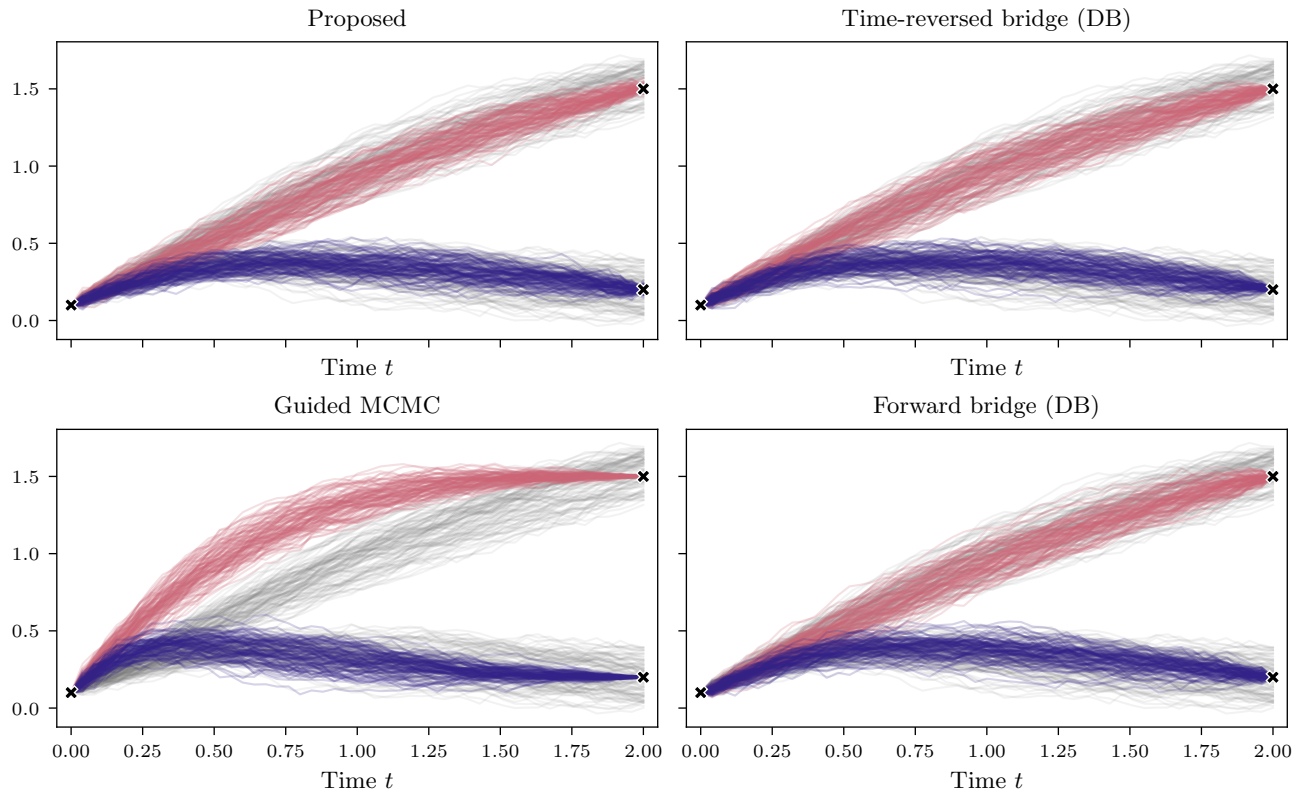

Figure 5: We plot 100 conditioned trajectories with the methods *Proposed, Guided MCMC* and *Forward bridge (DB)* and the time-reversed bridge with *Time-reversed bridge*. In grey, we plot some unconditioned trajectories that end in a given window about the point $(1.5, 0.2)$. For details see 5.3.

