Differentiable Plug-And-Play Particle Filtering

Kevin Tan

Thesis Advisor
Professor Edward Ionides

Department of Statistics, University of Michigan

Abstract:
Many approaches to inference in highly nonlinear stochastic dynamical systems assume access to the probability density of next states given the current state. This is a problem in some critical applications, such as disease modeling, where the models are complex enough that obtaining the density is intractable. However, the particle filter, a popular method for solving the filtering problem in partially-observed dynamical systems, does not require evaluation of the transition density of the latent Markov process, enabling an arbitrary model simulator to be plugged into the algorithm in a feature known as the plug-and-play property. Still, maximum likelihood parameter estimation can be challenging, especially when the Monte Carlo variance of the evaluation is high and the number of parameters is not small. Algorithmic differentiation potentially facilitates numerical optimization, but currently its use for particle filters is limited. We investigate ways to use algorithmic differentiation of particle filters within the confines of the plug-and-play property, with the goal of enhancing current inference capabilities for general POMP models.

Keywords and phrases: Hidden Markov model, sequential Monte Carlo, state space model.

1. Introduction

The particle filter, or sequential Monte Carlo (SMC), is a popular method for solving the filtering problem in partially-observed Markov processes (POMPs), obtaining the most likely sequence of hidden states and an approximate likelihood. It possesses several desirable properties, such as runtime linear in the number of particles J and timesteps T, the lack of a requirement of access to the transition density (which may in general be unavailable), and a central limit theorem [6].

The issue: Particle filters provide convenient approaches to evaluating the log-likelihood function for partially observed Markov process (POMP) models. However, using this evaluator to obtain a maximum likelihood parameter estimate can be challenging – especially when the Monte Carlo variance of the evaluation is high and the number of parameters is not small. Empirically, methods such as the improved iterated filtering algorithm (IF2) from Ionides et. al. [11] rapidly converge to a neighborhood of the optimum but struggle at finding the exact optimum due to Monte Carlo variance, even with an annealing random walk standard deviation.

A potential solution to this could lie in auto-differentiation (AD). This would allow for the use of first and second-order iterative optimization techniques. However, though AD could potentially facilitate numerical optimization, its use for plug-and-play particle filters has so far been limited. This is potentially because particle filtering methods are inherently non-differentiable due to the resampling step that may take place in between iterations.

The importance of the plug-and-play property: Performing inference in highly nonlinear stochastic dynamical systems is a challenging problem. Although many methods for inference assume access to the density of state transitions, this is often not available, especially in critical applications like epidemiology.

Basic particle filtering algorithms do not require evaluation of the transition density of the latent Markov process, in a feature known as the **plug-and-play property** [3] since it enables an arbitrary model simulator to be plugged into the algorithm. We investigate ways to use algorithmic differentiation of particle filters within the confines of the plug-and-play property, with the goal of enhancing current inference capabilities for general POMP models.

Other potential applications: This has applications beyond the obvious one of learning model parameters via first or second-order optimization routines. For example, it could be a step towards developing very general Hamiltonian Monte Carlo methods for particle MCMC, as Rosato et. al. [22] do by using previous work such as [27, 19] (and we conjecture that the seed-fixing derivatives of Rosato et. al. are the same as these as an immediate consequence of section 3.5) to differentiate the particle filter.

1.1. Previous Literature

Previous research has attempted to make the particle filter differentiable. Poyiadjis et. al. [19] began the discussion on estimating the gradient of the log-likelihood (score) through a recursive expression relying on the Fisher identity, and show that this has variance quadratic in horizon. Jonschkowski et. al. [12] provide a method where the gradients do not flow through the resampling steps. Naesseth et. al. [18] propose an extension to variational inference.

Corenflos et. al. [7] use the Sinkhorn algorithm (which is differentiable) to approximately solve an optimal transport problem in $O(J^2)$ time, where J is the number of particles, between the particles before and after resampling. This amounts to modifying the particle filter itself (in the forward pass) in order to obtain a derivative of a relaxation of the resampling obtained by optimal transport.

Scibior and Wood [27] show how one can obtain the unbiased estimates relying on the Fisher identity obtained by Poyiadjis et. al. [19] with AD. Their main contribution lies in showing how these can be recovered by auto-differentiation without having to modify the forward pass, or the computation of the likelihood and posterior approximation, itself.

Singh et. al. [23] estimate the gradient of the log-likelihood online for the purposes of autonomous driving, with the caveat that the process model has to decompose into a policy that takes actions conditional on states and a deterministic, differntiable and injective motion model.

Many useful theoretical results have been proved for the particle filter. Chopin [6] provides a central limit theorem for particle filters that holds for multinomial and residual resampling, while Del Moral and Rio [17] develop concentration inequalities for the concentration of measure of functions of the prediction particles to the expectation of the function under the prediction distribution.

1.2. Our Contributions

In this paper, we (1) develop a new framework for reasoning about Monte Carlo estimates of the likelihood and their derivatives, (2) propose methods for numerical optimization of the log-likelihood in a POMP model that take advantage of AD for PF, (3) provide theoretical guarantees for the above, and (4) showcase numerical experiments that validate the theory and methods.

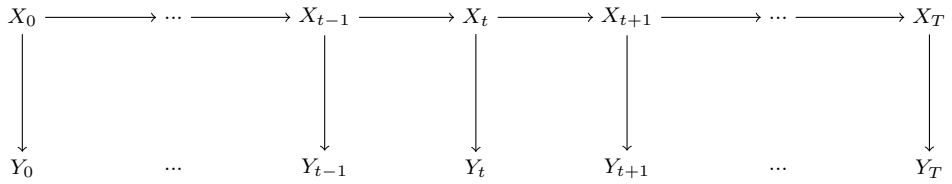

FIG 1. Directed acyclic graph (DAG) of a POMP model.

