Efficient Learning of the Parameters of Non-Linear Models using Differentiable Resampling in Particle Filters

Conor Rosato Lee Devlin Vincent Beraud Paul Horridge

Thomas B. Schön Simon Maskell

June 2022

Abstract

It has been widely documented that the sampling and resampling steps in particle filters cannot be differentiated. The *reparameterisation trick* was introduced to allow the sampling step to be reformulated into a differentiable function. We extend the *reparameterisation trick* to include the stochastic input to resampling therefore limiting the discontinuities in the gradient calculation after this step. Knowing the gradients of the prior and likelihood allows us to run particle Markov Chain Monte Carlo (p-MCMC) and use the No-U-Turn Sampler (NUTS) as the proposal when estimating parameters.

We compare the Metropolis-adjusted Langevin algorithm (MALA), Hamiltonian Monte Carlo with different number of steps and NUTS. We consider three state-space models and show that NUTS improves the mixing of the Markov chain and can produce more accurate results in less computational time.

1 Introduction

State-Space Space Models (SSMs) have been used to model dynamical systems in a wide range of research fields (see [1] for numerous examples). SSMs are represented by two stochastic processes: $\{X_t\}_{t\geq 0}$ and $\{Y_t\}_{t\geq 0}$ where X_t indicates the hidden state which evolves according to a Markov process $p\left(x_t|x_{t-1},\theta\right)$ and Y_t is the observation (both at time $t \geq 0$), such that

$$X_t|X_{t-1} \sim \; p\left(x_t|x_{t-1},\theta\right), \tag{1}$$

$$Y_t|X_t \sim \; p\left(y_t|x_t,\theta\right). \tag{2}$$

The initial latent state X_0 has initial density denoted $\mu_\theta(x_0)$. The SSM is parameterised by an unknown static parameter θ contained in the parameter space Θ. The transition and observation

[*] C. Rosato, L. Devlin, V. Beraud, P. Horridge and S. Maskell are with the Department of Electrical Engineering and Electronics, University of Liverpool, United Kingdom, e-mail: e-mail: ({c.m.rosato}, {ljdevlin}, {vincent.beraud}, {p.horridge}, {smaskell}@liverpool.ac.uk). Thomas B. Schön is with Uppsala University, Sweden, e-mail: (thomas.schon@it.uu.se)

densities are given by (1) and (2), respectively. In this paper we focus on Bayesian parameter estimation in SSMs using Particle Markov Chain Monte Carlo (p-MCMC), as first proposed in [2]. This approach combines two Monte Carlo methods that use repeated sampling techniques to obtain numerical estimates of a target distribution $\pi(\theta)$, for which exact inference is intractable. The two methods are Markov Chain Monte Carlo (MCMC), as described in [3–5], and Sequential Monte Carlo (SMC) i.e. a particle filter, as described in [6–8].

MCMC algorithms such as Metropolis-Hastings (M-H) often use random walk sampling within the proposal. Such proposals can struggle to enable the MCMC to reach the stationary distribution when estimating large numbers of parameters. A related issue can occur with Gibbs samplers when the correlation between parameters is high. These issues can result in the sampler getting stuck in local maxima within $\pi(\theta)$. Hamiltonian Monte Carlo (HMC), as described in [9], is an approach that simulates from a problem-specific Hamiltonian system to generate the samples used in the MCMC. HMC has been seen to be effective when estimating parameters in models when the target distribution is complex or multi-modal but is sensitive to hyperparameters which have to be determined by the user. An adaptive version of HMC called the No-U-Turn Sampler (NUTS) [10] automates the selection of these hyperparameters. Probabilistic programming languages (ppls) such as Stan [11] and PyMC3 [12] are tools that have been developed to allow users to define and make inferences about probabilistic models using NUTS.

Particle filters have been used in many areas of research, such as finance [13], disease modelling [14] and multiple target tracking [15] to infer time-dependent hidden states. The original contribution of [2] uses a particle filter to calculate an unbiased estimate of the often intractable likelihood for θ. A M-H algorithm with a random walk proposal was used to sample from $\pi(\theta)$. Using such a proposal in p-MCMC will inherit the same issues as described above in the context of MCMC generically. To make use of more sophisticated proposals the gradient of the log-posterior of the parameter, θ, needs to be estimated.

Extensions of the original p-MCMC algorithm have focused on including gradient information when proposing new parameters. Reference [16] shows how to estimate the score (gradient) of the log-likelihood and the observed information matrices at θ in SSMs using particle filter methods. The two methods proposed run with computational complexity $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$, respectively. The first has a linear computational cost but the performance decreases over time. The second has a computational cost that increases quadratically with the number of particles N but performance does not deteriorate over time, with [17] theoretically substantiating this claim. [18] built on this work to compute these terms with computational complexity $\mathcal{O}(N)$ and avoids the quadratically increasing variance caused by particle degeneracy. In [19–22] the authors utilise the previous work of [18] to recursively estimate the score (gradient) of the log-likelihood at θ. References [19] and [20] include Langevin Monte Carlo (LMC) methods seen in [23] whilst [21] and [22] include first- and second-order Hessian information about the posterior in the proposal. Use of the Hessian is shown to improve the mixing of the Markov chain at the stationary phase and decrease the length of burn-in. However, calculating a $d \times d$ matrix of the second-order partial derivatives can become infeasible when the dimensionality, d, becomes large. While [21], [22] do mention using HMC dynamics within p-MCMC, to the best of the authors' knowledge, no implementation of this approach has been described in the literature up to now.

We aim to complement the recent literature seen in machine learning that addresses the problem of differentiating resampling (see section 6)). In order to obtain the gradient of the log-likelihood w.r.t θ, the particle filter needs to differentiated. However, it has been noted in [24–26] that the stochastic nature of both the sampling and resampling steps, that are inherently part of the particle filter, are not differentiable. As will be explained in more detail later in this paper (in Sections 3 and 4.1), the *reparameterisation trick* was proposed in [27] to reformulate the sampling operation into a differentiable function by sampling a noise vector in advance and defining the likelihood for θ as being a deterministic function of this sampled noise vector. However, resampling remains problematic since after resampling all weights are equal. More specifically, the gradients cannot be calculated since the new particles' states are not differentiable w.r.t. the weights that are input to resampling. Recent work in machine learning has focused on how to modify the resampling step to make it differentiable.

Using the *reparameterisation trick* for resampling has been described in [28] in the context of fixing the random number seed in every simulation to produce common random numbers (CRN). Our core contribution is to fix the random numbers used within the resampling step so we can condition the input to resampling which results in the subsequent particle derivative calculations being a function of the parent particle. The gradients can then be efficiently estimated and utilised within the framework of p-MCMC and specifically used to calculate gradient-based proposals for θ: more specifically, this allows us to use NUTS as the proposal. Another novel contribution, relative to the previous work on differentiable particle filters in the neural network community, is that we provide full Bayesian parameter estimates (including variances). This differs from the present literature on differentiable particle filtering which focuses exclusively on point-estimates of parameters. We also compare NUTS' performance with that achieved by HMC and Metropolis-adjusted Langevin algorithm (MALA).

An outline of the paper is as follows: in Section 2 we describe a generic particle filter followed by a description of the difficulties associated with differentiating the sampling and resampling steps. We describe how to calculate the likelihood and gradients in Section 3, the methods to propagate the derivatives of the particle weights in Section 4 and how we extend the *reparameterisation trick* to include the stochastic input to resampling in section 5. We test the likelihood and gradient estimates, explain particle-HMC (p-HMC) and particle-NUTS (p-NUTS) and detail comparative numerical results in the context of three applications in Section 8. Concluding remarks are described in Section 9.

2 Particle filtering background

Assume we have considered t timesteps, obtaining data at each increment of t given by $y_{1:t}$. The state sequence $x_{1:t}$ grows with time where x_t has n_x dimensions. The dynamics and likelihood are parameterised by θ (which has n_θ dimensions) such that

$$\begin{aligned} p\left(y_{1:t}, x_{1:t}|\theta\right) = p(y_1|x_1,\theta)p\left(x_1|\theta\right) & \\ \times \prod_{\tau=2}^{t} p\left(y_\tau|x_\tau,\theta\right) p\left(x_\tau|x_{\tau-1},\theta\right). & \end{aligned} \tag{3}$$

If θ is known, we can run a (conventional) particle filter.

2.1 Particle Filter

At every timestep t, the particle filter draws N samples (particles) from a proposal distribution, $q\left(x_{1:t}|y_{1:t},\theta\right)$, which is parameterised by the sequence of states and measurements. The samples are seen as statistically independent and each represents a different hypothesis for the sequence of states of the system. The ith sample has an associated weight, $w_t^{(\theta,i)}$, which indicates the relative importance of each of the corresponding sample. The weights at $t = 0$ are set to be $1/N$. The proposal distribution is constructed recursively as

$$q\left(x_{1:t}|y_{1:t},\theta\right) = q\left(x_1|y_1,\theta\right)\prod_{\tau=2}^{t} q\left(x_\tau|x_{\tau-1},y_\tau,\theta\right), \tag{4}$$

such that we can pose an estimate with respect to the joint distribution, $p\left(y_{1:t},x_{1:t}|\theta\right)$, as follows:

$$\int p\left(y_{1:t},x_{1:t}|\theta\right) f\left(x_{1:t}\right) dx_{1:t} \approx \frac{1}{N}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)} f\left(x_{1:t}^{(i)}\right). \tag{5}$$

This is an unbiased estimate, where (for t >1)

$$\begin{aligned} w_{1:t}^{(\theta,i)} =& \frac{p\left(y_1|x_1^{(\theta,i)},\theta\right)p\left(x_1^{(\theta,i)}|\theta\right)}{q\left(x_1^{(\theta,i)}|y_1,\theta\right)} \\ &\times \frac{\prod_{\tau=2}^{t} p\left(y_\tau|x_\tau^{(\theta,i)},\theta\right)p\left(x_\tau^{(\theta,i)}|x_{\tau-1}^{(\theta,i)},\theta\right)}{\prod_{\tau=2}^{t} q\left(x_\tau^{(\theta,i)}|x_{\tau-1}^{(\theta,i)},y_\tau,\theta\right)} \qquad (6) \\ =& w_{1:t-1}^{(\theta,i)}\frac{p\left(y_t|x_t^{(\theta,i)},\theta\right)p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)}{q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right)}, \qquad (7) \end{aligned}$$

and is a recursive formulation for the unnormalised weight, $w_{1:t}^{(\theta,i)}$, with incremental weight

$$\sigma\left(x_t^{(\theta,i)},x_{t-1}^{(\theta,i)},\theta\right) = \frac{p\left(y_t|x_t^{(\theta,i)},\theta\right)p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)}{q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right)}. \tag{8}$$

For t=1

$$\sigma\left(x_{1:1}^{(\theta,i)}\right) = \frac{p\left(y_1|x_1^{(\theta,i)},\theta\right)p\left(x_1^{(\theta,i)}|\theta\right)}{q\left(x_1^{(\theta,i)}|y_1\right)}. \tag{9}$$

2.2 Choice of proposal

Three commonly used options for the proposal distribution are:

1. Using the dynamic model as the proposal

$$q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right) = p\left(x_t^{(i)}|x_{t-1}^{(\theta,i)},\theta\right), \tag{10}$$

   which simplifies the weight update to

$$w_{1:t}^{(\theta,i)} = p\left(y_t|x_t^{(\theta,i)},\theta\right)w_{1:t-1}^{(\theta,i)}. \tag{11}$$

2. In certain situations it is possible to use the "optimal" proposal which is

$$q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right)=p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right) \tag{12}$$

with weights updated according to

$$w_{1:t}^{(\theta,i)}=p\left(y_t|x_{t-1}^{(\theta,i)},\theta\right)w_{1:t-1}^{(\theta,i)}. \tag{13}$$

Note that the term "optimal" in this context of the particle proposal means that the variance of the incremental particle weights at the current timestep is minimized. In fact, this variance is zero since the weight in (13) is independent of x_t (as explained in [29]).

