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 [Andersson, 2013, Andersson et al., 2019, Pearlmutter, 1995, Pontryagin, 2018]. Recently, it was combined with modern reverse-mode automatic differentiation packages, enabling ODEs with millions of parameters to be fit to data [Chen et al., 2018] and allowing more flexible density estimation and time-series models [Grathwohl et al., 2019, Jia and Benson, 2019, Rubanova et al., 2019].

Stochastic differential equations (SDEs) generalize ODEs, adding instantaneous noise to their dynamics [Øksendal, 2003, Särkkä, 2013, Särkkä and Solin, 2019]. They are a natural model for phenomena governed by many small and unobserved interactions, such as motion of molecules in a liquid [Brown, 1828], allele frequencies in a gene pool [Ewens, 2012], or prices in a market [Shreve, 2004]. Previous attempts on fitting SDEs mostly relied on methods with poor scaling properties. The *pathwise approach* [Gobet and Munos, 2005, Yang and Kushner, 1991], 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 [Giles and Glasserman, 2006] 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
Pathwise approach	$\mathcal{O}(1)$	$\mathcal{O}(LD)$
Backprop through solver	$\mathcal{O}(L)$	$\mathcal{O}(L)$
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.

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

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

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, 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 [Chen et al., 2018, Rubanova et al., 2019], Gaussian state-space models [Kitagawa and Gersch, 1996, Turner et al., 2010], and deep Kalman filters [Krishnan et al., 2017], 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

We present background on stochastic flows of diffeomorphisms, which justifies "running SDEs backwards" and inspired our new adjoint method. Due to space constraints, we include background on backward Stratonovich SDEs and the adjoint method for ODEs in Appendix 9.

2.1 Stochastic Flow of Diffeomorphisms

It is well known that an ODE defines a flow of diffeomorphisms [Arnold, 1978]. Here we consider the stochastic analog for the Stratonovich SDE

$$Z_T = z_0 + \int_0^T b(Z_t, t)\ \mathrm{d}t + \int_0^T \sigma(Z_t, t) \circ \mathrm{d}W_t. \quad (1)$$

Throughout, 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 from 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 ([Kunita, 2019, Thm. 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 (1).

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

$$\overleftarrow{\Psi}_{s,t}(z) = z - \int_s^t b(\overleftarrow{\Psi}_{u,t}(z), u)\ \mathrm{d}u - \int_s^t \sigma(\overleftarrow{\Psi}_{u,t}(z), u) \circ \mathrm{d}\overleftarrow{W}_u, \quad (2)$$

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

The coefficients in (1) and (2) differ by only a negative sign. This symmetry is due to our use of the Stratonovich integral (see Figure 1).

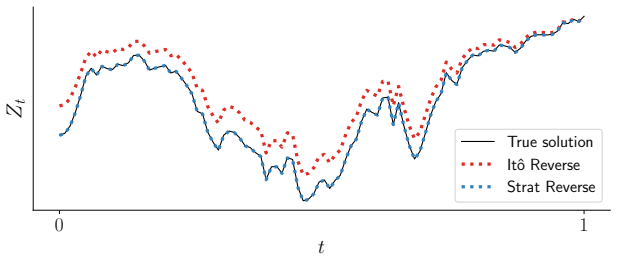


Figure 1: Negating the drift and diffusion functions for an Itô SDE and simulating backwards from the end state gives the wrong solution. Negating the drift and diffusion functions for the converted Stratonovich SDE gives the correct path when simulated backwards.

3 Sensitivity via Stochastic Adjoint

We present our main contribution: a stochastic analog of the adjoint sensitivity method for SDEs. We use (2) 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 $Z_t = \overleftarrow{\Psi}_{t,T}(Z_T)$ 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* [Rogers and Williams, 2000]. 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 (2) as in Theorem 2.1(b). Letting $J_s(z) := \nabla_z \breve{\Psi}_{s,T}(z)$, we have*

$$J_{s,t}(z) = I_d - \int_s^t \nabla b(\breve{\Psi}_{r,t}(z), r) J_{r,t}(z)\ \mathrm{d}r - \int_s^t \nabla \sigma(\breve{\Psi}_{r,t}(z), r) J_{r,t}(z) \circ \mathrm{d}\widetilde{W}_r, \quad (3)$$

