8 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 Additional Background

9.1 Adjoint Sensitivity Method

The adjoint sensitivity method is an efficient approach to solve control problems relying on the adjoint (co-state) system [Pontryagin, 2018]. Chen et al. [2018] 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 [Chang et al., 2017, 2018, Haber and Ruthotto, 2017, Lu et al., 2017, Ruthotto and Haber, 2018]. The method, shown in Algorithm 2, 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 [Rumelhart et al., 1988].

Algorithm 2 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 3 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 6: 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.

9.2 Stochastic Differential Equations

Here we briefly define stochastic differential equations: 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 $\mathbb{T} = [0, T] \subseteq \mathbb{R}$ 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

$$Z_T = z_0 + \int_0^T b(Z_t, t)\ \mathrm{d}t + \sum_{i=1}^m \int_0^T \sigma_i(Z_t, t)\ \mathrm{d}W_t^{(i)}, \tag{10}$$

where $z_0 \in \mathbb{R}^d$ is the starting value, 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, and note

that our contributions can be easily generalized to cases where $m > 1$. Here, the second integral on the right hand side of (10) is the Itô stochastic integral [Øksendal, 2003]. When the coefficients are globally Lipschitz in both the state and time, there exists a unique strong solution to the SDE [Øksendal, 2003].

9.3 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* [Jia and Benson, 2019, Liu et al., 2019, Tzen and Raginsky, 2019a,b].

Amongst work on neural SDEs, none has enabled an efficient training framework. In particular, Tzen and Raginsky [2019a] and Liu et al. [2019] 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 [Gobet and Munos, 2005, Yang and Kushner, 1991]. 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 automatic differentiation [Abadi et al., 2016, Frostig et al., 2018, Maclaurin et al., 2015, Paszke et al., 2017].

9.4 Backward Stratonovich Integral

Our stochastic adjoint sensitivity method involves stochastic processes running 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. [Platen, 1999, Sec. 2]).

Following the treatment of Kunita [Kunita, 2019], we introduce the forward and backward Stratonovich integrals. Let $\{\mathcal{F}_{s,t}\}_{s\leq t; s,t\in\mathbb{T}}$ be a *two-sided filtration*, where $\mathcal{F}_{s,t}$ is the σ-algebra generated by $\{W_v - W_u : s \leq u \leq v \leq t\}$ for $s, t \in \mathbb{T}$ such that $s \leq t$. 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

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

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* $\{\overleftarrow{W}_t\}_{t\in\mathbb{T}}$ defined as $\overleftarrow{W}_t = W_t - W_T$ 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 $\overleftarrow{Y}_t$ adapted to the backward filtration, the *backward Stratonovich integral* is

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

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

9.5 Proof of Theorem 3.1

Proof of Theorem 3.1. We have $J_{s,t}(z) = \nabla \overleftarrow{\Psi}_{s,t}(z)$, where $\overleftarrow{\Psi}_{s,t}(z)$ is defined in (2). 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 [Kunita, 2019, Proposition 2.4.3]. Theorem 3.4.3 Kunita [2019] 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 (3), we have (4). □

9.6 Proof of Theorem 3.3

Proof of Theorem 3.3. By the triangle inequality,

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

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 $W_\cdot$.

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

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

By Lemma 2.1(iv) of Ocone and Pardoux [1989] which can be easily adapted to our context, there exists a positive random variable C_1, finite almost surely, such that $\sup_{|z|\le 2M_1}|\nabla_z\mathsf{F}|\le C_1$, 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

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

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

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

On this event, we have

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

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

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

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

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

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

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

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

9.7 Euler-Maruyama Scheme Satisfies Condition (ii)

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 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 [Ocone and Pardoux, 1989, 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 [Milstein and Tretyakov, 2013, 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 [Milstein and Tretyakov, 2013] 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*

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

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 $\nabla_z Z_t^z$ [Kunita, 2019]. 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 $(p_1, p_2) = (1.5, 1.0)$ [Milstein and Tretyakov, 2013, 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 [Adams, 1975, 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,

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

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

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

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

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

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

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

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

We have shown that $\mathbb{E}\left[\left\|\mathsf{H}_h^\alpha\right\|_{1,2}^2\right] = O(h)$. 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. [Kloeden and Platen, 2013, Chapter 10]).

