Scalable Gradients for Stochastic Differential Equations

Xuechen Li[*] Google Research
Ting-Kam Leonard Wong University of Toronto
Ricky T. Q. Chen University of Toronto, Vector Institute
David Duvenaud University of Toronto, Vector Institute

Abstract

The adjoint sensitivity method scalably computes gradients of solutions to ordinary differential equations. We generalize this method to stochastic differential equations, allowing time-efficient and constant-memory computation of gradients with high-order adaptive solvers. Specifically, we derive a stochastic differential equation whose solution is the gradient, a memory-efficient algorithm for caching noise, and conditions under which numerical solutions converge. In addition, we combine our method with gradient-based stochastic variational inference for latent stochastic differential equations. We use our method to fit stochastic dynamics defined by neural networks, achieving competitive performance on a 50-dimensional motion capture dataset.

1 Introduction

Deterministic dynamical systems can often be modeled by ordinary differential equations (ODEs). The adjoint sensitivity method can efficiently compute gradients of ODE solutions with constant memory cost. This method was well-known in the physics, numerical analysis, and control communities for decades [3, 4, 60, 65]. Recently, it was combined with modern reverse-mode automatic differentiation packages, enabling ODEs with millions of parameters to be fit to data [12] and allowing more flexible density estimation and time-series models [23, 32, 72].

Stochastic differential equations (SDEs) generalize ODEs, adding instantaneous noise to their dynamics [55, 77, 78]. They are a natural model for phenomena governed by many small and unobserved interactions, such as motion of molecules in a liquid [8], allele frequencies in a gene pool [15], or prices in a market [79]. Previous attempts on fitting SDEs mostly relied on methods with poor scaling properties. The *pathwise approach* [22, 89], a form of forward-mode automatic differentiation, scales poorly in time with the number of parameters and states in the model. On the other hand, simply differentiating through the operations of an SDE solver [19] scales poorly in memory.

In this work, we generalize the adjoint method to stochastic dynamics defined by SDEs. We give a simple and practical algorithm for fitting SDEs with tens of thousands of parameters, while allowing the use of high-order adaptive time-stepping SDE solvers. We call this approach the *stochastic adjoint sensitivity method*.

Method	Memory	Time
Forward pathwise [22, 89]	$\mathcal{O}(1)$	$\mathcal{O}(LD)$
Backprop through solver [19]	$\mathcal{O}(L)$	$\mathcal{O}(L)$
Stochastic adjoint (ours)	$\mathcal{O}(1)$	$\mathcal{O}(L \log L)$

Table 1: Asymptotic complexity comparison. L is the number of steps used in a fixed-step solve, and D is the number of state and parameters. Both memory and time are expressed in units of the cost of evaluating the drift and diffusion functions once each.

There are two main difficulties in generalizing the adjoint formulation for ODEs to SDEs. The first is mathematical: SDEs are defined using nonstandard integrals that usually rely on Itô calculus. The adjoint method requires solving the dynamics backwards in time from the end state. However, it is not clear exactly what "running the SDE backwards" means in the context of stochastic calculus, and when it correctly reconstructs the forward trajectory. We address this problem in Section 3, deriving a backward Stratonovich SDE whose dynamics compute the necessary gradient.

The second difficulty is computational: To retrace the steps, one needs to reconstruct the noise sampled on the forward pass, ideally without storing it. In Section 4, we give an algorithm that allows querying a Brownian motion sample at any time point arbitrarily-precisely,

[*]A portion of work done during AI Residency.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

$$X = \underline{b}\, t + \underline{\sigma}\, B_t$$

mean

$$= b(X, t) \cdot t + \cdots$$

$$\frac{dX}{dt} = f(X) \Rightarrow dX = f(X)\,dt$$

$$dX = f(X, t) \cdot t + \sigma(X, t)\, B_t$$

$$\Downarrow$$

$$X_0, X_1, X_2, \cdots$$

while only storing a single random seed.

We combine our adjoint approach with a gradient-based stochastic variational inference scheme for efficiently marginalizing over latent SDE models with arbitrary differentiable likelihoods. This model family generalizes several existing families such as latent ODEs [12, 72], Gaussian state-space models [36, 81], and deep Kalman filters [40], and can naturally handle irregularly-sampled times series and missing observations. We train latent SDEs on toy and real datasets, demonstrating competitive performance compared to existing approaches for dynamics modeling.

2 Background: Stochastic Flows

2.1 Adjoint Sensitivity Method

The adjoint sensitivity method is an efficient approach to solve control problems relying on the adjoint (co-state) system [65]. Chen et al. [12] used this method to compute the gradient with respect to parameters of a *neural ODE*, which is a particular model among many others inspired by the theory of dynamical systems [10, 11, 26, 44, 46, 74, 86]. The method, shown in Algorithm 1, is scalable, since the most costly computation is a vector-Jacobian product defining its backwards dynamics. In addition, since the gradient is obtained by solving another ODE, no intermediate computation is stored as in the case of regular backpropagation [73].

2.2 Stochastic Differential Equations

We briefly define SDEs: Consider a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\in\mathbb{T}}, P)$ on which an m-dimensional adapted Wiener process (aka Brownian motion) $\{W_t\}_{t\in\mathbb{T}}$ is defined. For a fixed terminal time $T > 0$, we denote by $\mathbb{T} = [0, T]$ the time horizon. We denote the ith component of W_t by $W_t^{(i)}$. A stochastic process $\{Z_t\}_{t\in\mathbb{T}}$ can be defined by an Itô SDE

$$Z_T = z_0 + \int_0^T b(Z_t, t)\ \mathrm{d}t + \sum_{i=1}^{m} \int_0^T \sigma_i(Z_t, t)\ \mathrm{d}W_t^{(i)} \quad (1)$$

where $z_0 \in \mathbb{R}^d$ is the starting state, and $b : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and $\sigma_i : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ are the drift and diffusion functions, respectively. For ease of presentation, we let $m = 1$ in the following unless otherwise stated. Our contributions can be easily generalized to cases where $m > 1$. Here, the second integral on the right hand side of (1) is the Itô stochastic integral [55]. When the coefficients are globally Lipschitz in both the state and time, there exists a unique strong solution to the SDE [55].

2.3 Neural Stochastic Differential Equations

Similar to neural ODEs, one can consider drift and diffusion functions defined by neural networks, a model known as the *neural SDE* [32, 45, 82, 83].

Amongst work on neural SDEs, none has enabled an efficient training framework. In particular, Tzen and Raginsky [82] and Liu et al. [45] considered computing the gradient by simulating the forward dynamics of an explicit Jacobian matrix. This Jacobian has size of either the square of the number of parameters, or the number of parameters times the number of states, building on the pathwise approach [22, 89]. In contrast, our approach only requires a small number of cheap vector-Jacobian products, independent of the dimension of the parameter and state vectors. These vector-Jacobian products have the same asymptotic time cost as evaluating the drift and diffusion functions, and can be easily computed by modern automatic differentiation libraries [1, 16, 49, 59].

2.4 Backward Stratonovich Integral

Our stochastic adjoint sensitivity method involves stochastic processes running both forward and backward in time. The Stratonovich stochastic integral, due to its symmetry, gives nice expressions for the backward dynamics and is more convenient for our purpose. Our results can be straightforwardly applied to Itô SDEs as well, using a simple conversion (see e.g. [64, Sec. 2]).

Following the treatment of Kunita [41], we introduce the forward and backward Stratonovich integrals. Let $\{\mathcal{F}_{s,t}\}_{s\le t; s,t\in\mathbb{T}}$ be a *two-sided filtration*, where $\mathcal{F}_{s,t}$ is the σ-algebra generated by $\{W_v - W_u : s \le u \le v \le t\}$ for $s, t \in \mathbb{T}$ such that $s \le t$. For a continuous semimartingale $\{Y_t\}_{t\in\mathbb{T}}$ adapted to the forward filtration $\{\mathcal{F}_{0,t}\}_{t\in\mathbb{T}}$, the *Stratonovich stochastic integral* is

$$\int_0^T Y_t \circ \mathrm{d}W_t = \lim_{|\Pi|\to 0} \sum_{k=1}^{N} \frac{(Y_{t_k} + Y_{t_{k-1}})}{2} \left(W_{t_k} - W_{t_{k-1}}\right),$$

where $\Pi = \{0 = t_0 < \cdots < t_N = T\}$ is a partition of the interval $\mathbb{T} = [0, T]$, $|\Pi| = \max_k t_k - t_{k-1}$ denotes the size of largest segment of the partition, and the limit is to be interpreted in the L^2 sense. The Itô integral uses instead the left endpoint Y_{t_k} rather than the average. In general, the Itô and Stratonovich integrals differ by a term of finite variation.

To define the backward Stratonovich integral, we consider the *backward Wiener process* $\{\widetilde{W}_t\}_{t\in\mathbb{T}}$ defined as $\widetilde{W}_t = W_t - W_T$ for all $t \in \mathbb{T}$ that is adapted to the backward filtration $\{\mathcal{F}_{t,T}\}_{t\in\mathbb{T}}$. For a continuous semimartingale $\breve{Y}_t$ adapted to the backward filtration,

Algorithm 1 ODE Adjoint Sensitivity

Input: Parameters θ, start time t_0, stop time t_1, final state z_{t_1}, loss gradient $\partial\mathcal{L}/z_{t_1}$, dynamics $f(z,t,\theta)$.

def $\overline{f}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented dynamics
  $v = f(z_t, -t, \theta)$
  return $[-v,\ a_t\partial v/\partial z,\ a_t\partial v/\partial\theta\,]$

$$\begin{bmatrix} z_{t_0} \\ \partial\mathcal{L}/\partial z_{t_0} \\ \partial\mathcal{L}/\partial\theta \end{bmatrix} = \texttt{odeint}\left(\begin{bmatrix} z_{t_1} \\ \partial\mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_p \end{bmatrix}, \overline{f}, -t_1, -t_0\right)$$

return $\partial\mathcal{L}/\partial z_{t_0}, \partial\mathcal{L}/\partial\theta$

Algorithm 2 SDE Adjoint Sensitivity (Ours)

Input: Parameters θ, start time t_0, stop time t_1, final state z_{t_1}, loss gradient $\partial\mathcal{L}/z_{t_1}$, drift $f(z,t,\theta)$, diffusion $\sigma(z,t,\theta)$, Wiener process sample $w(t)$.

def $\overline{f}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented drift
  $v = f(z_t, -t, \theta)$
  return $[-v,\ a_t\partial v/\partial z,\ a_t\partial v/\partial\theta]$

def $\overline{\sigma}([z_t, a_t, \cdot],\ t,\ \theta)$: ▷ Augmented diffusion
  $v = \sigma(z_t, -t, \theta)$
  return $[-v, a_t\partial v/\partial z, a_t\partial v/\partial\theta]$

def $\overline{w}(t)$: ▷ Replicated noise
  return $[-w(-t), -w(-t), -w(-t)]$

$$\begin{bmatrix} z_{t_0} \\ \partial\mathcal{L}/\partial z_{t_0} \\ \partial\mathcal{L}/\partial\theta \end{bmatrix} = \texttt{sdeint}\left(\begin{bmatrix} z_{t_1} \\ \partial\mathcal{L}/\partial z_{t_1} \\ \mathbf{0}_p \end{bmatrix}, \overline{f}, \overline{\sigma}, \overline{w}, -t_1, -t_0\right)$$

return $\partial\mathcal{L}/\partial z_{t_0}, \partial\mathcal{L}/\partial\theta$

Figure 1: Pseudocode of the (ODE) adjoint sensitivity method (*left*), and our generalization to Stratonovich SDEs (*right*). Differences are highlighted in blue. Square brackets denote vector concatenation.

the *backward Stratonovich integral* is

$$\int_s^T \breve{Y}_t \circ \mathrm{d}\breve{W}_t = \lim_{|\Pi|\to 0} \sum_{k=1}^{N} \frac{\left(\breve{Y}_{t_k} + \breve{Y}_{t_{k-1}}\right)}{2} \left(\breve{W}_{t_{k-1}} - \breve{W}_{t_k}\right),$$

where $\Pi = \{0 = t_N < \cdots < t_0 = T\}$ is the partition.

2.5 Stochastic Flow of Diffeomorphisms

It is well known that an ODE defines a flow of diffeomorphisms [6]. Here we consider the stochastic analog for the Stratonovich SDE

$$Z_T = z_0 + \int_0^T b(Z_t, t)\ \mathrm{d}t + \int_0^T \sigma(Z_t, t) \circ \mathrm{d}W_t. \quad (2)$$

Throughout the paper, we assume that both b and σ have infinitely many bounded derivatives w.r.t. the state, and bounded first derivatives w.r.t. time, i.e. $b, \sigma \in C_b^{\infty,1}$, so that the SDE has a unique strong solution. Let $\Phi_{s,t}(z) := Z_t^{s,z}$ be the solution at time t when the process is started at z at time s. Given a realization of the Wiener process, this defines a collection of continuous maps $\mathcal{S} = \{\Phi_{s,t}\}_{s\le t; s,t\in\mathbb{T}}$ from $\mathbb{R}^d$ to itself.

The following theorem shows that these maps are diffeomorphisms (after choosing a suitable modification) and that they satisfy backward SDEs.

