A DERIVATION OF THE ROUGH ADJOINT EQUATION

In this section, we will present a "rough path" derivation of the adjoint equation for Neural SDEs. Since rough path theory is a well developed field, much of our analysis involves quoting key results. To begin, we recall the informal statement of the theorem that we wish to prove:

Theorem (Informal). *Consider the Stratonovich SDE*

$$\mathrm{d}X_t = \mu_\theta(t, X_t)\,\mathrm{d}t + \sigma_\theta(t, X_t) \circ \mathrm{d}W_t, \tag{4}$$

where μ_θ and $\sigma_\theta = \{\sigma_\theta^i\}_{1\leq i\leq d}$ are sufficiently regular vector fields. Let L be a scalar loss on X_T. Then the adjoint process $a_t = \mathrm{d}L(X_T)/\mathrm{d}X_t$ is a strong solution of the linear Stratonovich SDE

$$\mathrm{d}a_t = -(a_t \cdot \nabla)\,\mu_\theta(t, X_t)\,\mathrm{d}t - (a_t \cdot \nabla)\,\sigma_\theta(t, X_t) \circ \mathrm{d}W_t \tag{5}$$

for $t \in [0, T]$. In particular W_t is the same Brownian noise as used in the forward pass.

Whilst the above theorem looks simple enough, it provides us with three main challenges to address:

The first challenge in proving this theorem is that Brownian sample paths are not differentiable and thus the adjoint process will not be differentiable. In particular, we cannot use the proof given by Chen et al. (2018) where the derivative of the adjoint process is approximated using a Taylor series.

The second challenge is more subtle and relates to fact that Brownian sample paths do not have bounded variation. In particular, this means that we cannot define integrals with respect to Brownian sample paths in the Riemann-Stieltjes sense (this is discussed in Section 1.5 of Lyons et al. (2007)).

The third challenge is purely technical in that the vector fields of the adjoint equation (5) do not satisfy certain technical conditions. Typical assumptions in rough path theory are that the vector fields are either bounded (and with some smoothness) or linear. However the adjoint vector fields are linear in a but nonlinear in X; overall they are unbounded and nonlinear. Therefore our analysis will involve separating the linear part of the adjoint equation from the bounded nonlinear part.

The outline of this section is as follows. In subsection A.1, we will derive the adjoint equation for systems where the "driving path" has bounded variation but can be non-differentiable (Challenge 1). In subsection A.2, we will discuss some aspects of rough path theory – which provides a "pathwise" integration theory for SDEs (Challenge 2). Finally, in subsection A.3, we shall put the various pieces together and derive the *rough adjoint equation* for Stratonovich SDEs (Challenge 3).

A.1 THE ADJOINT FOR CONTROLLED DIFFERENTIAL EQUATIONS

Before we consider SDEs and Brownian motion, we first derive the adjoint equation for a slightly more manageable class of differential equation – namely the *controlled differential equation*.[3] A CDE takes a similar form to an SDE, except the system is "controlled" by a continuous path X instead of Brownian motion with time (that is, we write $\mathrm{d}X_t$ instead of $\mathrm{d}t$ or $\mathrm{d}W_t$). By assuming that X has bounded variation, we can use Riemann-Stieltjes integration to define well-posed CDEs (existence and uniqueness results for CDE solutions are given in Chapter 3 of Friz & Victoir (2010)).

Theorem A.1 (Adjoint equation for CDEs that are driven by bounded variation paths). *Consider the controlled differential equation,*

$$\mathrm{d}y_t = \sum_{i=1}^{d} f_\theta^i(y_t)\,\mathrm{d}X_t^i, \tag{6}$$

$$y_0 = \xi \in \mathbb{R}^n, \tag{7}$$

where $X : [0, T] \to \mathbb{R}^d$ is continuous bounded variation path and each $f_\theta^i : \mathbb{R}^n \to \mathbb{R}^n$ is bounded, differentiable and with bounded first derivatives. Let $L : \mathbb{R}^n \to \mathbb{R}$ be a differentiable loss function. Then the adjoint process

$$a_t := \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_t}, \tag{8}$$

[3] Also referred to in the literature as a rough differential equation.

satisfies the following linear CDE

$$\mathrm{d}a_t = -\sum_{i=1}^{d} a_t \nabla f_\theta^i(y_t)\,\mathrm{d}X_t^i. \tag{9}$$

Proof. For $s \le t$, let $\Psi_{s,t} : \mathbb{R}^n \to \mathbb{R}^n$ be the "time-reversed" flow map for the CDE (7) on $[s,t]$. So for $y \in \mathbb{R}^n$, $\Psi_{s,t}(y)$ is the solution of the CDE (7) at time s so that its future value at time t is y. Since X has bounded variation, $\Psi_{s,t}$ is well-defined (via Riemann-Stieltjes integration) and satisfies

$$y = \Psi_{s,t}(y) + \sum_{i=1}^{d} \int_s^t f_\theta^i\big(\Psi_{u,t}(y)\big)\,\mathrm{d}X_u^i. \tag{10}$$

It was shown by Theorem 4.4 in Friz & Victoir (2010) that CDE flows have directional derivatives. As a result of this theorem, taking the gradient of (10) is possible and rearranging gives

$$\nabla\Psi_{s,t}(y) = \mathrm{Id} - \sum_{i=1}^{d} \int_s^t \nabla f_\theta^i\big(\Psi_{u,t}(y)\big)\nabla\Psi_{u,t}(y)\,\mathrm{d}X_u^i. \tag{11}$$

Applying the chain rule to the adjoint process $a_t = \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_t}$ gives

$$a_t = \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_t} = \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_s}\frac{\mathrm{d}y_s}{\mathrm{d}y_t}, \tag{12}$$

where $\frac{\mathrm{d}y_s}{\mathrm{d}y_t}$ is the Jacobian matrix given by $\nabla\Psi_{s,t}(y_t)$, and so

$$a_t = a_s\nabla\Psi_{s,t}(y_t). \tag{13}$$

Thus, substituting (11) into the above yields

$$a_t = a_s - a_s\bigg(\sum_{i=1}^{d}\int_s^t \nabla f_\theta^i\big(y_u\big)\nabla\Psi_{u,t}(y_t)\,\mathrm{d}X_u^i\bigg).$$