2. Background

2.1. Partially-Observed Markov Processes

Let (Ω, Σ, μ) be a probability space. Consider random functions $X(\omega)$ and $Y(\omega)$, $\omega \in \Omega$, taking values in a sequence of measurable spaces $(E_t)_{0 \le t \le T}$ each equipped with a sigma-field $\mathcal{F}_t$. Let $\mathcal{P}(E_t)$ be the set of probability measures on E_t. Write for each E_t, $E_t = \mathcal{X}_t \times \mathcal{Y}_t$, where we call $\mathcal{X}_t$ the state space and $\mathcal{Y}_t$ the observation space.

Let $\Theta \subseteq \mathbb{R}^p$. We specify that $X(\omega)$ is a Markov process $\{X_t(\omega) : 0 \le t \le T\}$ on $(\mathcal{X}_t)_{0 \le t \le T}$, governed by transitions $p(x_{t+1}|x_t, \theta)$ for some $\theta \in \Theta$. We also specify that $Y(\omega)$ is another random process $\{Y_t(\omega) : 0 \le t \le T\}$ on $(\mathcal{Y}_t)_{0 \le t \le T}$, where each $Y_t(\omega)$ only depends on $X_t(\omega)$ according to the density $p(y_t|x_t, \theta)$. Observe that each Y_t is conditionally independent from Y_s, $s \ne t$, given each $X_0, ..., X_T$, that X_t only depends on X_{t-1}, and that Y_t depends only on X_t.

We will call $p(x_{t+1}|x_t, \theta)$ the process model, process(x_t, θ) the corresponding simulator that produces samples of states at the next timestep, $p(y_t|x_t, \theta)$ the measurement model, and will write y_t^* for the actual values of the observations that were observed.

The above defines a partially-observed Markov process (POMP).

2.2. The Weighted Particle Filter

Particle filters are a popular way to perform inference in partially-observed Markov processes. At a high level, a particle filter keeps track of a set of particles $x_{t,j}$, sequentially approximating (1) the distribution of $X_n|y^*_{1:n-1}$ with a mixture Dirac measure $\frac{1}{J}\sum_{j=1}^{J} \tilde{w}_{t,j}\delta_{x^P_{t,j}}$ via simulation from the process model $p(x_t|x_{t-1})$ and (2) the distribution of $X_n|y^*_{1:n}$ with another mixture Dirac measure $\frac{1}{J}\sum_{j=1}^{J} \bar{w}_{t,j}\delta_{x^P_{t,j}}$ via a reweighting of the particles in proportion to their likelihood.

The particle filter approximates the Bayes filter through these two sequences of mixture Dirac measures. The Bayes filter is the "optimal" filter that updates the prediction distribution $p(x_t^P|y^*_{1:t-1})$ with the prediction formula [13]

$$p(x_t^P|y^*_{1:t-1}) = \int p(x_{t-1}^F|y^*_{1:t-1})p(x_t|x_{t-1})dx_{t-1} \tag{1}$$

and the filtering distribution $p(x_t^F|y^*_{1:t})$ via Bayes' theorem [13],

$$p(x_t^F|y^*_{1:t}) = \frac{p(y_t^*|x_t)p(x_t|y^*_{1:t-1})}{\int p(y_t^*|x_t)p(x_t|y^*_{1:t-1})dx_t}. \tag{2}$$

Particle filters are plug-and-play, requiring only a simulator and a measurement model to approximate the posterior distribution of the sequence of hidden states given a set of observations. Here are a few problems that they can solve:

1. Forecasting, i.e. finding the distribution of $X_n|y^*_{1:n-1}$, which we call the prediction distribution $p(x^P_t|y^*_{1:t-1})$
2. Filtering, i.e. finding the distribution of $X_n|y^*_{1:n}$, $p(x^F_t|y^*_{1:t})$, which we call the filtering distribution
3. Smoothing, i.e. finding the distribution of $X_t|y^*_{1:T}$, which we call the smoothing distribution $p(x_t|y^*_{1:T})$
4. Finding the likelihood $\mathcal{L}(\theta) = p(y^*_{1:T}|\theta)$

where this list is non-exhaustive.

In Algorithm 1, we provide pseudocode for a relatively general particle filter that keeps track of the importance weight of each particle at every timestep, accumulating these weights at every timestep. It is useful to note that the vanilla particle filter or bootstrap filter is a special case of the weighted particle filter where resampling takes place at every step and every post-resampling particle has weight $1/J$.

Algorithm 1: Weighted Particle Filter

Input: Number of particles J, timesteps T, measurement model $p(y^*_t|x_t, \theta)$, simulator process(x_t, θ) parameters θ
Initialize particles $x^F_{0,j}$, weights $w_{t,j} = 1/J$, normalized weights $\bar{w}_{t,j} = 1/J$
for $t = 1, ..., T$ **do**
    Get prediction particles $x^P_{t,j} \sim$ process$(x^F_{t-1,j}, \theta)$
    if resampling condition fulfilled: **then**
        Draw indices $k_j \sim p(y^*_t|x^P_{t,j}, \theta)$
        Assign filtering particles $x^F_{t,j} := x^P_{t,k_j}$, $\tilde{w}_{t,j} := 1/J$
    else
        Trivially assign filtering particles $x^F_{t,j} := x^P_{t,j}$, $\tilde{w}_{t,j} := \bar{w}_{t-1,j}$
    end if
    Obtain weights $w_{t,j} := \tilde{w}_{t,j} \cdot p(y^*_t|x^P_{t,j}, \theta)$
    Obtain likelihood at timestep t, $\mathcal{L}_t(\theta) := \sum_{j=1}^J w_{t,j}$
    Normalize particle weights, $\bar{w}_{t,j} := w_{t,j}/\mathcal{L}_t(\theta)$
end for
Obtain likelihood $\prod_{t=1}^T \mathcal{L}_t(\theta)$, filtering distribution of weighted prediction particles $\{(x^P_{t,j}, \bar{w}_{t,j})\}$

2.3. Likelihood of a POMP Model

As mentioned earlier, the bootstrap filter is a special case of the weighted particle filter, where resampling occurs at every step. In this case, the prediction particles always have importance weights $p(y^*_t|x^P_{t,j}, \theta)/J$, so in this special case we can obtain the likelihood at timestep t with $\mathcal{L}_t(\theta) = \frac{1}{J}\sum_{i=1}^J p(y^*_t|x^P_{t,j}, \theta)$.