Theorem 2.1 ([41, Theorem 3.7.1]). *(a) With probability* 1*, the collection* $\mathcal{S} = \{\Phi_{s,t}\}_{s\le t; s,t\in\mathbb{T}}$ *satisfies the flow property*

$$\Phi_{s,t}(z) = \Phi_{u,t}(\Phi_{s,u}(z)), \quad s \le u \le t,\ z \in \mathbb{R}^d.$$

Moreover, each $\Phi_{s,t}$ *is a smooth diffeomorphism from* $\mathbb{R}^d$ *to itself. We thus call* $\mathcal{S}$ *the stochastic flow of diffeomorphisms generated by the SDE (2).*

(b) The backward flow $\breve{\Psi}_{s,t} := \Phi_{s,t}^{-1}$ *satisfies the backward SDE:*

$$\breve{\Psi}_{s,t}(z) = z - \int_s^t b(\breve{\Psi}_{u,t}(z), u)\ \mathrm{d}u - \int_s^t \sigma(\breve{\Psi}_{u,t}(z), u) \circ \mathrm{d}\breve{W}_u, \quad (3)$$

for all $z \in \mathbb{R}^d$ *and* $s, t \in \mathbb{T}$ *such that* $s \le t$.

The coefficients in (2) and (3) differ by only a negative sign. This symmetry is due to our use of the Stratonovich integral (see Figure 2).

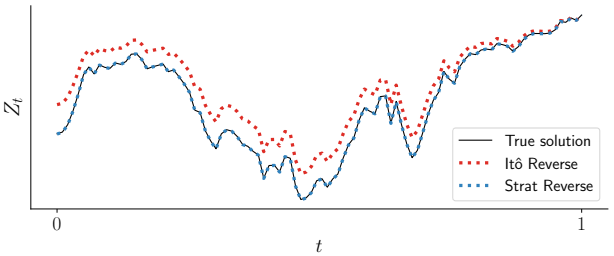


Figure 2: Negating the drift and diffusion functions for an Itô SDE and simulating backwards from the end state gives the wrong reconstruction. Negating the drift and diffusion functions for the converted Stratonovich SDE gives the same path when simulated backwards.

3 Sensitivity via Stochastic Adjoint

We present our main contribution: a stochastic analog of the adjoint sensitivity method for SDEs. We use (3) to derive another backward Stratonovich SDE, which we call the *stochastic adjoint process*. The direct implication is a gradient computation algorithm that works by solving a set of dynamics in reverse time, and relies on cheap vector-Jacobian products without storing any intermediate quantities.

3.1 Stochastic Adjoint Process

The goal is to derive a stochastic adjoint process $\{\partial\mathcal{L}/\partial Z_t\}_{t\in\mathbb{T}}$ that can be simulated by evaluating only vector-Jacobian products, where $\mathcal{L} = \mathcal{L}(Z_T)$ is a scalar loss of the terminal state from the forward flow $Z_T = \Phi_{0,T}(z_0)$.

We first derive a backward SDE for the process $\{\partial Z_T/\partial Z_t\}_{t\in\mathbb{T}}$, assuming that $Z_t = \breve{\Psi}_{t,T}(Z_T)$ follows the inverse flow from a deterministic end state $Z_T \in \mathbb{R}^d$ that does not depend on the realized Wiener process (Lemma 3.1). We then extend to the case where $Z_T = \Phi_{0,T}(z_0)$ is obtained by the forward flow starting from a deterministic initial state $z_0 \in \mathbb{R}^d$ (Theorem 3.2). This latter part is unconventional, and the resulting value cannot be interpreted as the solution to a backward SDE anymore due to loss of adaptedness. Instead, we will formulate the result with the *Itô map* [69]. Finally, it is straightforward to extend the state Z_t to include parameters of the drift and diffusion functions such that the desired gradient can be obtained for stochastic optimization; we comment on this step in Section 3.3.

We first present the SDE for the Jacobian matrix of the backward flow.

Lemma 3.1 (Dynamics of $\partial Z_T/\partial Z_t$)**.** *Consider the stochastic flow generated by the backward SDE (3) as in Theorem 2.1(b). Letting $J_{s,t}(z) := \nabla\breve{\Psi}_{s,t}(z)$, we have*

$$J_{s,t}(z) = I_d - \int_s^t \nabla b(\breve{\Psi}_{r,t}(z), r) J_{r,t}(z)\ \mathrm{d}r - \int_s^t \nabla\sigma(\breve{\Psi}_{r,t}(z), r) J_{r,t}(z) \circ \mathrm{d}\breve{W}_r, \quad (4)$$

for all $s \le t$ and $x \in \mathbb{R}^d$. Furthermore, letting $K_{s,t}(z) = [J_{s,t}(z)]^{-1}$, we have

$$K_{s,t}(z) = I_d + \int_s^t K_{r,t}(z)\nabla b(\breve{\Psi}_{r,t}(z), r)\ \mathrm{d}r + \int_s^t K_{r,t}(z)\nabla\sigma(\breve{\Psi}_{r,t}(z), r) \circ \mathrm{d}\breve{W}_r, \quad (5)$$

for all $s \le t$ and $x \in \mathbb{R}^d$.

The proof included in Appendix 9.1 relies on Itô's lemma in the Stratonovich form [41, Theorem 2.4.1]. We stress that this lemma considers only the case where the endpoint z is fixed and deterministic.

Now, we extend to the case where the endpoint is not deterministic, but rather computed from the forward flow. To achieve this, we compose the state process and the loss function. Consider $A_{s,t}(z) = \partial\mathcal{L}(\Phi_{s,t}(z))/\partial z$. The chain rule gives $A_{s,t}(z) = \nabla\mathcal{L}(\Phi_{s,t}(z))\nabla\Phi_{s,t}(z)$. Let

$$\breve{A}_{s,t}(z) := A_{s,t}(\breve{\Psi}_{s,t}(z)) = \nabla\mathcal{L}(z)\nabla\Phi_{s,t}(\breve{\Psi}_{s,t}(z)) = \nabla\mathcal{L}(z)K_{s,t}(z). \quad (6)$$

Note that $A_{s,t}(z) = \breve{A}_{s,t}(\Phi_{s,t}(z))$. Since $\nabla\mathcal{L}(z)$ is a constant, $(\breve{A}_{s,t}(z), \breve{\Psi}_{s,t}(z))$ satisfies the augmented backward SDE system

$$\begin{aligned}\breve{A}_{s,t}(z) =& \nabla\mathcal{L}(z) + \int_s^t \breve{A}_{r,t}(z)\nabla b(\breve{\Psi}_{r,t}(z), r)\ \mathrm{d}r + \int_s^t \breve{A}_{r,t}(z)\nabla\sigma(\breve{\Psi}_{r,t}(z), r) \circ \mathrm{d}\breve{W}_r,\\ \breve{\Psi}_{s,t}(z) =& z - \int_s^t b(\breve{\Psi}_{r,t}(z), r)\ \mathrm{d}r - \int_s^t \sigma(\breve{\Psi}_{r,t}(z), r) \circ \mathrm{d}\breve{W}_r.\end{aligned} \quad (7)$$

Since the drift and diffusion functions of this augmented system are $C_b^{\infty,1}$, the system has a unique strong solution. Let $s = 0$ and $t = T$. Since (7) admits a strong solution, we may write

$$\breve{A}_{0,T}(z) = \mathsf{F}(z, W_\cdot), \quad (8)$$

where $W_\cdot = \{W_t\}_{0\le t\le T}$ denotes the path of the Wiener process and

$$\mathsf{F} : \mathbb{R}^d \times C([0,1], \mathbb{R}^m) \to \mathbb{R}^d$$

is a deterministic measurable function (the Itô map) [69, Chapter V, Definition 10.9]. Intuitively, F can be thought as a black box that computes the solution to the backward SDE system (7) given the position z at time T and the realized Wiener process sample. Similarly, we let G be the solution map for the forward flow (2). The next theorem follows immediately from (6) and the definition of F.

Theorem 3.2. *For P-almost all $\omega \in \Omega$, we have*

$$A_{0,T}(z) = \breve{A}_{0,T}(\mathsf{G}(z, W_\cdot)) = \mathsf{F}(\mathsf{G}(z, W_\cdot), W_\cdot),$$

where $\mathsf{G}(z, W_\cdot) = \Phi_{0,T}(z)$.

Proof. This is a consequence of composing $A_{0,T}(z) = \breve{A}_{0,T}(\Phi_{0,T}(z))$ and (8). □

This shows that one can obtain the gradient by "composing" the backward SDE system (7) with the original forward SDE (2) and ends our continuous-time analysis.

3.2 Numerical Approximation

In practice, we compute solutions to SDEs with numerical solvers F_h and G_h, where $h = T/L$ denotes the mesh size of a fixed grid. The approximate algorithm thus outputs $\mathsf{F}_h(\mathsf{G}_h(z, W_\cdot), W_\cdot)$. The following theorem provides sufficient conditions for convergence.

Theorem 3.3. *Suppose the schemes F_h and G_h satisfy the following conditions: (i) $\mathsf{F}_h(z, W_\cdot) \to \mathsf{F}(z, W_\cdot)$ and $\mathsf{G}_h(z, W_\cdot) \to \mathsf{G}(z, W_\cdot)$ in probability as $h \to 0$, and (ii) for any $M > 0$, we have $\sup_{|z| \le M} |\mathsf{F}_h(z, W_\cdot) - \mathsf{F}(z, W_\cdot)| \to 0$ in probability as $h \to 0$. Then, for any starting point z of the forward flow, we have*

$$\mathsf{F}_h(\mathsf{G}_h(z, W_\cdot), W_\cdot) \to \mathsf{F}(\mathsf{G}(z, W_\cdot), W_\cdot) = A_{0,T}(z)$$

in probability as $h \to 0$.

See Appendix 9.2 for the proof. Usual schemes such as the Euler-Maruyama scheme (more generally Itô-Taylor schemes) converge pathwise (i.e. almost surely) from any fixed starting point [38] and satisfies (i). While (ii) is strong, we note that the SDEs considered here have smooth coefficients, and thus their solutions enjoy nice regularity properties in the starting position. Therefore, it is reasonable to expect that the corresponding numerical schemes to also behave nicely as a function of *both* the mesh size and the starting position. To the best of our knowledge, this property is not considered at all in the literature on numerical methods for SDEs (where the initial position is fixed), but is crucial in the proof of Theorem 3.3. In Appendix 9.3, we prove that condition (ii) holds for the Euler-Maruyama scheme. Detailed analysis for other schemes is beyond the scope of this paper.

3.3 The Algorithm

So far we have derived the gradient of the loss with respect to the initial state. We can extend these results to give gradients with respect to parameters of the drift and diffusion functions by treating them as an additional part of the state whose dynamics has zero drift and diffusion. We summarize this in Algorithm 2, assuming access only to a black-box solver `sdeint`. All terms in the augmented dynamics, such as $a_t \partial f/\partial\theta$ and $a_t \partial\sigma/\partial\theta$ can be cheaply evaluated by calling $\texttt{vjp}(a_t, f, \theta)$ and $\texttt{vjp}(a_t, \sigma, \theta)$, respectively.

Difficulties with non-diagonal diffusion. In principle, we can simulate the forward and backward adjoint dynamics with any high-order solver of choice. However, for general matrix-valued diffusion functions σ, to obtain a numerical solution with strong order[1] beyond 1/2, we need to simulate multiple integrals of the Wiener process such as $\int_0^t \int_0^s \mathrm{d}W_u^{(i)}\, \mathrm{d}W_s^{(j)}$, $i, j \in [m], i \neq j$. These random variables are difficult to simulate and costly to approximate [87].

Fortunately, if we restrict our SDE to have diagonal noise, then even though the backward SDE for the stochastic adjoint will not in general have diagonal noise, it will satisfy a commutativity property [70]. In that case, we can safely adopt certain numerical schemes of strong order 1.0 (e.g. Milstein [52] and stochastic Runge-Kutta [71]) without approximating multiple integrals or the Lévy area during simulation. We formally show this in Appendix 9.4.

One may also consider numerical schemes with high weak order [39]. However, analysis of this scenario is beyond the current scope.

3.4 Software and Implementation

We have implemented several common SDE solvers in PyTorch [59] with adaptive time-stepping using a PI controller [9, 30]. Following `torchdiffeq` [12], we have created a user-friendly subclass of `torch.autograd.Function` that facilitates gradient computation using our stochastic adjoint framework for SDEs that are subclasses of `torch.nn.Module`. We include a short code snippet covering the main idea of the stochastic adjoint in Appendix 9.12. The complete codebase can be found at https://github.com/google-research/torchsde.

4 Virtual Brownian Tree

Our formulation of the adjoint can be numerically integrated efficiently, since simulating its dynamics only requires evaluating cheap vector-Jacobian products, as opposed to whole Jacobians. However, the backward-in-time nature introduces a new difficulty: The same Wiener process sample path used in the forward pass must be queried again during the backward pass. Naïvely storing Brownian motion increments implies a large memory consumption and complicates the usage of adaptive time-stepping integrators, where the evaluation times in the backward pass may be different from those in the forward pass.

To overcome this issue, we combine Brownian trees with splittable pseudorandom number generators (PRNGs) to give an algorithm that can query values of a Wiener

[1] A numerical scheme is of strong order p if $\mathbb{E}\left[|X_T - X_{N\eta}|\right] \le C\eta^p$ for all $T > 0$, where X_t and $X_{N\eta}$ are respectively the coupled true solution and numerical solution, N and η are respectively the iteration index and step size such that $N\eta = T$, and C is independent of η.

process sample path at arbitrary times. This algorithm, which we call the *virtual Brownian tree*, has $\mathcal{O}(1)$ memory cost, and time cost logarithmic with respect to the inverse error tolerance.

