The Neural Moving Average Model for Scalable Variational Inference of State Space Models

Thomas Ryder[1,2] **Dennis Prangle**[1] **Andrew Golightly**[1] **Isaac Matthews**[1]

[1]School of Mathematics Statistics and Physics, Newcastle University, Newcastle, United Kingdom
[2]Huawei Noah's Ark Lab

Abstract

Variational inference has had great success in scaling approximate Bayesian inference to big data by exploiting mini-batch training. To date, however, this strategy has been most applicable to models of independent data. We propose an extension to state space models of time series data based on a novel generative model for latent temporal states: the neural moving average model. This permits a subsequence to be sampled without drawing from the entire distribution, enabling training iterations to use mini-batches of the time series at low computational cost. We illustrate our method on autoregressive, Lotka-Volterra, FitzHugh-Nagumo and stochastic volatility models, achieving accurate parameter estimation in a short time.

1 INTRODUCTION

State space models (SSMs) are a flexible and interpretable model class for sequential data, popular in areas including engineering [Elliott et al., 2008], economics [Zeng and Wu, 2013], epidemiology [Fasiolo et al., 2016] and neuroscience [Paninski et al., 2010]. SSMs assume a latent Markov chain x with states $x_1, x_2, \ldots, x_T$, with data as noisy observations of some or all of these.

Standard inference methods for the parameters, θ, of an SSM require evaluating or estimating the likelihood under various choices of θ e.g. using a Kalman or particle filter [Särkkä, 2013]. Each such evaluation has $O(T)$ cost at best, and even larger costs may be required to control the variance of likelihood estimates. These methods can thus be impractically expensive for long time series (e.g. $T \gg 10^6$), which are increasingly common in applications such as genomics [Foti et al., 2014] and geoscience [Foreman-Mackey et al., 2017].

In contrast, for models of independent data, one can estimate the log-likelihood using a short *mini-batch*, at an $O(1)$ cost only. This allows scalable inference methods based on stochastic gradient optimisation e.g. maximum likelihood or variational inference. The latter introduces a family of approximate densities for the latent variables indexed by ϕ. One then selects ϕ to minimise the Kullback-Leibler divergence from the approximate density to the posterior.

We propose a mini-batch variational inference method for SSMs, for the case of continuous states i.e. $x_i \in \mathbb{R}^d$. This requires a family of variational approximations $q(\theta, x; \phi)$ with a crucial *locality* property. It must be possible to sample a subsequence $(x_i)_{a \leq i \leq b}$, to be used as a mini-batch, from the middle of the x sequence at a $O(1)$ cost. Much existing work on flexibly modelling sequence data (e.g. van den Oord et al. 2016, Ryder et al. 2018, Radev et al. 2020) uses an **autoregressive model** for x. Here x_i is generated from some or all x_js with $i < j$, so sampling x_a requires sampling $(x_i)_{i<a}$, and the locality property is not met.

To achieve the locality property we introduce the **neural moving average (nMA) model**. This is a generative model for sequence data in which a learnable convolutional neural network (CNN) processes (1) an underlying sequence of base $N(0, 1)$ variables and (2) the sequence of observed data. The CNN's receptive field has a limited size, rather than encompassing the entire input sequences. Therefore a sample from the nMA model is a type of moving average of the input sequences (1) and (2): x_i is produced from a window of values in the input sequences close to position i. This achieves the locality property. Also, by viewing the nMA model as a type of normalising flow [Rezende and Mohamed, 2015, Papamakarios et al., 2019], we show later that the mini-batch samples can be used to unbiasedly estimate the log-density of the whole x chain, which is crucial to implement variational inference.

A trade-off for producing the locality property is that, under a nMA model, x_i and x_j are independent for $|i - j|$ sufficiently large i.e. if they are far enough apart that their

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

CNN receptive fields do not overlap. Hence using a nMA as the variational approximation assumes no long-range dependence in the posterior for x. Despite this, we demonstrate that our approach works well in several examples. These include various challenging observation regimes: sparse observation times, partial observation of x_i, low observation variance and a large number of observations. Our flexible variational family produces good posterior estimates in these examples: at best our variational output is indistinguishable from the true posterior.

The remainder of our paper is as follows. Section 2 describes state space models. Section 3 reviews relevant material on normalising flows and presents the nMA model. Section 4 sets out our variational inference method. The nMA model and the resulting inference algorithm are our novel methodological contribution. Section 5 presents our experiments, and Section 6 gives conclusions and opportunities for future work. Code for the paper is available at `https://github.com/Tom-Ryder/VIforSSMs`.

Related Work Bayesian inference for SSMs commonly uses sampling-based Markov chain Monte Carlo (MCMC) methods, involving repeated use of Kalman or particle filters [Doucet et al., 2001, Cappé et al., 2010, Särkkä, 2013]. As discussed above, these methods typically become expensive for long time series, with each likelihood estimate requiring an $O(T)$ pass through the data. A recent $O(1)$ sampling-based scheme using a related strategy to ours for scalable SSM inference is Aicher et al. [2019]. This approach uses stochastic gradient MCMC with *buffered* gradient estimates, which are based on running a particle filter on a short subsequence of data. Like our contribution, this approach neglects long-range dependence.

Aicher et al. [2019], in common with several other papers discussed here, requires an observation for each x_i. However many applications involve missing or sparsely observed data. Our generative model can be applied to such settings as it learns to impute x_i values between observations.

Several stochastic optimisation variational inference methods for SSMs have previously been proposed, with different variational families for x, including: a multivariate normal distribution with tridiagonal covariance structure [Archer et al., 2016], a recurrent neural network [Krishnan et al., 2017], an autoregressive distribution [Karl et al., 2014, Ryder et al., 2018], a particle filter [Hirt and Dellaportas, 2019]. However, all of these methods have an $O(T)$ cost for each iteration of training and/or require storing $O(T)$ parameters.