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

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

Recall the Stratonovich SDE (1) 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 (4) and (5), 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:

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

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 (2). 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:

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

Recall, for an SDE with diffusion function $\Sigma(x) \in \mathbb{R}^{d\times m}$, it is said to satisfy the commutativity property [Rößler, 2004] 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 [Ilie et al., 2015]:

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

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.

9.9 Background on Latent SDE

9.9.1 Setup

We consider a simple scenario where the posterior process that we intend to learn is governed by the SDE:

$$\mathrm{d}X_t = b(X_t)\, \mathrm{d}t + \mathrm{d}W_t, \quad X_0 = x_0 \in \mathbb{R}^d,$$

where $\{W_t\}_{t\geq 0}$ is defined on the probability space $(\Omega, \mathcal{F}, P)$ and adapted to the filtration $\{\mathcal{F}_t\}_{t\geq 0}$. Suppose we use a Wiener process as the prior process that is started at x_0, i.e. defined by the SDE

$$\mathrm{d}\tilde{X}_t = \mathrm{d}W_t, \quad \tilde{X}_0 = x_0 \in \mathbb{R}^d.$$

By Girsanov Theorem I [Oksendal, 2013, Theorem 8.6.3], the posterior process is a Wiener process under a different probability measure Q on the measurable space $(\Omega, \mathcal{F})$ defined by

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

where Z_T is the Radon-Nikodym derivative for a finite terminal time T. More generally, by the theorem, $\{Z_t\}_{t\geq 0}$ is the Radon-Nikodym derivative process given by

$$Z_t = \exp\left(-\int_0^t \frac{1}{2}|b(X_s)|^2 \, \mathrm{d}s - \int_0^t b(X_s)^\top \, \mathrm{d}W_s\right), \quad 0 \leq t \leq T.$$

9.9.2 Derivation of the Variational Bound

Let $y_{t_1}, \ldots, y_{t_N}$ be observation data at times $t_1, \ldots, t_N$, whose conditional distributions only depend on the respective latent states $X_{t_1}, \ldots, X_{t_N}$. Since $\{X_t\}_{t\geq 0}$ is a Wiener process under Q and $\{\tilde{X}_t\}_{t\geq 0}$ is a Wiener process under P,

$$\begin{aligned}
&\log \mathbb{E}_P\left[\prod_{i=1}^N p(y_{t_i}|\tilde{X}_{t_i})\right] \\
=&\log \mathbb{E}_Q\left[\prod_{i=1}^N p(y_{t_i}|X_{t_i})\right] \\
=&\log \mathbb{E}_P\left[\prod_{i=1}^N p(y_{t_i}|X_{t_i}) Z_T\right] \\
\geq&\mathbb{E}_P\left[\sum_{i=1}^N \log p(y_{t_i}|X_{t_i}) + \log Z_T\right] \\
=&\mathbb{E}_P\left[\sum_{i=1}^N \log p(y_{t_i}|X_{t_i}) - \int_0^T \frac{1}{2}|b(X_t)|^2 \, \mathrm{d}t - \int_0^T b(X_t)^\top \, \mathrm{d}W_t\right] \\
=&\mathbb{E}_P\left[\sum_{i=1}^N \log p(y_{t_i}|X_{t_i}) - \int_0^T \frac{1}{2}|b(X_t)|^2 \, \mathrm{d}t\right],
\end{aligned}$$

where the second equality follows from the definition of Q and the inequality follows from Jensen's. The derived result is the evidence lower bound (9) for simple case of the constant one function being the diffusion coefficient.

The derivation can be extended to processes with non-constant diffusion coefficients and prior processes with non-zero drift functions (see e.g. [Oksendal, 2013, Theorems 8.6.4, 8.6.5]), so long as certain regularity conditions of the coefficients hold and that the prior and posterior processes have the *same* diffusion coefficient.

9.10 Stochastic Adjoint for Latent SDE

Note that the variational free energy (9) can be derived from Girsanov's change of measure theorem citepopper2019variational. 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 citepoksendal2003stochastic is satisfied, then the process defined by the Itô integral $\int_0^t V_s \, \mathrm{d}W_s$ is a Martingale citepoksendal2003stochastic. Hence, $\mathbb{E}\big[\int_0^T u(Z_t, t)^\top \, \mathrm{d}W_t\big] = 0$, and