3. Using the Unscented Transform, as explained in [30].

2.3 Estimation with respect to the posterior

It is often the case that we wish to calculate estimates with respect to the posterior, $p\left(x_{1:t}|y_{1:t},\theta\right)$, which we can calculate as follows:

$$\int p\left(x_{1:t}|y_{1:t},\theta\right)f\left(x_{1:t}\right)dx_{1:t} \tag{14}$$

$$=\int\frac{p\left(y_{1:t},x_{1:t}|\theta\right)}{p\left(y_{1:t}|\theta\right)}f\left(x_{1:t}\right)dx_{1:t}, \tag{15}$$

$$p\left(y_{1:t}|\theta\right)=\int p\left(y_{1:t},x_{1:t}|\theta\right)dx_{1:t}\approx\frac{1}{N}\sum_{i=1}^{N}w_{1:t}^{(\theta,i)} \tag{16}$$

in line with (5), such that

$$\int p\left(x_{1:t}|y_{1:t},\theta\right)f\left(x_{1:t}\right)dx_{1:t}$$

$$\approx\frac{1}{\frac{1}{N}\sum_{i=1}^{N}w_{1:t}^{(\theta,i)}}\frac{1}{N}\sum_{i=1}^{N}w_{1:t}^{(\theta,i)}f\left(x_{1:t}^{(\theta,i)}\right) \tag{17}$$

$$=\sum_{i=1}^{N}\tilde{w}_{1:t}^{(\theta,i)}f\left(x_{1:t}^{(\theta,i)}\right), \tag{18}$$

where

$$\tilde{w}_{1:t}^{(\theta,i)}=\frac{w_{1:t}^{(\theta,i)}}{\sum_{j=1}^{N}w_{1:t}^{(\theta,j)}} \tag{19}$$

are the normalised weights.

Equation (18) is a biased estimate since it is a ratio of estimates, in contrast with (5).

2.4 Resampling

The algorithm described up to now is called the sequential importance sampling (SIS) algorithm. As time evolves, the normalised weights will become increasingly skewed such that one of the weights given by (19) becomes close to unity and the others approach zero. This is an inevitability and cannot be avoided [31]. As well as the number of effective samples, N_{eff}, eventually becoming 1, most of the computational effort will be expended on particles that have very little contribution to the overall estimate.

It is often suggested that monitoring N_{eff}, can be used to identify the need to resample, where

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}_{1:t}^{(\theta,i)} \right)^2}. \tag{20}$$

There are many resampling methods, some of which are outlined and evaluated in [32] but they all share the same purpose—stochastically replicate particles with higher weights whilst eliminating ones with lower weights. Multinomial resampling is commonly used and involves drawing from the current particle set N times proportional to its weight. The associated distribution is defined by

$$\tilde{w}_{1:t}^{(\theta,i)} \quad \text{for} \quad i = 1, \ldots, N. \tag{21}$$

To keep the total unnormalised weight constant (such that the approximation (16) is the same immediately before and after resampling), we assign each newly-resampled sample an unnormalised weight

$$\frac{1}{N} \sum_{i=1}^{N} w_{1:t}^{(\theta,i)}. \tag{22}$$

Note this is such that the normalised weights after resampling are $\frac{1}{N}$.

3 Calculating the likelihood and gradients

We pose the calculation of the likelihood of the parameter as the calculation of the approximation in (16), (ie the sum of unnormalised particle filter weights), with $t = T$.

Differentiating the weights gives an approximation to the gradient of the likelihood[1]:

$$\frac{d}{d\theta} p(y_{1:t}|\theta) \quad \approx \quad \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\theta} w_{1:t}^{(\theta,i)}. \tag{23}$$

For numerical stability, it is typically preferable to propagate values in logs. Applying the Chain Rule to (16) and (23) gives

$$\begin{aligned} \frac{d}{d\theta} \log p(y_{1:t}|\theta) &\approx \frac{1}{N} \frac{1}{p(y_{1:t}|\theta)} \sum_{i=1}^{N} w_{1:t}^{(\theta,i)} \frac{d}{d\theta} \log w_{1:t}^{(\theta,i)} \qquad (24) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_{1:t}^{(\theta,i)} \frac{d}{d\theta} \log w_{1:t}^{(\theta,i)}. \qquad (25) \end{aligned}$$

[1] We note that this approach differs from that advocated in [20–22], which use a Fixed-Lag filter (with a user-specified lag) to approximate the derivatives. In contrast, we explicitly calculate the derivatives of the approximation to the likelihood.

Note that in (25), the weights outside the logs are normalised, while the weights inside are not. The log weights can be calculated recursively as

$$\log w_{1:t}^{(\theta,i)} = \log w_{1:t-1}^{(\theta,i)} + \log \sigma \left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta \right), \tag{26}$$

so

$$\frac{d}{d\theta} \log w_{1:t}^{(\theta,i)} = \frac{d}{d\theta} \log w_{1:t-1}^{(\theta,i)} + \frac{d}{d\theta} \log \sigma \left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta \right), \tag{27}$$

where

$$\begin{aligned} &\frac{d}{d\theta} \log \sigma \left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta \right) \\ &= \frac{d}{d\theta} \log p \left(x_t^{(\theta,i)} | x_{t-1}^{(\theta,i)}, \theta \right) + \frac{d}{d\theta} \log p \left(y_t | x_{t-1}^{(\theta,i)} \right) \\ &\quad - \frac{d}{d\theta} \log q \left(x_t^{(\theta,i)} | x_{t-1}^{(\theta,i)}, \theta, y_t \right). \end{aligned} \tag{28}$$

So, if we can differentiate the single measurement likelihood, transition model and proposal, we can calculate (an approximation to) the derivative of the log-likelihood for the next time step, thus recursively approximating the log-likelihood derivatives for each time step. While this is true, there are some challenges involved, which we now discuss in turn.

If the particle filter is using the transition model as the dynamics (as in (10)), the likelihood in the weight update does not explicitly depend on θ and we might initially suppose that $d \log \sigma / d\theta = 0$. If this were the case, an induction argument using (27) would show that the weight derivatives were always zero and therefore give an approximation of zero for the gradient of the likelihood for θ. This seems intuitively incorrect. Indeed, the flaw in this reasoning is that, in fact, the likelihood (somewhat implicitly) does depend on θ since $x_t^{(\theta,i)}$ depends on θ. Applying the Chain Rule gives

$$\frac{d}{d\theta} \log p \left(y_t | x_t^{(\theta,i)} \right) = \frac{d}{dx} \log p \left(y_t | x \right) \bigg|_{x = x_t^{(\theta,i)}} \frac{d}{d\theta} x_t^{(\theta,i)}. \tag{29}$$

Since $x_t^{(\theta,i)}$ is a random variable sampled from the proposal, we use the *reparameterisation trick* [27]: we consider the derivative for a fixed random number seed. More precisely, let $\epsilon_t^{(i)}$ be the vector of standard $\mathcal{N}(0,1)$ random variables used when sampling from the proposal such that, if $\epsilon_t^{(i)}$ is known, then $x_t^{(\theta,i)}$ is a deterministic function (that can be differentiated) of $x_{t-1}^{(\theta,i)}$. We then consider

$$\begin{aligned} \frac{d}{d\theta} p \left(y_{1:t} | \theta \right) &= \frac{d}{d\theta} \int p \left(y_{1:t}, \epsilon_{1:t} | \theta \right) d\epsilon_{1:t} \qquad (30) \\ &= \int \frac{d}{d\theta} p \left(y_{1:t}, \epsilon_{1:t} | \theta \right) d\epsilon_{1:t} \qquad (31) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\theta} p \left(y_{1:t} | \epsilon_{1:t}^{(i)}, \theta \right) \qquad (32) \end{aligned}$$

where $\epsilon_{1:t}^{(i)} \sim p(\epsilon_{1:t})$ is considered fixed and, most importantly, (32) then involves differentials that can be calculated.

As a simple example, consider sampling from the dynamics with a random walk proposal and θ being the standard deviation of the process noise. This is such that

$$x_t^{(\theta,i)} = x_{t-1}^{(\theta,i)} + \theta\epsilon_t^{(i)} \tag{33}$$

so

$$\frac{d}{d\theta}x_t^{(\theta,i)} = \frac{d}{d\theta}x_{t-1}^{(\theta,i)} + \epsilon_t^{(i)} \tag{34}$$

which can be calculated recursively and then used to calculate (29).

More generally, the derivatives of the weight are non-zero and, to calculate these derivatives, we have to propagate the particle derivatives $dx_t^{(\theta,i)}/d\theta$.

4 Calculating the derivatives

In order to propagate the derivatives of the particle weights we need to calculate:

- The particle derivatives,

$$\frac{dx_t^{(\theta,i)}}{d\theta}. \tag{35}$$

- The derivatives of the proposal pdfs,

$$\frac{d}{d\theta}\log q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\right). \tag{36}$$

- The derivatives of the prior log pdfs,

$$\frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right). \tag{37}$$

- The derivatives of the single measurement likelihood log pdfs,

$$\frac{d}{d\theta}\log p\left(y_t|x_{t-1}^{(\theta,i)}\right). \tag{38}$$

In this section, we show how to calculate these derivatives in turn.

4.1 Derivative of New Particle States

We now describe how to calculate $dx_{t-1}^{(\theta,i)}/d\theta$. Suppose the proposal takes the following form:

$$\begin{aligned} & q\Big(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\Big) \\ & \qquad = \mathcal{N}\left(x_t^{(\theta,i)};\mu\left(x_{t-1}^{(\theta,i)},\theta,y_t\right),C\left(x_{t-1}^{(\theta,i)},\theta,y_t\right)\right) \end{aligned} \tag{39}$$

where $\mu(.)$ and $C(.)$ are functions of the old particle state, the measurement and the parameter. Such a generic description can articulate sampling from the prior, or defining a proposal using a Kalman filter with the predicted mean and covariance given by the motion model.

If we sample the proposal noise $\epsilon_t^{(i)} \sim \mathcal{N}(\cdot; 0, I_{n_X})$ in advance, the new particle states can be written as a deterministic function

$$\begin{aligned} x_t^{(\theta,i)} &= f(x_{t-1}^{(\theta,i)}, \theta, y_t, \epsilon_t^{(i)}) \\ &\triangleq \mu(x_{t-1}^{(\theta,i)}, \theta, y_t) + \sqrt{C(x_{t-1}^{(\theta,i)}, \theta, y_t)} \times \epsilon_t^{(i)}. \end{aligned} \tag{40}$$

We would like to compute the derivative of this w.r.t. the parameter. Care must be taken however, since $x_{t-1}^{(\theta,i)}$ is itself a function of θ (if a different θ was chosen, a different $x_{t-1}^{(\theta,i)}$ would have been sampled).

$$\frac{dx_t^{(\theta,i)}}{d\theta} = \frac{d}{d\theta} f(x_{t-1}^{(\theta,i)}, \theta, y_t, \epsilon_t^{(i)}) \tag{41}$$

$$= \frac{\partial f}{\partial x_{t-1}^{(\theta,i)}} \frac{dx_{t-1}^{(\theta,i)}}{d\theta} + \frac{\partial f}{\partial \theta} \frac{d\theta}{d\theta} \tag{42}$$

$$= \frac{\partial f}{\partial x_{t-1}^{(\theta,i)}} \frac{dx_{t-1}^{(\theta,i)}}{d\theta} + \frac{\partial f}{\partial \theta}. \tag{43}$$

Note that $df/d\theta$ in (41) is not the same as $\partial f/\partial\theta$ in (42) — see Appendix A for the distinction. Note also that the terms here are matrix-valued in general: $\partial f/\partial x_{t-1}^i$ is an $n_X \times n_X$ matrix, and $dx_{t-1}^i/d\theta$ and $\partial f/\partial\theta$ are $n_X \times n_\Theta$ matrices.