5.4 Shape models

We continue increasing the difficulty of the problems, and apply our proposed method to stochastic shape analysis (Arnaudon et al., 2023). This is an interesting test since it involves conditioning SDEs on shape spaces, where the stochastic processes are non-linear and have non-trivial diffusion functions. Such a model is found in evolutionary biology when studying how the morphometry of a species changes as the species evolves (Baker et al., 2024; Yang et al., 2024). Therefore, this experiment will give insights into what we can expect when applying the method to evolutionary biology, though we apply it to non-biological shapes.

For a set of points $\{x_i\}_{i=1}^n \subset \mathbb{R}^2$ representing the discretisation of a shape, we consider the SDE

$$\mathrm{d}X_t = \sum_{i=1}^{m} k(X(t), z_i)\mathrm{d}W_i(t), \tag{29}$$

with the initial constraint $X(0) = \{x_i\}_{i=1}^n \in \mathbb{R}^{n\times 2}$, where W_i are independent Wiener processes in $\mathbb{R}^2$, $\{z_i\}_{i=1}^m \subset \mathbb{R}^2$ is a grid on a compact subset of $\mathbb{R}^2$ and k is the Gaussian function defined as $k(x, y) = \alpha \exp -\frac{\|x-y\|^2}{2\sigma^2}$ for some scalars σ, α. This SDE represents the position of the points x_i at time t.

We use the neural network architecture of Yang et al. (2024). They use Fourier neural operators and learn the term $\nabla \log p(0, x_0; t, x_t)$, using the method proposed by Heng et al. (2022) to get a time-reversal of Y_t. This is applied to butterfly data to bridge between the outlines of butterflies. Instead, we apply the proposed method to the outlines of two emojis, conditioning such that the points reach a "smiley face" at time $t = 1$.

In Figure 1 we plot one trajectory of the SDE Equation (29). At each time point, $X(t) \in \mathbb{R}^{n\times 2}$ can be interpreted as a stochastic change in the original shape. We have conditioned the SDE in Equation (50) using Algorithm 1 such that at time $t = 1$, the configuration of points is the smiley face shown in blue. For this trajectory, we start it from the sad face shown in red. In this case, we cannot evaluate the learnt score directly, since the true score is unknown. However, we see that indeed the sample paths end at the wanted boundary points as expected, even for non-trivial diffusion functions and trivially chosen shapes.

6 PROS, CONS, AND CONCLUDING REMARKS

We believe the main drawback of this method is conditioning on unlikely events, that are not covered in the data. An obvious example would be conditioning a 1-dimensional Wiener process $W(t), W(0) = 0$ such that $W(1) = 100$. In this case, learning either the time-reversal or the bridge with proposed score-matching methods will struggle, since the generated trajectories are highly unlikely to cover this state. We envisage this problem could be compounded using the adjoints. For example, consider conditioning on a state that is highly likely to be reached from a large range of starting states e.g. conditioning the Ornstein-Uhlenbeck process on 0 at time 100. Then trajectories of the adjoint process will fan out to cover the wide range of likely start states. This will be a disadvantage, if one is only interested in a small range of specific start points. However, it could also be an advantage to some extent, since a larger area will be covered by the data.

We have shown that we can use adjoint processes to learn the score term $\nabla \log p(t, x; T, y)$ for given T, y, which we use to sample from the bridged diffusion processes. This seems to perform well compared to using MCMC methods to sample trajectories. When compared to the method in Heng et al. (2022) for simulating forward-in-time bridges, we have demonstrated that using the adjoints can lead to an estimation with lower error and negates the need for training twice.

Acknowledgements

The work presented in this article was done at the Center for Computational Evolutionary Morphometry and is partially supported by the Novo Nordisk Foundation grant NNF18OC0052000, by the Chalmers AI Research Centre (CHAIR) ("Stochastic Continuous-Depth Neural Network") as well as a research grant (VIL40582) from VILLUM FONDEN and UCPH Data+ Strategy 2023 funds for interdisciplinary research.

References

Arnaudon, A., Holm, D., and Sommer, S. (2023). *Stochastic Shape Analysis*, page 1325–1348. Springer International Publishing, Cham.