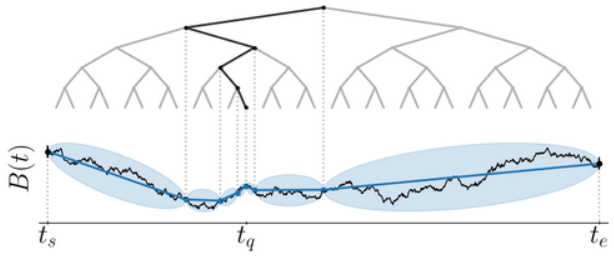


Figure 3: Evaluating a Brownian motion sample at time t_q using a virtual Brownian tree. Our algorithm repeatedly bisects the interval, sampling from a Brownian bridge at each halving to determine intermediate values. Each call to the random number generator uses a unique key whose value depends on the path taken to reach it.

4.1 Brownian Bridges and Brownian Trees

Lévy's *Brownian bridge* [67] states that given a start time t_s and end time t_e along with their respective Wiener process values w_s and w_e, the marginal of the process at time $t \in (t_s, t_e)$ is a normal distribution:

$$\mathcal{N}\left(\frac{(t_e - t)w_s + (t - t_s)w_e}{t_e - t_s}, \frac{(t_e - t)(t - t_s)}{t_e - t_s} I_d\right). \quad (9)$$

We can recursively apply this formula to evaluate the process at the midpoint of any two distinct timestamps where the values are already known. Constructing the whole sample path of a Wiener process in this manner results in what is known as the *Brownian tree* [17]. Storing this tree would be memory-intensive, but we show how to reconstruct any node in this tree as desired.

4.2 Brownian Trees using Splittable Seeds

We assume access to a splittable PRNG [14], which has an operation `split` that deterministically generates two keys from an existing key. Given a key, the function `BrownianBridge` samples deterministically from (9). To obtain the Wiener process value at a specific time, we must first know or sample the values at the initial and terminal times. Then, the virtual Brownian tree recursively samples from the midpoint of Brownian bridges, each sample using a key split from that of its parent node. The algorithm terminates when the most recently sampled time is close enough to the desired time. We outline the full procedure in Algorithm 3.

Algorithm 3 Virtual Brownian Tree

Input: Seed s, query time t, error tolerance ϵ, start time t_s, start state w_s, end time t_e, end state w_e.
  $t_{\text{mid}} = (t_s + t_e)/2$
  $w_{\text{mid}} = \texttt{BrownianBridge}(t_s, w_s, t_e, w_e, t_{\text{mid}}, s)$
  while $|t - t_{\text{mid}}| > \epsilon$ **do**
    $s_l, s_r = \texttt{split}(s)$
    if $t < t_{\text{mid}}$ **then** $t_e, x_e, s = t_{\text{mid}}, w_{\text{mid}}, s_l$
    else $t_s, x_s, s = t_{\text{mid}}, w_{\text{mid}}, s_r$
    end if
    $t_{\text{mid}} = (t_s + t_e)/2$
    $w_{\text{mid}} = \texttt{BrownianBridge}(t_s, w_s, t_e, w_e, t_{\text{mid}}, s)$
  end while
  return w_{mid}

This algorithm has constant memory cost. For a fixed-step-size solver taking L steps, the tolerance that the tree will need to be queried at scales as $1/L$. Thus the per-step time complexity scales as $\log L$. Our implementation uses an efficient *count-based PRNG* [76] which avoids passing large random states, and instead simply passes integers. Table 1 compares the asymptotic time complexity of this approach against existing alternatives.

5 Latent Stochastic Differential Equations

The algorithms presented in Sections 3 and 4 allow us to efficiently compute gradients of scalar objectives with respect to SDE parameters, letting us fit SDEs to data. This raises the question: Which loss to optimize?

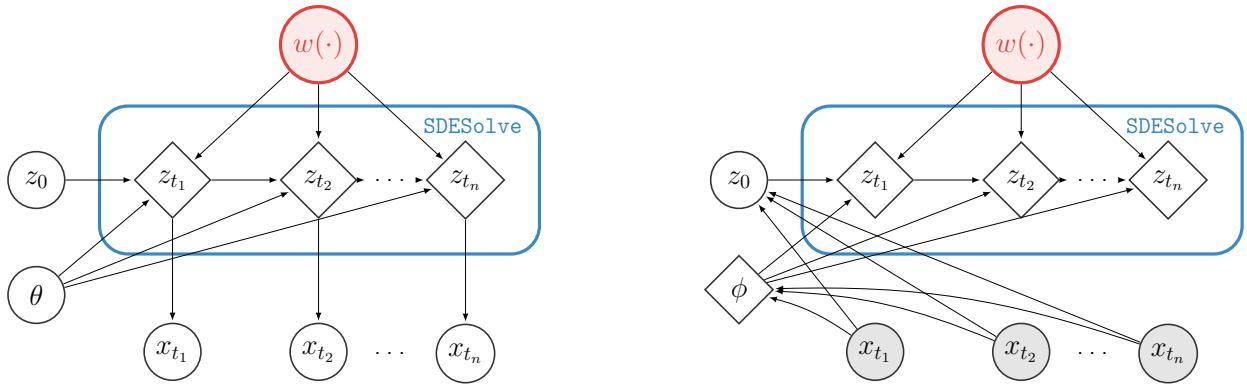
Simply fitting SDE parameters to maximize likelihood will in general cause overfitting, and will result in the diffusion function going to zero. In this section, we show how to do efficient variational inference in SDE models, and optimize the marginal log-likelihood to fit both prior (hyper-)parameters and the parameters of a tractable approximate posterior over functions.

In particular, we can parameterize both a prior over functions and an approximate posterior using SDEs:

$$\mathrm{d}\tilde{Z}_t = h_\theta(\tilde{Z}_t, t)\ \mathrm{d}t + \sigma(\tilde{Z}_t, t)\,\mathrm{d}W_t, \quad \text{(prior)}$$
$$\mathrm{d}Z_t = h_\phi(Z_t, t)\ \mathrm{d}t + \sigma(Z_t, t)\ \mathrm{d}W_t, \quad \text{(approx. post.)}$$

where h_θ, h_ϕ, and σ are Lipschitz in both arguments, and both processes have the same starting value: $\tilde{Z}_0 = Z_0 = z_0 \in \mathbb{R}^d$.

If both processes share the same diffusion function σ, then the KL divergence between them is finite (under additional mild regularity conditions; see Appendix 9.6), and can be estimated by sampling paths from the posterior process. Then, the evidence lower



(a) Generation (b) Recognition

Figure 4: Graphical models for the generative process (decoder) and recognition network (encoder) of the latent stochastic differential equation model. This model can be viewed as a variational autoencoder with infinite-dimensional noise. Red circles represent entire function draws from Brownian motion. Given the initial state z_0 and a Brownian motion sample path $w(\cdot)$, the intermediate states $z_{t_1}, \ldots, z_{t_n}$ are deterministically approximated by a numerical SDE solver.

bound (ELBO) can be written as:

$$\log p(x_1, x_2, \ldots, x_N|\theta) \geq \tag{10}$$
$$\mathbb{E}_{Z_t}\left[\sum_{i=1}^{N} \log p(x_{t_i}|z_{t_i}) - \int_0^T \frac{1}{2}|u(z_t,t)|^2 \, \mathrm{d}t\right],$$

where $u : \mathbb{R}^d \times [0,T] \to \mathbb{R}^m$ satisfies

$$\sigma(z,t)u(z,t) = h_\phi(z,t) - h_\theta(z,t),$$

and the expectation is taken over the approximate posterior process defined by (approx. post.). The likelihoods of the observations $x_1, \ldots, x_N$ at times $t_1, \ldots, t_N$ depend only on the latent states z_t at corresponding times.

To compute the gradient with respect to prior parameters θ and variational parameters ϕ, we need only augment the forward SDE with an extra scalar variable whose drift function is $\frac{1}{2}|u(Z_t,t)|^2$ and whose diffusion function is zero. The backward dynamics can be derived analogously using (7). We include a detailed derivation in Appendix 9.6. Thus, a stochastic estimate of the gradients of the loss w.r.t. all parameters can be computed in a single pair of forward and backward SDE solves.

The variational parameters ϕ can either be optimized individually for each sequence, or if multiple time series are sharing parameters, then an encoder network can be trained to input the observations and output ϕ. This architecture, shown in figure 4, can be viewed as an infinite-dimensional variational autoencoder [35, 68].

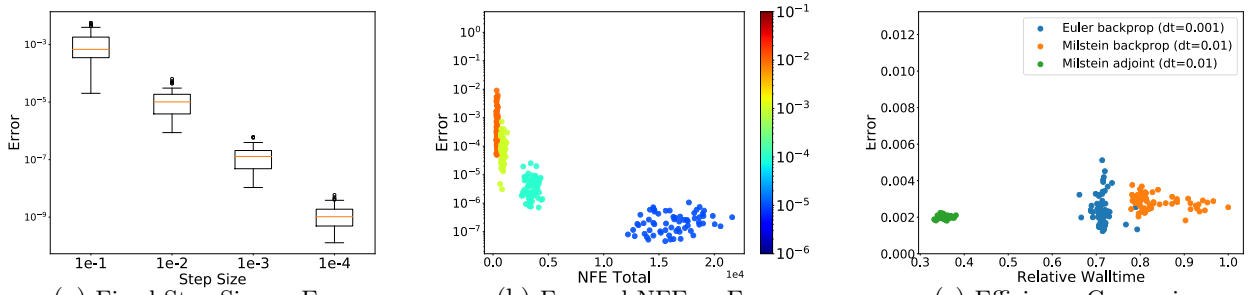
6 Related Work

Sensitivity Analysis for SDEs. Gradient computation is closely related to sensitivity analysis. Computing gradients with respect to parameters of vector fields of an SDE has been extensively studied in the stochastic control literature [42]. In particular, for low dimensional problems, this is done effectively using dynamic programming [7] and finite differences [20, 43]. However, both approaches scale poorly with the dimensionality of the parameter vector.

Analogous to REINFORCE (or the score-function estimator) [21, 37, 88], Yang and Kushner [89] considered deriving the gradient as $\nabla\mathbb{E}[\mathcal{L}(Z_T)] = \mathbb{E}[\mathcal{L}(Z_T)H]$ for some random variable H. However, H usually depends on the density of Z_T with respect to the Lebesgue measure which can be difficult to compute. Gobet and Munos [22] extended this approach by weakening a non-degeneracy condition using Mallianvin calculus [53].

Closely related to the current approach is the pathwise method [89], which is also a continuous-time analog of the reparameterization trick [35, 68]. Existing methods in this regime [22, 45, 82] all require simulating a (forward) SDE where each step requires computing entire Jacobian matrices. This computational cost is prohibitive for high-dimensional systems with a large number of parameters.

Based on the Euler discretization, Giles and Glasserman [19] considered simply performing reverse-mode automatic differentiation through all intermediate steps. They named this method the *adjoint approach*, which, by modern standards, is a form of "backpropagation through the operations of a numerical solver". This approach, widely adopted in the field of finance for calibrating market models [19], has high memory cost, and relies on a fixed Euler-Maruyama discretization. Recently, this approach was also used by Hegde et al. [27] to learn parameterized drift and diffusion functions



(a) Fixed Step Size vs Error (b) Forward NFE vs Error (c) Efficiency Comparison

Figure 5: (a) Same fixed step size used in both forward and reverse simulation. Boxplot generated by repeating the experiment with different Brownian motion sample paths 64 times. (b) Colors of dots represent tolerance levels and correspond to the colorbar on the right. Only `atol` was varied and `rtol` was set to 0.

of an SDE. In scientific computing, Innes et al. [31] considered backpropagating through high-order implicit SDE solvers.

In the machine learning literature, Ryder et al. [75] perform variational inference over the state and parameters for Euler-discretized latent SDEs and optimize the model with regular backpropagation. This approach should not be confused with the formulation of variational inference for non-discretized SDEs presented in previous works [25, 57, 82] and our work, as it is unclear whether the limit of their discretization corresponds to that obtained by operating with continuous-time SDEs using Girsanov's theorem.

Backward SDEs. Our stochastic adjoint process relies on the notion of backward SDEs devised by Kunita [41], which is based on two-sided filtrations. This is different from the more traditional notion of backward SDEs where only a single filtration is defined [58, 62]. Based on the latter notion, forward-backward SDEs (FBSDEs) have been proposed to solve stochastic optimal control problems [63]. However, simulating FBSDEs is costly due to the need to estimate conditional expectations in the backward pass [58].

Bayesian Learning of SDEs. Recent works considered the problem of inferring an approximate posterior SDE given observed data under a prior SDE with the same diffusion coefficient [25, 57, 82]. The special case with constant diffusion coefficients was considered more than a decade ago [5]. Notably, computing the KL divergence between two SDEs over a finite time horizon was well-explored in the control literature [33, 80]. We include background on this topic in Appendix 9.5.

Bayesian learning and parameter estimation for SDEs have a long history [24]. Techniques which don't require positing a variational family such as theh extended Kalman filter and Markov chain Monte Carlo have been considered in the literature [50].

7 Experiments

The aim of this section is threefold. We first empirically verify our theory by comparing the gradients obtained by our stochastic adjoint framework against analytically derived gradients for problems having closed-form solutions. We then fit latent SDE models with our framework on two synthetic datasets, verifying that the variational inference framework allows learning a generative model of time series. Finally, we learn dynamics parameterized by neural networks with a latent SDE from a motion capture dataset, demonstrating competitive performance compared to existing approaches.