Also note that for $t \geq 2$, $x_{t-1}^{(\theta,i)}$ implicitly depends on $x_{t-2}^{(\theta,i)}$, which itself depends on θ. Hence we need the total derivative $dx_{t-1}^{(\theta,i)}/d\theta$.

4.2 Derivative of Proposal

To differentiate the log proposal pdf, we note that we can write it as

$$\log q\left(x_t^{(\theta,i)} | x_{t-1}^{(\theta,i)}, \theta, y_t\right) = Q\left(x_{t-1}^{(\theta,i)}, \theta, y_t, \epsilon_t^{(i)}\right) \tag{44}$$

where (dropping the fixed values $\epsilon_t^{(i)}$ and y_t for notational convenience)

$$Q\left(x_{t-1}^{(\theta,i)}, \theta\right) \triangleq \log q\left(f(x_{t-1}^{(\theta,i)}, \theta) | x_{t-1}^{(\theta,i)}\right) \tag{45}$$

$$= \log \mathcal{N}\left(f(x_{t-1}^{(\theta,i)}, \theta); \mu(x_{t-1}^{(\theta,i)}, \theta), C(x_{t-1}^{(\theta,i)}, \theta)\right) \tag{46}$$

where we emphasise again that we assume the proposal is Gaussian. We then get

$$\frac{d}{d\theta}Q(x_{t-1}^{(\theta,i)},\theta) = \frac{\partial}{\partial f}\log\mathcal{N}(f;\mu,C)\left(\frac{df}{d\theta}+\frac{d\mu}{d\theta}+\frac{dC}{d\theta}\right) \tag{47}$$

$$\begin{aligned} &= \frac{\partial}{\partial f}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}\right) \\ &\quad + \frac{\partial}{\partial\mu}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial\mu}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial\mu}{\partial\theta}\right) \\ &\quad + \frac{\partial}{\partial C}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial C}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial C}{\partial\theta}\right). \end{aligned} \tag{48}$$

where we denote $\mu = \mu(x_{t-1}^{(\theta,i)},\theta)$ and $C = C(x_{t-1}^{(\theta,i)},\theta)$ for brevity. The derivatives of $\log\mathcal{N}(f;\mu,C)$ are given in Appendix C.

4.3 Derivative of the Prior

We now describe how to calculate $\frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)$. Let

$$P(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}) \triangleq \log p\left(f\left(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}\right)|x_{t-1}^{(\theta,i)},\theta\right) \tag{49}$$

$$= \log\mathcal{N}\left(f\left(x_{t-1}^{(\theta,i)}\right);a\left(x_{t-1}^{(\theta,i)},\theta\right),\Sigma(\theta)\right) \tag{50}$$

where here we assume that the transition model has additive Gaussian noise that is independent of $x_{t-1}^{(\theta,i)}$. Then

$$\begin{aligned} \frac{d}{d\theta}P(x_{t-1}^{(\theta,i)},\theta) = &\frac{\partial}{\partial f}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}\right) \\ &+ \frac{\partial}{\partial a}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial a}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial a}{\partial\theta}\right) \\ &+ \frac{\partial}{\partial\Sigma}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial\Sigma}{\partial\theta}\right). \end{aligned} \tag{51}$$

where we denote that $a = a\left(x_{t-1}^{(\theta,i)},\theta\right)$ and $\Sigma = \Sigma(\theta)$ for brevity. Note that this makes clear that since the realisation of the sampled particles, $x_t^{(\theta,i)}$, are, in general, dependent on θ, (51) includes $\frac{dx_{t-1}^{(\theta,i)}}{d\theta}$. Also note that these derivatives of $\log\mathcal{N}(f;a,\Sigma)$ are evaluated at $a(x_{t-1}^{(\theta,i)})$ (the prior mean) and not at μ (the proposal mean) as was the case in (48).

4.4 Derivative of the Likelihood

We now describe how to calculate $\frac{d}{d\theta}\log p\left(y_t|x_{t-1}^{(\theta,i)}\right)$. Let

$$L(x_t^{(\theta,i)},\theta,y_t) \triangleq \log p\left(y_t|x_t^{(\theta,i)},\theta\right) \tag{52}$$

$$= \log\mathcal{N}\left(y_t;h(x_t^{(\theta,i)},\theta),R(\theta)\right) \tag{53}$$

where we assume that the likelihood is Gaussian with a variance that is independent of $x_t^{(\theta,i)}$. Then

$$\begin{aligned} \frac{d}{d\theta} L\left(x_t^{(\theta,i)}, \theta, y_t\right) =& \frac{\partial}{\partial h} \log \mathcal{N}(y_t; h, R) \left(\frac{\partial h}{\partial x_t^{(\theta,i)}} \frac{d x_t^{(\theta,i)}}{d\theta} + \frac{\partial h}{\partial \theta} \right) \\ &+ \frac{\partial}{\partial R} \log \mathcal{N}(y_t; h, R) \frac{dR}{d\theta} \end{aligned} \tag{54}$$

where we denote $h = h(x_t^{(\theta,i)}, \theta)$ and $R = R(\theta)$ for brevity.

5 Resampling for a Differentiable Particle Filter

Unlike a standard particle filter, we also need to resample the weight derivatives

$$\frac{d}{d\theta} w_{1:t}^{(\theta,i)} \tag{55}$$

as well as the particle derivatives

$$\frac{d}{d\theta} x_t^{(\theta,i)}. \tag{56}$$

Let

$$c_t^{(\theta,i)} = \frac{\sum_{j=1}^{i} w_{1:t}^{(j,\theta)}}{\sum_{j=1}^{N} w_{1:t}^{(j,\theta)}}, \tag{57}$$

be the normalised cumulative weights and the index sampled for particle i be given by

$$\kappa_i = \kappa\left(\nu_t^i, w_{1:t}^{1:N}\right) = \sum_{j=0}^{N-1} \left[\nu_t^i > c_t^{(j,\theta)}\right] \tag{58}$$

where $\nu_t^i \sim \text{Uniform}((0,1])$ are independent for each particle and time step. Note that the particle indices are sampled according to a Categorical distribution giving a Multinomial resampler, where each index is resampled with probability proportional to its weight. Other resampling schemes are possible and could reduce the variance of any estimates.

The resampled weights are set to be the same as each other, while preserving the original sum:

$$x_t'^{(\theta,i)} = x_t^{(\theta,\kappa_i)}, \tag{59}$$

$$w_{1:t}'^{(\theta,i)} = \frac{1}{N} \sum_{j=1}^{N} w_{1:t}^{(\theta,j)}. \tag{60}$$

From (60), it is clear that

$$\frac{d}{d\theta} w_{1:t}'^{(\theta,i)} = \frac{1}{N} \sum_{j=1}^{N} \frac{d}{d\theta} w_{1:t}^{(\theta,j)}. \tag{61}$$

In order to convert this to log weights, applying the Chain Rule gives

$$\frac{d}{d\theta}\log w_{1:t}'^{(\theta,i)} = \frac{1}{N}\frac{1}{w_{1:t}'^{(\theta,i)}}\sum_{j=1}^{N} w_{1:t}^{(\theta,j)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,j)} \tag{62}$$

$$= \sum_{j=1}^{N}\tilde{w}_{1:t}^{(\theta,j)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,j)} \tag{63}$$

where $\tilde{w}_{1:t}^{(\theta,j)}$ are the normalised weights.

To get the particle gradient note that (where κ is differentiable),

$$\frac{d}{d\theta}x_t'^{(\theta,i)} = \frac{\partial}{\partial\kappa}x_t\left(\theta,\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right)\right)\frac{\partial}{\partial\theta}\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right) + \frac{d}{d\theta}x_t\left(\theta,\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right)\right) \tag{64}$$

Since $\partial\kappa/\partial\theta = 0$ except where

$$\nu_t^i = c_t^{(\theta,j)} \text{ for some } i,j = 1,\ldots,N \tag{65}$$

then

$$\frac{d}{d\theta}x_t'^{(\theta,i)} = \frac{d}{d\theta}x_t^{(\theta,\kappa_i)} \tag{66}$$

almost surely, so the derivative is obtained by taking the derivative of the parent particle.

5.1 Discontinuities after a Resampling Realisation

Sampling with CRN results in a "deterministic" function $f(\theta)$ i.e. evaluating the function twice results in the same output. Sampling without CRN would result in different outputs. A discontinuity occurs in the estimate of the log-likelihood in Figure 1 (a) when two values of θ cause a different resampling realisation to occur. On one side of the discontinuity, for some range of values of θ, the resampling happens at the same times and all particles have the same parents at all resampling events: the particles for different values of θ all share a single family tree. When θ is changed to the other side of the discontinuity the resampling realisations change and so the family tree also changes. Since the approximation to the likelihood (and its gradient) is a function of the family tree, the change in family tree results in a discontinuity in the likelihood approximation.

The best approach to limiting these discontinuities is to come up with a high-performance proposal. This is exemplified when comparing results obtained using the prior with those obtained using the optimal proposal (which can, in some settings, be derived using a Kalman Filter). We choose such a simple example to demonstrate this point. Suppose that the state is a single real number, with a motion model which is a random walk with zero initial mean and the standard deviation of both the initial state and each subsequent propagation is θ. Given the state, the measurements are not dependent on θ and are equal to the target state plus errors of known and fixed variance, R:

$$p\left(x_1\right) = \mathcal{N}\left(x_1; 0, \theta^2\right), \tag{67}$$

$$p\left(x_k \mid x_{k-1},\theta\right) = \mathcal{N}\left(x_k; x_{k-1}, \theta^2\right), \tag{68}$$

$$p\left(y_k \mid x_k, \theta\right) = \mathcal{N}\left(y_k; x_k, R\right). \tag{69}$$

We show in Appendix B how to calculate the mean and covariance of the optimal proposal for each particle x_{t-1}^i, as well as the necessary derivatives.

We run Algorithm 1 and compute an estimate of the log-likelihood and associated gradient across a range of 500 values of θ, equally spaced from 1 to 4. The true value is θ=2. We consider $N = 2000$ and $T = 250$ observations.

The log-likelihood and gradient of the log-likelihood w.r.t. θ, at each instance of θ, can be seen in Figures 1 (a) and (b), respectively. Figures 1 (c) and (d) are zoomed in instances of these plots that include results from the Kalman Filter, two runs of multinomial resampling (to indicate the difference in results when running the simulation twice) and CRN resampling. From looking at Figure 1 (a), there are no obvious differences in the graphs of the log-likelihood for the estimates given by the Kalman Filter and the estimates produced by the particle filter when using different combinations of the prior and optimal proposal with using CRN and multinomial resampling. However, there is a notable difference when comparing the gradient of the log-likelihood w.r.t. θ. Using the prior as the proposal results in large discontinuities in the estimate when compared with the optimal proposal. Figures 1 (c) and (d) show that using CRN and the optimal proposal produces piece-wise continuous estimates. This is in contrast to using multinomial resampling where there a lot of fluctutations are apparent. We therefore advocate using an optimal proposal, or an approximation to such a proposal, in conjunction with CRN resampling. A good proposal will minimise the variance of the incremental weights at the current timestep. The process of selecting a good proposal can be time consuming but, as outlined in [33], can be critical in obtaining good results in other contexts.

We also note that, while optimisation algorithms are likely to be sensitive to discontinuities, we are focused on sampling θ and using MCMC to correct for the disparity between the proposal and the target.

6 Differentiable Particle Filters

6.0.1 Soft Resampling

Soft resampling was introduced in [34] and utilised in [24] and considers an approximation which involves drawing from the distribution $q(n) = \alpha w_{1:t}^{(\theta,i)} + (1-\alpha)1/N$, with $\alpha \in [0,1]$, representing a trade-off parameter. If $\alpha = 1$, regular resampling is used and if $\alpha = 0$ the algorithm performs subsampling. The new weights are calculated by

$$w_{1:t}^{\prime(\theta,i)} = \frac{w_{1:t}^{(\theta,i)}}{\alpha w_{1:t}^{(\theta,i)} + (1-\alpha)1/N}. \tag{70}$$

This gives non-zero estimates of the gradient because the dependency on the previous weights is maintained. By changing α, this method trades resampling quality for biased gradient estimates.