Arnaudon, A., Holm, D. D., Pai, A., and Sommer, S. (2017). A stochastic large deformation model for computational anatomy. In *Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings 25*, pages 571–582. Springer.

Baker, E. L., Yang, G., Severinsen, M. L., Hipsley, C. A., and Sommer, S. (2024). Conditioning non-linear and infinite-dimensional diffusion processes. arXiv:2402.01434.

Baudoin, F. (2002). Conditioned stochastic differential equations: theory, examples and application to finance. *Stochastic Processes and their Applications*, 100(1-2):109–145.

Bayer, C. and Schoenmakers, J. (2013). Simulation of forward-reverse stochastic representation for conditional diffusions. *The Annals of Applied Probability*, 24.

Bladt, M. and Sørensen, M. (2014). Simple simulation of diffusion bridges with application to likelihood inference for diffusions. *Bernoulli*, pages 645–675.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Corstanje, M., van der Meulen, F., and Schauer, M. (2023). Conditioning continuous-time markov processes by guiding. *Stochastics*, 95(6):963–996.

De Bortoli, V., Thornton, J., Heng, J., and Doucet, A. (2021). Diffusion schrödinger bridge with applications to score-based generative modeling. *Advances in Neural Information Processing Systems*, 34:17695–17709.

Delyon, B. and Hu, Y. (2006). Simulation of conditioned diffusion and application to parameter estimation. *Stochastic Processes and their Applications*, 116(11):1660–1675.

Durham, G. B. and Gallant, A. R. (2002). Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–316.

Haussmann, U. G. and Pardoux, E. (1986). Time reversal of diffusions. *The Annals of Probability*, pages 1188–1205.

Heng, J., De Bortoli, V., Doucet, A., and Thornton, J. (2022). Simulating diffusion bridges with score matching. arXiv:2111.07243.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709.

Milstein, G. N., Schoenmakers, J. G., and Spokoiny, V. (2004). Transition density estimation for stochastic differential equations via forward-reverse representations. *Bernoulli*, 10(2).

Papaspiliopoulos, O. and Roberts, G. (2012). Importance sampling techniques for estimation of diffusion models. *Statistical methods for stochastic differential equations*, 124:311–340.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pedersen, A. R. (1995). Consistency and asymptotic normality of an approximate maximum likelihood estimator for discretely observed diffusion processes. *Bernoulli*, pages 257–279.

Schauer, M., Van Der Meulen, F., and Van Zanten, H. (2017). Guided proposals for simulating multi-dimensional diffusion bridges. *Bernoulli*, 23(4A).

Schilling, R. L. and Partzsch, L. (2014). *Brownian motion: an introduction to stochastic processes*. De Gruyter textbook. de Gruyter, Berlin ; Boston, second edition.

Singhal, R., Goldstein, M., and Ranganath, R. (2024). What's the score? Automated denoising score matching for nonlinear diffusions. In *Forty-first International Conference on Machine Learning*.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Cambridge University Press, 1 edition.

Van der Meulen, F. and Schauer, M. (2020). Automatic backward filtering forward guiding for Markov processes and graphical models. 2010.03509v2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Wang, J., Zhang, K., Xu, L., and Wang, E. (2011). Quantifying the Waddington landscape and biological paths for development and differentiation. *Proceedings of the National Academy of Sciences*, 108(20):8257–8262.

Yang, G., Baker, E. L., Severinsen, M. L., Hipsley, C. A., and Sommer, S. (2024). Simulating infinite-dimensional nonlinear diffusion bridges. *arXiv preprint arXiv:2405.18353*.

Zhou, L., Lou, A., Khanna, S., and Ermon, S. (2024). Denoising diffusion bridge models. In *The Twelfth International Conference on Learning Representations*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes: all theorems contain a statement of the assumptions and settings.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes: We include the time complexity for the algorithm.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes: the anonymized source code is included with a read me file containing details on how to install everything.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes: We include these in the statements of the theorems
   (b) Complete proofs of all theoretical results. Yes: The full proof are included in the appendix
   (c) Clear explanations of any assumptions. Yes: the assumptions or where to find them are included in the theorem statements.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes: To reproduce the experiments, we have an experiment folder in our codebase with all the code needed to reproduce.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes: We include the hyperparameters in the paper itself, including in the appendix, but the exact setup for the experiments are also in the code base including all hyperparameters.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes: where used, these are explained in our experiments section.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes: This can be found in the appendix with the further experiment details.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes: Code that we use are cited.