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

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 (6), the backward SDEs of the adjoint processes become

$$\begin{aligned} A_t^z =& A_T^z + \int_t^T \left(\frac{\partial b(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z + \frac{1}{2}\frac{\partial \|u(z,s)\|_2^2}{\partial z}\bigg|_{z=Z_s} A_s^l \right) \mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial z}\bigg|_{z=Z_s}^\top A_s^z \circ \check{\mathrm{d}}W_s^{(i)}, \\ A_t^\theta =& A_T^\theta + \int_t^T \left(\frac{\partial b(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z + \frac{1}{2}\frac{\partial \|u(z,s)\|_2^2}{\partial \theta}\bigg|_{z=Z_s} A_s^l \right) \mathrm{d}t + \sum_{i=1}^m \int_t^T \frac{\partial \sigma_i(z,s)}{\partial \theta}\bigg|_{z=Z_s}^\top A_s^z \circ \check{\mathrm{d}}W_s^{(i)}, \\ A_t^l =& A_T^l. \end{aligned} \tag{16}$$

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 (16) satisfies the commutativity property (15).

9.11 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:

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

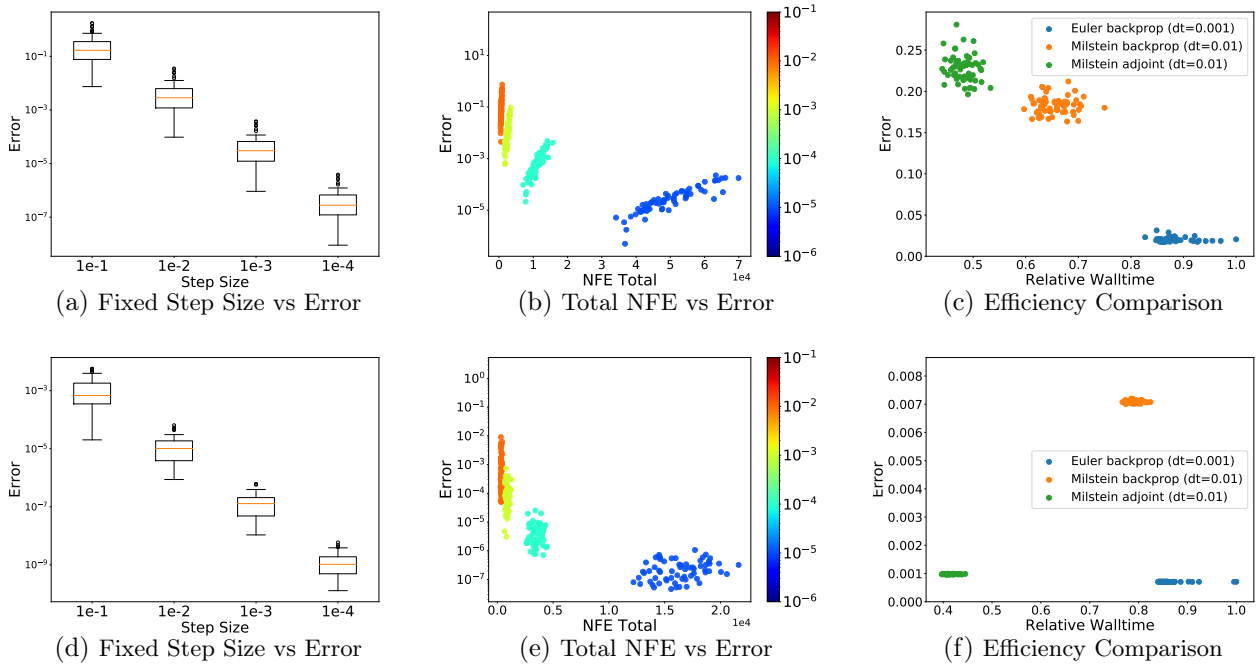
Example 3.

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

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.

9.12 Results for Example 1 and 3

9.13 Toy Datasets Configuration

9.13.1 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 [Cho et al., 2014] 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 [Kingma and Ba, 2014] 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.

9.13.2 Stochastic Lorenz Attractor

Consider a stochastic Lorenz attractor SDE with diagonal noise:

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

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.