So by the above equation along with the triangle inequality, we have

$$\begin{aligned}
&\bigg\|a_t - \bigg(a_s - \sum_{i=1}^{d}\int_s^t a_u\nabla f_\theta^i(y_u)\,\mathrm{d}X_u^i\bigg)\bigg\| \\
&\quad\le \bigg\|a_t - \bigg(a_s - \sum_{i=1}^{d}\int_s^t a_s\nabla f_\theta^i(y_u)\,\mathrm{d}X_u^i\bigg)\bigg\| + \bigg\|\sum_{i=1}^{d}\int_s^t (a_u - a_s)\nabla f_\theta^i(y_u)\,\mathrm{d}X_u^i\bigg\| \\
&\quad= \bigg\|a_s\bigg(\sum_{i=1}^{d}\int_s^t \nabla f_\theta^i\big(y_u\big)\big(\nabla\Psi_{u,t}(y_t) - \mathrm{Id}\,\big)\,\mathrm{d}X_u^i\bigg)\bigg\| + \bigg\|\sum_{i=1}^{d}\int_s^t (a_u - a_s)\nabla f_\theta^i(y_u)\,\mathrm{d}X_u^i\bigg\|.
\end{aligned} \tag{14}$$

In order to estimate these terms, we consider the matrix-valued path $M^{t,y} : [s,t] \to \mathbb{R}^{n\times n}$ given by

$$M_u^{t,y} := -\sum_{i=1}^{d}\int_u^t \nabla f_\theta^i(\Psi_{v,t}(y))\,\mathrm{d}X_v^i,$$

so that equation (14) becomes

$$\begin{aligned}
&\bigg\|a_t - \bigg(a_s - \sum_{i=1}^{d}\int_s^t a_u\nabla f_\theta^i(y_u)\,\mathrm{d}X_u^i\bigg)\bigg\| \\
&\quad\le \bigg\|a_s\int_s^t \mathrm{d}M_u^{t,y_t}\big(\nabla\Psi_{u,t}(y_t) - \mathrm{Id}\,\big)\bigg\| + \bigg\|\int_s^t (a_u - a_s)\,\mathrm{d}M_u^{t,y_t}\bigg\|.
\end{aligned} \tag{15}$$

We use the notation $\|\gamma\|_{1\text{-var};[s,t]}$ to denote the total variation (or 1-variation) of a path $\gamma : [s,t] \to \mathbb{R}^k$,

$$\|\gamma\|_{1\text{-var};[s,t]} := \sup_{\mathcal{D}} \sum_i \|\gamma_{t_{i+1}} - \gamma_{t_i}\|,$$

where $\|\cdot\|$ is a norm on $\mathbb{R}^k$ (we use $k = d, n^2$). The supremum is taken over all partitions $\mathcal{D}$ of $[s,t]$.

It is worth noting that since $u \mapsto \nabla f_i(y_u, \theta)$ is continuous, it is bounded for $u \in [s,t]$. As a result, M^{u,y_u} has bounded variation on $[s,u]$ and there exists a constant C_1 (depending only on t) such that

$$\|M^{u,y_u}\|_{1\text{-var};[s,u]} \leq C_1 \|X\|_{1\text{-var};[s,t]}, \tag{16}$$

for $u \in [s,t]$ with s and t sufficiently close together.

We can rewrite (11) as the following linear CDE:

$$\mathrm{d}z_u = -\big(\mathrm{d}M_u^{v,y_v}\big) z_u\,,$$
$$z_0 = \mathrm{Id},$$

where $z_u := \nabla \Psi_{u,v}(y_v)$ for $s \leq u \leq v \leq t$. Since the path M^{v,y_v} has bounded variation, by Davie's lemma for linear CDEs (Lemma 10.56 in Friz & Victoir (2010)), there exists a constant C_2 such that

$$\big\|z_u - z_0\big\| \leq C_2 \|M^{v,y_v}\|_{1\text{-var};[s,v]}.$$

for $s \leq u \leq v \leq t$ whenever s is sufficiently close to t and we note that $z_u - z_0 = \nabla \Psi_{u,v}(y_v) - \mathrm{Id}$.

Hence by the total variation estimate (16), there exists a constant C_3 depending only on t, such that

$$\big\|\nabla \Psi_{u,v}(y_v) - \mathrm{Id}\,\big\| \leq C_3 \|X\|_{1\text{-var};[s,t]}, \tag{17}$$

for $s \leq u \leq v \leq t$ whenever s is sufficiently close to t.

Since a is continuous, it is bounded on $[s,t]$ and so it follows from (13) with the estimate (17) that

$$\begin{aligned}
\left\| \int_s^t (a_u - a_s)\, \mathrm{d}M_u^{t,y_t} \right\| &\leq \sup_{u\in[s,t]} \Big(\|a_u - a_s\| \Big) \big\| M^{t,y_t} \big\|_{1\text{-var};[s,t]} \\
&\leq \sup_{u\in[s,t]} \Big(\big\| a_s \nabla \Psi_{s,u}(y_u) - a_s \big\| \Big) \big\| M^{t,y_t} \big\|_{1\text{-var};[s,t]} \\
&\leq \sup_{u\in[s,t]} \Big(\|a_s\| \big\| \nabla \Psi_{s,u}(y_u) - \mathrm{Id}\,\big\| \Big) \big\| M^{t,y_t} \big\|_{1\text{-var};[s,t]} \\
&\leq C_4 \|X\|^2_{1\text{-var};[s,t]},
\end{aligned}$$

and

$$\begin{aligned}
\left\| a_s \int_s^t \mathrm{d}M_u^{t,y_t} \Big(\nabla \Psi_{u,t}(y_t) - \mathrm{Id} \Big) \right\| &\leq \sup_{u\in[s,t]} \Big(\|a_s\| \, \big\| \nabla \Psi_{u,t}(y_t) - \mathrm{Id}\,\big\| \Big) \big\| M^{t,y_t} \big\|_{1\text{-var};[s,t]} \\
&\leq C_5 \|X\|^2_{1\text{-var};[s,t]},
\end{aligned}$$

where the constants C_4 and C_5 only depends on t (provided that $\epsilon := t - s$ is sufficiently small). Therefore equation (15) for the adjoint process becomes

$$\left\| a_t - \left(a_s - \sum_{i=1}^d \int_s^t a_u \nabla f_\theta^i(y_u)\, \mathrm{d}X_u^i \right) \right\| \leq (C_4 + C_5) \|X\|^2_{1\text{-var};[s,t]}.$$