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

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

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

The proof included in Appendix 9.5 relies on Itô's lemma in the Stratonovich form [Kunita, 2019, 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

$$\breve{A}_{s,t}(z) := A_{s,t}(\breve{\Psi}_{s,t}(z)) = \nabla \mathcal{L}(z) \nabla \Phi_{s,t}(\breve{\Psi}_{s,t}(z)) = \nabla \mathcal{L}(z) K_{s,t}(z). \quad (5)$$

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

$$\begin{aligned} \breve{A}_{s,t}(z) =& \nabla \mathcal{L}(z) + \int_s^t \nabla b(\breve{\Psi}_{r,t}(z), r)^\top \breve{A}_{r,t}(z)\ \mathrm{d}r + \int_s^t \nabla \sigma(\breve{\Psi}_{r,t}(r), r)^\top \breve{A}_{r,t}(z) \circ \mathrm{d}\widetilde{W}_r, \\ \breve{\Psi}_{s,t}(z) =& z - \int_s^t b(\breve{\Psi}_{r,t}(z), r)\ \mathrm{d}r - \int_s^t \sigma(\breve{\Psi}_{r,t}(z), r) \circ \mathrm{d}\widetilde{W}_r. \end{aligned} \quad (6)$$

Since the drift and diffusion functions of this augmented SDE system are $C_b^{\infty,1}$, it has a unique strong solution. Let $t = 0$. Since (6) admits a strong solution, we may write

$$\breve{A}_0(z) = \mathsf{F}(z, W_\cdot), \quad (7)$$

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) [Rogers and Williams, 2000, Chapter V, Definition 10.9]. Intuitively, F can be thought as a black box that computes the solution to the backward SDE system (6) 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 (1). The next theorem follows immediately from (5) and the definition of F.

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

$$A_{0,T}(z) = \breve{A}_0(z) = \mathsf{F}(\mathsf{G}(z, W_\cdot(\omega)), W_\cdot(\omega)),$$

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

Proof. This is a consequence of composing $A_{0,T}(z) = \breve{A}_{0,T}(\Phi_{0,T}(z))$ and (7). □

This shows that one can obtain the gradient by "composing" the backward SDE system (6) with the original forward SDE (1) 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 $\mathsf{G}_h(z, W_\cdot) \to \mathsf{G}(z, W_\cdot)$ converge to 0 in probability as $h \to 0$, and (ii) for any $M > 0$, we have*

$\sup_{|z|\leq M} |\mathsf{F}_h(z, W_\cdot) - \mathsf{F}(z, W_\cdot)| \to 0$ *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 9.6 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 [Kloeden and Neuenkirch, 2007], satisfying (i). While (ii) is strong, we note that the SDEs considered here have smooth coefficients and thus 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 9.7, we prove 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 3, assuming access only to a black-box solver `sdeint`. All terms required for the augmented dynamics, such as $a_t^\top \partial b/\partial\theta$ and $a_t^\top \partial\sigma/\partial\theta$ can be cheaply evaluated by calling $\texttt{vjp}(a_t, b, \theta)$ 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 [Wiktorsson et al., 2001].

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 [Rößler, 2004]. In that case, we can safely adopt certain numerical schemes of strong order 1.0 (e.g. Milstein [Milstein, 1994] and stochastic Runge-Kutta [Rößler, 2010]) without approximating multiple integrals or the Lévy area during simulation. We formally show this in Appendix 9.8.

One may also consider numerical schemes with high weak order [Kloeden and Platen, 2013]. However, analysis of this scenario is beyond the current scope.

3.4 Software and Implementation

We have implemented several common SDE solvers in PyTorch [Paszke et al., 2017] with adaptive time-stepping using a PI controller [Burrage et al., 2004, Ilie et al., 2015]. Following `torchdiffeq` [Chen et al., 2018], 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 9.16 and plan to release remaining code.

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 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.