The code that we ran in the experiments is linked to.

(b) The license information of the assets, if applicable. Yes: Links to the code are included which contain any license information.

(c) New assets either in the supplemental material or as a URL, if applicable. Not Applicable

(d) Information about consent from data providers/curators. Not Applicable

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. Not Applicable

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

Score matching for bridges without time-reversals

A PROOFS

A.1 Proof of Theorem 4.1

Theorem. Let X be a diffusion process as defined in Equation (1) and suppose further that f, σ are C^2 and that X admits C^2 transition densities. Let $\bar{X}$ be the conditioned diffusion satisfying Equation (3). Let $\bar{\mathbb{P}}$ be the path probability of $\bar{X}$ and $\mathbb{Q}^s$ the path probability of X^s in Equation (15) for a function s. Then for a given starting point x_0

$$\arg\inf_s \mathrm{KL}(\bar{\mathbb{P}} \parallel \mathbb{Q}^s) = \arg\inf_s \mathcal{L}(s, x_0), \tag{30}$$

where the infimum is over functions $s\colon [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$.

Proof. The Kullback-Leibler divergence is defined by

$$\mathrm{KL}(\bar{\mathbb{P}} \parallel \mathbb{Q}^s) = \int \log \frac{\mathrm{d}\bar{\mathbb{P}}}{\mathrm{d}\mathbb{Q}^s} \mathrm{d}\bar{\mathbb{P}}(X). \tag{31}$$

By Girsanov's theorem, (Schilling and Partzsch, 2014, Theorem 18.8)

$$\frac{\mathrm{d}\bar{\mathbb{P}}}{\mathrm{d}\mathbb{Q}^s}(X) = \exp\left(-\frac{1}{2}\int_0^T \|\sigma^\top(s - s_\theta)(t, X_t)\|^2 \mathrm{d}t + \int_0^T \sigma^\top(s - s_\theta)(t, X_t)\mathrm{d}B_t\right). \tag{32}$$

Since the expectation of a martingale is equal to 0, we see that

$$\begin{aligned}
\mathrm{KL}(\bar{\mathbb{P}} \parallel \mathbb{Q}^s) &= \int_{X\in\mathcal{X}} \left[\frac{1}{2}\int_0^T \|(s - s_\theta)(t, X_t)\|^2_{\sigma\sigma^\top} \mathrm{d}t\right] \mathrm{d}\mathbb{P}^y(X) && \text{(33a)}\\
&= \frac{1}{2}\int_0^T \bar{\mathbb{E}}^y \left[\|(s - s_\theta)(t, X_t)\|^2_{\sigma\sigma^\top}\right] \mathrm{d}t && \text{(33b)}\\
&= \frac{1}{2}\int_0^T \mathbb{E}\left[\|(s - s_\theta)(t, X_t)\|^2_{\sigma\sigma^\top} \mid X_{0,x_0}(T) = y\right] \mathrm{d}t. && \text{(33c)}
\end{aligned}$$

By (Bayer and Schoenmakers, 2013, Theorem 3.7), this is equal to

$$\frac{1}{2p(0, x_0; T, y)} \int_0^T \lim_{\epsilon\to 0} \mathbb{E}[\|(s - s_\theta)(t, Y(T-t))\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon(Y(T), x_0)] \mathrm{d}t. \tag{34}$$