According to [13], the likelihood in a POMP model is given by:

$$\hat{\mathcal{L}}(\theta) = f(y^*_{1:N}|\theta) \approx \frac{1}{J}\sum_{j=1}^{J}\prod_{n=1}^{N} f(y^*_n|x_{n,j}, \theta) \tag{3}$$

If we resample the particles according to the filtering distribution at each timestep, the particle filter estimate of the log-likelihood is:

$$\hat{\ell}(\theta) = \sum_{t=1}^{T}\log(\mathcal{L}_t(\theta)) = \sum_{t=1}^{T}\log\left(\frac{1}{J}\sum_{j=1}^{J} f(y^*_t|x^P_{t,j}, \theta)\right). \tag{4}$$

If we do not, it is:

$$\hat{\ell}(\theta) = \sum_{t=1}^{T}\log\left(\sum_{j=1}^{J}\bar{w}_{t,j}\right) \tag{5}$$

where $\bar{w}_{t,j}$ are the normalized weights from the weighted particle filter, $x^P_{t,j} \sim \text{process}(x^F_{t-1,j}, \theta)$ is a sample from the process model on particles drawn from the filtering distribution $X_t|y^*_{1:t}$ on the previous timestep (which was in turn drawn from the prediction distribution $X_t|y^*_{1:t-1}$ on the previous timestep via resampling according to particle likelihood).

2.4. Central Limit Theorem for Particle Filters

Chopin [6] provides a central limit theorem for particle filters under multinomial and residual resampling. Let $\varphi : \mathcal{X} \to \mathbb{R}^d$ be a measurable function from the state space to $\mathbb{R}^d$, i.e. a function of the particles. Let $\tilde{\pi}_t = p(x_t|y^*_{1:t-1})$ be the prediction distribution at time t, and $\pi_t = p(x_t|y^*_{1:t})$ be the filtering distribution at time t. Define $\tilde{V}_0(\varphi) = \text{Var}_{\tilde{\pi}_0}(\varphi)$, and

$$\tilde{V}_t(\varphi) = \hat{V}_{t-1}(\mathbb{E}_{x_t|x_{t-1}}\varphi + \mathbb{E}_{\pi_{t-1}}\text{Var}_{x_t|x_{t-1}}(\varphi)) \tag{6}$$

$$V_t(\varphi) = \tilde{V}_t(w_t(\varphi - \mathbb{E}_{\pi_t}\varphi)) \tag{7}$$

$$\hat{V}_t(\varphi) = V_t(\varphi) + \text{Var}_{\pi_t}(\varphi) \tag{8}$$

Then, for $\frac{1}{J}\sum_{j=1}^J \varphi(x^F_{t,j})$ and $\sum_{j=1}^J \bar{w}_{t,j}\varphi(x^P_{t,j})$, under multinomial resampling

$$\sqrt{J}\left(\sum_{j=1}^J \bar{w}_{t,j}\varphi(x^P_{t,j}) - \mathbb{E}_{\pi_t}\varphi\right) \to \mathcal{N}(0, V_t(\varphi)) \tag{9}$$

$$\sqrt{J}\left(\frac{1}{J}\sum_{j=1}^J \varphi(x^F_{t,j}) - \mathbb{E}_{\pi_t}\varphi\right) \to \mathcal{N}(0, \hat{V}_t(\varphi)) \tag{10}$$

in distribution.

2.5. A Concentration Inequality for Particle Filters

A concentration inequality for particle filters can be found in the work of Del Moral and Rio [17]. Assume resampling at every timestep. Let $\hat{\tilde{\pi}}_t = \frac{1}{J}\sum_{j=1}^J \delta_{x^P_{t,j}}$ be the mixture Dirac approximation of the prediction distribution. Write $E_t = \mathcal{X}_t \times \mathcal{Y}_t$ for the product space of the particles and observations. Let $\tilde{\pi}_t$ be the prediction distribution. Let $\mathcal{B}(E_t)$ be the Banach space of bounded and measurable functions of particles and observations equipped with the sup norm and $\mathcal{P}(E_t)$ the set of measures on the state space and observations at each timestep. Denote the Dobrushin coefficient of a bounded integral operator M by $\beta(M) := \sup\{\text{osc}(M(f))|\text{osc}(f) \leq 1\}$. Write Φ_t for the mapping between the prediction distribution and fixed observation at time $t-1$ to the prediction distribution and fixed observation at time t, and $\Phi_{s,t}$ for the same between the prediction distributions and fixed observations at time s to time t, where $s \leq t$. Then, Del Moral and Rio [17] have shown that for any measurable $\varphi_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with oscillations $\text{osc}(\varphi) \leq 1$, if for any two signed measures η and μ on the particles (and fixed observation) at time t there is a first-order integral operator $D_\mu\Phi_t$ from $\mathcal{B}(E_t) \to \mathcal{B}(E_{t-1})$ such that $D_\mu\Phi_t(1) = 0$ and a decomposition $\Phi_t(\eta) - \Phi_t(\mu) \approx (\eta - \mu)D_\mu\Phi_n$ (plus a remainder term), we can bound

$$\left|\frac{1}{J}\sum_{j=1}^{J}\varphi_\theta(x_{t,j}^P, y_t^*) - \mathbb{E}_{\tilde{\pi}_t}\varphi_\theta(x_t, y_t^*)\right| \leq \frac{r_t}{J}(1+h^{-1}(t)) + \sqrt{\frac{2t}{J}}\beta_t \tag{11}$$

with probability at least $1-\exp(-t/2)$. Here, $h(t) = \frac{1}{2}(t-\log(1+t))$. $\beta_t^2 = \sum_{s=0}^{t}\sup_{\eta\in\mathcal{P}(E_s)}\beta(D_\eta\Phi_{s,t})^2$, and r_t is a parameter whose values only depend on the amplitude of some second-order remainder terms on the decomposition $\Phi_t(\eta)-\Phi_t(\mu)\approx(\eta-\mu)D_\mu\Phi_n$.