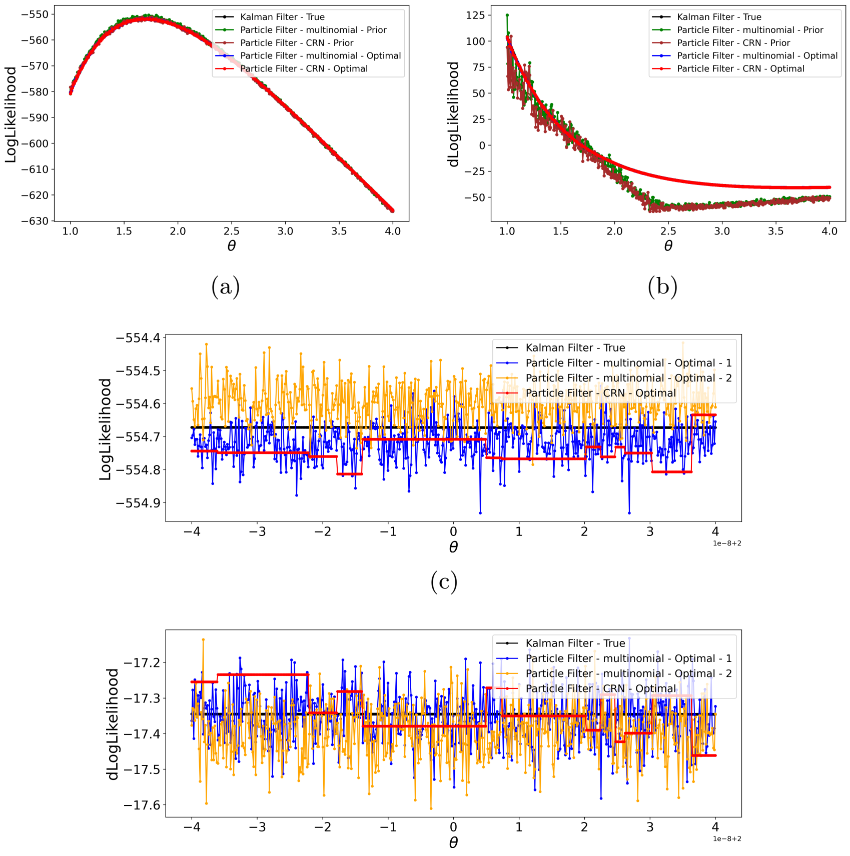
6.0.2 Gumbel Softmax

The Gumbel-Max trick [35] provides a way to sample a variable, z, from a categorical distribution that contains class probabilities, π_x. If we assume the categorical samples are one-hot vectors, z can be sampled by

$$z = \text{ onehot } \left(argmax_i \left\{G_i + \log\left(\pi_i\right)\right\}\right) \tag{71}$$

Algorithm 1: Particle Filter

```
Input: $\theta$, $y_{1:T}$
Initialise: $x_0^i$, $\frac{d}{d\theta}x_0^i$, $\log(w_0^i)$, $\frac{d}{d\theta}\log(w_0^i)$
for $t$ = 1, ..., $T$ do
    If necessary, resample $x_{t-1}^i$, $\log(w_{1:k-1}^i)$, $dx_{t-1}^i/d\theta$, $d\log(w_{1:t-1}^i)/d\theta$ as described in
     Section 5.
    Sample the new particles $x_t^i$ and calculate the partial derivatives
        $$\frac{\partial}{\partial\theta}f(x_{t-1}^i,\theta), \frac{\partial}{dx_{t-1}^i}f(x_{t-1}^i,\theta),$$
    .
    Get the proposal mean, $\mu(x_{t-1}^i)$ and covariance, $C(x_{t-1}^i)$ for each particle $x_{t-1}^i$, as well
     as their derivatives
        $$\frac{\partial}{\partial\theta}\mu(x_{t-1}^i), \frac{\partial}{x_{t-1}^i}\mu(x_{t-1}^i), \frac{\partial}{\partial\theta}C(x_{t-1}^i), \frac{\partial}{x_{t-1}^i}C(x_{t-1}^i),$$
    seen in Section 4.1.
    Get the particle gradients $\frac{d}{d\theta}x_t^i$ using Subsection 4.1.
    Get the derivatives of the prior, proposal and likelihood using Subsections 4.2, 4.3, 4.4.
    Evaluate the new log weights $\log w_{1:k}^i$ and log weight derivatives$\frac{d}{d\theta}\log w_{1:t}^i$ using (26)
     and (27), respectively.
end
Evaluate the final log likelihood, $\log p(y_{1:T}|\theta)$, and associated derivative, $\frac{d}{d\theta}\log p(y_{1:T}|\theta)$,
 using (16) and (25), respectively.
```



(a)

(b)

(c)

(d)

Figure 1: Plots of the log-likelihood (a), gradient of the log-likelihood w.r.t. θ (b), a zoomed in section of the log-likelihood plot (c) and associated gradient (d). All plots: The true values given from the Kalman Filter (black), particle filter using *reparameterisation trick* resampling (red) and multinomial resampling (blue and orange).

where $x_i = G_i + \log(\pi_i)$ and G_i are independently sampled from $Gumbel(0,1)$. The summation in (71) is similar to the reparametrisation trick which is described previously however the *argmax* function is not differentiable. The work outlined in [36] and [37] describes a differentiable approximation to *argmax* called *softmax* which is defined to be

$$z = \frac{\exp\left(\frac{x_k}{\lambda}\right)}{\sum_{i=1}^{n} \exp\left(\frac{x_i}{\lambda}\right)}. \tag{72}$$

The temperature parameter, λ, is defined by the user and controls how closely the resulting Gumbel-softmax distribution approximates the categorical distribution.

6.0.3 Optimal Transport

A fully differentiable particle filter is described in [38] that resamples by using Optimal Transport ideas seen in [39]. The method ensures stability by having unbiased gradient estimates whilst adding a little bias in the log-likelihood. The method needs additional hyper-parameters and runs in $O(N^2)$ time complexity so can be computationally expensive.

6.0.4 Fisher's identity to calculate gradient of log-likelihood

As described in section 1, [16] recursively computes the gradient of the log-likelihood using Fisher's Identity, which can be summarised as follows:

$$\begin{aligned}
\frac{d}{d\theta} \log p\left(y_{1:t}|\theta\right) =& \frac{1}{p\left(y_{1:t}|\theta\right)} \frac{d}{d\theta} p\left(y_{1:t}|\theta\right) && (73)\\
=& \frac{1}{p\left(y_{1:t}|\theta\right)} \frac{d}{d\theta} \int p\left(y_{1:t}, x_{1:t}|\theta\right) dx_{1:t} && (74)\\
=& \frac{1}{p\left(y_{1:t}|\theta\right)} \int \frac{d}{d\theta} p\left(y_{1:t}, x_{1:t}|\theta\right) dx_{1:t} && (75)\\
=& \int \frac{p\left(y_{1:t}, x_{1:t}|\theta\right)}{p\left(y_{1:t}|\theta\right)} \frac{d}{d\theta} \log p\left(y_{1:t}, x_{1:t}|\theta\right) dx_{1:t} && (76)\\
=& \int p\left(x_{1:t}|y_{1:t}, \theta\right) \frac{d}{d\theta} \log p\left(x_{1:t}, y_{1:t}|\theta\right) dx_{1:t} && (77)\\
\approx& \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_{1:t}^{(\theta,i)} \underbrace{\frac{d}{d\theta} \log p\left(x_{1:t}^{(\theta,i)}, y_{1:t}|\theta\right)}_{\alpha_n^{(\theta,i)}} && (78)
\end{aligned}$$

Note that we can calculate the term inside the sum recursively as follows:

$$\alpha_n^{(\theta,i)} = \alpha_{n-1}^{(\theta,i)} + \frac{d}{d\theta} \log p\left(y_t|x_{t-1}^{(\theta,i)}\right) + \frac{d}{d\theta} \log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, \theta\right), \tag{79}$$

where $\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, \theta\right)$ is the derivative of the prior and $\frac{d}{d\theta} \log p\left(y_t|x_{t-1}^{(\theta,i)}\right)$ is the derivative of the log-likelihood (see (8), (4.3) and (4.4), respectively).

Recent work in [40] has outlined how to calculate this in the framework of Pytorch [41] without having to modify the forward pass. They include a "*stop-gradient*" operator that stops the gradients of the weights flowing into the resampling distribution. Much like the method described in this paper, minimal changes to the particle filtering algorithm need to be employed.

References [16] and [17] focus on having an unbiased estimator of the gradient of the log likelihood: we approximate with the gradient of the logarithm of the expectation whereas they calculate the expectation of the gradient of the logarithm. A feature of our approximation, perhaps surprisingly, is that it uses all samples from all iterations whereas the approach in [16] and [17] is recursively calculated along the trajectory of each particle. If, as is likely, the particle trajectories are degenerate, the approach in [16] and [17] will have a limited ability to use the diversity of states early in the trajectory to inform the gradient estimate. In short, we perceive that choosing between using the two approximations becomes a bias-variance trade-off.

7 Estimation of parameters

If we have a prior, $p(\theta)$, for which the likelihood, $p(y_{1:T}|\theta)$, or log-likelihood[2] can be calculated, we can run p-MCMC to estimate $p(\theta|y_{1:T}) \propto p(\theta)\, p(y_{1:T}|\theta)$. The gradient of the log-posterior of θ is given by

$$\nabla \log p(\theta|y_{1:t}) = \nabla \log p(\theta) + \nabla \log p(y_{1:t}|\theta), \tag{80}$$

where $\nabla \log p(\theta)$ is the gradient of the log-prior and $\nabla \log p(y_{1:t}|\theta)$ is the gradient of the log-likelihood. If we know $\nabla \log p(\theta|y_{1:t})$, it is possible to guide proposals to areas of higher probability within $\pi(\theta)$.

7.1 Hamiltonian Monte Carlo

HMC is a gradient based algorithm which uses Hamilton's equations to generate new proposals. Since it uses gradients, it is better at proposing samples than a random-walk proposal. It was first developed in the late 1980s [42] and in the last decade it has become a popular approach when implementing MCMC [9]. In the following section we give a high level conceptual overview of HMC and direct the reader to [43] for a more thorough explanation. Hamilton's equations are a pair of differential equations that describe a system in terms of its position and momentum where the potential of the system is defined by $U = -\log(\pi(\theta))$. HMC introduces a momentum vector m which moves samples at θ on a trajectory to θ'. The total energy or Hamiltonian of a system can be expressed as $H(\theta, m) = U(\theta) + K(m)$, and is comprised of the sum of the Kinetic energy $K(m)$, which is dependent on where in the parameter space the samples are, and the potential energy $U(\theta)$, which is independent on the momentum m.

Hamilton's equations describe how the system evolves as a function of time and are:

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial m}, \frac{dm}{dt} = -\frac{\partial H}{\partial \theta}. \tag{81}$$

The joint density is

$$\begin{aligned} p(\theta, m) \propto \exp(-H(\theta, m)) &= \exp(-U(\theta)) \cdot \exp(-K(m)) \\ &= p(\theta)p(m), \end{aligned} \tag{82}$$

[2] The log-likelihood is likely to be more stable numerically.

therefore θ and m are independent samples from the joint density so m can be sampled from any distribution. For simplicity a Gaussian is often used, and we make that choice here.