In other words, for a fixed t, we have

$$a_t = a_s - \sum_{i=1}^d \int_s^t a_u \nabla f_\theta^i(y_u)\, \mathrm{d}X_u^i + O\Big(\|X\|^2_{1\text{-var};[s,t]} \Big),$$

provided that s is sufficiently close to t. Thus, letting $s \to t^-$ gives

$$\mathrm{d}a_t = -\sum_{i=1}^d a_t \nabla f_\theta^i(y_t)\, \mathrm{d}X_t^i,$$

as required. □

A.2 THE ROUGH PATH APPROACH TO STOCHASTIC DIFFERENTIAL EQUATIONS

In this subsection, we shall briefly outline the "pathwise solution" theory for SDEs that was made possible by the advent of rough path theory (originally proposed in Lyons (1998)). Whilst rough path theory extends beyond the SDE setting, this is not within the scope of this paper.

Let $\big(\Omega, \mathcal{F}, \mathbb{P}\,; \{\mathcal{F}_t\}_{t\geq 0}\big)$ be a filtered probability space containing a d-dimensional Brownian motion. Since the Brownian motion $W : \Omega \times [0,\infty) \to \mathbb{R}^n$ corresponds to a certain Gaussian measure on (infinite-dimensional) path space, it must be discretised in order to be used in SDE simulation. Moreover, by constructing a sequence of approximations converging to the Brownian path we can extend the adjoint equation from the bounded variation setting (see Theorem A.1) to the SDE setting.

To begin, we give a few key definitions (the signature, p-variation metric and geometric rough path).

Definition A.2. The (depth-2) *signature* of a continuous bounded variation path $X : [0,T] \to \mathbb{R}^d$ is $S^2(X) = \big\{S^2_{s,t}(X)\big\}_{0\leq s\leq t\leq T}$ where $S^2_{s,t}(X)$ is a collection of increments and integrals given by

$$S^2_{s,t}(X) := \left(1, \big\{X^i_t - X^i_s\big\}_{1\leq i\leq d}, \left\{\int_s^t \big(X^i_u - X^i_s\big)\,\mathrm{d}X^j_u\right\}_{1\leq i,j\leq d}\right), \tag{18}$$

where the above is defined using Riemann-Stieltjes integration.

Therefore $S^2(X) : \triangle_T \to \mathbb{R}^{1+d+d^2}$ where $\triangle_T = \{(s,t) \in [0,T]^2 : s < t\}$ is a rescaled 2-simplex.

Definition A.3. For $p \in [2,3)$, the *p-variation metric* between functions $Z^1, Z^2 : \triangle_T \to \mathbb{R}^{1+d+d^2}$ is

$$d_p\big(Z^1, Z^2\big) := \max_{k=1,2} \sup_{\mathcal{D}} \left(\sum_{t_i\in\mathcal{D}} \Big\|\pi_k\big(Z^1_{t_i,t_{i+1}}\big) - \pi_k\big(Z^2_{t_i,t_{i+1}}\big)\Big\|^{\frac{p}{k}}\right)^{\frac{k}{p}}, \tag{19}$$

where π_k denotes the projection map from $\mathbb{R}^{1+d+d^2}$ onto $\mathbb{R}^{d^k}$ (for $k = 1,2$) and the above supremum is taken over all partitions $\mathcal{D}$ of $[0,T]$ and the norms $\|\cdot\|$ must satisfy (up to a constant)

$$\|a \otimes b\| \leq \|a\|\|b\|,$$

for $a, b \in \mathbb{R}^d$. For example, we could use the standard L^2 (operator) norms for vectors and matrices.

Definition A.4. For $p \in [2,3)$, we say that a sequence of continuous bounded variation paths $X^N : [0,T] \to \mathbb{R}^d$ converges in the *p-variation* sense to a continuous map $\boldsymbol{X} : \triangle_T \to \mathbb{R}^{1+d+d^2}$ if

$$d_p\big(S^2\big(X^N\big), \boldsymbol{X}\big) \to 0, \tag{20}$$

as $N \to \infty$. When such a sequence exists, we can refer to the limit $\boldsymbol{X}$ as a geometric p-rough path.

We now state the following result from rough path theory (Corollary 13.22 in Friz & Victoir (2010)).

Theorem A.5 (Brownian motion as a geometric rough path). *Let W be a standard d-dimensional Brownian motion and W^N be the piecewise linear path with N pieces that coincides with W on the uniform partition $\mathcal{D}_N := \{0 = t_0 < t_1 < \cdots < t_N = T\}$ with $t_k := kh$ and mesh size $h := \frac{T}{N}$. Then there exists a random geometric p-rough path $\boldsymbol{W}$ ($p \in (2,3)$) such that for almost all $\omega \in \Omega$,*

$$d_p\Big(S^2\big(W^N\big)(\omega), \boldsymbol{W}(\omega)\Big) \to 0, \tag{21}$$

as $N \to \infty$ for any $p \in (2,3)$ and

$$\boldsymbol{W}(\omega) = \left\{\left(1, \big(W_t - W_s\big)(\omega), \left(\int_s^t \big(W_r - W_s\big) \otimes \circ\,\mathrm{d}W_r\right)(\omega)\right)\right\}_{0\leq s\leq t\leq T}. \tag{22}$$

Hence the geometric rough path $\boldsymbol{W}$ is often referred to as Stratonovich enhanced Brownian motion.