Foti et al. [2014] also propose an $O(1)$ variational inference method based on mini-batch updates. They consider *hidden Markov models* – SSMs with discrete states – which allows the x posterior to be derived using a forward-backward algorithm [Rabiner, 1989]. This would usually require forward and backward passes over the full dataset, at cost $O(T)$, but they show that approximating these on short subsequences suffices to perform variational inference. In contrast our paper explores SSMs with continuous states, where a forward-backward algorithm is not available in general [Briers et al., 2010]. Another difference is that Foti et al. [2014] use a variational approximation with independence between θ and x, while our approach avoids this strong assumption.

Parallel Wavenet [van den Oord et al., 2018] similarly uses a normalising-flow-based generative model for sequence data. This incorporates long-range dependence using dilated convolutions, while we use only short range dependence to allow mini-batch inference. Our local normalising flow is also similar at a high level to a *masked convolutional generative flow* (MACOW) [Ma et al., 2019]. The novelty of our approach is that we develop this idea to allow fast variational inference for time series, while Ma et al. [2019] focus on density estimation and sampling for image data.

Finally, Ward et al. [2020] successfully apply our method to mechanistic models with Gaussian process priors placed on unobserved forcing terms, including a multi-output system using real-world data, and Gaussian process regression using a Poisson observation model.

2 STATE SPACE MODELS

Notation Throughout we use x_i to denote an individual state, x to denote the whole sequence of states and $x_{a:b}$ to denote a subsequence $(x_i)_{a\leq i\leq b}$. We use similar notation for sequences represented by other letters. More generally we use $a{:}b$ to represent the sequence $(a, a+1, \ldots, b)$.

2.1 DEFINITION

A SSM is based on a latent Markov chain $x = x_{1:T}$. We focus on the case of continuous states $x_i \in \mathbb{R}^d$. States evolve through a *transition density* $p(x_i|x_{i-1}, \theta)$, with parameters $\theta \in \mathbb{R}^p$. We assume the initial state is $x_0(\theta)$, a deterministic function of θ. (This allows examples with initial state known – x_0 is a constant – or unknown – x_0 depends on unknown parameters.) Observations $y_i \in \mathbb{R}^{d_y}$ are available for $i \in \mathcal{S} \subseteq 0{:}T$ following an *observation density* $p(y_i|x_i, \theta)$.

In the Bayesian framework, after specifying a prior density $p(\theta)$, interest lies in the posterior density

$$p(\theta, x|y) \propto p(\theta, x, y) = p(\theta) \prod_{i=1}^{T} p(x_i|x_{i-1}, \theta) \prod_{i\in\mathcal{S}} p(y_i|x_i, \theta). \tag{1}$$

2.2 DISCRETISED STOCHASTIC DIFFERENTIAL EQUATIONS

One application of SSMs, which we use in our examples, is as discrete approximations to stochastic differential equations (SDEs), as follows:

$$x_{i+1} = x_i + \alpha(x_i, \theta)\Delta t + \sqrt{\beta(x_i, \theta)\Delta t}\epsilon_i, \qquad (2)$$

where $\epsilon_i \sim N(0, I_d)$ are independent random vectors. Here α is a d-dimensional *drift vector*, β is a $d \times d$ positive-definite *diffusion matrix* and $\sqrt{\beta}$ denotes its Cholesky factor. The state x_i approximates the state of the SDE process at time $i\Delta t$. Taking the limit $\Delta t \to 0$ in an appropriate way recovers the exact SDE [Øksendal, 2003, Särkkä and Solin, 2019].

3 THE NEURAL MOVING AVERAGE MODEL

Section 1 gave an intuitive description of the nMA model. In this section we present a formal description. First Section 3.1 presents background material on inverse autoregressive flows (IAFs). Then Sections 3.2–3.3 describe the nMA model as a special case of an IAF. Section 3.2 describes the case where x_i (a state of the SSM) is scalar, and Section 3.3 extends this to the multivariate case.

3.1 INVERSE AUTOREGRESSIVE FLOWS

A normalising flow represents a random object x as $g_m \circ \ldots g_2 \circ g_1(z)$: a composition of learnable bijections of a base random object z. Here we suppose $x = x_{1:T}$ and $x_i \in \mathbb{R}$. (Later we consider x_i as a vector.) We take $z = z_{1:T}$ as independent $N(0, 1)$ variables. By the standard change of variable result, the log-density of x is

$$\log q(x) = \sum_{i=1}^{T} \varphi(z_i) - \sum_{j=1}^{m} \log|\det J_j| \qquad (3)$$

where φ is the $N(0, 1)$ log-density function and J_j is the Jacobian matrix of transformation g_j given input $g_{j-1} \circ \ldots g_2 \circ g_1(z)$.

The bijections in an IAF are mainly *affine layers*, which transform input z^{in} to output z^{out} by

$$z_i^{\text{out}} = \mu_i(z_{1:i-1}^{\text{in}}) + \sigma_i(z_{1:i-1}^{\text{in}})z_i^{\text{in}}, \qquad (4)$$

with $\sigma_i > 0$. This transformation scales and shifts each z_i^{in}. The shift and scale shift values, μ_i and σ_i, are typically neural network outputs. An efficient approach is to use a single neural network to output all the μ_i, σ_i values for a particular affine layer. This network uses *masked dense layers* so that (μ_i, σ_i) depends only on $z_{1:i-1}^{\text{in}}$ as required [Germain et al., 2015, Kingma et al., 2016, Papamakarios et al., 2017]. In the resulting IAF each affine layer is based on a different neural network of this form. We'll refer to this as a *masked IAF*.