Many numerical integration methods exist which discretise Hamilton's equations and can be seen in [44] with the leapfrog method being the go-to method for HMC. Leapfrog is a symplectic method which means the Hamiltonian remains close to its initial value, though not equal to it exactly, as the system is simulated. This means samples are generated with a high accept/reject ratio so the target is explored efficiently. Leapfrog is also a reversible method which allows detailed balance to be maintained. Finally, Leapfrog is a low-order method which uses relatively few gradient evaluations per step and is therefore computationally cheap. Algorithm 1 is run every time a gradient evaluation is made within the Leapfrog numerical integrator. We note that the Pseudocode for Algorithm 1 is specific to the problem described in section 5.1 but can be easily applied to other models.

The samples generated are governed by a predetermined number of steps L of size ϵ, decided by the user. HMC is highly sensitive to the choice of these parameters, particularly L. If too large, computation time can be wasted as the trajectory can end close to where it started and, if too small, the proposal can exhibit random-walk behaviour. In some cases it has been shown that randomising L can be beneficial to avoid periodicities in the underlying Hamiltonian dynamics [45].

7.2 No-U-Turn sampler

NUTS [10] is an extension of HMC which adaptively finds the optimal number of leapfrog steps to take, for a given stepsize, and therefore eliminates the need to tune this number. NUTS does this by sampling from a trajectory of generated states in such a way that detailed balance holds. We outline how this trajectory is built in Section 7.2.1 and describe the stopping criteria in Section 7.2.2. Sections 7.2.3 and 7.2.4 outline how to sample from the generated trajectory and why detailed balance holds, respectively. We also provide relevant further details in Section 7.2.5.

7.2.1 Generating a trajectory

NUTS generates samples both forwards and backwards in time from the initial state θ^0 using leapfrog, by first sampling an initial momentum m^0. To ensure reversibility, a Bernoulli trial with probability 0.5 is undertaken to pick an initial direction d (either forwards $+\epsilon$, or backwards $-\epsilon$) after which a leapfrog step is taken. Until a U-turn is detected, a new direction is then sampled after every 2^j leapfrog steps where j, initialized to zero, is incremented by one after selecting a new direction. In doing so, NUTS builds a full binary tree of height j where the number of steps taken between tests for a U-turn is double the amount taken since the previous such test and each leaf node represents states along the trajectory.

7.2.2 Testing for U-turns

The doubling process described above stops when the position and momentum states at the far left (θ^-, m^-) and far right (θ^+, m^+) of the tree satisfy either of the following conditions:

$$\left(\theta^+ - \theta^-\right) \cdot m^- < 0 \quad \text{or} \quad \left(\theta^+ - \theta^-\right) \cdot m^+ < 0. \tag{83}$$

These conditions are met when the trajectory begins to double back on itself, i.e. begins to U-turn. When building a trajectory, once one of these conditions is satisfied the current subtree being built is discarded, and a sample from the resulting full tree (without the discarded subtree) is taken. This process of discarding means that any state in the tree can move to any other state without breaching the U-turn: This is important with respect to ensuring that detailed balance is maintained.

7.2.3 Drawing a Sample from the Trajectory

In the original description of NUTS, slice sampling was undertaken to sample a new state from the trajectory. This was done to account for the fact that by using leapfrog, the Hamiltonian dynamics are approximated by numerical integration. Practically, an auxiliary variable, u, is introduced which is sampled uniformly between 0 and $H(\theta^0, m^0)$. Once a U-turn is detected (and the relevant subtree has been discarded) a sample is then uniformly selected from the states which satisfy: $H(\theta, m) > u$. This effectively prevents samples being selected for which the dynamics were poorly modelled (and therefore would have had a poor acceptance probability had HMC been used).

7.2.4 Pertinent Elements of the Proof

In following the steps above, the algorithm satisfies detailed balance and is therefore a valid MCMC proposal. For a complete description we point the reader to Section 3.1.1 in [10]. However we summarise the more pertinent points here.

For reversibility, a pair of states (θ, m) and (θ', m') must be able to transition to each other. We require that the subset of states that could be transitioned to $\mathcal{B}$, where $\mathcal{B}$ is a subset of a potentially larger set of all states observed $\mathcal{C}$, to satisfy:

$$p(\mathcal{B}, \mathcal{C}|\theta, m) = p(\mathcal{B}, \mathcal{C}|\theta', m') \tag{84}$$

This is true if the states in NUTS are generated deterministically, i.e. through doubling the number of states. In this instance there will always be a series of directions that will produce the required tree. Furthermore, by discarding the samples in the subtree being generated which contain a U-turn we remove the possibility of extending the sequence of states such that it contains pairs of states for which the transition is possible from one to the other but not in the other direction.

7.2.5 Further details

The numerical integrator used to simulate the Hamiltonian dynamics is chosen to preserve the geometric structure of the dynamics, i.e. has a unit determinant. NUTS uses the leapfrog method, like HMC, which has this property. As a result, NUTS avoids the situation where, due to numerical errors, some states in the trajectory are unlikely to be sampled.

To avoid excessively large trees, which can result from choosing too small a stepsize, it is necessary to upper bound the tree depth. To find a reasonable initial stepsize we used the heuristic approach in [10] though we do not use dual averaging to automatically tune ϵ between NUTS iterations. We also elect to not adapt the mass matrix and set the mass matrix to be the identity matrix. However, we anticipate adaption could improve performance.

Similarly to HMC, were NUTS able to run with an exact integrator, all the states would be equally likely to be sampled. As the leapfrog step is volume-preserving, running HMC and NUTS does not require computing of determinants related to the Jacobian. Since using CRN in calculating the gradient from a particle filter results in gradients that are a deterministic function of the parameter, we note that these Jacobian terms still cancel (and we do not need to integrate over the possible dynamic states).

An alternative approach to the slice sampling step is to use multinomial sampling where each leaf node in the tree is attributed a transition probability. This approach is performed in the probabilistic programming language Stan [11]. While we have not used this approach in this work we believe it should be considered in the future.

The pseudo-code for p-HMC and p-NUTS can be seen in Algorithm 2. We implement the Algorithms 1 and 3 in [10] for HMC and NUTS respectively and note that each time a gradient evaluation is made during the leapfrog step, we run Algorithm 1.

7.3 Metropolis-Adjusted Langevin Algorithm (MALA)

MALA is a M-H proposal that includes gradient information about the log-posterior (as seen in [22]):

$$\theta' = \mathrm{N}\left(\theta + \frac{1}{2}\Gamma\nabla\log p(\theta|y_{1:T}), \Gamma\right), \tag{85}$$

where $\Gamma = \gamma^2 I_d$, and γ is the stepsize. In a similar way to [21], we run the algorithm with different stepsizes and chose the one that provides an acceptance rate of around 0.3 in the stationary phase. Although we have presented MALA as an independent proposal it is a special case of HMC when $L = 1$.

Algorithm 2: Particle - HMC or NUTS

```
Input: $\theta_0$, $y_{1:T}$, $M$, $L$
$\ell(\theta)$, $\nabla\mathcal{L}(\theta)$ = Run Algorithm 1
$\epsilon$ = Find Reasonable $\epsilon(\theta_0)$
for $i$ = 1 to $M$ do
    HMC = Algorithm 1 in [10] or
    NUTS = Algorithm 3 in [10]
end
Function Leapfrog($\theta$, $m$, $y_{1:T}$, $\nabla\mathcal{L}(\theta)$):
    $m' = m + 0.5 \cdot \epsilon \cdot \nabla\mathcal{L}(\theta)$
    $\theta' = \theta + \epsilon \cdot m'$
    $\ell(\theta)$, $\nabla\mathcal{L}(\theta)'$ = Run Algorithm 1
    $m' = m' + 0.5 \cdot \epsilon \cdot \nabla\mathcal{L}(\theta)'$
    return $\theta'$, $m'$, $\ell(\theta)'$, $\nabla\mathcal{L}(\theta)'$
```

8 Numerical Experiments

8.1 Linear Gaussian State Space Model

We consider the Linear Gaussian State Space (LGSS) model seen in Section 4 of [46] which is given by

$$x_t \mid x_{t-1} \sim \mathcal{N}\left(x_t; \phi x_{t-1}, \sigma_v^2\right), \tag{86}$$

$$y_t \mid x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{87}$$

where $\theta = \{\phi, \sigma_v, \sigma_e\}$ are parameters with prior densities Normal$(0, 1)$, Gamma$(1, 1)$ and Gamma$(1, 1)$, respectively. The "optimal" proposal is used (see (12)) and can be derived from the properties of (86) and (87). This results in

$$q\left(x_t | x_{t-1}, y_t\right) = \mathcal{N}\left(x_t; \sigma^2\left[\sigma_e^{-2} y_t + \sigma_v^{-2} \phi x_{t-1}\right], \sigma^2\right), \tag{88}$$

with $\sigma^{-2} = \sigma_v^{-2} + \sigma_e^{-2}$.

The weights are updated using (13), which can be shown to be

$$w_{1:t}^{(\theta,i)} = \mathcal{N}\left(y_t; \phi x_t, \sigma_v^2 + \sigma_e^2\right) w_{1:t-1}^{(\theta,i)}. \tag{89}$$

8.1.1 Results

First, we compare the different differentiable particle filters described in section 6 in terms of computational run-time and the MSE between the true and inferred values of $\theta = \{\phi, \sigma_v\}$ in the LGSSM described in (86) and (87). The true values of ϕ and σ_v are 0.7 and 1.2 respectively. We use NUTS as the proposal and run with different numbers of particles, N, and observations, T, over $M = 50$ MCMC iterations. The results are presented in Table 1 with the MSE and the time taken in seconds being averages over 10 runs (with different random number seeds).

Using CRN and FI (Fisher's identity) consistently results in the lowest time taken to complete the different experiments. One reason for this is that minimal changes to the resampling step are introduced when compared to the other methods in Table 1. When $T = 25$ and $T = 100$, using CRN results in the lowest MSE when running with $N = 32$, 64 and 128 and $N = 16$, 32, 64, and 128, respectively. Using optimal transport resampling results in lower MSE estimates in some experiments but the computation time is considerably higher than CRN and FI.

Next, we compare the different proposals outlined in section 7. It has been explained previously that using NUTS eliminates the need to manually select the length parameter, L, in HMC by adaptively choosing this parameter at every iteration. Therefore it is likely that NUTS will make more than one target evaluation per iteration so is more computationally costly than MALA, where only one evaluation is made, and HMC with certain values for L. We therefore include the number of gradient evaluations as well as the average MSE between the true and inferred values of $\theta = \{\phi, \sigma_v, \sigma_e\}$ and the computation time in seconds for the LGSSM described in (86) and (87). The true values of ϕ, σ_v and σ_e are 0.7, 1.2 and 1, respectively. The setup of the experiment was as follows: $T = 100$, $M = 10$, $N = 512$ and 1024 and CRN resampling was used. Table 2 exemplifies the benefit of NUTS over HMC in not having to optimise the number of steps, L: NUTS obtains a lower MSE in a shorter run-time than HMC.