To put simply, this theorem states that Brownian motion can be approximated (in a rough path sense) by a sequence of bounded variation paths. This is particularly helpful within stochastic analysis as it allows one to construct pathwise solutions for SDEs governed by sufficiently regular vector fields. The central result within rough path theory that makes this possible is the *Universal Limit Theorem*. To counter the roughness of Brownian motion, this requires vector fields to have *Lip(γ) regularity*.

Definition A.6 ($\mathrm{Lip}(\gamma)$ functions)**.** A function $f : \mathbb{R}^n \to \mathbb{R}^n$ is said to be $\mathrm{Lip}(\gamma)$ with $\gamma > 1$ if it is bounded with $\lfloor \gamma \rfloor$ bounded derivatives, the last being Hölder continuous with exponent $(\gamma - \lfloor \gamma \rfloor)$. Equivalently, f is $\mathrm{Lip}(\gamma)$ if the following norm is finite:

$$\|f\|_{\mathrm{Lip}(\gamma)} := \max_{0 \le k \le \lfloor \gamma \rfloor} \left\| D^k f \right\|_\infty \vee \left\| D^{\lfloor \gamma \rfloor} f \right\|_{(\gamma - \lfloor \gamma \rfloor)\text{-Höl}}, \tag{23}$$

where $D^k f$ is the k-th (Fréchet) derivative of f and $\|\cdot\|_{\alpha\text{-Höl}}$ is the standard α-Hölder norm for $\alpha \in (0, 1)$. We say that f is $\mathrm{Lip}(1)$ if it is bounded and Lipschitz continuous. That is, if the norm

$$\|f\|_{\mathrm{Lip}(1)} := \left\| f \right\|_\infty \vee \sup_{\substack{x, y \in \mathbb{R}^n \\ x \neq y}} \frac{\left\| f(x) - f(y) \right\|}{\|x - y\|}, \tag{24}$$

is finite.

Theorem A.7 (Universal Limit Theorem for RDEs (Theorem 5.3 in Lyons et al. (2007))). *Let $p \in (2, 3)$ and $x^N : [0, T] \to \mathbb{R}^d$ be a sequence of continuous bounded variation paths which converge in p-variation to a geometric p-rough path $\boldsymbol{x}$. Let $\{f_i\}_{1 \le i \le d}$ denote a collection of $\mathrm{Lip}(\gamma)$ functions on $\mathbb{R}^n$ with $\gamma > p$ and consider the controlled differential equation (CDE)*

$$\begin{aligned} \mathrm{d}y_t^N &= \sum_{i=1}^d f_i(y_t^N)\, \mathrm{d}\big(x^N\big)_t^i, \\ y_0^N &= \xi, \end{aligned}$$

where $\xi \in \mathbb{R}^n$ and the above differential equation is defined using Riemann-Stieltjes integration. Then there exists a unique geometric p-rough path $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y}) : \triangle_T \to \mathbb{R}^{1 + (d+n) + (d+n)^2}$ such that y^N converges to $\boldsymbol{y}$ in p-variation. Moreover, the "universal limit" $\boldsymbol{y}$ depends only on $\boldsymbol{x}$, $\{f_i\}$ and ξ.

Definition A.8. We shall refer to $\boldsymbol{y}$ as the solution of the *rough differential equation* (RDE),

$$\mathrm{d}\boldsymbol{y}_t = \sum_{i=1}^d f_i(\boldsymbol{y}_t)\, \mathrm{d}\boldsymbol{x}_t^i. \tag{25}$$

Remark A.9. Theorem A.7 and Definition A.8 also apply when the vector fields $\{f_i\}$ are linear (Theorem 10.57 in Friz & Victoir (2010)).

Importantly for us, the above theory applies directly to (Stratonovich) SDEs as Brownian motion can be viewed as a geometric p-rough path with $p \in (2, 3)$ by Theorem A.5. We refer the reader to Section 17.2 of Friz & Victoir (2010) for a detailed account of the "rough path approach" to Stratonovich theory. For our purposes, we state a *Universal Limit Theorem for Stratonovich SDEs*

Theorem A.10 (Remark 17.5 in Friz & Victoir (2010)). *Suppose that μ is a $\mathrm{Lip}(1)$ function on $\mathbb{R}^{n+1}$ and $\{\sigma^k\}_{1 \le k \le d}$ are $\mathrm{Lip}(2)$ functions on $\mathbb{R}^{n+1}$. Let $\{W^N\}_{N \ge 1}$ be a sequence of piecewise linear paths converging to the Stratonovich enhanced Brownian motion $\boldsymbol{W}$ given by Theorem A.5. Let $\{y^N\}_{N \ge 1}$ be the sequence of solutions to the following controlled differential equations (CDEs),*

$$\begin{aligned} \mathrm{d}y_t^N &= \mu(t, y_t^N)\, dt + \sum_{i=1}^d \sigma^i(t, y_t^N)\, \mathrm{d}\big(W^N\big)_t^i, \\ y_0 &= \xi \in \mathbb{R}^n, \end{aligned}$$

Then y^N converges in p-variation to a geometric p-rough path $\boldsymbol{y}$ with $p \in (2, 3)$ and the process $y : [0, T] \to \mathbb{R}^n$ given by $y_t := \xi + \pi_1(\boldsymbol{y}_{0,t})$ coincides with the strong solution of the Stratonovich SDE

$$\begin{aligned} \mathrm{d}y_t &= \mu(t, y_t)\, dt + \sum_{i=1}^d \sigma^i(t, y_t) \circ \mathrm{d}W_t^i, \\ y_0 &= \xi, \end{aligned} \tag{26}$$

almost surely.