2.6. Auto-differentiation of the Likelihood

Let us take a gradient of the likelihood estimate of the particle filter, using the bootstrap filter to simplify calculations. This yields:

$$\nabla_\theta\hat{\ell}(\theta) = \sum_{t=1}^{T}\nabla_\theta\log(\mathcal{L}_t(\theta)) = \sum_{t=1}^{T}\frac{\nabla_\theta\mathcal{L}_t(\theta)}{\mathcal{L}_t(\theta)} = \sum_{t=1}^{T}\frac{\frac{1}{J}\sum_{j=1}^{J}\nabla_\theta f(y_t^*|x_{t,j}^P,\theta)}{\frac{1}{J}\sum_{j=1}^{J}f(y_t^*|x_{t,j}^P,\theta)} \tag{12}$$

Auto-differentiation takes care of every operation in the above but those involving $\nabla_\theta f(y_n^*|x_{n,j}^P,\theta)$, for any timestep n. To deal with this, recall that the vector $x_{n,j}^P$ happens to be obtained from $x_{1,j}^P$, having gone through n alternating resampling steps and simulations from the process model. We can unroll these operations, to find that $x_{n,j}^P = (\text{process}\circ\text{resample})^n(x_{1,j})$, where we use 'process' to denote a simulation from the process model, and 'resample' to denote a resampling operation). This suggests a strategy, to keep track of the gradients as nodes in the stochastic computational graph with regard to each simulation and resampling operation while performing them.

We then have

$$\nabla_\theta f(y_n^*|x_{n,j}^P,\theta) = \nabla_\theta f(y_n^*|(\text{process}\circ\text{resample})^n(x_{1,j}^P)) \tag{13}$$

and can differentiate through this recursively.

Therefore, to obtain a gradient estimate, one needs (1) a differentiable measurement density $f(y|x)$, (2) a differentiable 'process', and (3) a differentiable 'resample'.

2.7. The Differentiable Particle Filter (DPF) from Scibior and Wood [27]

To calculate the same estimate for $\nabla_\theta\hat{\ell}(\theta)$ that Poyiadjis et. al. derived [19], Scibior and Wood [27] make only one minor modification to the standard particle filter. After resampling, the standard particle filter assigns weight $1/J$ to each particle. DPF, on the other hand, assigns weight $\frac{1}{J}\bar{w}_{t,k_j}/\text{stop}(w_{t,k_j})$, enabling gradients to propagate while having the expression evaluate to $1/J$ on the forward pass. For future reference, we will use k_j to refer to the index of resampled particle j, $x_{1:t,j}^A$ to refer to the ancestral trajectory of $x_{t,j}^F$ and $\bar{w}_{t,j}$ to refer to the normalized weight of particle j at observation t.

2.8. The Score Function and Path Difference Gradient Estimators

Finally, for future reference, we provide the formulas for two popular stochastic derivative estimators. The score function derivative is defined as

$$\nabla_\theta \mathbb{E}_x[f(x)] = \mathbb{E}_x[f(x)\nabla_\theta \log p(x;\theta)] \tag{14}$$

and the path derivative is defined as

$$\nabla_\theta \mathbb{E}_\epsilon[f(x(\epsilon,\theta)] = \mathbb{E}_\epsilon[\nabla_\theta f(x(\epsilon,\theta)] \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(x(\epsilon,\theta)) \tag{15}$$

.

3. A New Framework for Stochastic Derivative Estimates

One may ask: Exactly what does AD compute? Having simulated forth an array of particles at each timestep corresponding to a sequence of observations, reverse-mode differentiation recursively computes the derivative of the log-likelihood, a function of the observations and simulated particles, with respect to the model parameters. But post-filtering, the observations and simulated particles are deterministic quantities, which correspond to numerical simulations from some random seed.

Does AD then compute the derivative of the log-likelihood as a deterministic quantity conditional on a fixed seed? If so, why not then fix a seed, so that the particles and likelihood are deterministic quantities for each θ, and optimize the likelihood?

In the following section, we seek to answer these questions and consider how the stochastic derivative estimators and previous work mentioned in the literature review relate to this idea.

Remark: The idea of seed-fixing and computing the derivative of an otherwise-stochastic quantity is not new. Rosato et al [21] do so for a very specific case where the transition dynamics are Gaussian. Our contribution here is to address the more general case.

3.1. Seed-fixing for maximization with Monte Carlo evaluation

Definition 1 (Seed-Fixing)**.** Consider a probability space (Ω, Σ, μ) governing the particles and observations. As the particle filter is unbiased for the likelihood, write the likelihood as

$$\mathcal{L}(\theta) = \int_\Omega \mathcal{L}(\theta, \omega, J)\, d\mu(\omega) \tag{16}$$

where $\mathcal{L}(\theta, \omega, J)$ is the Monte-Carlo estimate of the likelihood given the realization of J particles corresponding to $\omega \in \Omega$. We call ω a **seed**, and the deterministic quantity $\mathcal{L}(\theta, \omega, J)$ the **seed-fixed estimate**.