Finally, we show results when using NUTS and CRN resampling and present a commonly used diagnostic to determine if three independent chains that had different initial starting values for $\theta = \{\phi, \sigma_v, \sigma_e\}$ have converged. The particle filter was initialised with $N = 750$, $T = 250$ observations, $M = 500$ (with the first 100 discarded as burn-in) and the true values were $\theta = [0.7, 1.2, 1]$. The trace plots and density plots of the accepted samples of θ can be seen on the left and right of Figure 2 (a), respectively. These plots give an indication of how well the chains have converged to their stationary distribution. Figure 2(b)-(d) shows the 1-dimensional histograms, plotted using [47], for the same three chains and the uncertainties associated with these estimates of θ. The mean estimate of θ from the three chains is $[0.67, 1.23, 1.04]$. A numerical method for determining if multiple chains have converged is the Gelman-Rubin diagnostic [48] which compares the variances between chains. It is a commonly used diagnostic in the probabilistic programming language Stan [11] (where it

		T=25					**T=50**					**T=100**				
		CRN	SR	GS	OT	FI	CRN	SR	GS	OT	FI	CRN	SR	GS	OT	FI
N=16	MSE	0.151	1.446	0.179	0.141	**0.135**	0.075	0.438	0.076	**0.025**	**0.025**	**0.035**	0.056	0.045	0.040	0.047
	Time (sec)	593	730	554	2062	582	728	1423	765	6256	1590	939	2684	1018	4396	1005
N=32	MSE	**0.112**	0.899	0.174	0.124	0.121	0.077	0.297	0.078	**0.020**	0.021	**0.035**	0.050	0.045	**0.035**	0.036
	Time (sec)	558	816	587	1667	589	790	1478	784	6839	1339	941	2601	1583	5292	1014
N=64	MSE	**0.121**	1.091	0.165	**0.121**	0.126	0.074	0.452	0.084	0.069	**0.025**	**0.025**	0.396	0.050	0.044	0.049
	Time (sec)	552	747	852	1647	608	750	1481	1229	2303	1369	1952	2571	2959	5281	1022
N=128	MSE	**0.129**	1.576	0.135	0.142	0.162	0.078	0.318	0.074	0.082	**0.027**	**0.026**	0.825	0.043	0.044	0.044
	Time (sec)	583	880	1152	1827	582	738	1452	1423	2592	1383	1586	2621	3034	5651	1020

Table 1: Time in seconds and the average MSE of $\theta = \{\phi, \sigma_v\}$ in the LGSS model for different numbers of N, T and differentiable particle filters outlined in section 6. CRN = us, SR = soft resampling (see section 6.0.1), GS = gumbel-softmax (see section 6.0.2), OT = optimal transport (see section 6.0.3) and FI = Fisher's identity (see section 6.0.4). The results are an average over 10 runs using different random number seeds and NUTS was used as the proposal.

		N = 512			**N = 1024**		
		T (Secs)	*MSE*	*NGE*	*T (Secs)*	*MSE*	*NGE*
	MALA	4.342	0.306	9	6.288	0.300	9
HMC	L2	3.70	0.36±0.01	18	14.21	0.36±0.02	18
	L4	7.26	0.31±0.02	36	32.34	0.31±0.04	36
	L6	11.39	0.27±0.03	54	57.84	0.26±0.06	54
	L8	18.86	0.25±0.05	72	54.18	0.25±0.07	72
	L10	24.46	0.22±0.06	90	103.80	0.23±0.08	90
	NUTS	35.77	0.23±0.08	106	69.91	0.19±0.07	66

Table 2: The time taken in seconds, number of gradient evaluations (NGE) and MSE and standard deviation of $\theta = \{\phi, \sigma_v, \sigma_e\}$ in the LGSS model for different numbers of N and MCMC proposals. $T = 100$ observations, $M = 10$ MCMC iterations and CRN resampling was used. The results were averaged over 10 runs using different random number seeds.

is referred to as $\hat{R}$) to ascertain if the sampler has correctly sampled from the posterior. Stan's documentation states an $\hat{R}$ value below 1.05 passes their internal diagnostic check. The calculated Gelman-Rubin for parameters for ϕ, σ_v and σ_e where 1.0091, 1.007 and 1.0094, respectively.

8.2 Stochastic Volatility Model

Stochastic volatility (SV) models are widely used to evaluate financial securities and prices [49], as the variance of the latent process changes over time and is not constant. This is modelled with the state x_t being proportional to the observation noise. More specifically, we use the same model as seen in [46], which is defined as follows:

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{90}$$

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \mu + \phi\left(x_t - \mu\right), \sigma_v^2\right), \tag{91}$$

$$y_t \mid x_t \sim \mathcal{N}\left(y_t; 0, \exp\left(x_t\right)\right), \tag{92}$$

where $\theta = [\mu, \phi, \sigma_v]$ are parameters with prior densities Normal$(0,1)$, Normal$(0,1)$ and Gamma$(2,10)$, respectively. The log-returns (observations), y_t, are modelled using the formula,

$$y_t = 100 \log\left[\frac{s_t}{s_{t-1}}\right] = 100\left[\log\left(s_t\right) - \log\left(s_{t-1}\right)\right] \tag{93}$$

where s_t is the price. We note that the data used here is real data and is the daily closing prices of the NASDAQ OMXS30 index, i.e., a weighted average of the 30 most traded stocks at the Stockholm stock exchange. The data is extracted from Quandl[3] for the period between January 2, 2012 and January 2, 2014.

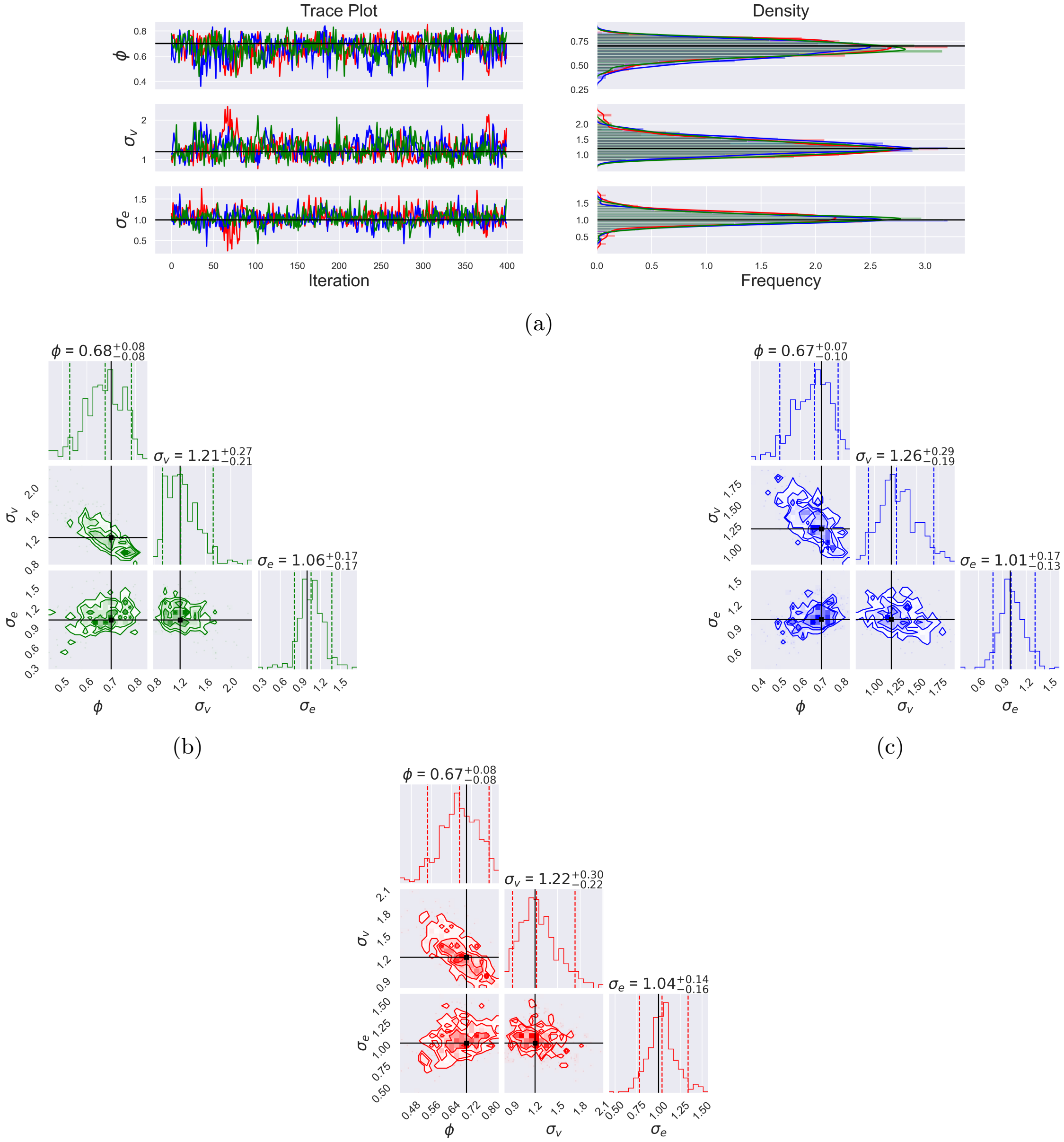
We use the prior as the proposal (see (10)), which is such that the incremental log weight becomes

$$\log \sigma\left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta\right) = \log p\left(y_t | x_t^{(\theta,i)}\right), \tag{94}$$

and the associated gradient is

$$\frac{d}{d\theta} \log \sigma\left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta\right) = \frac{d}{d\theta} \log p\left(y_t \mid x_{t-1}^{(\theta,i)}\right). \tag{95}$$

[3] The data can be downloaded from `https://data.nasdaq.com/data/NASDAQOMX/OMXS30`



(a)

(b)

(c)

(d)

Figure 2: (a) The trace plots (left) and density estimates (right) of three independent chains with different initial starting positions for θ. The horizontal black lines indicate the true values. (b)-(d) 1-Dimensional corner plots of the three independent chains seen in Figure 2 for the LGSS model. The black lines indicate the true values.

8.2.1 Results

It is here we compare the different proposals outlined in section 7 when inferring the parameters of the stochastic volatility model outlined in (90) - (92). Each simulation was initialised with the same values of θ, $N = 5000$, $T = 500$ observations, $M = 5000$ MCMC iterations with the first 2000 discarded as burn-in and CRN resampling.

The first and second rows of figure 3 shows the histograms and trace plots of the accepted values of σ_v, respectively. These plots give a visual indication of how well each of the samplers perform but should not be solely used to asses convergence. Table 3 outlines a number of MCMC diagnostics that determine if the sampler has converged to equilibrium. They include the mean of the posterior samples which on its own is not very informative, especially if the parameter being inferred is not known.

By looking at the trace plots, it is evident that for MALA and some values of L in HMC, there is a lot of serial correlation between consecutive draws. This results in the parameter space being poorly explored. However using NUTS can be seen to have the least serial correlation between MCMC draws. This observation is backed up by looking at the third row, which shows auto-correlation function (ACF) plots for the same simulations. These plots show the auto-correlation for a Markov chain up to a user-specified number of lags, which in this case is chosen to be 100. An ideal ACF plot is large at short lags but quickly drops towards zero. For MALA and a number of the HMC simulations the ACF plots do not reach 0 within the specified 100 lags. The Integrated Auto-Correlation Time (IACT) is a measure of the area under the ACF plot. The aim is to minimise this value since it gives an indication of the mixing within the Markov chain: IACT estimates the number of iterations it takes to draw an independent sample. It is evident by looking at Table 3 that using NUTS results in lower IACT scores for parameters μ and σ_v and is comparable with HMC with $L = 6$ for ϕ.

The effective sample size (ESS) is also shown in Table 3. This gives an indication of the number of independent samples it would take to have the same estimation power as a set of auto-correlated samples: larger values for ESS are to be preferred. Much like the IACT scores for parameters μ and σ_v, NUTS provides better results than the other samplers and is slightly worse than HMC with $L = 6$ for ϕ.

As explained previously, randomising the L parameter within HMC can avoid periodicities in the underlying Hamiltonian dynamics. To do this we draw an L value from an exponential distribution with a mean parameter of 2.5. To ensure this value is an integer we round up. It is evident when looking at Table 3, randomising the L parameter in HMC provides better results for μ when compared with HMC with fixed L but is still worse than NUTS.

The mean estimates of θ differ slightly in each simulation to those presented in [46], which were $[-0.23, 0.97, 0.15]$. We believe this disparity stems from [46] using the same particle filter but a M-H random walk proposal for the parameters. However, when a reparameterised model (described on page 29 of [46]) is used, they obtain estimates equal to $[-0.16, 0.96, 0.17]$, which are very similar to those seen in Table 3 when using NUTS, $[-0.17, 0.96, 0.18]$.