A.3 THE ADJOINT FOR STOCHASTIC DIFFERENTIAL EQUATIONS

Ideally, we would like to just replace the path X in Theorem A.1 with a Brownian motion (coupled with time, that is to say (t, W_t).). However, that result required that X have bounded variation, whilst sample paths of Brownian motion have infinite total variation. Resolving this difficulty is one of the essential reasons that rough path theory exists (see also in Section 1.5 of Lyons et al. (2007)).

As mentioned previously, the main challenge in applying rough path theory here is that the adjoint equation (29) has nonlinear unbounded vector fields, which are not $\mathrm{Lip}(\gamma)$ functions in (a, y). Our trick is to first derive an adjoint equation for the stochastic process corresponding to the Jacobian (which satisfies assumptions of boundedness), and then to drive the adjoint equation by this Jacobian-valued stochastic process (which satisfies assumptions of linearity).

Theorem A.11 (Adjoint equation for Stratonovich SDEs). *Suppose that μ_θ and $\{\sigma_\theta^k\}_{1\leq k\leq d}$ are bounded functions on $\mathbb{R}^{n+1}$ such that*

- *The drift vector field μ_θ is continuously differentiable with bounded first derivative.*
- *Each noise vector field σ_θ^k is a $\mathrm{Lip}(\gamma)$ function with $\gamma > 2$.*

Consider the (Stratonovich) stochastic differential equation,

$$\begin{aligned} \mathrm{d}y_t &= \mu_\theta(t, y_t)\,\mathrm{d}t + \sum_{i=1}^{d} \sigma_\theta^i(t, y_t) \circ \mathrm{d}W_t^i, \\ y_0 &= \xi \in \mathbb{R}^n, \end{aligned} \tag{27}$$

and let $L : \mathbb{R}^n \to \mathbb{R}$ denote a differentiable loss function. Then the adjoint process

$$a_t := \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_t}, \tag{28}$$

coincides with the strong solution of the linear Stratonovich SDE

$$\mathrm{d}a_t = -a_t \nabla \mu_\theta(t, y_t)\,\mathrm{d}t - \sum_{i=1}^{d} a_t \nabla \sigma_\theta^i(t, y_t) \circ \mathrm{d}W_t^i. \tag{29}$$

almost surely.

Proof. Let $\{y^N\}_{N\geq 1}$ be the sequence of solutions to the following controlled differential equations,

$$\begin{aligned} \mathrm{d}y_t^N &= \mu_\theta(t, y_t^N)\,\mathrm{d}t + \sum_{i=1}^{d} \sigma_\theta^i(t, y_t^N)\,\mathrm{d}\big(W^N\big)_t^i, \\ y_0^N &= \xi, \end{aligned} \tag{30}$$

where $\{W^N\}_{N\geq 1}$ are the piecewise linear paths converging to $\boldsymbol{W}$ in p-variation by Theorem A.5. Hence by Theorem A.10, we have that the corresponding sequence of CDE solutions $\{y^N\}_{N\geq 1}$ converges almost surely in p-variation to the solution y of the Stratonovich SDE (27).

Let L be a differentiable loss function so that by Theorem A.1, each adjoint process

$$a_t^N = \frac{\mathrm{d}L\big(y_T^N\big)}{\mathrm{d}y_t^N}, \tag{31}$$

satisfies the linear CDE

$$\mathrm{d}a_t^N = -a_t^N \nabla \mu_\theta(t, y_t^N)\,\mathrm{d}t - \sum_{i=1}^{d} a_t^N \nabla \sigma_\theta^i(t, y_t^N)\,\mathrm{d}\big(W^N\big)_t^i,$$

Just as in the proof of Theorem A.1, we can rewrite the adjoint equation for a^N as

$$\mathrm{d}a_t^N = -a_t^N\,\mathrm{d}M_t^N,$$

where the matrix-valued path $M^N : [0, T] \to \mathbb{R}^{n\times n}$ is given by

$$M_t^N := -\int_t^T \nabla\mu_\theta\big(s, y_s^N\big)\,\mathrm{d}s - \sum_{i=1}^{d}\int_t^T \nabla\sigma_\theta^i\big(s, y_s^N\big)\,\mathrm{d}\big(W^N\big)_s^i. \tag{32}$$

Since the vector fields $\nabla\mu_\theta$ and $\{\nabla\sigma_\theta^i\}_{1\le i\le n}$ are bounded, we see that M^N has bounded variation.

Recall from the universal limit theorem (Theorem A.7) that $(\boldsymbol{x}, \boldsymbol{y})$ was a geometric p-rough path. This carries over to our setting and thus we define the (random) geometric p-rough path $\boldsymbol{z} = (\boldsymbol{W}, \boldsymbol{y})$. Then by Proposition 17.1 in Friz & Victoir (2010), we have that the following rough integral exists

$$\int_0^t \varphi(W, y) \circ \mathrm{d}(W, y) = \pi_1\bigg(\int_0^t \varphi(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z}\bigg),$$

for all $t \in [0, T]$ with probability one, provided that $\varphi = \{\varphi^i\}$ is a collection of $\mathrm{Lip}(\gamma-1)$ functions with $\gamma > p$. Since each vector field σ_θ^i is $\mathrm{Lip}(\gamma)$, it follows that each gradient $\nabla\sigma_\theta^i$ is $\mathrm{Lip}(\gamma - 1)$. Therefore we can apply Proposition 17.1 in Friz & Victoir (2010) to the $\mathrm{d}W^N$ integrals in (32) and, since $\nabla\mu_\theta$ is continuous and bounded, it is clear that the $\mathrm{d}t$ integral in equation (32) also converges.