The parameter is $\theta \in \Theta \subseteq \mathbb{R}^p$. Rather than seeking to differentiate $\ell(\theta)$, we shall focus on to the task of differentiating $\mathcal{L}(\theta, \omega, J)$ with respect to θ with AD. If one can optimize $\mathcal{L}(\theta, \omega, J)$ readily for fixed ω this is a potentially powerful tool for optimizing $\mathcal{L}(\theta)$, especially when the Monte Carlo sample size is large enough that $\mathcal{L}(\theta, \omega, J)$ is close to $\mathcal{L}(\theta)$ for all or most values of ω.

Combining various values of $\hat{\theta}(\omega_i) = \arg\max \mathcal{L}(\theta, \omega_i)$ is not entirely trivial, but can be put into the context of Monte Carlo adjusted confidence interval construction [10] or the argmax theorem. Thus, here we focus on the task of obtaining $\hat{\theta}(\omega)$.

A problem that arises is that $\mathcal{L}(\theta, \omega, J)$ may be far from smooth even when $\mathcal{L}(\theta)$ is smooth. This leads to a widely observed tradeoff between optimizing the smooth expectation of a stochastic function or a noisy deterministic function. This is the case for basic particle filter algorithms, due to the discontinuity of

resampling. The smoothness of $\mathcal{L}(\theta, \omega, J)$ may be obtained by weighting the particles in such a way that the resampling is fixed as a function of θ for a given ω. This loses the plug-and-play property. Worse, it leads to weights that are unstable with time, with instability increasing as θ deviates from some value $\phi \in \Theta$ for which weights are well balanced. We investigate the extent to which these problems affect differentiation at ϕ, which requires consideration of parameters only in an arbitrarily small neighborhood of ϕ.

3.2. A Smooth Extension to the Particle Filter

We modify the problem to one that is locally smooth by replacing (16) with what we call an off-policy likelihood.

Definition 2 (Off-Policy Particle Filter Likelihood)**.** Let $\mathcal{L}(\theta, \phi, \omega, J)$ denote the Monte-Carlo estimate of the likelihood evaluated at $\theta \in \Theta$ with J particles, but where the system evolves according to another $\phi \in \Theta$ and conditional on a fixed seed $\omega \in \Omega$. We then have

$$\mathcal{L}(\theta) = \mathbb{E}[\mathcal{L}(\theta, \phi, \Omega, J)] = \int_{\Omega} \mathcal{L}(\theta, \phi, \omega, J)\, d\mu(\omega) \tag{17}$$

We want $\phi \in \Theta$ to be a point in parameter space around which $\mathcal{L}(\theta, \phi, \omega, J)$ is differentiable with respect to θ and does not have excessive Monte Carlo variance. This leads us to our first assumption.

Assumption 1 (Smooth Neighborhood)**.** There exists a neighborhood $\mathcal{N}(\phi)$ around any $\phi \in \Theta$ where for all $\theta \in \mathcal{N}(\phi)$ and almost every $\omega \in \Omega$, the Monte Carlo estimate of the likelihood at $\theta \in \mathcal{N}(\phi)$ with the system evolving according to ϕ conditional on ω, $\mathcal{L}(\theta, \phi, \omega, J)$, is twice differentiable in θ.

3.2.1. Evaluating the Off-Policy Particle Filter Likelihood

Evaluating the off-policy likelihood is possible with a weighted particle filter by using resampling weight $p(y_n^*|x_n; \phi)$ and making two corrections in the particle weight, through multiplying $w_{t,j}$ by the two likelihood ratios below,

$$s = p(y_n^*|x_n; \theta)/p(y_n^*|x_n; \phi) \tag{18}$$

and

$$r = p(x_n|x_{n-1}; \theta)/p(x_n|x_{n-1}; \phi). \tag{19}$$

We make a couple of important remarks below.

Justification for Assumption 1: This enables us to justify Assumption 1 by noting that the particles for $\mathcal{L}(\theta, \phi, \omega, J)$ and $\mathcal{L}(\phi, \phi, \omega, J)$ are the same, eliminating any discontinuities from resampling, and that for small enough $\mathcal{N}(\phi)$ the likelihood ratios are bounded. The particle weights $w_{t,j}$, and therefore the likelihood $\sum_{j=1}^{J} \mathcal{L}_t(\theta)$, only change by a factor of the likelihood ratios, which are bounded and smooth in θ if the densities are.

Plug-and-Play Property: Having to multiply $w_{t,j}$ by (19) would typically break the plug-and-play property, but as we only evaluate this with respect to $\phi = \theta$, r evaluates to 1 and the plug-and-play property is maintained, as we show later in Lemma 2.

The following result showcases the correctness of corrections 18 and 19.

Proposition 1. *For θ, ϕ that satisfy Assumption 1, the following correction in the particle filter weights,*

$$w_{t,j} := \tilde{w}_{t,j} \cdot p(y^*|x^P_{t,j}, \phi) \cdot r \cdot s \tag{20}$$

successfully recovers $\mathcal{L}(\theta)$ in expectation.

Proof. Recall $\mathcal{L}(\theta) = \mathbb{E}[\mathcal{L}(\theta, \phi, \Omega, J)] = \int_\Omega \mathcal{L}(\theta, \phi, \omega, J)\, d\mu(\omega)$ and $\mathcal{L}(\theta) = \int_\Omega \mathcal{L}(\theta, \omega, J) d\mu(\omega)$. We want to evaluate

$$\begin{aligned}
\mathcal{L}(\theta, \omega, J) &= \mathbb{E}\left[\prod_{t=1}^{T} \mathcal{L}_t(\theta) \middle| \omega\right] = \mathbb{E}\left[\prod_{t=1}^{T} p(y^*_t|y^*_{1:t-1}, \theta) \middle| \omega\right] \\
&= \mathbb{E}\left[\prod_{t=1}^{T} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j}, \theta) p(x^P_{t,j}|y^*_{1:t-1}, \theta) \middle| \omega\right] \\
&= \mathbb{E}\left[\prod_{t=1}^{T} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j}, \theta) p(x^P_{t,j}|x^F_{t-1}, \theta) \middle| \omega\right]
\end{aligned}$$

where $p(x^P_t|y^*_{1:t-1}, \theta)$ is the prediction distribution of particles, and the filtering distribution of particles is proportional to $p(y^*_t|x^P_{t,j}, \theta)$, the likelihood of particle j at time t.