The shift and scale functions for z_i^{out} in (4) have an *autoregressive property*: they depend on z^{in} only through z_j^{in} with $j < i$. Hence the Jacobian matrix of the transformation is diagonal with non-zero entries $\sigma_{1:T}$. The log-density of an IAF made of m affine layers is

$$\log q(x) = \sum_{i=1}^{T} \varphi(z_i) - \sum_{j=1}^{m}\sum_{i=1}^{T} \log \sigma_i^j \qquad (5)$$

where σ_i^j is the shift value for the ith input to the jth affine layer.

IAFs typically alternate between affine layers and *permutation layers*, using order reversing or random permutations. Such layers have Jacobians with absolute determinant 1. Thus the log-density calculation is unchanged (interpreting j in (5) to index the jth *affine layer* not the jth layer of any type). The supplement (Section G.1) details methods to restrict the output of a IAF e.g. to ensure all x_is are positive.

IAFs are flexible and, for small T, allow fast sampling and calculation of a sample's log-density. However they are expensive for large T as large neural networks are needed to map between length T sequences.

3.2 THE NEURAL MOVING AVERAGE MODEL

Our neural moving average (nMA) model reduces the number of weights that IAFs require by using a CNN to calculate the μ_i and σ_i values in an affine layer. Thus it can be thought of as a kind of local IAF. Here we explain the main idea by presenting a version for scalar x_i. Section 3.3 extends this to the vector x_i case.

To define the nMA model we describe how a single affine layer produces its shift and scale values. The affine layer uses a CNN with input z^{in}, a vector of length T. Let h^k represent the kth hidden layer of the CNN, a matrix of dimension (T, n_k) where n_k is a tuning choice. The first layer applies a convolution with receptive field length ℓ. This is an *off-centre convolution* so that row i of h^1 is a transformation of $z_{i-\ell:i-1}^{\text{in}}$. We use zero-padding by taking $z_i^{\text{in}} = 0$ for $i < 0$. The following hidden layers are length-1 convolutions, so row i of h^{k+1} is a transformation of row i of h^k. The output, h^n, is a matrix of dimension $(T, 2)$ whose ith row contains μ_i and σ_i. The final layer applies a softplus activation to produce the σ_i values, ensuring they are positive. An identity activation is used to produce the μ_i values. The μ_i and σ_i values are used in (4) to produce the output of the affine layer.

A nMA model composes several affine layers of the form just described. Some properties of the distribution for the output sequence x are:

1. No long-range dependence: x_i and x_j are independent if $|i-j| > m\ell$, where m is the number of affine layers.
2. Stationary local dependence: the distributions of $x_{i:j}$ and $x_{i+a:j+a}$ are the same for most choices of a. (Subsequences near to the start of x can differ due to zero-padding.)

To improve the flexibility of the nMA model, affine layers can be alternated with order-reversing permutations. (Random permutations would not be suitable, as they would disrupt our ability to sample subsequences quickly, as described in Section 3.4.) Throughout the paper we consider nMA models *without* order reversing permutation layers, as we found these models already sufficiently flexible for our examples. (The supplement, Section C, details how to incorporate these layers.)

We relax stationary local dependence by injecting *local side information* to the CNN i.e. giving an extra feature vector s_i as input for each position i in the first CNN layer. We also use *global* side information to allow x to depend on the parameter values θ i.e. giving θ as extra input for every position i. See the supplement (Sections D, E) for details of the side information we use in practice.

3.3 MULTIVARIATE CASE

Here we generalise the nMA model to the case where $x_i \in \mathbb{R}^d$. We now let z be a sequence $z_1, z_2, \ldots, z_T$ of independent random $N(0, I_d)$ vectors. A nMA affine layer makes the transformation

$$z_i^{\text{out}} = \mu_i + \sigma_i \odot z_i^{\text{in}}, \tag{6}$$

scaling the vector z_i^{in} (elementwise multiplication by vector σ_i) then shifting it (adding vector μ_i).

In the scalar case it was important to allow complex dependencies *between* z_i^{out} values. Now we must also allow dependencies *within* each z_i^{out} vector. To do so we use *coupling layers* as in Dinh et al. [2016].

We use an extra k subscript to denote the kth component of a vector e.g. z_{ik}^{in}. We select some $a \approx d/2$. For $k \leq a$, we take $\mu_{ik} = 0$ and $\sigma_{ik} = 1$, so that $z_{ik}^{\text{out}} = z_{ik}^{\text{in}}$. For $k > a$, we compute μ_{ik} and σ_{ik} using a CNN, modifying the scalar case as follows. Now row i of h^1 is a transformation of $z_{i-\ell:i-1}^{\text{in}}$ (the ℓ vectors preceding z_i^{in}), and also z_{ik}^{in} for $k \leq a$ (the part of z_i^{in} not being modified). The output h^n is now a tensor of dimension $(T, d-a, 2)$ containing μ_{ik} and σ_{ik} values for $k > a$.

This affine layer does not transform the first a components of z_i^{in}. To allow different components to be transformed in each layer, we permute components between affine layers. For example, a $d = 2$ permutation layer transforms z^{in} to z^{out} by $z_{i1}^{\text{out}} = z_{i2}^{\text{in}}$, $z_{i2}^{\text{out}} = z_{i1}^{\text{in}}$. The log-density is now

$$\log q(x) = \sum_{i=1}^{T} \lambda_i, \quad \lambda_i = \varphi(z_i) - \sum_{k=1}^{d}\sum_{j=1}^{m} \log \sigma_{ik}^j, \tag{7}$$

where φ is the $N(0, I_d)$ log-density function and σ_{ik}^j is the kth entry of the shift vector for position i output by the jth affine layer. Decomposing $\log q(x)$ into λ_i contributions will be useful in Section 4.