9.14 Additional Visualization

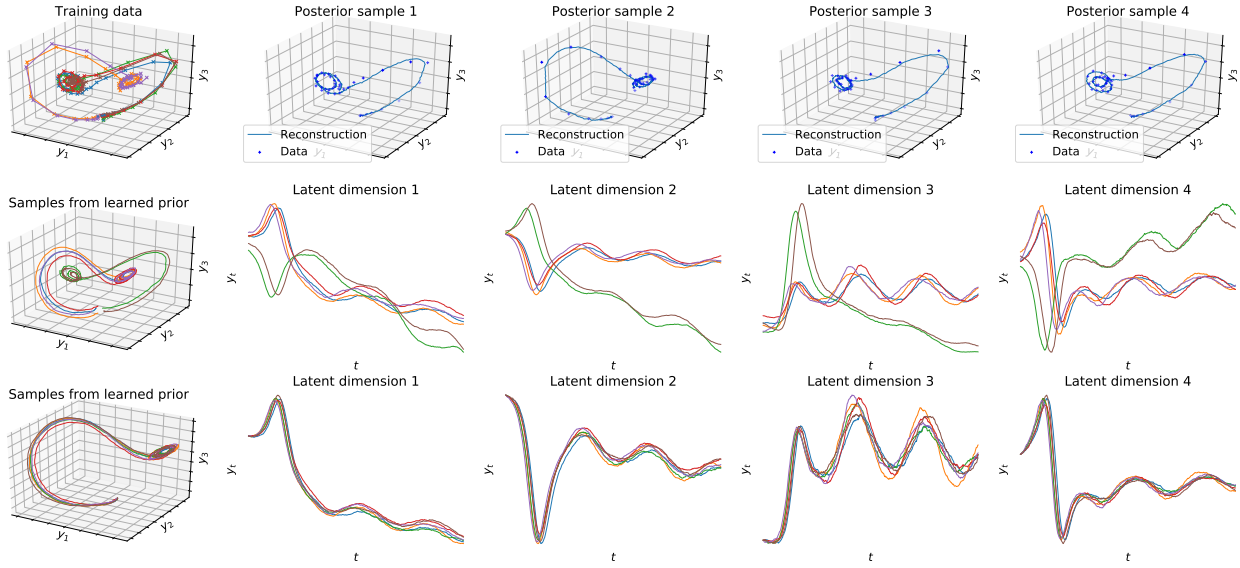


Figure 8: Additional visualizations of learned posterior and prior dynamics on the synthetic stochastic Lorenz attractor dataset. First row displays the true data and posterior reconstructions. Second row displays samples with initial latent state for each trajectory is sampled independently. Third row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

9.15 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 [Yıldız et al., 2019] 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

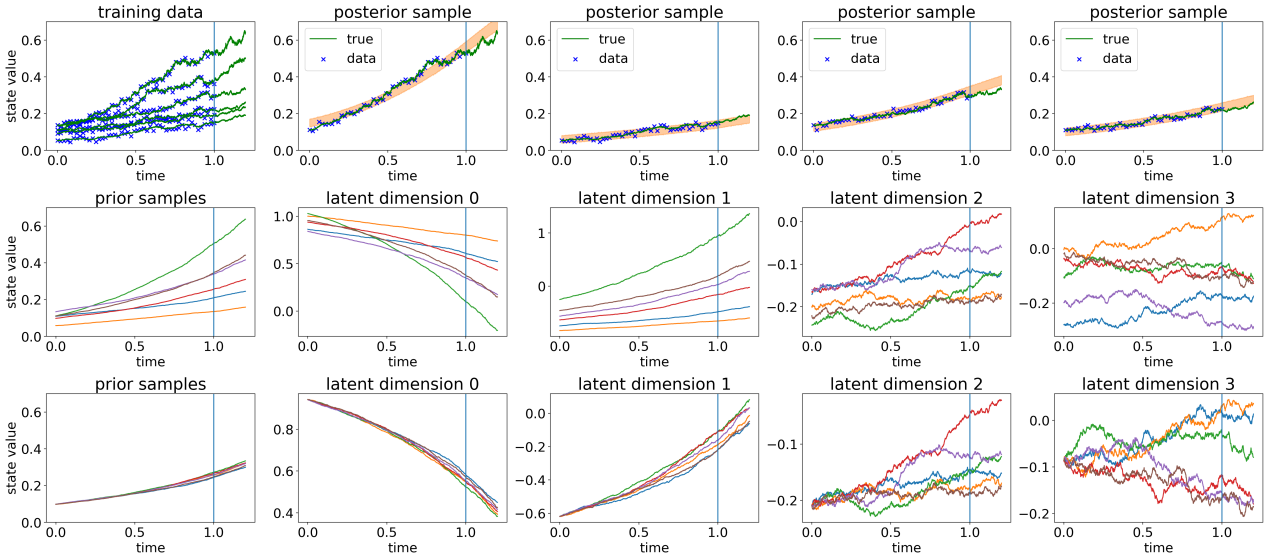


Figure 9: Visualizations of learned posterior and prior dynamics on the synthetic geometric Brownian motion dataset. First row displays the true data and posterior reconstructions. Orange contour covers 95% of 512 samples. Second row displays samples with initial latent state for each trajectory is sampled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