4.1 Brownian Bridges and Brownian Trees

Lévy's *Brownian bridge* [Revuz and Yor, 2013] 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:

$$\mathcal{N}\left(\frac{(t_e - t)w_s + (t - t_s)w_e}{t_e - t_s}, \frac{(t_e - t)(t - t_s)}{t_e - t_s} I_d\right). \quad (8)$$

[1] A numerical scheme is of strong order p if $\mathbb{E}\left[|X_T - X_{N\eta}|\right] \leq C\eta^p$ 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.

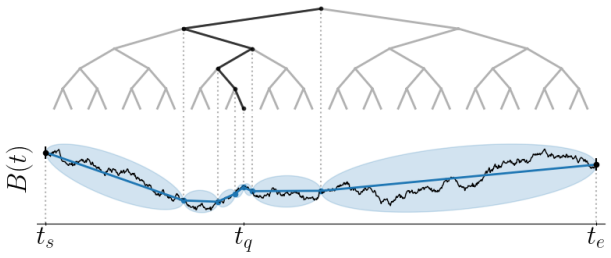


Figure 2: Evaluating a Brownian motion sample at time t_q using a virtual Brownian tree. Our algorithm repeatedly bisects the interval, sampling from a Brownian bridge at each halving to determine intermediate values. Each call to the random number generator uses a unique key whose value depends on the path taken to reach it.

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* [Gaines and Lyons, 1997]. 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 [Claessen and Pałka, 2013], which has an operation `split` that deterministically generates two keys from an existing key. Given a key, the function `BrownianBridge` samples deterministically from (8). 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 1.

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* [Salmon et al., 2011] which avoids passing large random states, and instead simply passes integers. Table 1 compares the asymptotic time complexity of this approach against existing alternatives.

Algorithm 1 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}

5 Latent SDE

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:

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

where h_θ, h_ϕ, and σ are Lipschitz in both arguments, and both processes have the same starting value.

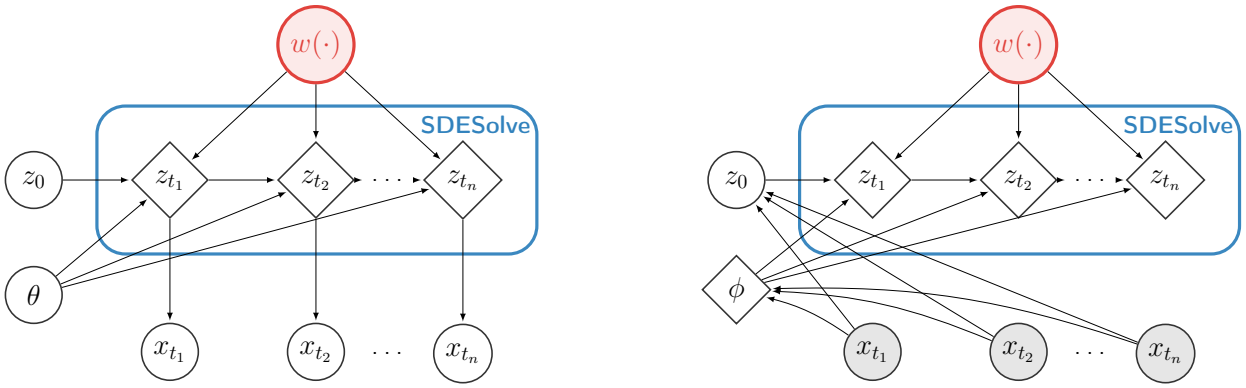
If both processes share the same diffusion function σ, then the KL divergence between them is finite, and can be estimated by sampling paths from the posterior process. Then, the evidence lower bound (ELBO) [Opper, 2019] can be written as:

$$\log p(x_1, x_2, \ldots, x_N|\theta) \geq \qquad (9)$$
$$\mathbb{E}_{Z_t}\left[\sum_{i=1}^{N} \log p(x_{t_i}|z_{t_i}) - \int_0^T \frac{1}{2}|u(z_t, t)|^2\,\mathrm{d}t\right],$$

where

$$u(z, t) = \sigma(z, t)^{-1}\left(h_\theta(z, t) - h_\phi(z, t)\right),$$

$\sigma(z, t)^{-1}$ is the left inverse, and the expectation is taken over the approximate posterior process defined by (approx. posterior). 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 the corresponding times.