Thus, due to the regularity of μ_θ and σ_θ, we see that the sequence $\{M^N\}$ converges in p-variation to a geometric p-rough path $\boldsymbol{M}$ and the limiting (matrix-valued) process $M_t := \pi_1\big(\boldsymbol{M}_{t,T}\big)$ satisfies

$$M_t = -\int_t^T \nabla\mu_\theta\big(s, y_s\big)\,\mathrm{d}s - \sum_{i=1}^{d}\int_t^T \nabla\sigma_\theta^i\big(s, y_s\big) \circ \mathrm{d}W_s^i, \tag{33}$$

almost surely. We now have all the ingredients needed to construct the rough adjoint equation (29),

1. Each CDE (30) admits a unique solution y^N and the resulting sequence $\{y^N\}$ converges to the solution y of the SDE (27) almost surely.
2. Each CDE (30) admits a unique adjoint process a^N satisfying a linear CDE driven by M^N.
3. The sequence $\{M_N\}$ converges in p-variation to a geometric p-rough path (almost surely).
4. By Theorem 10.57 in Friz & Victoir (2010), we have that the Universal Limit Theorem (Theorem A.7) also holds for linear RDEs.

Therefore the sequence $\{a^N\}$ converges in p-variation to a geometric p-rough path $\boldsymbol{a}$ almost surely and the process $a_t := \frac{\mathrm{d}L(y_T)}{\mathrm{d}y_0} + \pi_1(\boldsymbol{a}_{0,t})$ coincides with the strong solution to the Stratonovich SDE

$$\begin{aligned}\mathrm{d}a_t &= -a_t \circ \mathrm{d}M_t \\ &= -a_t\nabla\mu_\theta(t, y_t)\,\mathrm{d}t - \sum_{i=1}^{d} a_t\nabla\sigma_\theta^i(t, y_t) \circ \mathrm{d}W_t^i.\end{aligned}$$

almost surely. The fact that a is the adjoint process follows from (31) and the continuity of ∇L. □

Remark A.12. The above argument can also extend to an RDE driven by a geometric p-rough path. In this case, the vector fields governing the differential equation would have to be $\mathrm{Lip}(\gamma)$ with $\gamma > p$.

B Sampling Brownian motion

B.1 Algorithm

We begin with providing the complete traversal and splitting algorithm needed to find or create all intervals in the Brownian Interval, as in Section 3.2. We discuss its operation in the next section.

Here, *List* is an ordered data structure that needs to be appended to, and iterated over sequentially. For example a linked list would suffice. We let `split` denote a splittable PRNG as in Salmon et al. (2011); Claessen & Pałka (2013). We use $*$ to denote an unfilled part of the data structure, equivalent to `None` in Python or a null pointer in C/C++; in particular this is used as a placeholder

for the (nonexistent) children of leaf nodes. We use $=$ to denote the creation of a new local variable, and $\leftarrow$ to denote in-place modification of a variable.

Algorithm 2: Definition of `traverse`

def `bisect`*(I : Node, x : $\mathbb{R}$)*:
    # Only called on leaf nodes
    Let $I = ([a, b], s, I_{\texttt{parent}}, *, *)$
    $s_{\texttt{left}}, s_{\texttt{right}} = \texttt{split}(s)$
    $I_{\texttt{left}} = ([a, x], s_{\texttt{left}}, J, *, *)$
    $I_{\texttt{right}} = ([x, b], s_{\texttt{right}}, J, *, *)$
    $I \leftarrow ([a, b], s, I_{\texttt{parent}}, I_{\texttt{left}}, I_{\texttt{right}})$

def `traverse`*(I : Node, $[c, d]$: Interval,* nodes *: List[Node])*:
    Let $I = ([a, b], s, I_{\texttt{parent}}, I_{\texttt{left}}, I_{\texttt{right}})$

    # Outside our jurisdiction - pass to our parent
    if $c < a$ or $d > b$ **then**
        `traverse`($I_{\texttt{parent}}, [c, d]$, nodes)
        return
    # It's I that is sought. Add I to the list and return.
    if $c = a$ and $d = b$ **then**
        nodes.append(I)
        return
    # Check if I is a leaf or not.
    if I_{left} is $*$ **then**
        # I is a leaf
        if $a = c$ **then**
            # If the start points align then create children and add on the left child.
            # (Which is created in `bisect`.)
            `bisect`(I, d)
            nodes.append($I_{\texttt{left}}$) # nodes is passed by reference
            return
        # Otherwise create children and pass on to our right child.
        # (Which is created in `bisect`.)
        `bisect`(I, c)
        `traverse`($I_{\texttt{right}}, [c, d]$, nodes)
        return
    else
        # I is not a leaf.
        Let $I_{\texttt{left}} = ([a, m], s_{\texttt{left}}, I, I_{ll}, I_{lr})$
        if $d \leq m$ **then**
            # Strictly our left child's problem.
            `traverse`($I_{\texttt{left}}, [c, d]$, nodes)
            return
        if $c \geq m$ **then**
            # Strictly our right child's problem.
            `traverse`($I_{\texttt{right}}, [c, d]$, nodes)
            return
        # A problem for both of our children.
        `traverse`($I_{\texttt{left}}, [c, m]$, nodes)
        `traverse`($I_{\texttt{right}}, [m, d]$, nodes)
        return

B.2 Discussion

The function `traverse` is simply a depth-first tree traversal for locating an interval within a binary tree. The search may split into multiple (potentially parallel) searches (on the last few lines) if the target interval crosses the intervals of multiple existing leaf nodes. If its target is not found then additional nodes are created if needed.

Sections 3.2 and B.1 now between them define the algorithm in technical detail.

There are some further technical considerations worth mentioning. Recall that the context we are explicitly considering is when sampling Brownian motion to solve an SDE forwards in time, then the adjoint backwards in time, and then discarding the Brownian motion. This motivates several of the choices here.

Small intervals First, the access patterns of SDE solvers are quite specific. Queries will be over relatively small intervals: the step that the solver is making. This means that the list of nodes populated by `traverse` is typically small. In our experiments we observed it usually only consisting of a single element; occasionally two. In contrast if the Brownian Interval has built up a reasonable tree of previous queries, and was then queried over $[0, s]$ for $s \gg 0$, then a long (inefficient) list would be returned. It is the fact that SDE solvers do not make such queries that means this is acceptable.

Searching from $\widehat{J}$ Moreover, the queries are either just ahead (fixed-step solvers; accepted steps of adaptive-step solvers) or just before (rejected steps of adaptive-step solvers) previous queries. Thus in Algorithm 1, we keep track of the most recent node $\widehat{J}$, so that we begin `traverse` near to the correct location.