Consider first the case with no resampling. Let $\mathcal{L}^j(\theta)$, $\mathcal{L}^j(\phi)$ be the likelihood of trajectory j under θ and ϕ respectively. As the prediction and filtering particles are then the same, we use $x_{t,j}$ to refer to both. As such, conditional on ω,

$$\begin{aligned}
\mathcal{L}^j(\theta) &= \prod_{t=1}^{T} p(y^*_t|x_{t,j}, \theta) p(x_{t,j}|x_{t-1,j}, \theta) \\
&= \prod_{t=1}^{T} p(y^*_t|x_{t,j}, \phi) \frac{p(y^*_t|x_{t,j}, \theta)}{p(y^*_t|x_{t,j}, \phi)} p(x_{t,j}|x_{t-1,j}, \theta) \frac{p(x_{t,j}|x_{t-1,j}, \theta)}{p(x_{t,j}|x_{t-1,j}, \phi)} \\
&= \prod_{t=1}^{T} p(y^*_t|x_{t,j}, \phi) \cdot s \cdot p(x_{t,j}|x_{t-1,j}, \phi) \cdot r
\end{aligned}$$

and therefore, if trajectories were proposed with equal probability with particles conditional on ω,

$$\mathcal{L}(\theta, \omega, J) = \frac{1}{J} \sum_{j=1}^{J} \prod_{t=1}^{T} p(y^*_t|x_{t,j}, \phi) \cdot s \cdot p(x_{t,j}|x_{t-1,j}, \phi) \cdot r$$

The correction is therefore sufficient for recovering $\mathcal{L}(\theta)$, with trajectories collected with the process model at ϕ and measurements evaluated with the measurement model at ϕ.

Consider now the case with resampling. Without loss of generality we can resample every timestep. Here, $x^F_{t,j}(\phi) \sim p(y^*|x^P_{t,j}, \phi)$ and $x^P_{t,j}(\phi) \sim p(x_t|x^F_{t-1}, \phi)$, and similarly for θ. We use this notation to emphasize that **both** the measurement model and the previously drawn particles depend on ϕ or θ.

$$
\begin{aligned}
\mathcal{L}_t(\theta) &= \mathbb{E}_{x^P_{t,j} \sim p(x_t|x^F_{t-1},\theta)} \left[\frac{1}{J} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j},\theta) \right] \\
&= \mathbb{E}_{x^P_{t,j} \sim p(x_t|x^F_{t-1},\theta)} \left[\frac{1}{J} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j},\phi) \frac{p(y^*_t|x^P_{t,j},\theta)}{p(y^*_t|x^P_{t,j},\phi)} \right] \\
&= \mathbb{E}_{x^P_{t,j} \sim p(x_t|x^F_{t-1},\phi)} \left[\frac{1}{J} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j},\phi) \frac{p(y^*_t|x^P_{t,j},\theta)}{p(y^*_t|x^P_{t,j},\phi)} \frac{p(x_{t,j}|x_{t-1,j},\theta)}{p(x_{t,j}|x_{t-1,j},\phi)} \right] \\
&= \mathbb{E}_{x^P_{t,j} \sim p(x_t|x^F_{t-1},\phi)} \left[\frac{1}{J} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j},\phi) \cdot s \cdot r \right] \\
&= \mathbb{E}_{\omega} \mathbb{E}_{x^P_{t,j} \sim \text{process}(x^F_{t-1},\omega,\phi)} \left[\frac{1}{J} \sum_{j=1}^{J} p(y^*_t|x^P_{t,j},\phi) \cdot s \cdot r \middle| \omega \right]
\end{aligned}
$$

and again the correction is sufficient.

□

More Considerations: Likelihood ratios such as r can be singular for continuous-time continuous-state Markov processes, noting that in this case we would compute the ratio for the process $\{X_{(t)}, t_{n-1} \leq t \leq t_n\}$ over the continuous time interval. For such processes, the likelihood ratio r must be removed in the expression and some bias must be accepted when $\theta \neq \phi$.

We will also need r to be differentiable with respect to θ for fixed x_n and x_{n-1} at $\theta = \phi$. This leads us to our next assumption.

Assumption 2 (Differentiable Process Model Likelihood Ratios)**.** If X_n does not have a continuous distribution, the likelihood ratio

$$r = p(x_t|x_{t-1};\theta)/p(x_t|x_{t-1};\phi)$$

is differentiable with respect to θ for fixed x_t and x_{t-1} at $\theta = \phi \in \Theta$.

Lastly, we require an additional regularity condition.

Assumption 3 (Locally Bounded Derivative)**.** Let $\mathcal{M}$ be an open subset of Ω. Fix $J = 1$. There exists some function $G(\theta)$ and a constant $G = \sup_{\theta \in \mathcal{N}} G(\theta) < \infty$ such that

$$||\nabla \ell(\theta, \phi, \omega, J)||_2 < G(\theta) \leq G < \infty$$

for every $\phi \in \mathcal{M}$, θ in smooth neighborhood $\mathcal{N}(\phi)$, and almost every $\omega \in \Omega$.