We next focus on the contents of the expectation. Writing s out in full as $s = \nabla_x \log p(t, x; T, y)$, using K_ϵ in place of $K_\epsilon(Y(T), x_0)$ and expanding the square, we see

$$\begin{aligned}
&\int_0^T \mathbb{E}[\|(s - s_\theta)(t, Y(T-t))\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon(Y(T), x_0)] \mathrm{d}t && \text{(35a)}\\
&\qquad = \mathbb{E}\left[\int_0^T \|s_\theta(t, Y(T-t))\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon \mathrm{d}t\right] && \text{(35b)}\\
&\qquad + \mathbb{E}\left[\int_0^T \|\nabla \log p(t, Y(T-t); T, y)\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon \mathrm{d}t\right] && \text{(35c)}\\
&\qquad - 2\mathbb{E}\left[\sum_{l=1}^L \int_{t_{\ell-1}}^{t_\ell} \langle s_\theta(t, Y(T-t)), \nabla \log p(t, Y(T-t); T, y)\rangle_{\sigma\sigma^\top} \cdot \mathcal{Y}(T) K_\epsilon \mathrm{d}t\right]. && \text{(35d)}
\end{aligned}$$

We focus on the third term, Equation (35d). Multiplying this out further and using Bayer and Schoenmakers (2013, Theorem 3.6), we see that

$$\mathbb{E}\left[s_\theta(t, Y(T-t)) \cdot \nabla \log p(t, Y(T-t); T, y) \cdot \mathcal{Y}(T)\right] \tag{36a}$$

$$= \int s_\theta(t,x) \nabla_x \log p(t,x,T,y) p(t_0, x_0; t, x) p(t, x; T, y) \mathrm{d}x_0 \mathrm{d}x \tag{36b}$$

$$= \int s_\theta(t,x) \nabla_x p(t,x;T,y) p(t_0, x_0; t, x) \mathrm{d}x_0 \mathrm{d}x. \tag{36c}$$

Using Chapman-Kolmogorov it holds that for any $t < t_1 < T$,

$$p(t,x;T,y) = \int p(t,x;t_1,x_1) p(t_1,x_1;T,y) \mathrm{d}x_1, \tag{37}$$

and so, using the properties of the differential of the logarithm it holds

$$\nabla_x p(t,x;T,y) = \int \nabla_x \log p(t,x;t_1,x_1) p(t,x;t_1,x_1) p(t_1,x_1;T,y) \mathrm{d}x_1. \tag{38}$$

Putting this back into Equation (36c) and using Bayer and Schoenmakers (2013, Theorem 3.6) we get,

$$\int [s_\theta(t,x) \nabla_x \log p(t,x;t_1,x_1)] p(t_0,x_0;t,x) p(t,x;t_1,x_1) p(t_1,x_1;T,y) \mathrm{d}x_0 \mathrm{d}x \mathrm{d}x_1 \tag{39a}$$

$$= \mathbb{E}[s_\theta(t, Y(T-t)) \nabla \log p(t, Y(T-t); t_1, Y(T-t_1)) \mathcal{Y}(T)]. \tag{39b}$$

From this we see that Equation (35d) becomes

$$-2\mathbb{E}\left[\sum_{\ell=1}^{L} \int_{t_{\ell-1}}^{t_\ell} \langle s_\theta(t, Y(\hat{t})), \nabla \log p(t, Y(\hat{t}); t_\ell, Y(\hat{t_\ell})) \rangle_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon \mathrm{d}t\right]. \tag{40}$$

Then Equation (35a) is equal to

$$\sum_{\ell=1}^{L} \int_{t_{\ell-1}}^{t_\ell} \mathbb{E}\left[\|s_\theta(t, Y(\hat{t})) - \nabla \log p(t, Y(\hat{t}); t_\ell, Y(\hat{t_\ell}))\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon\right] \mathrm{d}t + \text{const.}, \tag{41}$$

where

$$\text{const.} = \sum_{\ell=1}^{L} \int_{t_{\ell-1}}^{t_\ell} \left[\|\nabla \log p(t, Y(\hat{t}); T, y)\|^2_{\sigma\sigma^\top} - \|\nabla \log p(t, Y(\hat{t}); t_\ell, Y(\hat{t_\ell}))\|^2_{\sigma\sigma^\top}\right] \mathcal{Y}(T) K_\epsilon \mathrm{d}t. \tag{42}$$