We report results based on an implementation of Brownian motion that stores all intermediate queries. The virtual Brownian tree allowed training with much larger batch sizes on GPUs, but was not necessary for our small-scale experiments. Notably, our adjoint approach, even when combined with the Brownian motion implementation that stores noise, was able to reduce the memory usage by 1/2-1/3 compared to directly backpropagating through solver operations on the tasks we considered.

7.1 Numerical Studies

We consider three test problems (examples 1-3 from [66]; details in Appendix 9.7), all of which have closed-form solutions. We compare the gradient computed from simulating our stochastic adjoint process using the Milstein scheme against the exact gradient. Figure 5(a) shows that for test example 2, the error between the adjoint gradient and analytical gradient decreases with step size.

For all three test problems, the mean squared error across dimensions tends to be smaller as the absolute

tolerance of the adaptive solver is reduced (e.g. see Fig. 5 (b)). However, the Number of Function Evaluations (NFEs) tends to be much larger than that in the ODE case [12].

Additionally, for two out of three test problems, we found that our adjoint approach with the Milstein scheme and fixed step size can be much more time-efficient than regular backpropagation through operations of the Milstein and Euler schemes (see e.g. Fig. 5(c)). Backpropagating through the Euler scheme gives gradients of higher error compared to the Milstein method. On the other hand, directly backpropagating through the Milstein solve requires evaluating high-order derivatives and can be costly.

Results for examples 1 and 3 are in Appendix 9.8.

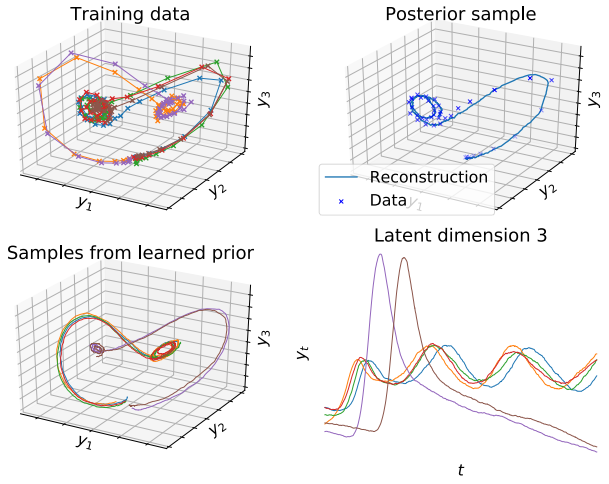


Figure 6: Learned posterior and prior dynamics on data from a stochastic Lorenz attractor. All samples from our model are continuous-time paths, and form a multi-modal, non-Gaussian distribution.

7.2 Synthetic Datasets

We trained latent SDEs with our adjoint framework to recover (1) a 1D Geometric Brownian motion, and (2) a 3D stochastic Lorenz attractor process. The main objective is to verify that the learned posterior can reconstruct the training data, and that the learned priors are not deterministic. We jointly optimize the evidence lower bound (10) with respect to parameters of the prior and posterior distributions at the initial latent state z_0, the prior and posterior drift, the diffusion function, the encoder, and the decoder. We include the details of datasets and architectures in Appendix 9.9.

For the stochastic Lorenz attractor, not only is the model able to reconstruct the data well, but also the learned prior process can produce bimodal samples in both data and latent space. This is showcased in the last row of Figure 6 where the latent and data space samples cluster around two modes. This is hard to achieve using a latent ODE with a unimodal Gaussian initial approximate posterior. We include additional visualizations in Appendix 9.10.

7.3 Motion Capture Dataset

To demonstrate that latent SDEs can learn complex dynamics from real-world datasets, we evaluated their predictive performance on a 50-dimensional motion capture dataset. The dataset, from Gan et al. [18], consists of 23 walking sequences of subject 35 partitioned into 16 training, 3 validation, and 4 test sequences. We follow the preprocessing of Wang et al. [85].

In designing the recognition network, we follow Yıldız et al. [90] and use a fully connected network to encode the first three observations of each sequence and thereafter predicted the remaining sequence. This encoder is chosen for fair comparison to existing models, and could be extended to a recurrent or attention model [84]. The overall architecture is described in Appendix 9.11 and is similar to that of ODE^{2}VAE [90], with a similar number of parameters. We also use a fixed step size 1/5 of smallest interval between any two observations [90].

We train latent ODE and latent SDE models with the Adam optimizer [34] and its default hyperparameter settings, with an initial learning rate of 0.01 that is exponentially decayed with rate 0.999 during each iteration. We perform validation over the number of training iterations, KL penalty [29], and KL annealing schedule. All models were trained for at most 400 iterations, where we start to observe severe overfitting for most model instances. We report the test MSE on future observations following Yıldız et al. [90]. We believe that the improved performance is due to the strong regularization in path space, as removing the KL penalty improve training error but caused validation error to deteriorate.

Table 2: Test MSE on 297 future frames averaged over 50 samples. 95% confidence interval reported based on t-statistic. †results from [90].

Method	Test MSE
DTSBN-S [18]	$34.86 \pm 0.02^\dagger$
npODE [28]	$22.96^\dagger$
NeuralODE [12]	$22.49 \pm 0.88^\dagger$
ODE^{2}VAE [90]	$10.06 \pm 1.4^\dagger$
ODE^{2}VAE-KL [90]	$8.09 \pm 1.95^\dagger$
Latent ODE [12, 72]	5.98 ± 0.28
Latent SDE (this work)	$\mathbf{4.03 \pm 0.20}$

8 Discussion

We presented a generalization of the adjoint sensitivity method to compute gradients through solutions of SDEs. In contrast to existing approaches, this method has nearly the same time and memory complexity as simply solving the SDE. We showed how our stochastic adjoint framework can be combined with a gradient-based stochastic variational inference scheme for training latent SDEs.

It is worthwhile to mention that SDEs and the commonly used GP models define two distinct classes of stochastic processes, albeit having a nonempty intersection (e.g. Ornstein-Uhlenbeck processes fall under both). Computationally, the cost of fitting GPs lies in the matrix inversion, whereas the computational bottleneck of training SDEs is the sequential numerical solve. Empirically, another avenue of research is to reduce the variance of gradient estimates. In the future, we may adopt techniques such as control variates or antithetic paths.

On the application side, our method opens up a broad set of opportunities for fitting any differentiable SDE model, such as Wright-Fisher models with selection and mutation parameters [15], derivative pricing models in finance, or infinitely-deep Bayesian neural networks [61]. In addition, the latent SDE model enabled by our framework can be extended to include domain knowledge and structural or stationarity constraints [48] in the prior process for specific applications.

On the theory side, there remain fundamental questions to be answered. Convergence rates of numerical gradients estimated with general schemes are unknown. Additionally, since our analyses are based on strong orders of schemes, it is natural to question whether convergence results still hold when we consider weak errors, and moreover if the method could be reformulated more coherently with rough paths theory [47].

Acknowledgements

We thank Yulia Rubanova, Danijar Hafner, Mufan Li, Shengyang Sun, Kenneth R. Jackson, Simo Särkkä, Daniel Lacker, and Philippe Casgrain for helpful discussions. We thank Çağatay Yıldız for helpful discussions regarding evaluation settings of the mocap task. We also thank Guodong Zhang, Kevin Swersky, Chris Rackauckas, and members of the Vector Institute for helpful comments on an early draft of this paper.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016.

[2] R Adams. *Sobolev Spaces*. Academic Press, 1975.

[3] Joel Andersson. *A general-purpose software framework for dynamic optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, 2013.

[4] Joel Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[5] Cédric Archambeau, Manfred Opper, Yuan Shen, Dan Cornford, and John S Shawe-Taylor. Variational inference for diffusion processes. In *Advances in Neural Information Processing Systems*, pages 17–24, 2008.

[6] VI Arnold. *Ordinary Differential Equations*. The MIT Press, 1978.

[7] Jonathan Baxter and Peter L Bartlett. Infinite-horizon gradient-based policy search. 2001.

[8] Robert Brown. ... microscopical observations ... on the particles contained in the pollen of plants. *The Philosophical Magazine*, 4(21):161–173, 1828.

[9] Pamela M Burrage, R Herdiana, and Kevin Burrage. Adaptive stepsize based on control theory for stochastic differential equations. *Journal of Computational and Applied Mathematics*, 170(2): 317–336, 2004.

[10] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*, 2017.

[11] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] Ricky Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[14] Koen Claessen and Michał H Pałka. Splittable pseudorandom number generators using crypto-

graphic hashing. In *ACM SIGPLAN Notices*, volume 48, pages 47–58. ACM, 2013.

[15] Warren J Ewens. *Mathematical population genetics 1: theoretical introduction*, volume 27. Springer Science & Business Media, 2012.

[16] Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing, 2018.

[17] Jessica G Gaines and Terry J Lyons. Variable step size control in the numerical solution of stochastic differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, 1997.

[18] Zhe Gan, Chunyuan Li, Ricardo Henao, David E Carlson, and Lawrence Carin. Deep temporal sigmoid belief networks for sequence modeling. In *Advances in Neural Information Processing Systems*, pages 2467–2475, 2015.

[19] Mike Giles and Paul Glasserman. Smoking adjoints: Fast Monte Carlo greeks. *Risk*, 19(1):88–92, 2006.

[20] Paul Glasserman and David D Yao. Some guidelines and guarantees for common random numbers. *Management Science*, 38(6):884–908, 1992.

[21] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

[22] Emmanuel Gobet and Rémi Munos. Sensitivity analysis using Itô–Malliavin calculus and martingales, and application to stochastic optimal control. *SIAM Journal on control and optimization*, 43(5): 1676–1713, 2005.

[23] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

[24] Narendra Gupta and Raman Mehra. Computational aspects of maximum likelihood estimation and reduction in sensitivity function calculations. *IEEE transactions on automatic control*, 19(6): 774–783, 1974.

[25] Jung-Su Ha, Young-Jin Park, Hyeok-Joo Chae, Soon-Seo Park, and Han-Lim Choi. Adaptive path-integral autoencoders: Representation learning and planning for dynamical systems. In *Advances in Neural Information Processing Systems*, pages 8927–8938, 2018.

[26] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

[27] Pashupati Hegde, Markus Heinonen, Harri Lähdesmäki, and Samuel Kaski. Deep learning with differential gaussian process flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1812–1821, 2019.

[28] Markus Heinonen, Cagatay Yildiz, Henrik Mannerström, Jukka Intosalmi, and Harri Lähdesmäki. Learning unknown ode models with gaussian processes. *arXiv preprint arXiv:1803.04303*, 2018.

[29] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[30] Silvana Ilie, Kenneth R Jackson, and Wayne H Enright. Adaptive time-stepping for the strong numerical solution of stochastic differential equations. *Numerical Algorithms*, 68(4):791–812, 2015.

[31] Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckus, Elliot Saba, Viral B Shah, and Will Tebbutt. Zygote: A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*, 2019.

[32] Junteng Jia and Austin R. Benson. Neural Jump Stochastic Differential Equations. *arXiv e-prints*, art. arXiv:1905.10403, May 2019.

[33] Hilbert Johan Kappen and Hans Christian Ruiz. Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162(5): 1244–1266, 2016.

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[35] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[36] Genshiro Kitagawa and Will Gersch. Linear gaussian state space modeling. In *Smoothness Priors Analysis of Time Series*, pages 55–65. Springer, 1996.

[37] Jack PC Kleijnen and Reuven Y Rubinstein. Optimization and sensitivity analysis of computer simulation models by the score function method. *European Journal of Operational Research*, 88(3): 413–427, 1996.

[38] Peter E Kloeden and Andreas Neuenkirch. The pathwise convergence of approximation schemes for stochastic differential equations. *LMS journal of Computation and Mathematics*, 10:235–253, 2007.

[39] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*,

volume 23. Springer Science & Business Media, 2013.

[40] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[41] Hiroshi Kunita. *Stochastic Flows and Jump-Diffusions*. Springer, 2019.

[42] Harold Kushner and Paul G Dupuis. *Numerical methods for stochastic control problems in continuous time*, volume 24. Springer Science & Business Media, 2013.

[43] Pierre L'Ecuyer and Gaétan Perron. On the convergence rates of ipa and fdc derivative estimators. *Operations Research*, 42(4):643–656, 1994.

[44] Qianxiao Li, Long Chen, Cheng Tai, and E Weinan. Maximum principle based algorithms for deep learning. *The Journal of Machine Learning Research*, 18(1):5998–6026, 2017.

[45] Xuanqing Liu, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise. *arXiv preprint arXiv:1906.02355*, 2019.

[46] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *arXiv preprint arXiv:1710.10121*, 2017.

[47] Terry J Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310, 1998.

[48] Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

[49] Dougal Maclaurin, David Duvenaud, M Johnson, and RP Adams. Autograd: Reverse-mode differentiation of native python. In *ICML workshop on Automatic Machine Learning*, 2015.

[50] Isambi S Mbalawata, Simo Särkkä, and Heikki Haario. Parameter estimation in stochastic differential equations with markov chain monte carlo and non-linear kalman filtering. *Computational Statistics*, 28(3):1195–1223, 2013.

[51] Grigori Noah Milstein and Michael V Tretyakov. *Stochastic Numerics for Mathematical Physics*. Springer Science & Business Media, 2013.

[52] Grigorii Noikhovich Milstein. *Numerical integration of stochastic differential equations*, volume 313. Springer Science & Business Media, 1994.