3.4 SAMPLING

Sampling from a nMA model is straightforward. First sample the base random object z. This is a sequence of length T (of scalars or vectors – the sampling process is similar in either case). Now apply the IAF's layers to this in turn. To apply an affine layer, pass the input (and any side information) through the layer's CNN to calculate shift and scale values, then apply the affine transformation. The final output is the sampled sequence x. The cost of sampling in this way is $O(T)$.

In the next section, we will often wish to sample a short subsequence $x_{u:v}$. It is possible to do this at $O(1)$ cost with respect to T. Algorithm A in the supplement gives the details. In brief, the key insight is that $x_{u:v}$ only depends on z through $z_{u-m\ell:v}$. Therefore we sample $z_{u-m\ell:v}$ and apply the layers to this subsequence. The output will contain the correct values of $x_{u:v}$.

4 VARIATIONAL INFERENCE FOR SSMS

This section describes how we use nMA models to perform variational inference (VI) efficiently for SSMs. Section 4.1 reviews standard details of VI. See e.g. Blei et al. [2017] for more details. We then present our novel VI derivation involving nMA models in Section 4.2 and the resulting algorithm in Section 4.3.

4.1 VARIATIONAL INFERENCE BACKGROUND

We wish to infer the joint posterior density $p(\theta, x|y)$. We introduce a family of approximations indexed by ϕ, $q(\theta, x; \phi)$. Optimisation is used to find ϕ minimising the Kullback-Leibler divergence $KL[q(\theta, x; \phi)||p(\theta, x|y)]$. This is equivalent to maximising the ELBO (evidence lower bound) [Jordan et al., 1999],

$$\mathcal{L}(\phi) = E_{\theta,x\sim q}[r(\theta, x, y, \phi)], \tag{8}$$

$$\text{for} \quad r(\theta, x, y, \phi) = \log p(\theta, x, y) - \log q(\theta, x; \phi). \tag{9}$$

Here r is a log-density ratio. The optimal $q(\theta, x; \phi)$ approximates the posterior density. It is typically overconcentrated,

unless the approximating family is expressive enough to allow particularly close matches to the posterior.

Optimisation for VI can be performed efficiently using the *reparameterisation trick* [Kingma and Welling, 2014, Rezende et al., 2014, Titsias and Lázaro-Gredilla, 2014]. That is, letting (θ, x) be the output of an invertible deterministic function $g(\varepsilon, \phi)$ for some random variable ε with a fixed distribution. Then the ELBO gradient and unbiased Monte Carlo estimate are

$$\nabla \mathcal{L}(\phi) = E_\varepsilon[\nabla r(\theta, x, y, \phi)], \tag{10}$$

$$\widehat{\nabla \mathcal{L}}(\phi) = \frac{1}{n}\sum_{j=1}^{n}[\nabla r(\theta^{(j)}, x^{(j)}, y, \phi)], \tag{11}$$

where $(\theta^{(j)}, x^{(j)}) = g(\varepsilon^{(j)}, \phi)$ and $\varepsilon^{(1)}, \ldots, \varepsilon^{(n)}$ are independent ε samples. This gradient estimate can be used in stochastic gradient optimisation algorithms.

4.2 ELBO DERIVATION

Our variational family for the SSM posterior (1) is

$$q(\theta, x; \phi) = q(\theta; \phi_\theta) q(x|\theta; \phi_x), \tag{12}$$

where $\phi = (\phi_\theta, \phi_x)$. We use a masked IAF for $q(\theta; \phi_\theta)$ and a nMA model for $q(x|\theta; \phi_x)$. For the latter we inject θ as side information. See the supplement (Sections D and E) for more details.

The masked IAF maps a base random vector z_θ to θ using parameters ϕ_θ, as described in Section 3.1. The nMA model maps θ and a sequence of vectors z_x to x using parameters ϕ_x as described in Section 3.3. Hence we have a mapping from $\varepsilon = (z_\theta, z_x)$ to $g(\varepsilon, \phi) = (\theta, x)$, allowing us to use the reparameterisation trick below.

This section derives a mini-batch optimisation algorithm to train ϕ based on sampling short x subsequences, so that the cost-per-training-iteration is $O(1)$. The algorithm is applicable for scalar or multivariate x_i. In this presentation we assume that $\mathcal{S} = 0{:}T$ i.e. there are observations for all i values. To relax this assumption remove any terms involving y_i for $i \notin \mathcal{S}$.

For our variational family (12), the ELBO is (8) with

$$r = \log p(\theta, x, y) - \log q(\theta; \phi_\theta) - \log q(x|\theta; \phi_x). \tag{13}$$

Substituting (1) and (7) into (13) gives

$$\begin{aligned} r = &\log p(\theta) - \log q(\theta; \phi_\theta) + \log p(y_0|x_0, \theta) + \\ &\sum_{i=1}^{T} \big\{ \log p(x_i|x_{i-1}, \theta) + \log p(y_i|x_i, \theta) - \lambda_i \big\}. \end{aligned} \tag{14}$$

Now introduce batches $B_1, B_2, \ldots, B_b$: length M sequences of consecutive integers partitioning $1{:}T$. Draw κ uniformly from $1{:}b$. Then an unbiased estimate of r is

$$\begin{aligned} r_\kappa = &\log p(\theta) - \log q(\theta; \phi_\theta) + \log p(y_0|x_0, \theta) + \\ &\frac{T}{M}\sum_{i \in B_\kappa} \big\{ \log p(x_i|x_{i-1}, \theta) + \log p(y_i|x_i, \theta) - \lambda_i \big\}. \end{aligned} \tag{15}$$

Hence an unbiased estimate of the ELBO gradient is

$$\widehat{\nabla \mathcal{L}}(\phi) = \frac{1}{n}\sum_{j=1}^{n} \nabla r_\kappa(\theta^{(j)}, x^{(j)}, y, \phi). \tag{16}$$

where $(\theta^{(j)}, x^{(j)}) = g(\varepsilon^{(j)}, \phi)$ and $\varepsilon^{(1)}, \ldots, \varepsilon^{(n)}$ are independent ε samples.