8.3 Lorenz-63

The Lorenz-63 model is a 3-dimensional (N_x) dynamical system widely used in data assimilation [50] that uses ordinary differential equations to propagate a state. The model has a state variable which

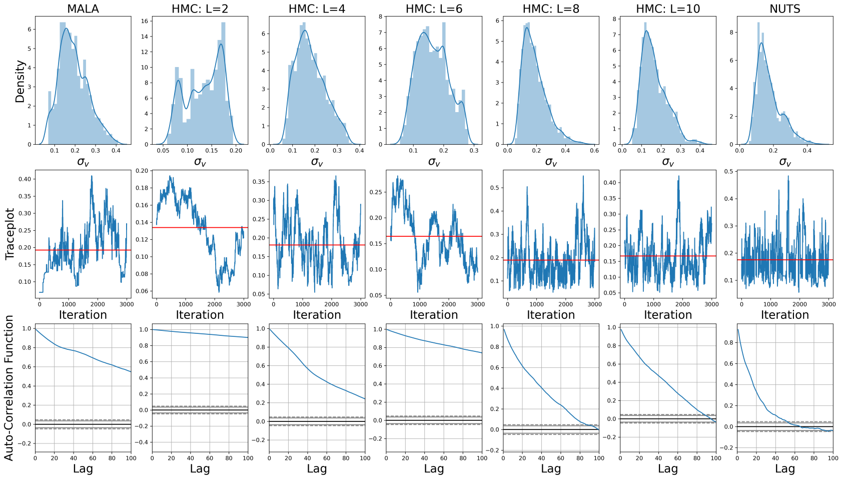


Figure 3: Columns: Simulations using MALA, HMC with different L, RHMC and NUTS. First row: Histograms of posterior estimate of σ_v. Second row: Trace plots of σ_v and the red horizontal line is the estimated mean. Third row: ACF plots for σ_v with lag = 100.

		MALA	**HMC**										**RHMC**	**NUTS**
			L1	L2	L3	L4	L5	L6	L7	L8	L9	L10		
Mean	μ	-0.27	-0.12	0.01	-0.34	-0.01	-0.16	-0.20	-0.24	-0.16	-0.30	-0.17	-0.16	-0.17
	ϕ	0.95	0.97	0.94	0.96	0.97	0.96	0.96	0.95	0.95	0.96	0.93	0.96	0.96
	σ_v	0.19	0.13	0.23	0.18	0.14	0.16	0.18	0.19	0.2	0.17	0.23	0.19	0.18
IACT	μ	131	173	190	182	164	192	154	125	163	167	174	101	66
	ϕ	125	158	175	100	98	139	35	68	79	73	144	63	36
	σ_v	148	191	181	111	137	173	44	74	89	81	146	66	35
ESS	μ	15	5	1	2	8	1	12	16	4	11	1	30	50
	ϕ	3	2	2	25	15	8	94	59	37	42	18	75	89
	σ_v	2	1	2	26	6	6	71	50	34	44	16	55	102
	Acc. rate	0.35	0.73	0.81	0.73	0.87	0.83	0.70	0.66	0.76	0.68	0.70	0.53	0.86

Table 3: IACT, ESS and mean estimates of θ in the SV model and the acceptance probability of the different algorithms. The same value of ϵ was used for each simulation of HMC, RHMC and NUTS, $N = 1000$, $M = 5000$ and CRN resampling was used.

is propagated with:

$$\frac{dx}{dt} = \sigma(y - x), \frac{dy}{dt} = x(r - z) - y, \frac{dz}{dt} = xy - bz, \tag{96}$$

where $\sigma = 10, r = 28, b = 8/3$, which represent the Prandtl number, the Rayleigh number and the physical dimensions of the layer respectively. This model is commonly used in meteorology and oceanography due to its non-linear chaotic behaviour. The assimilation cycle length is $dt = 0.05$ time units in every experiment. This corresponds to 50 integration time steps which are performed with a 4th-order Runge-Kutta algorithm.

		HMC					NUTS
		L2	L4	L6	L8	L10	
CRN	MSE	0.42	0.40	0.19	0.17	0.18	0.12
	Time (s)	98	360	1458	4860	5250	1560
GS	MSE	0.36	0.51	0.59	0.59	0.21	0.21
	Time (s)	83	644	1900	6600	12900	1255
SR	MSE	0.39	0.33	1.2	3.5	0.67	0.20
	Time (s)	119	617	2141	9180	52200	869

Table 4: Mean square errors (for the $\sigma_{\mathcal{Q}}$ parameter) and computational times for different MCMC proposals and resampling schemes. The same value of ϵ was used for each simulation of HMC and NUTS, $N = 500$, $M = 10$, $T = 50$

We are interested in the following model:

$$x_t \mid x_{t-1} \sim \mathcal{N}(x_t; \mathcal{M}(x_{t-1}), \mathcal{Q}) \tag{97}$$

$$y_t \mid x_t \sim \mathcal{N}(y_t; \mathcal{H}x_t, \mathcal{R}) \tag{98}$$

where $\mathcal{M}$ is the Lorenz propagation and $\mathcal{H}$ is the observational model. The parameters in the model and their prior densities are $\theta = \{\mathcal{Q}, \mathcal{R}\}$ and Normal$(0, 1)$ and Normal$(0, 1)$, respectively. The observational model, $\mathcal{H}$, is a $N_y \times N_x$ matrix where N_y represents the dimension of the observations with $N_y < N_x$. The motivation for having a partially observable model is to evaluate our proposed method in a realistic environment. We use the prior as the proposal as described in (10). The true values of the parameters are as follows: $N_x = 3$, $N_x = 2$, $\mathcal{Q} = \sigma_{\mathcal{Q}}\mathbb{I}_{N_x}$ and $\mathcal{R} = \sigma_{\mathcal{R}}\mathbb{I}_{N_y}$ where $\mathbb{I}_{N_x}$ and $\mathbb{I}_{N_y}$ are $N_x \times N_x$ and $N_y \times N_y$ identity matrices, respectively. $\sigma_{\mathcal{Q}}$ and $\sigma_{\mathcal{R}}$ are both set to 1.2.

8.3.1 Results

For this example we compare HMC with different L and NUTS when using the different differentiable particle filters described in section 6 in terms of the computational run-time and the MSE between the true and inferred values of $\theta = \sigma_{\mathcal{Q}}$. Note we do not include results for MALA and RHMC proposals because we make use of the HMC and NUTS implementations provided by PyTorch [41].

Each simulation was initialised with the same values of θ, $N = 500$, $T = 50$ observations, $M = 10$ MCMC iterations using CRN, Gumbel-Softmax resampling (GS) and Soft-resampling (SR). Note we do not include results for OT resampling due to the computation time being excessive. We also note that running the experiments on a GPU did not significantly reduce the computation time.

It is evident when looking at Table 4 that using NUTS and CRN resampling yields the lowest MSE. One explanation for Gumbel-softmax and soft-resampling giving worse results than CRN is because these functions are approximations of multinomial-resampling which is unbiased. We note that fine tuning α and λ in (70) and (71), respectively may result in more accurate estimates of θ.

9 Conclusion

We have outlined how to extend the *reparameterisation trick* to use common random numbers when performing the resampling step in a particle filter. This limits the discontinuities encountered when calculating gradients that are used in HMC and NUTS to propose new parameters within p-MCMC.

We have applied these algorithms to three problems and show that using NUTS in this context can improve the mixing of the Markov chain compared to using MALA or HMC. We also compare different methods for resampling and show that using CRN resampling can produce more accurate estimates in shorter run time.

Although we have only included the methods for estimating the derivatives of Gaussian models, we note that our method could be applied to other models as long as the derivatives can be calculated. Considering a variety of different models would be a sensible direction for future work.

In this piece of work we have not included any analysis using Hessian information about the log-posterior within MCMC proposals, as was considered in [21]: This is due to the generic complexity of having to compute the second-order partial and full derivatives of the equations seen in sections 3 and 4. An estimate of the state-dependent Hessian matrix could be made, using the gradients estimated in (80), via the Gaussian Process construction provided in [51]. Hence, an avenue for future work is to include the resulting Hessian matrix in the proposal (85), along the lines of what was done in [21], or as the mass matrix within NUTS. Doing so could provide additional performance gains over those reported herein.

An interesting direction for future work would involve a broader comparison of algorithms, that differ from p-MCMC, which can be applied to parameter estimation in SSMs. These include but are not limited to SMC^2 [52], Nudging the Particle Filter [53] and the Nested Particle Filter [54].

The nested particle filter and SMC^2 have two layers of SMC method: one (an SMC sampler with N_θ particles) estimates the pdf over the static parameters, θ, and the other (a particle filter with N_x particles) considers the dynamic states. The difference between the two methods is that the nested particle filter runs in a purely recursive manner. A detailed comparison of the nested particle filter and SMC^2 can be seen in [54] but the computational complexity of both methods is $O(N_\theta N_x T)$, just like the methods described in this paper (assuming we ran for N_θ MCMC steps). Future work would sensibly include a comparison with these alternative methods.

In [53] particles are *nudged* towards regions of the state space where the likelihood is deemed to be high. They also apply this method to the particle Metropolis-Hastings algorithm [2] and outline how gradient nudging steps can be used within the framework of differentiable likelihoods and automatic differentiation libraries. This would be applicable to the methods proposed in this work with the potential to offer improvements in performance.

We also note that there is a trade-off between the theoretical concerns related to convergence and the empirical performance achieved. An avenue for future work would be to derive proofs related to the regularity properties of the estimated derivatives considered in this paper.

Funding

This work was supported by a Research Studentship jointly funded by the EPSRC and the ESRC Centre for Doctoral Training on Quantification and Management of Risk and Uncertainty in Complex Systems Environments [EP/L015927/1], the EPSRC through the Big Hypotheses grant [EP/R018537/1], the project *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079), funded by the Swedish Research Council and by the *Kjell och Märta Beijer Foundation*, and a Research Studentship jointly funded by the EPSRC Centre for Doctoral Training in Distributed Algorithms [EP/S023445/1] and the UK Government.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments which helped improve this paper.

References

[1] A. Doucet, N. De Freitas, N. J. Gordon, *et al.*, *Sequential Monte Carlo methods in practice*, vol. 1. Springer, 2001.

[2] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[3] C. P. Robert and G. Casella, "The metropolis—hastings algorithm," in *Monte Carlo Statistical Methods*, pp. 231–283, Springer, 1999.

[4] D. Van Ravenzwaaij, P. Cassey, and S. D. Brown, "A simple introduction to markov chain monte–carlo sampling," *Psychonomic bulletin & review*, vol. 25, no. 1, pp. 143–154, 2018.

[5] G. O. Roberts and J. S. Rosenthal, "General state space markov chains and mcmc algorithms," *Probability surveys*, vol. 1, pp. 20–71, 2004.

[6] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F-radar and signal processing*, vol. 140, pp. 107–113, IET, 1993.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[8] A. Doucet, A. M. Johansen, *et al.*, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[9] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[10] M. D. Hoffman, A. Gelman, *et al.*, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo.," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[11] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A probabilistic programming language," *Journal of statistical software*, vol. 76, no. 1, pp. 1–32, 2017.

[12] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[13] D. Lautier, A. Javaheri, and A. Galli, "Filtering in finance," 2003.

[14] W. Yang, A. Karspeck, and J. Shaman, "Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics," *PLoS computational biology*, vol. 10, no. 4, p. e1003583, 2014.

[15] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull, "Multiple object tracking using particle filters," in *2006 IEEE Aerospace Conference*, pp. 8–pp, IEEE, 2006.

[16] G. Poyiadjis, A. Doucet, and S. S. Singh, "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation," *Biometrika*, vol. 98, no. 1, pp. 65–80, 2011.

[17] P. Del Moral, A. Doucet, and S. S. Singh, "Uniform stability of a particle approximation of the optimal filter derivative," *SIAM Journal on Control and Optimization*, vol. 53, no. 3, pp. 1278–1304, 2015.