□

A.2 Proof of Theorem 4.2

Theorem. We assume the same setup as in Theorem 4.1. Define $\bar{\mathcal{L}}(s) \coloneqq \int \mathcal{L}(s, x_0) \mathrm{d}x_0$. Then, letting I_ℓ be defined as in Equation (16b)

$$\bar{\mathcal{L}}(s) \coloneqq \mathbb{E}\left[\frac{\mathcal{Y}(T)}{p(0, Y(T); T, y)} \sum_{l=1}^{L} I_\ell\right]. \tag{43a}$$

Proof. By definition

$$\int \mathcal{L}(s,x) \mathrm{d}x = \int \frac{1}{p(0,x;T,y)} \lim_{\epsilon \to 0} \sum_{l=1}^{L} \int_{t_{l-1}}^{t_l} \mathbb{E}[\|s(t, Y(\hat{t})) - \nabla \log p(t, Y(\hat{t}); t_l, Y(\hat{t_l}))\|^2_{\sigma\sigma^\top} \mathcal{Y}(T) K_\epsilon(Y(T), x)] \mathrm{d}t \mathrm{d}x. \tag{44}$$

Swapping the order of integration, this is equal to

$$\lim_{\epsilon\to 0}\sum_{l=1}^{L}\int_{t_{l-1}}^{t_l}\mathbb{E}\left[\int\frac{\mathcal{Y}(T)K_\epsilon(Y(T),x)}{p(0,x;T,y)}\|s(t,Y(\hat{t}))-\nabla\log p(t,Y(\hat{t});t_l,Y(\hat{t_l}))\|^2_{\sigma\sigma^\top}\mathrm{d}x\right]\mathrm{d}t. \tag{45}$$

Now noting that the only term to depend on x is $p(0,x;T,y)$, and that

$$\lim_{\epsilon\to 0}\int\frac{1}{p(0,x;T,y)}K_\epsilon(Y(T),x)\mathrm{d}x=\frac{1}{p(0,Y(T);T,y)}, \tag{46}$$

gives the result. □

B ALGORITHM

This is the exact algorithm we use to condition a given SDE on a given fixed point y, as described in Section 4.2.

Algorithm 1: SDE Bridge: Condition an SDE on a fixed endpoint y

input: Number of batches N, endpoint y, time grid $\{t_\ell\}_{\ell=1}^L$
$\hat{t}_\ell \leftarrow T - t_{L-\ell}$
while *not converged* **do**
  For given y, compute N trajectories $\{Y^y(\hat{t}_\ell)\}_{\ell=1}^L$
  $Z_\ell^y \leftarrow Y^y(\hat{t}_{L-\ell})$
  $g_\ell \leftarrow g(t_\ell, Z_\ell^y, t_{\ell+1}, Z_{\ell+1}^y)$
  $\mathcal{L}_\theta \leftarrow \frac{1}{(\Delta t)N}\sum_{\ell=0}^{L-1}\sum_{Z_\ell^y}\|[s_\theta(t_\ell,Z_\ell^y)-g_\ell]\|^2_{\sigma\sigma^\top}$
  Compute $\frac{\mathrm{d}\mathcal{L}_\theta}{d\theta}$ and use it to update s_θ
end

For a batch of N trajectories, and L time points per trajectory, with the SDE having dimension d the computational complexity is $O(N\cdot L\cdot d^3)$. The d^3 complexity comes from the covariance matrix arithmetic.

C EXPERIMENT DETAILS

Here we provide some more details for the individual experiment setups. For all of the experiments one NVIDIA RTX 4090 GPU and one Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz were used.