corresponding dimension such that the posterior SDE has diagonal noise. We use the same observation likelihood as that of the ODE^{2}VAE model [Yıldız et al., 2019]. 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.

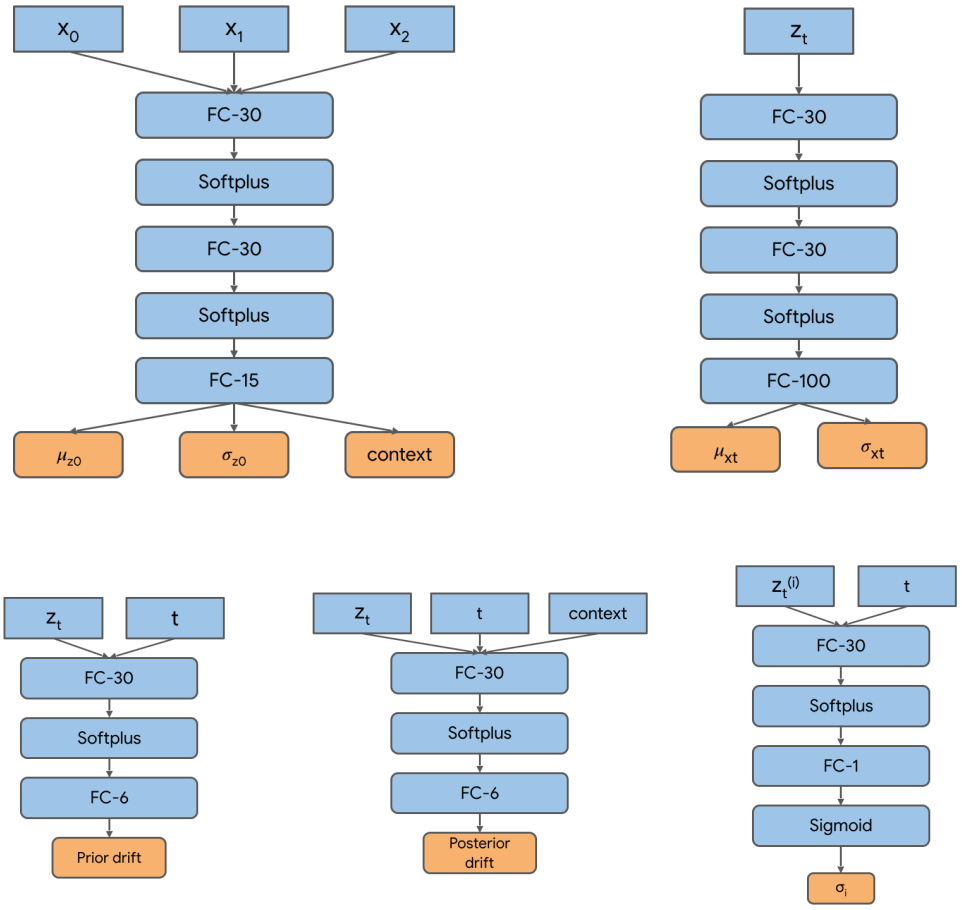


Figure 10: Architecture specifics for the latent SDE model used to train on the mocap dataset. First row from left to right are the encoder and decoder. Second row from left to right are the prior drift, posterior drift, and diffusion functions.

9.16 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)
```

10 Discussion

We presented a generalization of the adjoint method to compute gradients through solutions of SDEs. In contrast to existing approaches, this method has nearly the same time and memory complexity as solving the SDE itself. We showed how our 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. 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 opportunities for fitting any differentiable SDE model, such as Wright-Fisher models with selection and mutation parameters [Ewens, 2012], derivative pricing models in finance, or infinitely-deep Bayesian neural networks [Peluchetti and Favaro, 2019]. In addition, the latent SDE model enabled by our framework can be extended to include domain knowledge and structural or stationarity constraints [Ma et al., 2015] 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.