[53] Ivan Nourdin and Giovanni Peccati. *Normal approximations with Malliavin calculus: from Stein's method to universality*, volume 192. Cambridge University Press, 2012.

[54] Daniel Ocone and Étienne Pardoux. A generalized itô-ventzell formula. application to a class of anticipating stochastic differential equations. 25 (1):39–71, 1989.

[55] Bernt Øksendal. *Stochastic Differential Equations*. Springer, 2003.

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

[57] Manfred Opper. Variational inference for stochastic differential equations. *Annalen der Physik*, 531 (3):1800233, 2019.

[58] Etienne Pardoux and Shige Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In *Stochastic Partial Differential Equations and Their Applications*, pages 200–217. Springer, 1992.

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

[60] Barak A Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural networks*, 6(5):1212–1228, 1995.

[61] Stefano Peluchetti and Stefano Favaro. Neural stochastic differential equations. *arXiv preprint arXiv:1904.01681*, 2019.

[62] Shige Peng. A general stochastic maximum principle for optimal control problems. *SIAM Journal on Control and Optimization*, 28(4):966–979, 1990.

[63] Shige Peng and Zhen Wu. Fully coupled forward-backward stochastic differential equations and applications to optimal control. *SIAM Journal on Control and Optimization*, 37(3):825–843, 1999.

[64] Eckhard Platen. An introduction to numerical methods for stochastic differential equations. *Acta numerica*, 8:197–246, 1999.

[65] Lev Semenovich Pontryagin. *Mathematical Theory of Optimal Processes*. Routledge, 2018.

[66] Christopher Rackauckas and Qing Nie. Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. *Discrete and Continuous Dynamical Systems. Series B*, 22(7):2731, 2017.

[67] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media, 2013.

[68] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[69] L Chris G Rogers and David Williams. *Diffusions, Markov Processes and Martingales: Volume 2, Itô Calculus*, volume 2. Cambridge University Press, 2000.

[70] Andreas Rößler. Runge–Kutta methods for stratonovich stochastic differential equation systems with commutative noise. *Journal of Computational and Applied mathematics*, 164:613–627, 2004.

[71] Andreas Rößler. Runge–Kutta methods for the strong approximation of solutions of stochastic differential equations. *SIAM Journal on Numerical Analysis*, 48(3):922–952, 2010.

[72] Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. Latent odes for irregularly-sampled time series. *Neural Information Processing Systems*, 2019.

[73] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[74] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *arXiv preprint arXiv:1804.04272*, 2018.

[75] Thomas Ryder, Andrew Golightly, A Stephen McGough, and Dennis Prangle. Black-box variational inference for stochastic differential equations. *arXiv preprint arXiv:1802.03335*, 2018.

[76] John K Salmon, Mark A Moraes, Ron O Dror, and David E Shaw. Parallel random numbers: as easy as 1, 2, 3. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 16. ACM, 2011.

[77] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[78] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[79] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

[80] Evangelos Theodorou. Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations. *Entropy*, 17(5):3352–3375, 2015.

[81] Ryan Turner, Marc Deisenroth, and Carl Rasmussen. State-space inference and learning with gaussian processes. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 868–875, 2010.

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

[83] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. *Proceeings of the Conference on Learning Theory*, 2019.

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[85] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2007.

[86] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[87] Magnus Wiktorsson et al. Joint characteristic function and simultaneous simulation of iterated itô integrals for multiple independent brownian motions. *The Annals of Applied Probability*, 11(2): 470–487, 2001.

[88] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[89] Jichuan Yang and Harold J Kushner. A monte carlo method for sensitivity analysis and parametric optimization of nonlinear stochastic systems. *SIAM Journal on Control and Optimization*, 29 (5):1216–1249, 1991.

[90] Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. Ode2vae: Deep generative second order odes with bayesian neural networks. *arXiv preprint arXiv:1905.10994*, 2019.

9 Appendix

Notation. For a fixed terminal time $T > 0$, we denote by $\mathbb{T} = [0, T] \subseteq \mathbb{R}$ the time horizon. Let C^∞ be the class of infinitely differentiable functions from $\mathbb{R}^d$ to itself. Let $C^{p,q}$ be the class of functions from $\mathbb{R}^d \times \mathbb{T}$ to $\mathbb{R}^d$ that are p and q times continuously differentiable in the first and second component, respectively. Let $C_b^{p,q} \subseteq C^{p,q}$ be the subclass with bounded derivatives of all possible orders. For a positive integer m, we adopt the short hand $[m] = \{1, 2, \ldots, m\}$. We denote the Euclidean norm of a vector v by $|v|$. For $f \in C^{p,q}$, we denote its Jacobian with respect to the first component by ∇f.

9.1 Proof of Theorem 3.1

Proof of Theorem 3.1. We have $J_{s,t}(z) = \nabla \breve{\Psi}_{s,t}(z)$, where $\breve{\Psi}_{s,t}(z)$ is defined in (3). Now we take the gradient with respect to z on both sides. The solution is differentiable with respect to z and we may differentiate under the stochastic integral [41, Proposition 2.4.3]. Theorem 3.4.3 [41] is sufficient for the regularity conditions required. Since $K_{s,t}(z) = J_{s,t}(z)^{-1}$, applying the Stratonovich version of Itô's formula to (4), we have (5). □

9.2 Proof of Theorem 3.3

Proof of Theorem 3.3. By the triangle inequality,

$$\begin{aligned}&|\mathsf{F}(\mathsf{G}(z, W.), W.) - \mathsf{F}_h(\mathsf{G}_h(z, W.), W.)| \\ \leq &\underbrace{|\mathsf{F}(\mathsf{G}(z, W.), W.) - \mathsf{F}(\mathsf{G}_h(z, W.), W.)|}_{I_h^{(1)}} + \underbrace{|\mathsf{F}(\mathsf{G}_h(z, W.), W.) - \mathsf{F}_h(\mathsf{G}_h(z, W.), W.)|}_{I_h^{(2)}}.\end{aligned}$$

We show that both $I_h^{(1)}$ and $I_h^{(2)}$ converge to 0 in probability as $h \to 0$. For simplicity, we suppress z and $W.$.

Bounding $I_h^{(1)}$. Let $\epsilon > 0$ be given. Since $G_h \to G$ in probability, there exist $M_1 > 0$ and $h_0 > 0$ such that

$$\mathbb{P}(|G| > M_1) < \epsilon, \quad \mathbb{P}(|G_h| > 2M_1) < \epsilon, \quad \text{for all } h \leq h_0.$$

By Lemma 2.1 (iv) of Ocone and Pardoux [54], which can be easily adapted to our context, there exists a positive random variable C_1, finite almost surely, such that $\sup_{|z| \leq 2M_1} |\nabla_z \mathsf{F}| \leq C_1$, and there exists $M_2 > 0$ such that $\mathbb{P}(|C_1| > M_2) < \epsilon$. Given M_2, there exists $h_1 > 0$ such that

$$\mathbb{P}\left(|G - G_h| > \frac{\epsilon}{M_2}\right) < \epsilon, \quad \text{for all } h \leq h_1.$$

Now, suppose $h \leq \min\{h_0, h_1\}$. Then, by the union bound, with probability at least $1 - 4\epsilon$, we have

$$|G| \leq M_1, \quad |G_h| \leq 2M_1, \quad |C_1| \leq M_2, \quad |G - G_h| \leq \frac{\epsilon}{M_2}.$$

On this event, we have

$$I_h^{(1)} = |\mathsf{F}(\mathsf{G}) - \mathsf{F}(\mathsf{G}_h)| \leq C_1 |G - G_h| \leq M_2 \frac{\epsilon}{M_2} = \epsilon.$$

Thus, we have shown that $I_h^{(1)}$ converges to 0 in probability as $h \to 0$.

Bounding $I_h^{(2)}$. The idea is similar. By condition (ii), we have

$$\lim_{h \to 0} \sup_{|z_T| \leq M} |\mathsf{F}_h(z_T) - \mathsf{F}(z_T)| = 0$$

in probability. Using this and condition (i), for given $\epsilon > 0$, there exist $M > 0$ and $h_2 > 0$ such that for all $h \leq h_2$, we have

$$|G_h| \leq M \quad \text{and} \quad \sup_{|z_T| \leq M} |\mathsf{F}_h(z_T) - \mathsf{F}(z_T)| < \epsilon$$

with probability at least $1 - \epsilon$. On this event, we have

$$|\mathsf{F}(\mathsf{G}_h) - \mathsf{F}_h(\mathsf{G}_h)| \leq \sup_{|z_T| \leq M} |\mathsf{F}_h(z_T) - \mathsf{F}(z_T)| < \epsilon.$$

Thus $I_h^{(2)}$ also converges to 0 in probability as $h \to 0$. □

9.3 Euler-Maruyama Scheme Satisfies Local Uniform Convergence

Here we verify that the Euler-Maruyama scheme satisfies condition (ii) when $d = 1$. Our proof can be extended to the case where $d > 1$ assuming an L^p estimate of the error; see the discussion after the proof of Proposition 9.1.

Proposition 9.1. *Let $\mathsf{F}_h(z)$ be the Euler-Maruyama discretization of a 1-dimensional SDE with mesh size h of $\mathsf{F}(z)$. Then, for any compact $A \subset \mathbb{R}$, we have*

$$\operatorname{plim}_{h\to 0} \sup_{z\in A} |\mathsf{F}_h(z) - \mathsf{F}(z)| = 0.$$

Usual convergence results in stochastic numerics only control the error for a single fixed starting point. Here, we strengthen the result to local uniform convergence. Our main idea is to apply a Sobolev inequality argument [54, Part II]. To do so, we need some preliminary results about the Euler-Maruyama discretization of the original SDE and its derivative. We first recall a theorem characterizing the expected squared error for general schemes.

Theorem 9.2 (Mean-square order of convergence [51, Theorem 1.1])**.** *Let $\{Z_t^z\}_{t\geq 0}$ be the solution to an Itô SDE, and $\{\tilde{Z}_k^z\}_{k\in\mathbb{N}}$ be a numerical discretization with fixed step size h, both of which are started at $z \in \mathbb{R}^d$ and defined on the same probability space. Let the coefficients of the SDE be $C_b^{1,\infty}$. Furthermore, suppose that the numerical scheme has order of accuracy p_1 for the expectation of deviation and order of accuracy p_2 for the mean-square deviation. If $p_1 \geq p_2 + 1/2$ and $p_2 \geq 1/2$, then, for any $N \in \mathbb{N}$, $k \in [N]$, and $z \in \mathbb{R}^d$*

$$\mathbb{E}\left[|Z_{t_k}^z - \tilde{Z}_k^z|^2\right] \leq C\left(1 + |z|^2\right) h^{2p_2-1},$$

for a constant C that does not depend on h or z.

We refer the reader to [51] for the precise definitions of orders of accuracy and the proof. Given this theorem, we establish an estimate regarding errors of the discretization and its derivative with respect to the initial position.

Lemma 9.3. *We have*

$$\mathbb{E}\left[|\mathsf{F}(z) - \mathsf{F_h}(z)|^2 + |\nabla_z \mathsf{F}(z) - \nabla_z \mathsf{F_h}(z)|^2\right] \leq C_1(1 + |z|^2)h,$$

where C_1 is a constant independent of z and h.

Proof of Lemma 9.3. Since the coefficients of the SDE are of class $C_b^{\infty,1}$, we may differentiate the SDE in z to get the SDE for the derivative $\nabla_z Z_t^z$ [41]. Specifically, letting $Y_t^z = \nabla_z Z_t^z$, we have

$$Y_t^z = I_d + \int_0^t \nabla b(Z_s^z, s) Y_s^z \, \mathrm{d}s + \int_0^t \nabla \sigma(Z_s^z, s) Y_s^z \, \mathrm{d}W_s.$$

Note that the augmented process $(\mathsf{F}(z), \nabla_z \mathsf{F}(z))$ satisfies an SDE with $C_b^{\infty,1}$ coefficients. By the chain rule, one can easily show that the derivative of the Euler-Maruyama discretization $\mathsf{F}_h(z)$ is the discretization of the derivative process Y_t^z. Thus, $(\mathsf{F}_h(z), \nabla_z \mathsf{F}_h(z))$ is simply the discretization of $(\mathsf{F}(z), \nabla_z \mathsf{F}(z))$.

Since the Euler-Maruyama scheme has orders of accuracy $(p_1, p_2) = (1.5, 1.0)$ [51, Section 1.1.5], by Theorem 9.2, we have

$$\mathbb{E}\left[|\mathsf{F}(z) - \mathsf{F}_h(z)|^2 + |\nabla_z \mathsf{F}(z) - \nabla_z \mathsf{F}_h(z)|^2\right] \leq C_1(1 + |z|^2)h, \quad z \in \mathbb{R}^d$$

for some constant C_1 that does not depend on z or h. □

We also recall a variant of the Sobolev inequality which we will apply for $d = 1$.

Theorem 9.4 (Sobolev inequality [2, Theorem 5.4.1.c])**.** *For any $p > d$, there exists a universal constant c_p such that*

$$\sup_{x\in\mathbb{R}^d} |f(x)| \leq c_p \|f\|_{1,p},$$

where

$$\|f\|_{1,p}^p := \int_{\mathbb{R}^d} |f(x)|^p \, \mathrm{d}x + \int_{\mathbb{R}^d} |\nabla_x f(x)|^p \, \mathrm{d}x,$$

for all continuously differentiable $f : \mathbb{R}^d \to \mathbb{R}$.