LRU cache The fact that queries are often close to one another is also what makes the strategy of using an LRU (least recently used) cache work. Most queries will correspond to a node that have a recently-computed parent in the cache.

Backward pass The queries are broadly made left-to-right (on the forward pass), and then right-to-left (on the backward pass). (Other than the occasional rejected adaptive step.)

Left to its own devices, the forward pass will thus build up a highly imbalanced binary tree. At any one time, the LRU cache will contain only nodes whose intervals are a subset of some contiguous subinterval $[s, t]$ of the query space $[0, T]$. Letting n be the number of queries on the forward pass, then this means that the backward pass will consume $\mathcal{O}(n^2)$ time – each time the backward pass moves past s, then queries will miss the LRU cache, and a full recomputation to the root will be triggered, costing $\mathcal{O}(n)$. This will then hold only nodes whose intervals are subets of some contiguous subinterval $[u, s]$: once we move past u then this $\mathcal{O}(n)$ procedure is repeated, $\mathcal{O}(n)$ times. This is clearly undesirable.

This is precisely analogous to the classical problem of optimal recomputation for performing backpropagation, whereby a dependency graph is constructed, certain values are checkpointed, and a minimal amount of recomputation is desired; see Griewank (1992).

In principle the same solution may be applied: apply a snapshotting procedure in which specific extra nodes are held in the cache. This is a perfectly acceptable solution, but implementing it requires some additional engineering effort, carefully determining which nodes to augment the cache with.

Fortunately, we have an advantage that Griewank (1992) does not: we have some control over the dependency structure between the nodes, as we are free to prespecify any dependency structure we like. That is, we do not have to start the binary tree as just a stump. We may exploit this to produce an easier solution.

Given some estimate ν of the average step size of the SDE solver, a size of the LRU cache L, and *before a user makes any queries*, we simply make some queries of our own. These queries correspond to the intervals $[0, T/2], [T/2, T], [0, T/4], [T/4, T/2], \ldots$, so as to create a dyadic tree, such that the smallest intervals (the final ones in this sequence) are of size not more than νL. (In practice we use $0.8 \times \nu L$ as an additional safety factor.)

Letting $[s, t]$ be some interval at the bottom of this dyadic tree, where $t \approx s + 0.8\nu L$, then we are capable of holding every node within this interval in the LRU cache. Once we move past s on the backward pass, then we may in turn hold the entire previous subinterval $[u, s]$ in the LRU cache, and in particular the values of the nodes whose intervals lie within $[u, s]$ may be computed in only logarithmic time, due to the dyadic tree structure.

This is now analogous to the Brownian Tree of Gaines & Lyons (1997); Li et al. (2020). (Up to the use of intervals rather than points.) If desired, this approach may be loosely interpreted as placing a Brownian Interval on every leaf of a shallow Brownian Tree.

Recursion errors We find that for some problems, the recursive computations of `traverse` (and in principle also `sample`, but this is less of an issue due to the LRU cache) can occasionally grow very deep. In particular this occurs when crossing the midpoint of the pre-specified tree: for this particular query, the traversal must ascend the tree to the root, and then descend all the way down again. As such `traverse` should be implemented with trampolining and/or tail recursion to avoid maximum depth recursion errors.

CPU vs GPU memory We describe this algorithm as requiring only constant memory. To be more precise, the algorithm requires only constant GPU memory, corresponding to the fixed size of the LRU cache. As the Brownian Interval receives queries then its internal tree tracking dependencies will grow, and CPU memory will increase. For deep learning models, GPU memory is usually the limiting (and so more relevant) factor.

Stochastic integrals What we have not discussed so far is the simulation of integrals such as $\mathbb{W}_{s,t} = \int_s^t W_{s,r} \circ \mathrm{d}W_r$ and $H_{s,t} = \frac{1}{t-s}\int_s^t W_{s,r}\,\mathrm{d}r$ which are used in higher order SDEs solvers (such as the Runge-Kutta methods in Rößler (2010) and the log-ODE method in Foster et al. (2020)). Just like increments $W_{s,t}$, these integrals fit nicely into an interval-based data structure.

In general simulating the pair $(W_{s,t}, \mathbb{W}_{s,t})$ is known to be a difficult problem (Dickinson, 2007), and exact solutions are only known when W is one or two dimensional (Gaines & Lyons, 1994). However, the approximation developed in Davie (2014) and further analysed using rough path theory by Flint & Lyons (2015) constitutes a simple and computable solution. Their approach is to generate

$$\widetilde{\mathbb{W}}_{s,t} := \frac{1}{2}W_{s,t} \otimes W_{s,t} + H_{s,t} \otimes W_{s,t} - W_{s,t} \otimes H_{s,t} + \lambda_{s,t},$$

where $\lambda_{s,t}$ is an anti-symmetric matrix with independent entries $\lambda_{s,t}^{i,j} \sim \mathcal{N}\left(0, \frac{1}{12}(t-s)^2\right), i < j$.

In both papers, the authors input $\widetilde{\mathbb{W}}$ into an SDE solver (the Milstein and log-ODE methods respectively) and prove that the resulting approximation achieves a 2-Wasserstein convergence rate beyond $O\big(1/\sqrt{N}\big)$, where N is the number of steps. We have follow-up work planned on this topic.

C Experimental details

C.1 General notes

Code Our code is available at `[redacted]`.

Software We used PyTorch (Paszke et al., 2019) as an autodifferentiable framework. We used the [redacted] library to solve SDEs. We used the Signatory library (Kidger & Lyons, 2020) to calculate the signatures used in the MMD metric. We used the `torchcde` library (Kidger, 2020) for its interpolation schemes, and to solve the neural CDEs used in the classification and prediction metrics. We used the `torchdiffeq` library (Chen, 2018) to solve the neural ODEs used in the classification and prediction metrics, and for the ODE components of the Latent ODE and CTFP models.