4.3 OPTIMISATION ALGORITHM

Algorithm 1 presents our mini-batch training procedure. Each iteration of Algorithm 1 involves sampling a subsequence of x values of length $M + 1$. The cost is $O(1)$ with respect to the total length of the sequence T. This compares favourably to the $O(T)$ cost of sampling the entire x sequence. Discussion of implementation details is given in the supplement (Section E).

Algorithm 1 Mini-batch variational inference for state space models

1: Initialise ϕ_θ, ϕ_x.
2: **loop**
3: Sample a batch κ uniformly from $1{:}b$. Let u and v denote the endpoints of B_κ.
4: Calculate $\widehat{\nabla \mathcal{L}}(\phi)$ from (16), generating the terms in the sum as follows.
5: **for** $1 \le j \le n$ **do**
6: Sample $\theta^{(j)} \sim q(\theta; \phi_\theta)$.
7: Sample[1] $x^{(j)}_{u-1:v}$ from $q(x|\theta; \phi_x)$ (unless $u = 1$ in which case sample $x_{u:v}$), calculating corresponding $\lambda_{u:v}$ values. See Algorithm A in the supplement for details.
8: Calculate $\nabla r_\kappa(\theta^{(j)}, x^{(j)}, y, \phi)$ using automatic differentiation of (15).
9: **end for**
10: Update ϕ_θ, ϕ_x using stochastic gradient optimisation.
11: **end loop**

5 EXPERIMENTS

Below we apply our method to several examples. All results were obtained using an NVIDIA Titan XP and an 8 core

[1] Note this samples x_{u-1}, the state immediately *before* the current batch of interest. This is needed for the $p(x_i|x_{i-1}, \theta)$ term in (15) when $i = u$.

CPU. For tuning choices and experimental specifics see the supplement (Sections E, F). Sections 5.1–5.3 use simulated data so the results can be compared to true parameter values, while Section 5.4 uses real data.

5.1 AR(1) MODEL

First we consider the AR(1) model $x_{i+1} = \theta_1 + \theta_2 x_i + \theta_3 \epsilon$, with $\epsilon \sim N(0,1)$ and $x_0 = 10$. We assume observations $y_i \sim N(x_i, 1)$ for $i \in 0{:}T$, and independent $N(0, 10^2)$ priors on $\theta_1, \theta_2, \log\theta_3$. We use this model to investigate how our method scales with larger T, and the effect of receptive field length ℓ. To judge the accuracy of our results we compare to near-exact posterior inference using MCMC, as described in the supplement (Section A).

Effect of Observation Sequence Length We simulated a synthetic dataset for each of four T values: $5000, 10000, 50000, 100000$ under true parameter values $\theta = (5.0, 0.5, 3.0)$. We then inferred θ, fixing the hyperparameters so that the cost per iteration for each setting is constant.

Figure 1a plots the accuracy of our results against number of iterations performed. Accuracy is measured as Maximum Mean Discrepancy (MMD) [Gretton et al., 2012] between variational approximation and MCMC output. (We use MMD with a Gaussian kernel.) In all cases, variational inference approximates the posterior well. Also, the number of training iterations required remains similar as T increases. As a further check on the quality of the posterior approximation, Figure 1b shows a good match between marginal posteriors for MCMC and variational output for the case $T = 5000$. Here, as for other T values, the 10,000th iteration is achieved after ~ 3 minutes of computation. In comparison, the cost per iteration of MCMC is roughly proportional to T.

Effect of Receptive Field Length We consider again the $T = 5000$ dataset, and investigate the effect of ℓ. Figure 1 shows MMD against iteration and wall-clock time for $\ell \in \{5, 10, 50, 100, 200\}$. In all cases the variational output converges to a good approximation of the posterior. Convergence takes a similar number of iterations for all choices of ℓ, but wall-clock time per iteration increases with ℓ.

5.2 LOTKA-VOLTERRA

Next we test our method on short time series with complex dynamics. We use a version of the Lotka-Volterra model for predator-prey population dynamics under three events: prey reproduction, predation (in which prey are consumed and predators get resources to reproduce) and predator death. A SDE Lotka-Volterra model (for derivation see e.g. Golightly and Wilkinson [2011]) is defined by drift and diffusion

$$\alpha(x,\theta) = \begin{pmatrix} \theta_1 u - \theta_2 uv \\ \theta_2 uv - \theta_3 v \end{pmatrix}, \tag{17}$$

$$\beta(x,\theta) = \begin{pmatrix} \theta_1 u + \theta_2 uv & -\theta_2 uv \\ -\theta_2 uv & \theta_2 uv + \theta_3 v \end{pmatrix}, \tag{18}$$

where $x = (u, v)$ represents population sizes of prey and predators. The parameters $\theta = (\theta_1, \theta_2, \theta_3)$ control the rates of the three events described above.

We consider a discretised version of this SDE, as described in Section 2.2, with $\Delta t = 0.1$ and $u_0 = v_0 = 100$. We simulated realisations under parameters $\theta = (0.5, 0.0025, 0.3)$ of x_i for $i \in 1{:}500$, and construct two datasets with observations at: (a) $i = 0, 10, 20, \ldots, 500$ (dense); (b) $i = 0, 100, 200, \ldots, 500$ (sparse). We assume noisy observations $y_i \sim N(x_i, I_2)$ and independent $N(0, 10^2)$ priors for $\log\theta_1, \log\theta_2, \log\theta_3$.

Unlike previous examples, we needed to restrict u_i and v_i to be positive. Also, we found multiple posterior modes, and needed to pretrain carefully to control which we converged to. See Section G of the supplement for more details of both these issues.