3.3. Off-Policy Weighted Differentiable Particle Filter

We are finally ready to introduce the off-policy weighted differentiable particle filter. In the event where a seed is fixed, all expressions in Algorithm 2 can be considered to be conditional on $\omega \in \Omega$.

Algorithm 2: Off-Policy Weighted Differentiable Particle Filter

Input: Number of particles J, timesteps T, measurement model $p(y_t^*|x_t, \theta)$, simulator process$(x_{t+1}|x_t, \theta)$, evaluation parameter θ, behavior parameter ϕ, seed ω.
Initialize particles $x_{0,j}^F$, weights $w_{t,j} = 1/J$, normalized weights $\bar{w}_{t,j} = 1/J$. Fix ω.
for $t = 1, ..., T$ **do**
  Get prediction particles $x_{t,j}^P \sim \text{process}(x_{t-1,j}^F, \phi)$
  if resampling condition fulfilled: **then**
    Draw indices $k_j \sim p(y_t^*|x_{t,j}^P, \phi)$
    Assign filtering particles $x_{t,j}^F := x_{t,k_j}^P$, $\tilde{w}_{t,j} := \frac{1}{J} \frac{\bar{w}_{t,k_j}}{\text{stop}(\bar{w}_{t,k_j})}$
  else
    Trivially assign filtering particles $x_{t,j}^F := x_{t,j}^P$, $\tilde{w}_{t,j} := \bar{w}_{t-1,j}$
  end if
  Calculate $r = p(x_t|x_{t-1}; \theta)/p(x_t|x_{t-1}; \phi)$, $s = p(y_t^*|x_t; \theta)/p(y_t^*|x_t; \phi)$
  Obtain weights $w_{t,j} := \tilde{w}_{t,j} \cdot p(y_t^*|x_{t,j}^P, \phi) \cdot r \cdot s$
  Obtain likelihood at timestep t, $\mathcal{L}_t(\theta) := \sum_{j=1}^J w_{t,j}$
  Normalize particle weights, $\bar{w}_{t,j} := w_{t,j}/\mathcal{L}_t(\theta)$
end for
Obtain likelihood estimate $\hat{\mathcal{L}}(\theta) = \mathcal{L}(\theta, \phi, \omega, J) := \prod_{t=1}^T \mathcal{L}_t(\theta)$, filtering distribution of weighted prediction particles $\{(x_{t,j}^P, \bar{w}_{t,j})\}$

Fortunately, Algorithm 2 possesses the desirable plug-and-play property when evaluated at $\theta = \phi$, as we show in the following lemma.

Lemma 2. *When evaluated at $\theta = \phi$, obtaining derivatives from Algorithm 2 is plug-and-play.*

Proof. Fix $\theta = \phi$. We take advantage of the fact that we use the process model as the proposal that Scibior and Wood refer to, and so their $q_\phi(x_t|x_t{-}1) = p_\theta(x_t|x_{t-1})$. Let $L_{t,j} = \frac{p(y_{1:t}^*, x_{1:t,j}^A|\theta)}{p(x_{1:t,j}^A|\theta)} = p(y_{1:t}^*|x_{1:t,j}^A, \theta)$ be the importance weight of prediction particle j accumulated under its full ancestral trajectory, satisfying $L_{t,j} = C w_{t,j} L_{t-1,a_j}$ where a_j is the ancestor particle of particle j, C is some constant not dependent on θ, and $L_{0,j} = 1$, as in Scibior and Wood [27].

When $\theta = \phi$, without loss of generality Algorithm 2 reduces to the DPF algorithm from Scibior and Wood. Then, from their equation (62),

$$\text{eval}\left(\nabla_\theta \hat{\ell}(\theta)\right) = \sum_{j=1}^J \bar{w}_{T,j} \nabla_\theta \log L_{T,j} \tag{21}$$

and $L_{T,j}$ is a cumulative product of weight components along the ancestral trajectory, where the weights only depend on $p(y_t^*|x_{t,j}^P, \theta)$.

As the Hessian is the gradient of the gradient, obtaining the Hessian is plug-and-play as well. Explicitly, from equation (107) in Scibior and Wood,

$$\text{eval}\left(\nabla_\theta^2 \hat{\ell}(\theta)\right) \tag{22}$$

$$= \sum_{j=1}^{J} \bar{w}_{T,j}((\nabla_\theta \log L_{T,i})(\nabla_\theta \log L_{t,j})^T + \nabla_\theta^2 \log L_{T,j}) - \sum_{i,j=1}^{J} \bar{w}_{T,i}\bar{w}_{T,j}(\nabla_\theta \log L_{T,i})(\nabla_\theta \log L_{T,j})^T \tag{23}$$

$$= \sum_{j=1}^{J} \bar{w}_{T,j}((\nabla_\theta \log L_{T,i})(\nabla_\theta \log L_{T,j})^T + \nabla_\theta^2 \log L_{T,j}) - \text{eval}\left(\nabla_\theta \hat{\ell}(\theta)\right) \text{eval}\left(\nabla_\theta \hat{\ell}(\theta)\right)^T \tag{24}$$

and one can observe that this depends only on weights along each particle's ancestral trajectory.

Therefore when $\theta = \phi$ one can obtain the gradient and Hessian of the likelihood via Algorithm 2 while maintaining the plug-and-play property.

□

3.4. Non-Reparametrizable Process Models

There is a catch to the result in Lemma 2. When taking advantage of the simplification of $L_{t,j}$ to $p(y^*_{1:t}|x^A_{1:t,j}, \theta)$ and the cancellation of $r = p(x_t|x_{t-1};\theta)/p(x_t|x_{t-1};\phi)$ possible when $\theta = \phi$, the gradients may only propagate to the process model parameters if the reparametrization trick can be performed so each particle is a function of random noise ϵ and the parameters, $x_{t,j}(\epsilon, \theta)$. In this case, $L_{T,j} = \log p(y^*_{1:T}|x^A_{1:T,j}, \theta) = \log p(y^*_{1:T}|x^A_{1:T,j}(\theta), \theta)$, and the process model successfully gets updated in each iteration with AD.