C.1 Ornstein-Uhlenbeck

We use the same Ornstein-Uhlenbeck formulation for both experiments as detailed in Equation (25). The adjoint processes are explicitly computed as follows:

$$\mathrm{d}Y(t)=Y(t)\mathrm{d}t+\mathrm{d}W(t) \qquad Y(0)=y \tag{47a}$$
$$\mathrm{d}\mathcal{Y}(t)=\mathcal{Y}(t)\mathrm{d}t \qquad \mathcal{Y}(0)=1. \tag{47b}$$

Moreover, the score function for the unconditioned process X_t is given by

$$\nabla_x\log p(t,x;s,y)=\frac{\exp\{-(s-t)\}}{v(s-t)}\left(y-m(s-t,x)\right), \tag{48}$$

where the functions m, v are the mean and variance defined as

$$m(t,x)=x\exp(-t) \qquad v(t)=\frac{1-\exp(-2t)}{2}. \tag{49}$$

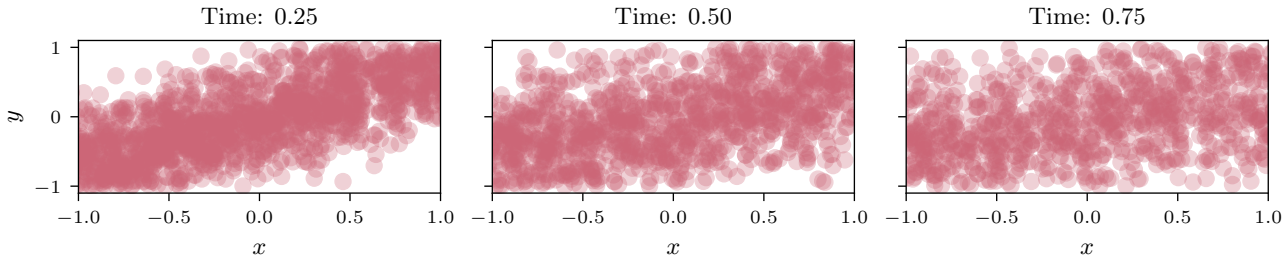

Figure 6: For the time points $t \in [0.25, 0.50, 0.75]$ we plot all the points $Y(t)$ of the generated adjoints that were used in training to learn the score $\nabla_x \log p(t, x; 1, y)$.

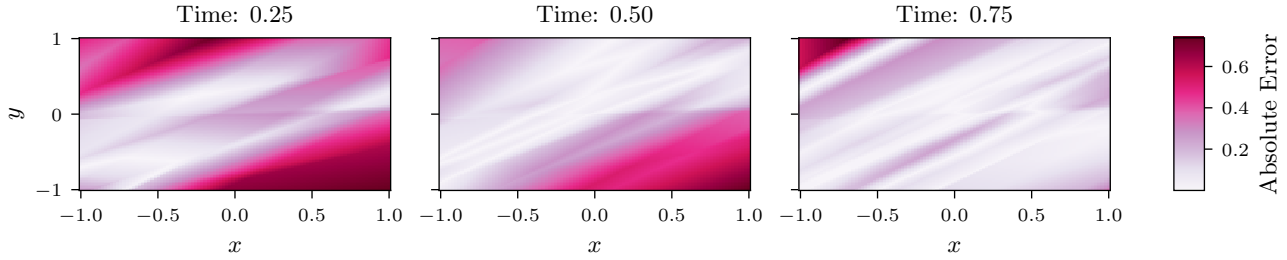

Figure 7: We plot the absolute error between the true score and the learned score from Figure 2.

Variable y The values y are sampled uniformly from the interval $[-1, 1]$. We train on 1000 iterations each with 1000 sample paths. These parameters are chosen following the similar experiment from Heng et al. (2022). For the neural network, we use the same as Heng et al. (2022): we use the sinusoidal embedding to encode the time steps (Vaswani et al., 2017), and use three multi-layer perceptron blocks with the Leaky ReLU activation function. In Figure 6 we plot the generated data at three different time steps, and in Figure 7 we plot the absolute error between the true score and learned score trained on said generated data. For smaller times, the data has not yet had so much time to disperse leading to higher errors in areas where there is a larger difference between x and y.