For observation setting (a), we compared our results to near-exact posterior samples from MCMC [Golightly and Wilkinson, 2008, Fuchs, 2013]. These papers use a Metropolis-within-Gibbs MCMC scheme with carefully chosen proposal constructs. Designing suitable proposals can be challenging, particularly in sparse observation regimes [Whitaker et al., 2017]. Consequently we were unable to use MCMC in setting (b).

Figure 2 (left plot) displays the visual similarity between marginal densities estimates from variational and MCMC output in setting (a). The VI output is taken from the 30,000th iteration, after ≈ 10 minutes of computation. Figure 2 (right plots) shows variational output for θ and x in setting (b) after ≈ 20 minutes of computation. These results are consistent with the ground truth parameter values and x path. VI using an autoregressive distribution for x has also performed well in a similar scenario [Ryder et al., 2018], but required more training time (roughly 2 hours).

5.3 FITZHUGH-NAGUMO

Here we test our method on a long time series with an unobserved component, using the FitzHugh-Nagumo model. A SDE version, following Jensen et al. [2012], van der Meulen and Schauer [2017], is defined by drift and diffusion

$$\alpha(x,\theta) = \begin{pmatrix} \theta_1 \left(-v^3 + v - w + \theta_2\right) \\ \theta_3 v - w + 1.4 \end{pmatrix}, \tag{19}$$

$$\beta(x,\theta) = \begin{pmatrix} \theta_4 & 0 \\ 0 & \theta_5 \end{pmatrix}, \tag{20}$$

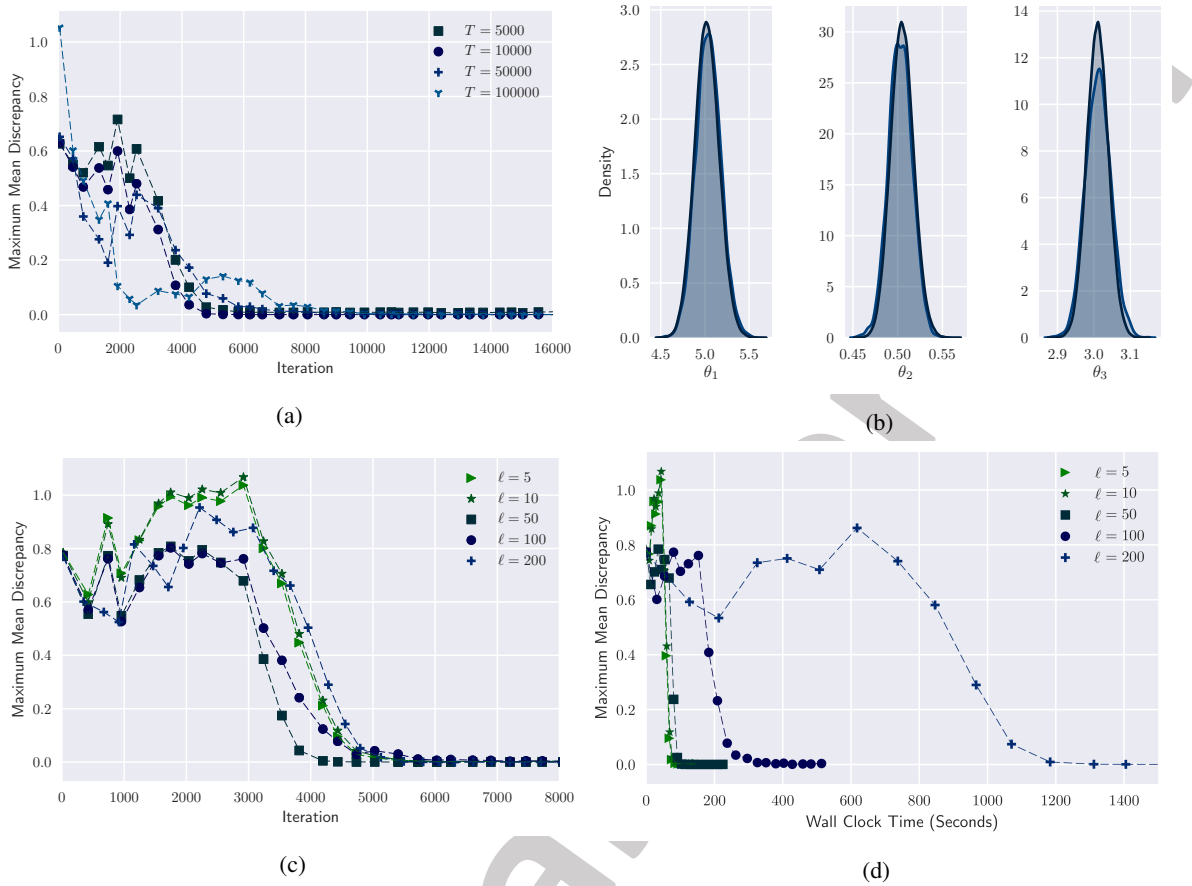

Figure 1: AR(1) results. (a) MMD between variational and MCMC output for θ. (b) Marginal posterior density plots of MCMC output (blue) and variational output after 10,000 iterations (black). (c,d) MMD between variational and MCMC output for θ for a range of receptive field lengths ℓ. The horizontal axis shows (c) number of training iterations (d) wall-clock training time.