Architectures By using similar differential equation models, we were able to use essentially the same parameterisation for every model's vector fields. We used essentially the same hyperparameters for every dataset.

To recap, the neural SDE has generator initial condition ζ_θ, generator drift μ_θ, generator diffusion σ_θ, discriminator initial condition ξ_ϕ, discriminator drift f_ϕ, and discriminator diffusion g_ϕ. All of these are parameterised as neural networks.

Meanwhile Latent ODEs have an ODE-RNN encoder (with a neural network vector field) and a neural ODE decoder (with a neural network vector field). The CTFP has an ODE-RNN encoder

(with a neural network vector field) and a continuous normalising flow (Chen et al., 2018; Grathwohl et al., 2019) (with a neural network vector field) Additionally Deng et al. (2020) condition the normalising flow on the time evolution of a neural ODE of some latent state, which requires another neural network vector field.

In every case, the neural network was parameterised as a feedforward network with 2 hidden layers, width 64, and softplus activations. The drift, diffusion and vector fields, for every model, all additionally had a tanh nonlinearity as their final operation. As per Kidger et al. (2020b) we found that this improved the performance of every model.

The neural SDE's generator has hidden state of size x and the discriminator has hidden state is of size h. These were both taken as $x = h = 96$. Note that this is larger than the width of each hidden layer within the neural networks, so that the first operation within each neural network is a map from $\mathbb{R}^{96} \to \mathbb{R}^{64}$. Somewhat anecdotally, we found that taking the state to be larger than the hidden width was beneficial for model performance.[4]

The Latent ODE likewise has evolving hidden state, which was also taken to be of size 96.

The Latent ODE samples noise from a normally distributed initial condition, we took to have 40 dimensions. The CTFP samples noise from a Brownian motion, which as a continuous normalising flow has dimension equal to the number of dimensions of target distribution.

The neural SDE samples noise from both a normally distributed initial condition and a Brownian motion. We took the initial condition to have 40 dimensions. The number of dimensions of the Brownian motion was dataset dependent, see below.

The CTFP included a latent context vector as described in Deng et al. (2020). This was taken to have 40 dimensions.

These hyperparameters were selected based on informal initial experiments with all models.

SDE solvers The SDEs used the midpoint method, without adaptive stepping. Recall that the target time series data was regularly sampled and linearly interpolated to make a path. We took the SDE solver to take a single step between each output data point.

ODE solvers The ODEs of the Latent ODE and CTFP models were solved using the midpoint method, for consistency with the SDE solvers.

CDE solvers The CDEs of the classification and prediction models were solved by reducing to ODEs as in Kidger et al. (2020b) and then using the midpoint method, for consistency with the SDE solvers.

Optimisers The CTFP, Latent ODE, and the generator of the neural SDE were all trained with Adam (Kingma & Ba, 2015) with a learning rate of 4×10^{-5}. The discriminator of the neural SDE was trained with RMSprop with a learning rate of 4×10^{-5}. The learning rates were chosen by starting at 4×10^{-4} (arbitrarily) and reducing until good performance was achieved. (In particular seeking to avoid oscillatory behaviour in training of the neural SDE.)

Training Every model was trained for 100 epochs. The discriminator of the neural SDE received five training steps for every step with which the generator was trained, as is usual; the number of epochs given at 100 is for the generator, for a fair comparison to the other models.

Batch sizes were picked based on what was the largest possible batch size that GPU memory allowed for; these vary by problem and are given below.

Classifier and predictor The classifier was taken to be a neural CDE with hidden state of size 32, and whose vector field was parameterised as a feedforward neural network with 2 hidden layers of width 32, with softplus activations and final tanh activation.

[4]This has some loose theoretical justification: a signature is a linear differential equation with very large state, and it is a universal approximator. (See Kidger et al. (2020b, Appendix B) and references within – this is a classical fact within rough analysis.) That is to say, it is a simple vector field with a large state, rather than a complicated vector field with a small state.

The predictor was taken to be a neural CDE/neural ODE encoder/decoder pair. Both had a hidden state of size 32, and vector fields parameterised as feedforward neural network with 2 hidden layers of width 32, with softplus activations and final tanh activation. 32 dimensions were used at the encoder/decoder interface.

The learning rate used was 10^{-4} for both models, for every dataset and generative model considered, with the one exception of CTFP on Beijing Air Quality, where we observed divergent training of the classifier; the learning rate was reduced to 10^{-5} for this case only.

In all cases they were trained for 50 epochs using Adam, with early stopping if the model failed to improve its training loss over 20 epochs.

The classifier took an 80%/20% train/test split of the dataset given by combining the underlying dataset and model-generated samples of equal size.

C.2 STOCKS

Each sample is of length 100.

The batch size was 2048 for every model.

For the neural SDE, the discriminator received 1 epoch of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over every training epoch. The Brownian motion from which noise was sampled had 3 dimensions.

The prediction metric was based on using the first 80% of the input to predict the last 20%.

C.3 WEIGHTS

Each sample is of length 100.

The batch size was 4096 for the neural SDE and latent ODE. This was reduced to 1024 for the CTFP, which we found to be a very memory intensive model on this problem.

For the neural SDE, the discriminator received 10 epochs of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over every training epoch. The Brownian motion from which noise was sampled had 3 dimensions.

The prediction metric was based on using the first 80% of the input to predict the last 20%.

C.4 BEIJING AIR QUALITY

Each sample is of length 24.

The data was normalised to have zero mean and unit variance.

The batch size was 1024 for every model.

For the neural SDE, the discriminator received 10 epochs of training before the main training (of both generator and discriminator simultaneously) commenced. The weight averaging (over both generator and discriminator) was over the final 40 epochs of training. (We realised that this was an obvious improvement over averaging every epoch, as was done for the previous two experiments.) The Brownian motion from which noise was sampled had 10 dimensions.

The prediction metric was based on using the first 50% of the input to predict the last 50%. (An accidental change from the 80%/20% split used in the other experiments; this was kept as it is fair, as it is the same for all models on this dataset.)