(a) Generation (b) Recognition

Figure 3: 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.

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 (6). We include a detailed derivation in Appendix 9.10. 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 3, can be viewed as an infinite-dimensional variational autoencoder.

6 Related Work

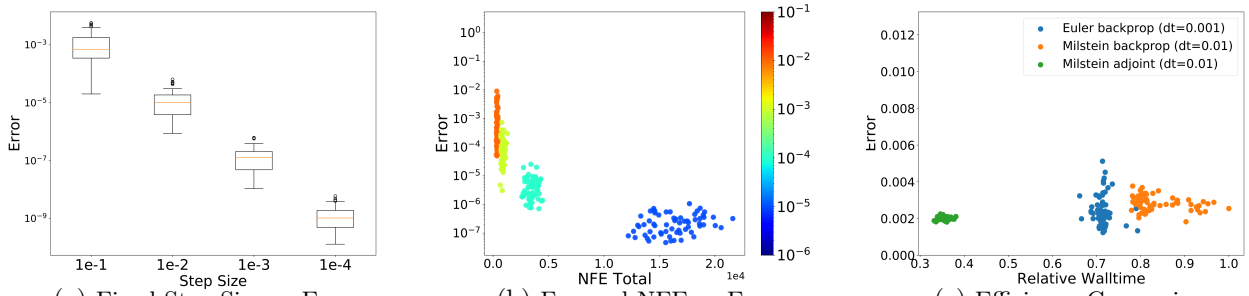
Sensitivity Analysis for SDEs. Gradient computation is closely related to sensitivity analysis. Computing gradients with respect to parameters of vector fields of an SDE has been extensively studied in the stochastic control literature [Kushner and Dupuis, 2013]. In particular, for low dimensional problems, this is done effectively using dynamic programming [Baxter and Bartlett, 2001] and finite differences [Glasserman and Yao, 1992, L'Ecuyer and Perron, 1994]. However, both approaches scale poorly with the dimensionality of the parameter vector.

Analogous to REINFORCE (or the score-function estimator) [Glynn, 1990, Kleijnen and Rubinstein, 1996, Williams, 1992], Yang and Kushner [1991] considered deriving the gradient as $\nabla \mathbb{E}\left[\mathcal{L}(Z_T)\right] = \mathbb{E}\left[\mathcal{L}(Z_T)H\right]$ 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 [2005] extended this approach by weakening a non-degeneracy condition using Mallianvin calculus.

Closely related to the current approach is the pathwise method [Yang and Kushner, 1991], which is also a continuous-time analog of the reparameterization trick [Kingma and Welling, 2013, Rezende et al., 2014]. Existing methods in this regime [Gobet and Munos, 2005, Liu et al., 2019, Tzen and Raginsky, 2019a] 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 [2006] 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 [Giles and Glasserman, 2006], has high memory cost, and relies on a fixed Euler-Maruyama discretization. Recently, this approach was also used by Hegde et al. [2019] to learn parameterized drift and diffusion functions of an SDE. In scientific computing, Innes et al. [2019] considered backpropagating through high-order implicit SDE solvers.

In the machine learning literature, Ryder et al. [2018] 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 vari-



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

Figure 4: (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.

ational inference for non-discretized SDEs presented in previous works [Ha et al., 2018, Opper, 2019, Tzen and Raginsky, 2019a] 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 [2019], 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 [Pardoux and Peng, 1992, Peng, 1990]. Based on the latter notion, forward-backward SDEs (FBSDEs) have been proposed to solve stochastic optimal control problems [Peng and Wu, 1999]. However, simulating FBSDEs is costly due to the need to estimate conditional expectations in the backward pass [Pardoux and Peng, 1992].

Bayesian Learning of SDEs. Recent works have considered the problem of inferring an approximate posterior SDE given observed data under a prior SDE with the same diffusion coefficient [Ha et al., 2018, Opper, 2019, Tzen and Raginsky, 2019a]. In particular, computing the KL divergence between two SDEs over a finite time horizon has been well-explored in the control literature [Kappen and Ruiz, 2016, Theodorou, 2015]. We include background on this topic in Appendix 9.9.