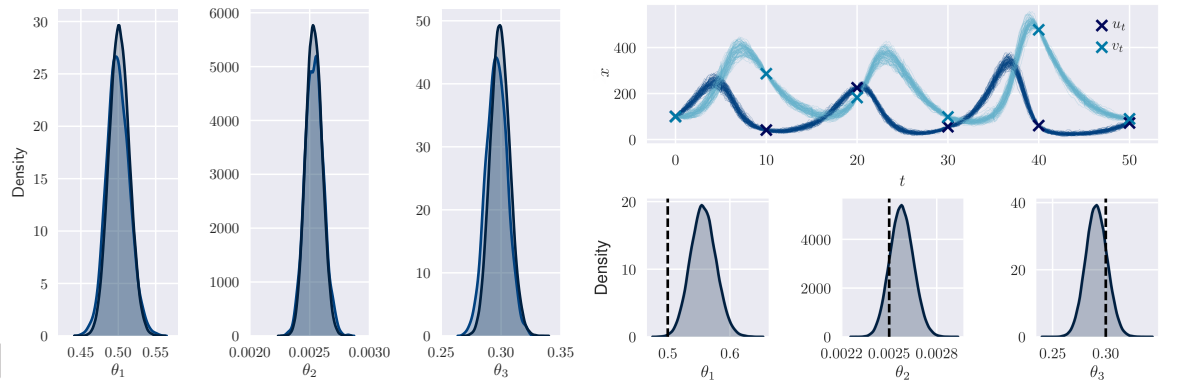

Figure 2: Lotka-Volterra results. Left: marginal density plots for setting (a) (dense observations), comparing MCMC (blue) to variational (black) output. Right: variational output for setting (b) (sparse observations). Right top: 100 x samples, with observations displayed as crosses. The horizontal axis shows $t = 0.1i$. Right bottom: marginal density plots for θ with true values displayed with dashed black lines. All variational results used 30,000 training iterations.

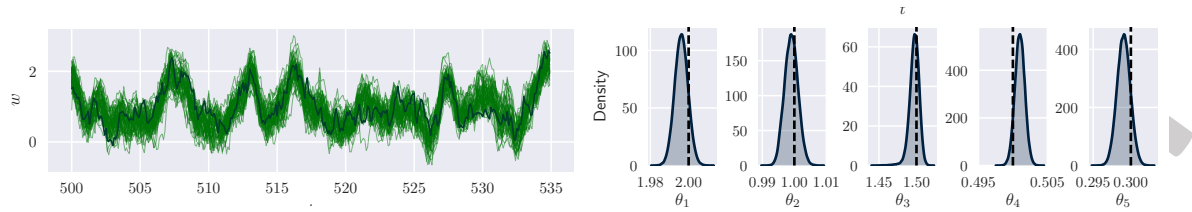

Figure 3: Fitzhugh-Nagumo results. Left: 100 variational posterior samples (light green) and true values (dark green) for unobserved coordinate w over a short time range. The x-axis shows $t = 0.1i$. Right: marginal density plots of variational output for θ and true values (dashed black lines).

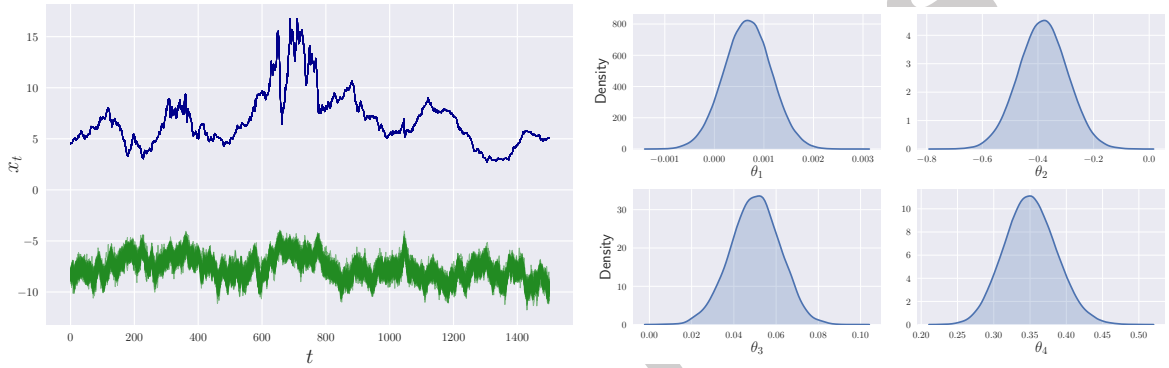

Figure 4: Stochastic volatility results. Left: the observed returns process (blue line) and 50 samples from the variational posterior for the latent volatility path (green lines). Right: Marginal density plots of the variational posterior for θ.

where $x = (v, w)$ represents the current membrane potential and latent recovery variables.

We consider a discretised version of this SDE, as in Section 2.2, with $\Delta t = 0.1, v_0 = 2, w_0 = 3$. We simulate synthetic data under parameter values $\theta = (2.0, 1.0, 1.5, 0.5, 0.3)$ up to $T = 1,000,000$, recording observations at every i to mimic a high frequency observation scenario. We assume independent observations $y_i \sim N(v_i, 0.1^2)$ and independent $N(0, 10^2)$ priors for $\log\theta_1, \theta_2, \theta_3, \log\theta_4, \log\theta_5$.

Figure 3 displays estimates of the unobserved component w, and marginal density estimates for θ. The results are consistent with the ground truth parameter values and w path. The approximate posterior is sampled after roughly 180 minutes of training.

5.4 LOG-GAUSSIAN STOCHASTIC VOLATILITY

We analyse a real data under a log-Gaussian stochastic volatility model presented as a discretised SDE with drift and diffusion

$$\alpha(x, \theta) = \begin{pmatrix} \theta_1 r \\ \theta_2 - \theta_3 z \end{pmatrix}, \quad \beta(x, \theta) = \begin{pmatrix} re^z & 0 \\ 0 & \theta_4^2 \end{pmatrix}, \quad (21)$$

where $x = (r, z)$ is the returns process and latent volatility factor, respectively. Similar discrete-time models have been analysed by Andersen and Lund [1997], Eraker [2001], Durham and Gallant [2002], but we use the form presented in Golightly and Wilkinson [2006].