Distributed y For all methods we train on 100 iterations of 1000 samples, and use the same MLP network architecture used by Heng et al. (2022), as described in Appendix C.1. We train with five different random seeds and plot the average mean square error and the standard deviation between the different trainings in Figure 3.

C.2 Cell model

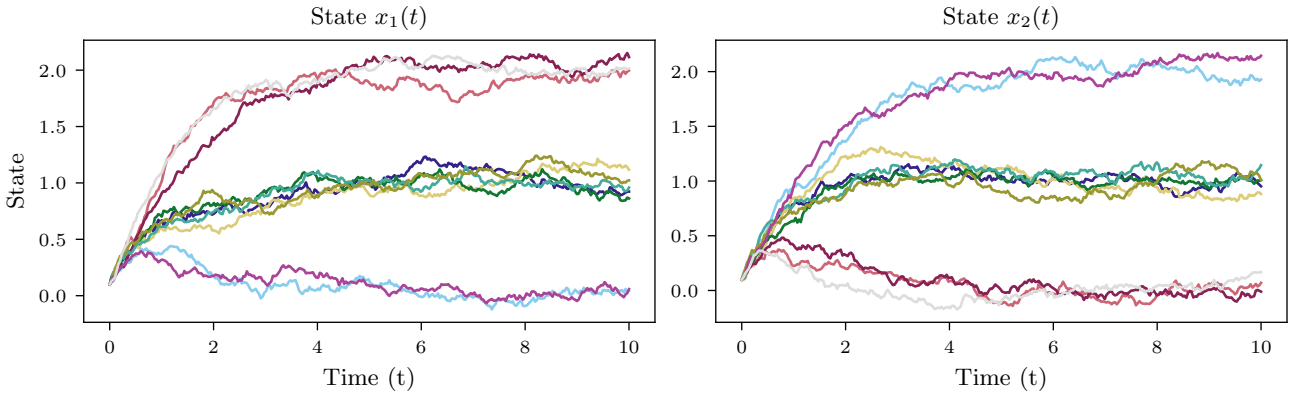

Figure 8: Trajectories from the unconditioned process of cell differentiation into three different states.

In Figure 8 we plot unconditional trajectories from the cell model defined in Equation (26). For the proposed method, the time-reversed bridge (DB (bw)) and the forward bridge (DB (fw)), we train on 10,000 trajectories in total with batch sizes of 100. We use the same network structure for both: again the architecture described in

Appendix C.1. However note the proposed method is written in Jax (Bradbury et al., 2018), whereas the others are implemented in Pytorch (Paszke et al., 2019), since we used the code from the authors[2]. For the guided method we use Schauer et al. (2017)'s code written in Julia [3]. We first simulate 1000 trajectories, without saving, and from then save every 100^{th} accepted trajectory, with $\rho = 0.7$.

C.3 Shape models

The SDE in Equation (29) represents the position of the points x_i at time t. We can equivalently look at the SDE representing the change of position, $Y(t) \coloneqq X(t) - x_0$, with the constraint $Y(0) = 0$:

$$Y(t) = \int \sum_{i=1}^{m} k(Y(t) + x_0, z_i) \mathrm{d}W(t). \tag{50}$$

In the shape experiment, we condition the process Y_t as Baker et al. (2024); Yang et al. (2024) do. This gives better results in training than conditioning the position X_t (Equation (29)) directly.

We train on a total of $409,600$ trajectories with a batch size of 128. We set $\sigma = 0.04$ and $\alpha = 1/\sigma$ for the Gaussian function k defining the SDE. For the grid points $\{z_i\}_{i=1}^m$ we take a 50×50 grid of evenly spaced values on $[-1.5, 1.5] \times [-1.5, 1.5]$. We train on 30 total points and sample 80 points after training. We use the neural network architecture proposed by Yang et al. (2024): a U-Net based Fourier neural operator architecture.

[2] `https://github.com/jeremyhengjm/DiffusionBridge`.

[3] Codebase at `https://github.com/mschauer/Bridge.jl` from file `project/partialbridge_fitzhugh.jl`