Bayesian learning and parameter estimation of SDEs has a long history [Gupta and Mehra, 1974]. Techniques which don't require positing a variational family such as extended Kalman filter and Markov chain Monte Carlo have been considered in the literature [Mbalawata et al., 2013].

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 [Rackauckas and Nie, 2017]; details in Appendix 9.11), 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 [Chen et al., 2018].

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 9.12.

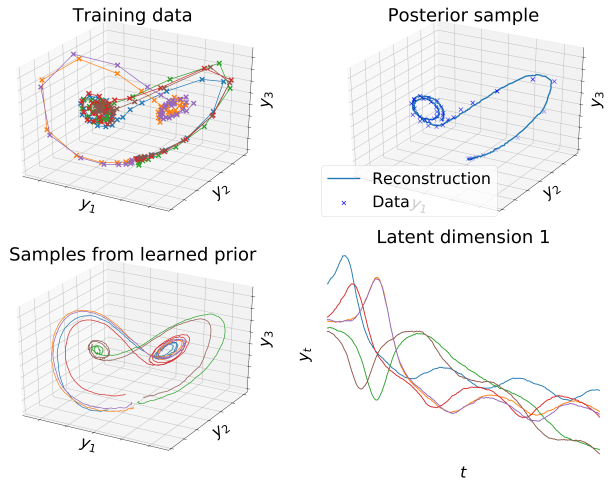


Figure 5: Learned posterior and prior dynamics on data from a stochastic Lorenz attractor. All samples from our model are continuous-time paths, and form a multi-modal, non-Gaussian distribution.

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 (9) 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 9.13.

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 5 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 9.14.

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. [2015], 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. [2007].

In designing the recognition network, we follow Yıldız et al. [2019] 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 [Vaswani et al., 2017]. The overall architecture is described in Appendix 9.15 and is similar to that of ODE^{2}VAE [Yıldız et al., 2019], with a similar number of parameters. We also use a fixed step size 1/5 of smallest interval between any two observations [Yıldız et al., 2019].

We train latent ODE and latent SDE models with the Adam optimizer [Kingma and Ba, 2014] 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 [Higgins et al., 2017], 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. [2019]. 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 [Yıldız et al., 2019].

Method	Test MSE
DTSBN-S [Gan et al., 2015]	$34.86 \pm 0.02^\dagger$
npODE [Heinonen et al., 2018]	$22.96^\dagger$
NeuralODE [Chen et al., 2018]	$22.49 \pm 0.88^\dagger$
ODE^{2}VAE [Yıldız et al., 2019]	$10.06 \pm 1.4^\dagger$
ODE^{2}VAE-KL [Yıldız et al., 2019]	$8.09 \pm 1.95^\dagger$
Latent ODE [Rubanova et al., 2019]	5.98 ± 0.28
Latent SDE (this work)	$\mathbf{4.03 \pm 0.20}$

Acknowledgements

We thank Yulia Rubanova, Danijar Hafner, Mufan Li, Shengyang Sun, Kenneth R. Jackson, Simo Särkkä, and Daniel Lacker 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

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.

R Adams. *Sobolev Spaces*. Academic Press, 1975.

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

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.

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

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

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

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.

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.

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.

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.

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.

Koen Claessen and Michał H Pałka. Splittable pseudorandom number generators using cryptographic hashing. In *ACM SIGPLAN Notices*, volume 48, pages 47–58. ACM, 2013.

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

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

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.

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.

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

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

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

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.

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.

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.

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.

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

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.

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

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.

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.

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.

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

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

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

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

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

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.

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.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

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.

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

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

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

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.

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.

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.

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

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.

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

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

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

Bernt Øksendal. *Stochastic Differential Equations*. Springer, 2003.

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

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

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.

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.

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

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

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

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.

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

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

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.

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

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

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

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

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.

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

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

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

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

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.

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

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

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

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

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.

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

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

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.

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.

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.

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

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.

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