Similarly to Golightly and Wilkinson [2006], we use 1508 weekly observations on the three-month U.S. Treasury bill rate for August 13, 1967 – August 30, 1996, and perform inference under independent $N(0, 10^2)$ priors for $\theta_1, \theta_2, \log\theta_3, \log\theta_4$ and z_0. We assume the returns process is fully observed without error and set $\Delta t = 1.0$. Our analysis took 20 minutes on a single GPU. Figure 4 shows the results, which are consistent with those obtained from the MCMC analysis in Golightly and Wilkinson [2006].

6 CONCLUSION

We present a variational inference method for state space models based on a neural moving average model. This is designed to model complex dependence in the conditional posterior $p(x|\theta, y)$ and be scalable to long time series. In particular, they allow mini-batch inference where each training iteration has $O(1)$ cost. We show that our method works well in several applications with challenging features including: an unobserved state component, sparse observation times, and a large number of observations. Further applications can be found in Ward et al. [2020].

Future work could investigate changing several aspects of the flow: alternating affine transformations with order revers-

ing permutations; allowing some long-range dependence using a multi-scale architecture; incorporating recently proposed ideas from the literature [Durkan et al., 2019].

Author Contributions

T. Ryder and D. Prangle conceived the idea and wrote the paper. T. Ryder and I. Matthews wrote the code and performed the experiments. A. Golightly advised on SSM models and methods, and wrote Supplementary Section A.

Acknowledgements

Thomas Ryder and Isaac Matthews are supported by the Engineering and Physical Sciences Research Council, Centre for Doctoral Training in Cloud Computing for Big Data (grant number EP/L015358/1).

We acknowledge with thanks an NVIDIA academic GPU grant for this project.

We thank Wil Ward and Stephen McGough for helpful discussions, and anonymous reviewers for useful comments.

References

Christopher Aicher, Yi-An Ma, Nicholas J. Foti, and Emily B Fox. Stochastic gradient MCMC for state space models. *SIAM Journal on Mathematics of Data Science*, 1(3):555–587, 2019.

Torben G. Andersen and Jesper Lund. Estimating continuous-time stochastic volatility models of the short-term interest rate. *Journal of Econometrics*, 77(2), 1997.

Evan. Archer, Il M. Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. In *International Conference on Learning Representations, Workshops*, 2016.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112:859–877, 2017.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61, 2010.

Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer, 2010.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Arnaud Doucet, Nando de Freitas, and Neil Gordon. *An Introduction to Sequential Monte Carlo Methods*. Springer New York, 2001.

Garland B Durham and A. Ronald Gallant. Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–338, 2002.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, 2019.

Robert J Elliott, Lakhdar Aggoun, and John B Moore. *Hidden Markov models: estimation and control*. Springer, 2008.

Bjørn Eraker. MCMC analysis of diffusion models with application to finance. *Journal of Business & Economic Statistics*, 19(2):177–191, 2001.

Matteo Fasiolo, Natalya Pya, and Simon N. Wood. A comparison of inferential methods for highly nonlinear state space models in ecology and epidemiology. *Statistical Science*, 31:96–118, 2016.

Daniel Foreman-Mackey, Eric Agol, Sivaram Ambikasaran, and Ruth Angus. Fast and scalable Gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, 154(6):220, 2017.

Nick Foti, Jason Xu, Dillon Laird, and Emily Fox. Stochastic variational inference for hidden Markov models. In *Advances in Neural Information Processing Systems*, 2014.

Christiane Fuchs. *Inference for Diffusion Processes: With Applications in Life Sciences*. Springer, 2013.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, 2015.

Andrew Golightly and Darren J. Wilkinson. Bayesian sequential inference for nonlinear multivariate diffusions. *Statistics and Computing*, 16(4), 2006.

Andrew Golightly and Darren J. Wilkinson. Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*, 52: 1674–1693, 2008.

Andrew Golightly and Darren J. Wilkinson. Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. *Interface focus*, 1:807–820, 2011.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

Marcel Hirt and Petros Dellaportas. Scalable Bayesian learning for state space models using variational inference with SMC samplers. In *Artificial Intelligence and Statistics*, 2019.

Anders Chr. Jensen, Susanne Ditlevsen, Mathieu Kessler, and Omiros Papaspiliopoulos. Markov chain Monte Carlo approach to parameter estimation in the FitzHugh-Nagumo model. *Physical Review E*, 86, 2012.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37: 183–233, 1999.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational Bayes filters: unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2014.

Durk P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Durk P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

Rahul Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Xuezhe Ma, Xiang Kong, Shanghang Zhang, and Eduard Hovy. Macow: Masked convolutional generative flow. In *Advances in Neural Information Processing Systems*, 2019.

Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext Universitext. Springer, 2003.

Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of computational neuroscience*, 29:107–126, 2010.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 2017.

George Papamakarios, Eric Nalisnick, Danilo J. Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Stefan T. Radev, Ulf K. Mertens, Andreass Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *arXiv preprint arXiv:2003.06281*, 2020.

Danilo J. Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Tom Ryder, Andrew Golightly, A. Stephen McGough, and Dennis Prangle. Black-box variational inference for stochastic differential equations. In *International Conference on Machine Learning*, 2018.

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

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

Michalis Titsias and Michalis K. Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, 2016.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, 2018.

Frank van der Meulen and Moritz Schauer. Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals. *Electronic Journal of Statistics*, 11:2358–2396, 2017.

Wil Ward, Tom Ryder, Dennis Prangle, and Mauricio Alvarez. Black-box inference for non-linear latent force models. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

Gavin A. Whitaker, Andrew Golightly, Richard J. Boys, and Chris Sherlock. Improved bridge constructs for stochastic differential equations. *Statistics and Computing*, 27:885–900, 2017.

Yong Zeng and Shu Wu. *State-space models: Applications in economics and finance*. Springer, 2013.