Proof of Proposition 9.1. Define $\mathsf{H}_h^\alpha : \Omega \times \mathbb{R} \to \mathbb{R}$, regarded as a random function $\mathsf{H}_h^\alpha(\omega) : \mathbb{R} \to \mathbb{R}$, by

$$\mathsf{H}_h^\alpha(z) = \frac{\mathsf{F}(z) - \mathsf{F}_h(z)}{(1+|z|^2)^{1/2+\alpha}},$$

where $\alpha > 1/2$ is a fixed constant. Since H_h^α is continuously differentiable a.s., by Theorem 9.4,

$$|\mathsf{F}(z) - \mathsf{F}_h(z)| \le c_2(1+|z|^2)^{1/2+\alpha} \left\|\mathsf{H}_h^\alpha\right\|_{1,2}, \quad \text{for all } z \in \mathbb{R} \quad a.s.$$

Without loss of generality, we may let the compact set be $A = \{z : |z| \le M\}$ where $M > 0$. Then,

$$\sup_{|z|\le M} |\mathsf{F}(z) - \mathsf{F}_h(z)| \le c_2(1+M^2)^{1/2+\alpha} \left\|\mathsf{H}_h^\alpha\right\|_{1,2}, \quad a.s. \tag{11}$$

It remains to estimate $\left\|\mathsf{H}_h^\alpha\right\|_{1,2}$. Starting from the definition of $\|\cdot\|_{1,p}$, a standard estimation yields

$$\|\mathsf{H}_h^\alpha\|_{1,2}^2 \le C_2 \int_{\mathbb{R}} \frac{|\mathsf{F}(z) - \mathsf{F}_h(z)|^2 + |\nabla_z \mathsf{F}(z) - \nabla_z \mathsf{F}_h(z)|^2}{(1+|z|^2)^{1+2\alpha}} \, \mathrm{d}z,$$

where C_2 is a deterministic constant depending only on α (but not z and h).

Now we take expectation on both sides. By Lemma 9.3, we have

$$\begin{aligned}\mathbb{E}\left[\|\mathsf{H}_h^\alpha\|_{1,2}^2\right] \le & C_2 \int_{\mathbb{R}} \frac{\mathbb{E}[|\mathsf{F}(z) - \mathsf{F}_h(z)|^2 + |\nabla_z \mathsf{F}(z) - \nabla_z \mathsf{F}_h(z)|^2]}{(1+|z|^2)^{1+2\alpha}} \, \mathrm{d}z, \\ \le & C_1 C_2 h \int_{\mathbb{R}} \frac{1}{(1+|z|^2)^{2\alpha}} \, \mathrm{d}z,\end{aligned}$$

where the last integral is finite since $\alpha > 1/2$.

We have shown that $\mathbb{E}\left[\|\mathsf{H}_h^\alpha\|_{1,2}^2\right] = O(h)$. Thus $\left\|\mathsf{H}_h^\alpha\right\|_{1,2} \to 0$ in L^2, and hence also in probability, as $h \to 0$. From equation 11, we have that $\sup_{z\in A} |\mathsf{F}_h(z) - \mathsf{F}(z)|$ converges to 0 in probability as $h \to 0$. □

It is clear from the above proof that we may generalize to the case where $d > 1$ and other numerical schemes if we can bound the expected $W^{1,p}$-norm of $\mathsf{F}_h - \mathsf{F}$ in terms of z and h, for $p > d$, where $W^{1,p}$ here denotes the Sobolev space consisting of all real-valued functions on $\mathbb{R}^d$ whose weak derivatives are functions in L^p. For the Euler scheme and $d > 1$, we need only bound the L^p norm of the discretization error in term of z and h for general p. To achieve this, we would need to make explicit the dependence on z for existing estimates (see e.g. [39, Chapter 10]).

Generically extending the argument to other numerical schemes, however, is technically non-trivial. We plan to address this question in future research.

9.4 Stochastic Adjoint has Commutative Noise when Original SDE has Diagonal Noise

Recall the Stratonovich SDE (2) with drift and diffusion functions $b, \sigma_1, \ldots, \sigma_m \in \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ governed by a set of parameters $\theta \in \mathbb{R}^p$. Consider the augmented state composed of the original state and parameters $Y_t = (Z_t, \theta)$. The augmented state satisfies a Stratonovich SDE with the drift function $f(y,t) = (b(z,t), \mathbf{0}_p)$ and diffusion functions $g_i(y,t) = (\sigma_i(z,t), \mathbf{0}_p)$ for $i \in [m]$. By (5) and (6), the dynamics for the adjoint process of the augmented state is characterized by the backward SDE:

$$A_t^y = A_T^y + \int_t^T \nabla f(Y_s, s)^\top A_s^y \, \mathrm{d}t + \sum_{i=1}^m \int_t^T \nabla g_i(Y_s, s)^\top A_s^y \circ \mathrm{d}\widetilde{W}_s^{(i)}.$$

By definitions of f and g_i, the Jacobian matrices $\nabla f(x,s)$ and $\nabla g_i(x,s)$ can be written as:

$$\nabla f(y,s) = \begin{pmatrix} \frac{\partial b(z,s)}{\partial z} & \mathbf{0}_{d\times p} \\ \mathbf{0}_{p\times d} & \mathbf{0}_{p\times p} \end{pmatrix} \in \mathbb{R}^{(d+p)\times(d+p)}, \quad \nabla g_i(y,s) = \begin{pmatrix} \frac{\partial \sigma_i(z,s)}{\partial z} & \mathbf{0}_{d\times p} \\ \mathbf{0}_{p\times d} & \mathbf{0}_{p\times p} \end{pmatrix} \in \mathbb{R}^{(d+p)\times(d+p)}.$$

Thus, we can write out the backward SDEs for the adjoint processes of the state and parameters separately:

$$A_t^z = A_T^z + \int_t^T \frac{\partial b(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z \,\mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z \circ \mathrm{d}\widetilde{W}_s^{(i)}, \tag{12}$$
$$A_t^\theta = A_T^\theta + \int_t^T \frac{\partial b(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z \,\mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z \circ \mathrm{d}\widetilde{W}_s^{(i)}.$$

Now assume the original SDE has diagonal noise. Then, $m = d$ and Jacobian matrix $\nabla\sigma_i(z)$ can be written as:

$$\frac{\partial \sigma_i(z)}{\partial z} = \begin{pmatrix} 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & \frac{\partial \sigma_{i,i}(z)}{\partial z_i} & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{pmatrix}. \tag{13}$$

Consider the adjoint process for the augmented state along with the backward flow of the backward SDE (3). We write the overall state as $X_t = (Z_t, A_t^z, A_t^\theta)$, where we abuse notation slightly to let $\{Z_t\}_{t\in\mathbb{T}}$ denote the backward flow process. Then, by (12) and (13), $\{X_t\}_{t\in\mathbb{T}}$ satisfies a backward SDE with a diffusion function that can be written as:

$$G(x) = \begin{pmatrix} -\sigma_{1,1}(z_1) & 0 & \dots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & 0 & -\sigma_{d,d}(z_d) \\ \hline \frac{\partial \sigma_{1,1}(z_1)}{\partial z_1} a_1^z & 0 & \dots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & 0 & \frac{\partial \sigma_{d,d}(z_d)}{\partial z_d} a_d^z \\ \hline \frac{\partial \sigma_{1,1}(z_1)}{\partial \theta_1} a_1^z & \dots & \dots & \dots & \frac{\partial \sigma_{d,d}(z_d)}{\partial \theta_1} a_d^z \\ \dots & \dots & \dots & \dots & \dots \\ \frac{\partial \sigma_{1,1}(z_1)}{\partial \theta_p} a_1^z & \dots & \dots & \dots & \frac{\partial \sigma_{d,d}(z_d)}{\partial \theta_p} a_d^z \end{pmatrix} \in \mathbb{R}^{(2d+p)\times d}. \tag{14}$$

Recall, for an SDE with diffusion function $\Sigma(x) \in \mathbb{R}^{d\times m}$, it is said to satisfy the commutativity property [70] if

$$\sum_{i=1}^d \Sigma_{i,j_2}(x) \frac{\partial \Sigma_{k,j_1}(x)}{\partial x_i} = \sum_{i=1}^d \Sigma_{i,j_1}(x) \frac{\partial \Sigma_{k,j_2}(x)}{\partial x_i}, \tag{15}$$

for all $j_1, j_2 \in [m]$ and $k \in [d]$. When an SDE has commutative noise, the computationally intensive double Itô integrals (and the Lévy areas) need not be simulated by having the numerical scheme take advantage of the following property of iterated integrals [30]:

$$\int_s^t \int_s^u \mathrm{d}W_r^{(i)} \,\mathrm{d}W_u^{(j)} + \int_s^t \int_s^u \mathrm{d}W_r^{(j)} \,\mathrm{d}W_u^{(i)} = \Delta W^{(i)} \Delta W^{(j)},$$

where the Brownian motion increment $\Delta W^{(i)} = W_t^{(i)} - W_s^{(i)}$ for $i \in [m]$ can be easily sampled.

To see that the diffusion function (14) indeed satisfies the commutativity condition (15), we consider several cases:

- $k = 1, \dots, d$: Both LHS and RHS are zero unless $j_1 = j_2 = k$, since for $\Sigma_{i,j_2}(x)\frac{\partial \Sigma_{k,j_1}(x)}{\partial x_i}$ to be non-zero, $i = j_1 = j_2 = k$.
- $k = d+1 \dots, 2d$: Similar to the case above.
- $k = 2d+1 \dots, 2d+p$: Write $k = 2d + l$, where $l \in [p]$. Both LHS and RHS are zero unless $j_1 = j_2 = l$, since for $\Sigma_{i,j_2}(x)\frac{\partial \Sigma_{k,j_1}(x)}{\partial x_i}$ to be non-zero $i = l$ or $i = d + l$ and $j_1 = j_2 = l$.

Since in all scenarios, LHS = RHS, we conclude that the commutativity condition holds.

Finally, we comment that the Milstein scheme for the stochastic adjoint of diagonal noise SDEs can be implemented such that during each iteration of the backward solve, `vjp` is only called a number of times independent respect to the dimensionality of the original SDE.

9.5 Background on Latent SDE

Consider a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{0\le t\le T}, P)$, where $[0,T]$ is a finite time horizon.

Recall the approximate posterior process that we intend to learn is governed by the SDE:

$$\mathrm{d}Z_t = h_\phi(Z_t,t)\ \mathrm{d}t + \sigma(Z_t,t)\ \mathrm{d}W_t, \quad Z_0 = z_0 \in \mathbb{R}^d, \tag{16}$$

Suppose there exists a measurable function $u(z,t)$ such that

- $\sigma(z,t)u(z,t) = h_\phi(z,t) - h_\theta(z,t)$, and
- $u(Z_t,t)$ satisfies Novikov's condition, i.e. $\mathbb{E}\left[\exp\left(\int_0^T \frac{1}{2}|u(Z_t,t)|^2\ \mathrm{d}t\right)\right] < \infty$.

Novikov's condition ensures that the process

$$M_t = \exp\left(-\int_0^t \frac{1}{2}|u(Z_s,s)|^2\ \mathrm{d}s - \int_0^t u(Z_s,s)^\top\ \mathrm{d}W_s\right), \quad 0\le t\le T,$$

is a P-martingale. By Girsanov Theorem II [56, Theorem 8.6.4], the process $\widehat{W}_t = \int_0^t u(Z_s,s)\ \mathrm{d}s + W_t$, $0\le t\le T$ is a Wiener process under the probability measure Q defined by

$$\mathrm{d}Q = M_T\,\mathrm{d}P,$$

Moreover, since a simple rewrite shows that

$$\mathrm{d}Z_t = h_\theta(Z_t,t)\ \mathrm{d}t + \sigma(Z_t,t)\,\mathrm{d}\widehat{W}_t, \quad Z_0 = z_0, \tag{17}$$

we conclude that the Q-law of (17) (or equivalently (16)) is the same as the P-law of the prior process.

9.5.1 Deriving the Variational Bound

Let $x_{t_1},\dots,x_{t_N}$ be observation data at times $t_1,\dots,t_N$, whose conditionals only depend on the respective latent states $z_{t_1},\dots,z_{t_N}$. Since the Q-law of the approximate posterior is the same as the P-law of the prior,

$$\begin{aligned}
\log p(x_{t_1},\dots,x_{t_N}) = \log \mathbb{E}_P\left[\prod_{i=1}^N p(x_{t_i}|\tilde{z}_{t_i})\right] &= \log \mathbb{E}_Q\left[\prod_{i=1}^N p(x_{t_i}|z_{t_i})\right] \\
&= \log \mathbb{E}_P\left[\prod_{i=1}^N p(x_{t_i}|z_{t_i}) M_T\right] \\
&\ge \mathbb{E}_P\left[\sum_{i=1}^N \log p(x_{t_i}|z_{t_i}) + \log M_T\right] \\
&= \mathbb{E}_P\left[\sum_{i=1}^N \log p(x_{t_i}|z_{t_i}) - \int_0^T \frac{1}{2}|u(Z_t,t)|^2\ \mathrm{d}t - \int_0^T u(Z_t)^\top\ \mathrm{d}W_t\right] \\
&= \mathbb{E}_P\left[\sum_{i=1}^N \log p(x_{t_i}|z_{t_i}) - \int_0^T \frac{1}{2}|u(Z_t,t)|^2\ \mathrm{d}t\right],
\end{aligned}$$

where the second line follows from the definition of Q and third line follows from Jensen's inequality. In the last equality we used the fact that the Itô integral $\int_0^\cdot u(Z_t)^\top dW_t$ is a martingale.