[18] C. Nemeth, P. Fearnhead, and L. Mihaylova, "Particle approximations of the score and observed information matrix for parameter estimation in state–space models with linear computational cost," *Journal of Computational and Graphical Statistics*, vol. 25, no. 4, pp. 1138–1157, 2016.

[19] C. Nemeth and P. Fearnhead, "Particle metropolis adjusted langevin algorithms for state space models," *arxiv. org*, 2014.

[20] J. Dahlin, F. Lindsten, and T. B. Schön, "Particle metropolis hastings using langevin dynamics," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6308–6312, IEEE, 2013.

[21] J. Dahlin, F. Lindsten, and T. B. Schön, "Second-order particle mcmc for bayesian parameter inference," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 8656–8661, 2014.

[22] J. Dahlin, F. Lindsten, and T. B. Schön, "Particle metropolis–hastings using gradient and hessian information," *Statistics and computing*, vol. 25, no. 1, pp. 81–92, 2015.

[23] M. Girolami and B. Calderhead, "Riemann manifold langevin and hamiltonian monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 2, pp. 123–214, 2011.

[24] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5101–5108, 2020.

[25] H. Wen, X. Chen, G. Papagiannis, C. Hu, and Y. Li, "End-to-end semi-supervised learning for differentiable particle filters," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5825–5831, IEEE, 2021.

[26] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," *arXiv preprint arXiv:1805.11122*, 2018.

[27] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[28] A. Lee, *Towards smooth particle filters for likelihood estimation with multivariate latent variables.* PhD thesis, University of British Columbia, 2008.

[29] C. Snyder, "Particle filters, the "optimal" proposal and high-dimensional systems," in *Proceedings of the ECMWF Seminar on Data Assimilation for atmosphere and ocean*, pp. 1–10, 2011.

[30] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," *Advances in neural information processing systems*, vol. 13, pp. 584–590, 2000.

[31] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[32] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *2006 IEEE nonlinear statistical signal processing workshop*, pp. 79–82, IEEE, 2006.

[33] J. Elfring, E. Torta, and R. van de Molengraft, "Particle filters: A hands-on tutorial," *Sensors*, vol. 21, no. 2, p. 438, 2021.

[34] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Conference on robot learning*, pp. 169–178, PMLR, 2018.

[35] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, vol. 33. US Government Printing Office, 1954.

[36] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[37] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[38] A. Corenflos, J. Thornton, G. Deligiannidis, and A. Doucet, "Differentiable particle filtering via entropy-regularized optimal transport," in *International Conference on Machine Learning*, pp. 2100–2111, PMLR, 2021.

[39] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[40] A. Ścibior and F. Wood, "Differentiable particle filtering without modifying the forward pass," *arXiv preprint arXiv:2106.10314*, 2021.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[42] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[43] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," *arXiv preprint arXiv:1701.02434*, 2017.

[44] N. Bou-Rabee and J. M. Sanz-Serna, "Geometric integrators and the hamiltonian monte carlo method," *Acta Numerica*, vol. 27, pp. 113–206, 2018.

[45] N. Bou-Rabee and J. M. Sanz-Serna, “Randomized hamiltonian monte carlo,” *The Annals of Applied Probability*, vol. 27, no. 4, pp. 2159–2194, 2017.

[46] J. Dahlin and T. B. Schön, “Getting started with particle metropolis-hastings for inference in nonlinear dynamical models,” *Journal of Statistical Software*, vol. 88, no. 1, pp. 1–41, 2019.

[47] D. Foreman-Mackey, “corner. py: Corner plots,” *Astrophysics Source Code Library*, pp. ascl–1702, 2017.

[48] A. Gelman and D. B. Rubin, “Inference from iterative simulation using multiple sequences,” *Statistical science*, vol. 7, no. 4, pp. 457–472, 1992.

[49] J. Hull and A. White, “The pricing of options on assets with stochastic volatilities,” *The journal of finance*, vol. 42, no. 2, pp. 281–300, 1987.

[50] T. J. Cocucci, M. Pulido, M. Lucini, and P. Tandeo, “Model error covariance estimation in particle and ensemble kalman filters using an online expectation–maximization algorithm,” *Quarterly Journal of the Royal Meteorological Society*, vol. 147, no. 734, pp. 526–543, 2021.

[51] A. G. Wills and T. B. Schön, “Stochastic quasi-newton with line-search regularisation,” *Automatica*, vol. 127, p. 109503, 2021.

[52] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos, “Smc2: an efficient algorithm for sequential analysis of state space models,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 397–426, 2013.

[53] Ö. D. Akyildiz and J. Míguez, “Nudging the particle filter,” *Statistics and Computing*, vol. 30, no. 2, pp. 305–330, 2020.

[54] D. Crisan and J. Miguez, “Nested particle filters for online parameter estimation in discrete-time state-space markov models,” *Bernoulli*, vol. 24, no. 4A, pp. 3039–3086, 2018.

Appendices

A Partial versus total derivatives

To try to avoid confusion, we use the partial derivative $\partial/\partial\theta$ to mean the derivative by only changing that function argument, and the total derivative $d/d\theta$ to mean also changing the other arguments depending on it, i.e. if θ is a scalar,

$$\frac{d}{d\theta}f(a(\theta),\theta) \triangleq \lim_{h\to 0}\frac{f(a(\theta+h),\theta+h)-f(a(\theta),\theta)}{h} \tag{99}$$

$$\frac{\partial}{\partial\theta}f(a(\theta),\theta) \triangleq \lim_{h\to 0}\frac{f(a(\theta),\theta+h)-f(a(\theta),\theta)}{h}. \tag{100}$$

B Differentiating a Kalman Filter

We have a transition kernel and a measurement model as follows, where θ is a parameter vector:

$$p(x'|x,\theta) = \mathcal{N}(x'; a(x,\theta), Q(x,\theta)) \tag{101}$$

$$p(y|x',\theta) = \mathcal{N}(y; h(x',\theta), R(x',\theta)). \tag{102}$$

Applying an Extended Kalman Filter gives a proposal for x' of the form

$$q(x'|x,\theta,y) = \mathcal{N}(x'; \mu(x,\theta,y), C(x,\theta,y)). \tag{103}$$

We wish to calculate the derivatives$\frac{\partial\mu}{\partial x}, \frac{\partial\mu}{\partial\theta}, \frac{\partial C}{\partial x}, \frac{\partial C}{\partial\theta}$. The standard Kalman filter equations are

$$\begin{aligned} S(x,\theta) &= H(a(x,\theta),\theta)Q(x,\theta)H(a(x,\theta),\theta)^T \\ &\quad + R(a(x,\theta),\theta) \end{aligned} \tag{104}$$

$$K(x,\theta) = Q(x,\theta)H(a(x,\theta),\theta)^T S(x,\theta)^{-1} \tag{105}$$

$$\mu(x,\theta,y) = a(x,\theta) + K(x,\theta)(y - h(a(x,\theta),\theta)) \tag{106}$$

$$C(x,\theta) = Q(x,\theta) - K(x,\theta)H(a(x,\theta),\theta)Q(x,\theta) \tag{107}$$

where

$$H(a,\theta) = \left(\frac{\partial h_i}{\partial a_j}(a,\theta)\right)_{ij} \tag{108}$$

is the Jacobian of the measurement function evaluated at the prior mean. We would like to differentiate these with respect to x and θ but the measurement model is defined in terms of the prior mean $a(x)$. Let

$$\tilde{h}(x,\theta) = h(a(x,\theta),\theta) \tag{109}$$

$$\tilde{H}(x,\theta) = H(a(x,\theta),\theta) \tag{110}$$

$$\tilde{R}(x,\theta) = R(a(x,\theta),\theta). \tag{111}$$

Then

$$S(x,\theta) = \tilde{H}(x,\theta)Q(x,\theta)\tilde{H}(x,\theta)^T + \tilde{R}(x,\theta) \tag{112}$$

$$K(x,\theta) = Q(x,\theta)\tilde{H}(x,\theta)^T S(x,\theta)^{-1} \tag{113}$$

$$\mu(x,\theta,y) = a(x,\theta) + K(x,\theta)(y - \tilde{h}(x,\theta))) \tag{114}$$

$$C(x,\theta) = Q(x,\theta) - K(x,\theta)\tilde{H}(x,\theta)Q(x,\theta). \tag{115}$$

To compute the derivatives of these from the derivatives in a, applying the chain rule gives

$$\frac{\partial \tilde{h}}{\partial x}(x,\theta) = H(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{116}$$

$$\frac{\partial \tilde{h}}{\partial \theta}(x,\theta) = H(a(x,\theta),\theta)\frac{\partial a}{\partial \theta}(x,\theta) + \frac{\partial h}{\partial \theta}(a(x,\theta),\theta) \tag{117}$$

$$\frac{\partial \tilde{R}}{\partial x}(x,\theta) = \frac{\partial R}{\partial a}(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{118}$$

$$\frac{\partial \tilde{R}}{\partial \theta}(x,\theta) = \frac{\partial R}{\partial a}(a(x,\theta),\theta)\frac{\partial a}{\partial \theta}(x,\theta) + \frac{\partial R}{\partial \theta}(a(x,\theta),\theta) \tag{119}$$

$$\frac{\partial \tilde{H}}{\partial x}(x,\theta) = \frac{\partial^2 h}{\partial a^2}(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{120}$$

$$\frac{\partial \tilde{H}}{\partial \theta}(x,\theta) = \frac{\partial^2 h}{\partial a^2}(a(x,\theta),\theta)\frac{\partial a}{\partial \theta}(x,\theta) + \frac{\partial^2 h}{\partial a \partial \theta}(a(x,\theta),\theta) \tag{121}$$

Hence to evaluate the derivatives of (114) and (115), we need

$$a(x,\theta), \frac{\partial a}{\partial x}, \frac{\partial a}{\partial \theta}, Q(x,\theta), \frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial \theta} \tag{122}$$

from the transition model and

$$h(a,\theta), \frac{\partial h}{\partial a}, \frac{\partial h}{\partial \theta}, \frac{\partial^2 h}{\partial a^2}, \frac{\partial^2 h}{\partial a \partial \theta}, R(a(x),\theta), \frac{\partial R}{\partial a}, \frac{\partial R}{\partial \theta} \tag{123}$$

from the measurement model. From this we apply the product rule and the inverse derivative in Appendix D.1.

C Derivatives of multivariate log normal

If

$$\mathcal{N}(x;\mu,C) = \frac{\exp\left(-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)\right)}{\sqrt{|2\pi C|}} \tag{124}$$

then

$$\frac{\partial}{\partial x}\log \mathcal{N} = -C^{-1}(x-\mu) \tag{125}$$

$$\frac{\partial}{\partial \mu}\log \mathcal{N} = C^{-1}(x-\mu) \tag{126}$$

$$\frac{\partial}{\partial C}\log \mathcal{N} = -\frac{1}{2}\left(C^{-1} - C^{-1}(x-\mu)(x-\mu)^T C^{-1}\right). \tag{127}$$

D Matrix derivatives

D.1 Derivative of a matrix inverse

Suppose U is an $N \times N$ invertible matrix with $N \times N$ derivative with respect to θ_r given by $dU/d\theta_r$. Then

$$\frac{\partial(U^{-1})}{\partial\theta_r} = -U^{-1}\left(\frac{\partial U}{\partial\theta_r}\right)U^{-1}. \tag{128}$$

If θ is an R-dimensional vector, $d(U^{-1})/d\theta$ is an $N \times N \times R$ tensor with slice r given by (128).

D.2 Derivative of a matrix square root

Suppose that A is the matrix square root of C, i.e.

$$C = AA. \tag{129}$$

Applying the product rule gives

$$\frac{\partial C}{\partial\theta} = A\frac{\partial A}{\partial\theta} + A\frac{\partial A}{\partial\theta} \tag{130}$$

hence

$$\frac{\partial A}{\partial\theta} = \frac{1}{2}A^{-1}\frac{\partial C}{\partial\theta}. \tag{131}$$