9.6 Stochastic Adjoint for Latent SDE

Note that the variational free energy (10) can be derived from Girsanov's change of measure theorem [57]. To efficiently Monte Carlo estimate this quantity and its gradient, we simplify the equation by noting that for a one-dimensional process $\{V_t\}_{t\in\mathbb{T}}$ adapted to the filtration generated by a one-dimensional Wiener process $\{W_t\}_{t\in\mathbb{T}}$,

if Novikov's condition [55] is satisfied, then the process defined by the Itô integral $\int_0^t V_s \, \mathrm{d}W_s$ is a Martingale [55]. Hence, $\mathbb{E}\big[\int_0^T u(Z_t,t)^\top \, \mathrm{d}W_t\big] = 0$, and

$$\mathcal{L}_{\mathrm{VI}} = \mathbb{E}\Big[\tfrac{1}{2}\int_0^T |u(Z_t,t)|^2 \, \mathrm{d}t - \textstyle\sum_{i=1}^N \log p(y_{t_i}|z_{t_i})\Big].$$

To Monte Carlo simulate the quantity in the forward pass along with the original dynamics, we need only extend the original augmented state with an extra variable L_t such that the new drift and diffusion functions for the new augmented state $Y_t = (Z_t^\top, \theta^\top, L_t)^\top$ are

$$f(x,t) = \begin{pmatrix} b(z,t) \\ \mathbf{0}_p \\ \frac{1}{2}|u(z,t)|_2^2 \end{pmatrix} \in \mathbb{R}^{d+p+1}, \quad g_i(x,t) = \begin{pmatrix} \sigma_i(z,t) \\ \mathbf{0}_p \\ 0 \end{pmatrix} \in \mathbb{R}^{d+p+1}, \quad i \in [m].$$

By (7), the backward SDEs of the adjoint processes become

$$\begin{aligned} A_t^z &= A_T^z + \int_t^T \left(\frac{\partial b(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z + \frac{1}{2}\frac{\partial \|u(z,s)\|_2^2}{\partial z}\bigg|_{z=Z_s} A_s^l \right) \mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z \circ \check{\mathrm{d}}W_s^{(i)}, \\ A_t^\theta &= A_T^\theta + \int_t^T \left(\frac{\partial b(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z + \frac{1}{2}\frac{\partial \|u(z,s)\|_2^2}{\partial \theta}\bigg|_{z=Z_s} A_s^l \right) \mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z \circ \check{\mathrm{d}}W_s^{(i)}, \\ A_t^l &= A_T^l. \end{aligned} \tag{18}$$

In this case, neither do we need to actually simulate the backward SDE of the extra variable nor do we need to simulate its adjoint. Moreover, when considered as a single system for the augmented adjoint state, the diffusion function of the backward SDE (18) satisfies the commutativity property (15).

9.7 Test Problems

In the following, α, β, and p are parameters of SDEs, and x_0 is a fixed initial value.

Example 1.

$$\mathrm{d}X_t = \alpha X_t \, \mathrm{d}t + \beta X_t \, \mathrm{d}W_t, \quad X_0 = x_0.$$

Analytical solution:

$$X_t = X_0 e^{\left(\beta - \frac{\alpha^2}{2}\right)t + \alpha W_t}.$$

Example 2.

$$\mathrm{d}X_t = -\left(p^2\right)^2 \sin\left(X_t\right)\cos^3\left(X_t\right) \, \mathrm{d}t + p\cos^2\left(X_t\right) \, \mathrm{d}W_t, \quad X_0 = x_0.$$

Analytical solution:

$$X_t = \arctan\left(pW_t + \tan\left(X_0\right)\right).$$

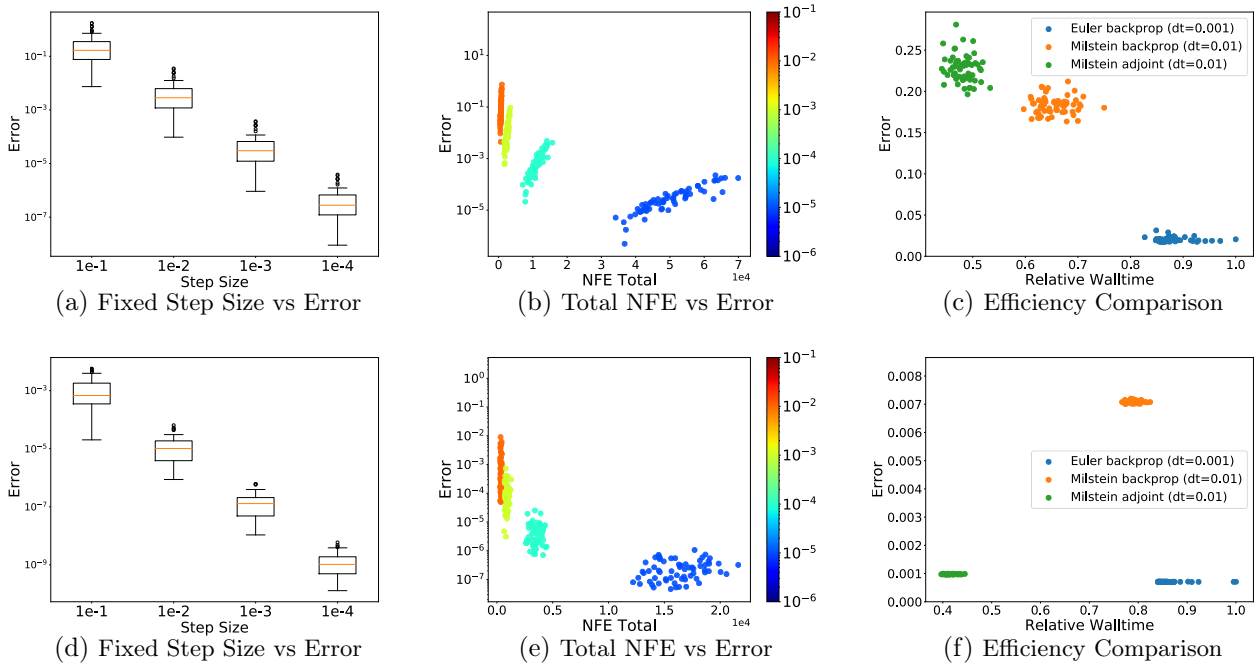
Example 3.

$$\mathrm{d}X_t = \left(\frac{\beta}{\sqrt{1+t}} - \frac{1}{2(1+t)}X_t\right) \mathrm{d}t + \frac{\alpha\beta}{\sqrt{1+t}} \, \mathrm{d}W_t, \quad X_0 = x_0.$$

Analytical solution:

$$X_t = \frac{1}{\sqrt{1+t}}X_0 + \frac{\beta}{\sqrt{1+t}}\left(t + \alpha W_t\right).$$

In each numerical experiment, we duplicate the equation 10 times to obtain a system of SDEs where each dimension had their own parameter values sampled from the standard Gaussian distribution and then passed through a sigmoid to ensure positivity. Moreover, we also sample the initial value for each dimension from a Gaussian distribution.



(a) Fixed Step Size vs Error (b) Total NFE vs Error (c) Efficiency Comparison

(d) Fixed Step Size vs Error (e) Total NFE vs Error (f) Efficiency Comparison

Figure 7: (a-c) Example 1. (d-f) Example 3.

9.8 Results for Example 1 and 3

9.9 Toy Datasets Configuration

9.9.1 Geometric Brownian Motion

Consider a geometric Brownian motion SDE:

$$\mathrm{d}X_t = \mu X_t \, \mathrm{d}t + \sigma X_t \, \mathrm{d}W_t, \quad X_0 = x_0.$$

We use $\mu = 1$, $\sigma = 0.5$, and $x_0 = 0.1 + \epsilon$ as the ground-truth model, where $\epsilon \sim \mathcal{N}(0, 0.03^2)$. We sample 1024 time series, each of which is observed at intervals of 0.02 from time 0 to time 1. We corrupt this data using Gaussian noise with mean zero and standard deviation 0.01.

To recover the dynamics, we use a GRU-based [13] latent SDE model where the GRU has 1 layer and 100 hidden units, the prior and posterior drift functions are MLPs with 1 hidden layer of 100 units, and the diffusion function is an MLP with 1 hidden layer of 100 hidden units and the sigmoid activation applied at the end. The drift function in the posterior is time-inhomogenous in the sense that it takes in a context vector of size 1 at each observation that is output by the GRU from running backwards after processing all future observations. The decoder is a linear mapping from a 4 dimensional latent space to observation space. For all nonlinearities, we use the softplus function. We fix the observation model to be Gaussian with noise standard deviation 0.01.

We optimize the model jointly with respect to the parameters of a Gaussian distribution for initial latent state distribution, the prior and posterior drift functions, the diffusion function, the GRU encoder, and the decoder. We use a fixed discretization with step size of 0.01 in both the forward and backward pass. We use the Adam optimizer [34] with an initial learning rate of 0.01 that is decay by a factor of 0.999 after each iteration. We use a linear KL annealing schedule over the first 50 iterations.

9.9.2 Stochastic Lorenz Attractor

Consider a stochastic Lorenz attractor SDE with diagonal noise:

$$\begin{aligned}
\mathrm{d}X_t &= \sigma\,(Y_t - X_t)\,\mathrm{d}t + \alpha_x\,\mathrm{d}W_t, \quad X_0 = x_0,\\
\mathrm{d}Y_t &= (X_t\,(\rho - Z_t) - Y_t)\,\mathrm{d}t + \alpha_y\,\mathrm{d}W_t, \quad Y_0 = y_0,\\
\mathrm{d}Z_t &= (X_t Y_t - \beta Z_t)\,\mathrm{d}t + \alpha_z\,\mathrm{d}W_t, \quad Z_0 = z_0.
\end{aligned}$$

We use $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, $(\alpha_x, \alpha_y, \alpha_z) = (.15, .15., .15)$, and (x_0, y_0, z_0) sampled from the standard Gaussian distribution as the ground-truth model. We sample 1024 time series, each of which is observed at intervals of 0.025 from time 0 to time 1. We normalize these samples by their mean and standard deviation across each dimension and corrupt this data by Gaussian noise with mean zero and standard deviation 0.01.

We use the same architecture and training procedure for the latent SDE model as in the geometric Brownian motion section, except that the diffusion function consists of four small neural networks, each for a single dimension of the latent SDE.

9.10 Additional Visualization

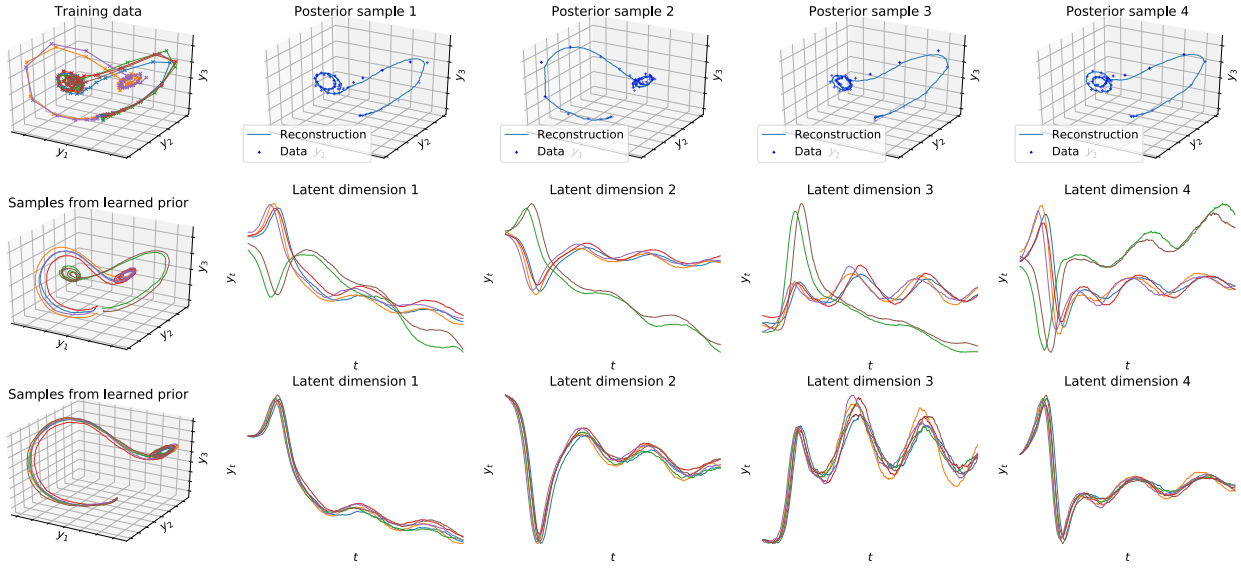


Figure 8: Additional visualizations of learned posterior and prior dynamics on the synthetic stochastic Lorenz attractor dataset. First row displays the true data and posterior reconstructions. Second row displays samples with initial latent state for each trajectory is sampled independently. Third row displays samples with initial latent state sampled and fixed to be the same for different trajectories.

See Figure 8 for additional visualization on the synthetic Lorenz attractor dataset. See Figure 9 for visualization on the synthetic geometric Brownian motion dataset. We comment that for the second example, the posterior reconstructs the data well, and the prior process exhibit behavior of the data. However, from the third row, we can observe that the prior process is learned such that most of the uncertainty is account for in the initial latent state. We leave the investigation of more interpretable prior process for future work.

9.11 Model Architecture for Learning from Motion Capture Dataset

We use a latent SDE model with an MLP encoder which takes in the first three frames and outputs the mean and log-variance of the variational distribution of the initial latent state and a context vector. The decoder has a similar architecture as that for the ODE^{2}VAE model [90] and projects the 6-dimensional latent state into the 50-dimensional observation space. The posterior drift function takes in a 3-dimensional context vector output by the encoder and the current state and time, whereas the prior drift only takes in the current state and time. The diffusion function is composed of multiple small neural nets, each producing a scalar for the corresponding

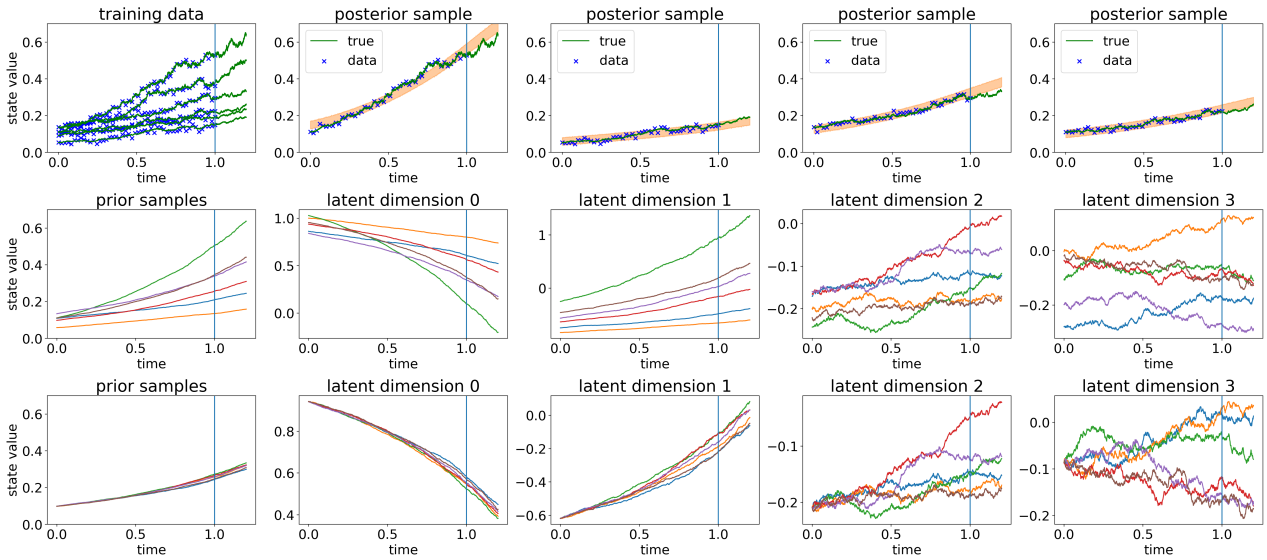


Figure 9: Visualizations of learned posterior and prior dynamics on the synthetic geometric Brownian motion dataset. First row displays the true data and posterior reconstructions. Orange contour covers 95% of 512 samples. Second row displays samples with initial latent state for each trajectory is sampled independently. Third row displays samples with initial latent state sampled and fixed to be the same for different trajectories.

dimension such that the posterior SDE has diagonal noise. We use the same observation likelihood as that of the ODE^{2}VAE model [90]. We comment that the overall parameter count of our model (11605) is smaller than that of ODE^{2}VAE for the same task (12157).

The latent ODE baseline was implemented with a similar architecture, except is does not have the diffusion and prior drift components, and its vector field defining the ODE does not take in a context vector. Therefore, the model has slightly fewer parameters (10573) than the latent SDE model. See Figure 10 for overall details of the architecture.

The main hyperparameter we tuned was the coefficient for reweighting the KL. For both the latent ODE and SDE, we considered training the model with a reweighting coefficient in $\{1, 0.1, 0.01, 0.001\}$, either with or without a linear KL annealing schedule that increased from 0 to the prescribed value over the first 200 iterations of training.

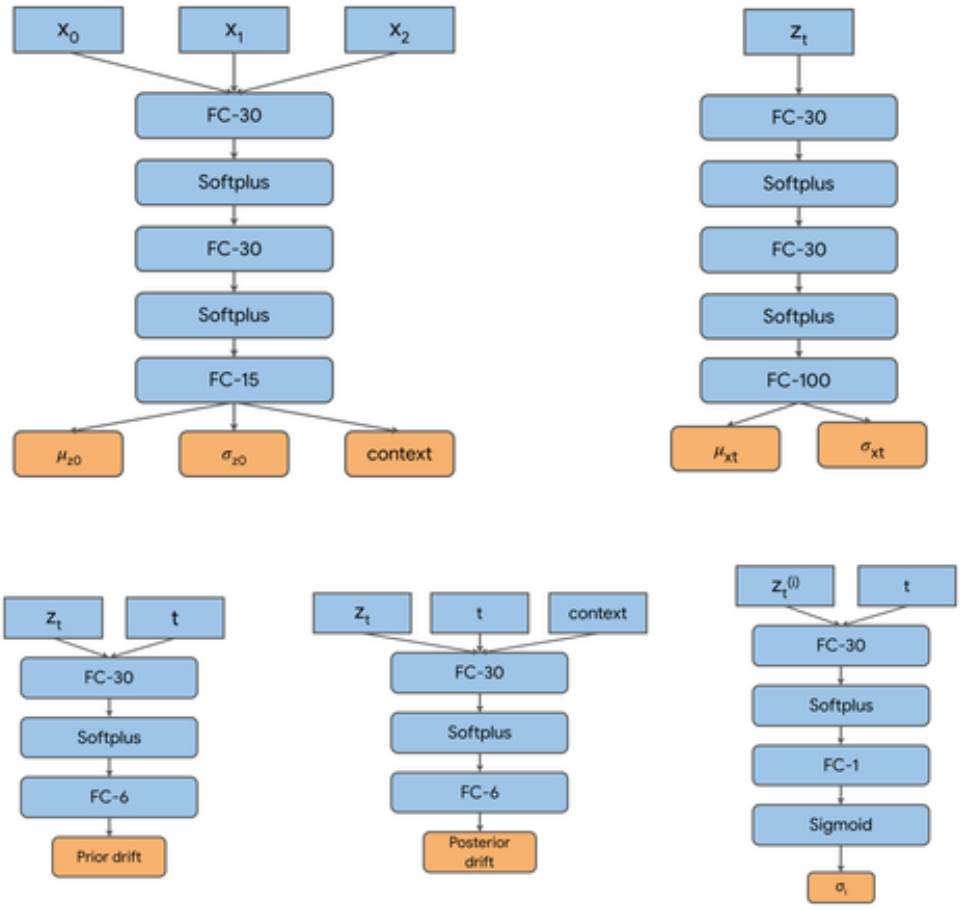


Figure 10: Architecture specifics for the latent SDE model used to train on the mocap dataset. First row from left to right are the encoder and decoder. Second row from left to right are the prior drift, posterior drift, and diffusion functions.

9.12 Stochastic Adjoint Implementation

We include the core implementation of the stochastic adjoint, assuming access to a callable Brownian motion `bm`, an Euler-Maruyama integrator `ito_int_diag` for diagonal noise SDEs, and several helper functions whose purposes can be inferred from their names.

```
class _SdeintAdjointMethod(torch.autograd.Function):

  @staticmethod
  def forward(ctx, *args):
    (y0, f, g, ts, flat_params_f, flat_params_g, dt, bm) = (
      args[:-8], args[-7], args[-6], args[-5], args[-4], args[-3], args[-2], args[-1])
    ctx.f, ctx.g, ctx.dt, ctx.bm = f, g, dt, bm

    def g_prod(t, y, noise):
      g_eval = g(t=t, y=y)
      g_prod_eval = tuple(
        g_eval_i * noise_i for g_eval_i, noise_i in _zip(g_eval, noise))
      return g_prod_eval

    with torch.no_grad():
      ans = ito_int_diag(f, g_prod, y0, ts, dt, bm)
    ctx.save_for_backward(ts, flat_params_f, flat_params_g, *ans)
    return ans

  @staticmethod
  def backward(ctx, *grad_outputs):
    ts, flat_params_f, flat_params_g, *ans = ctx.saved_tensors
    f, g, dt, bm = ctx.f, ctx.g, ctx.dt, ctx.bm
    f_params, g_params = tuple(f.parameters()), tuple(g.parameters())
    n_tensors = len(ans)

    def aug_f(t, y_aug):
      y, adj_y = y_aug[:n_tensors], y_aug[n_tensors:2 * n_tensors]

      with torch.enable_grad():
        y = tuple(y_.detach().requires_grad_(True) for y_ in y)
        adj_y = tuple(adj_y_.detach() for adj_y_ in adj_y)

        g_eval = g(t=-t, y=y)
        gdg = torch.autograd.grad(
          outputs=g_eval, inputs=y,
          grad_outputs=g_eval,
          create_graph=True)
        f_eval = f(t=-t, y=y)
        f_eval = _sequence_subtract(gdg, f_eval)  # -f + gdg.

        vjp_y_and_params = torch.autograd.grad(
          outputs=f_eval, inputs=y + f_params + g_params,
          grad_outputs=tuple(-adj_y_ for adj_y_ in adj_y),
          retain_graph=True, allow_unused=True)
        vjp_y = vjp_y_and_params[:n_tensors]
        vjp_f = vjp_y_and_params[-len(f_params + g_params):-len(g_params)]
        vjp_g = vjp_y_and_params[-len(g_params):]

        vjp_y = tuple(torch.zeros_like(y_)
          if vjp_y_ is None  else vjp_y_ for vjp_y_, y_ in zip(vjp_y, y))

        adj_times_dgdx = torch.autograd.grad(
          outputs=g_eval, inputs=y,
          grad_outputs=adj_y,
          create_graph=True)
        extra_vjp_y_and_params = torch.autograd.grad(
          outputs=g_eval, inputs=y + f_params + g_params,
          grad_outputs=adj_times_dgdx,
          allow_unused=True)
```

```
      extra_vjp_y = extra_vjp_y_and_params[:n_tensors]
      extra_vjp_f = extra_vjp_y_and_params[-len(f_params + g_params):-len(g_params)]
      extra_vjp_g = extra_vjp_y_and_params[-len(g_params):]

      extra_vjp_y = tuple(
        torch.zeros_like(y_) if extra_vjp_y_ is None
        else extra_vjp_y_ for extra_vjp_y_, y_ in zip(extra_vjp_y, y))

      vjp_y = _sequence_add(vjp_y, extra_vjp_y)
      vjp_f = vjp_f + extra_vjp_f
      vjp_g = vjp_g + extra_vjp_g

    return (*f_eval, *vjp_y, vjp_f, vjp_g)

  def aug_g_prod(t, y_aug, noise):
    y, adj_y = y_aug[:n_tensors], y_aug[n_tensors:2 * n_tensors]

    with torch.enable_grad():
      y = tuple(y_.detach().requires_grad_(True) for y_ in y)
      adj_y = tuple(adj_y_.detach() for adj_y_ in adj_y)

      g_eval = tuple(-g_ for g_ in g(t=-t, y=y))
      vjp_y_and_params = torch.autograd.grad(
        outputs=g_eval, inputs=y + f_params + g_params,
        grad_outputs=tuple(-noise_ * adj_y_ for noise_, adj_y_ in zip(noise, adj_y)),
        allow_unused=True)
      vjp_y = vjp_y_and_params[:n_tensors]
      vjp_f = vjp_y_and_params[-len(f_params + g_params):-len(g_params)]
      vjp_g = vjp_y_and_params[-len(g_params):]

      vjp_y = tuple(
        torch.zeros_like(y_) if vjp_y_ is None
        else vjp_y_ for vjp_y_, y_ in zip(vjp_y, y)
      )
      g_prod_eval = _sequence_multiply(g_eval, noise)

    return (*g_prod_eval, *vjp_y, vjp_f, vjp_g)

  def aug_bm(t):
    return tuple(-bmi for bmi in bm(-t))

  T = ans[0].size(0)
  with torch.no_grad():
    adj_y = tuple(grad_outputs_[-1] for grad_outputs_ in grad_outputs)
    adj_params_f = torch.zeros_like(flat_params_f)
    adj_params_g = torch.zeros_like(flat_params_g)

    for i in range(T - 1, 0, -1):
      ans_i = tuple(ans_[i] for ans_ in ans)
      aug_y0 = (*ans_i, *adj_y, adj_params_f, adj_params_g)
      aug_ans = ito_int_diag(
        f=aug_f, g_prod=aug_g_prod, y0=aug_y0,
        ts=torch.tensor([-ts[i], -ts[i - 1]]).to(ts),
        dt=dt, bm=aug_bm)
      adj_y = aug_ans[n_tensors:2 * n_tensors]
      adj_params_f, adj_params_g = aug_ans[-2], aug_ans[-1]

      # Take the result at the end time.
      adj_y = tuple(adj_y_[1] for adj_y_ in adj_y)
      adj_params_f, adj_params_g = adj_params_f[1], adj_params_g[1]

      # Accumulate gradients at intermediate points.
      adj_y = _sequence_add(
        adj_y, tuple(grad_outputs_[i - 1] for grad_outputs_ in grad_outputs)
      )
  return (*adj_y, None, None, None, adj_params_f, adj_params_g, None, None)
```