However, if the reparametrization trick cannot be used, one may have to resort to explicitly using the score-function derivative estimator or other stochastic derivative estimates for the process model. The former can be done through maintaining a separate surrogate loss function. For example, let $\xi \in \Xi \subseteq \Theta$, where Ξ spans a subspace of Θ that corresponds to random variables used in the simulator that cannot be reparametrized. Then $\nabla_\xi \log p(y^*_{1:T}|x^A_{1:T,j}) = 0 = \nabla_\xi w_{t,j}$, and gradients of the log-likelihood with respect to θ do not propagate back to ξ. One can then use the surrogate loss

$$s(\xi) := -\sum_{j=1}^{J} w_{T,j} \sum_{t=1}^{T} \log p(u^A_{t,j}|\xi) \tag{25}$$

where $u^A_{t,j} \sim p(u|\xi)$ is a discrete random variable drawn at each timestep for each surviving particle along its ancestral trajectory. The corresponding derivative is then $\nabla_x is(\xi) = -\sum_{j=1}^{J} w_{T,j} \sum_{t=1}^{T} \nabla_\xi \log p(u^A_{t,j}|\xi)$, and one can then update the process model parameters with the derivatives of the surrogate loss with respect to ξ.

3.5. Relationship to Scibior and Wood [27]

The derivative of the smooth particle filter in Sec. 3.2 was implemented using the stop gradient approach of Scibior and Wood [27]. In fact, the derivative of this algorithm matches the quantity calculated by the DPF algorithm of Scibior and Wood [27], which in turn matches the quantity calculated by Poyiadjis et. al. [19].

Therefore, Algorithm 2 has derivatives corresponding exactly to both a properly-weighted unbiased off-policy extension of a particle filter that maintains resampling events with a fixed seed and when $\theta = \phi$, the stochastic derivative estimates of Poyiadjis et. al. [19].

4. Properties of Auto-Differentiation for the Particle Filter

In this section, we highlight a few more interesting properties of AD for the particle filter.

4.1. Dependence only on Surviving Particles

Equation (62) of Scibior and Wood [27] makes a strong claim. Specifically,

$$\text{eval}\left(\nabla_\theta \hat{\ell}_{DPF}(\theta)\right) = \sum_{j=1}^{J} \bar{w}_{T,j} \nabla_\theta \log L_{T,j} = \sum_{j=1}^{J} \bar{w}_{T,j} \nabla_\theta \log p(y^*_{1:T} | x^A_{1:T,j}, \theta) \tag{26}$$

where $X^A_{1:N,j}$ is the ancestral trajectory of $X^F_{N,j}$ and $\bar{w}_{N,j}$ is the normalized weight of particle j at observation N. The last equality follows from the fact that we propose transitions with the process model. Equation (26) may be surprising because the right hand side depends only on ancestral particles of those particles surviving to time N. The identity in (26) is claimed to be true for all models, and all realizations of the Monte Carlo random variables involved in computing these quantities.

In Appendix A, we check this identity on a toy example.

4.2. Variance Dependence on Horizon

According to the above expression in (26), Scibior and Wood [27] show that the derivative obtained by ADPF is equivalent to the derivative calculated by Poyiadjis et. al. [19] over only the ancestors of particles surviving to the final time point, via a Fisher identity. This helps to clarify the mixing issues and scaling properties of the derivative. Intuitively, according to Poyiadjis et. al. [19], the variance of ADPF estimate scales at least quadratically with horizon because it is only dependent on the ancestors of surviving particles, and not every particle itself.

Other modifications that lower variance remain open as possibilities. These include both the optimal transport resampling from Corenflos et. al. [7] and the auto-derivative of the marginal particle filter from Scibior and Wood [27]. However, both of the above modify the forward pass, and have runtime quadratic in the number of particles. Additionally, the latter is not plug-and-play due to the requirement to take a weighted average of the process density.

4.3. Potential Speedup

On the other hand, as the Fisher identity implies a massive cancellation that requires one to track only the ancestors of surviving particles (looking at only approximately $N \log(J)$ rather than NJ items), this could lead to a large speedup in computation.

We conjecture that it is possible that existing AD libraries may implement this speedup by default. As reverse-mode backpropagation starts from the final output and recursively computes derivatives, it is likely that doing so these libraries do not track non-surviving particles when computing derivatives.

5. Numerical Optimization with ADPF

Automatic differentiation for the particle filter opens up a variety of attractive options. Gradient-based methods, second-order methods like Newton's method, two potential improvements to IF2 from [11], as

well as Hamiltonian Particle Markov Chain Monte Carlo are all possible. This is because of the following properties.

Lemma 3 (Unbiased Likelihood Derivatives)**.** *For $\theta \in \Theta$, $\phi \in \mathcal{N}(\theta)$, and $J \in \mathbb{N}$,*

$$\mathbb{E}_\omega \nabla_\theta \mathcal{L}(\theta, \phi, \omega, J) = \nabla_\theta \mathcal{L}(\theta) \text{ and } \mathbb{E}_\omega \nabla^2_\theta \mathcal{L}(\theta, \phi, \omega, J) = \nabla^2_\theta \mathcal{L}(\theta)$$

Proof. It is known that the weighted particle filter is unbiased for the likelihood. So for any properly computed importance weights w_j, $\mathbb{E}\hat{\mathcal{L}}(\theta) = \mathbb{E}\left[\prod_{t=1}^T \sum_{j=1}^J w_{t,j}\right] = \mathcal{L}(\theta)$. As the off-policy weighted particle filter is a special case, $\mathbb{E}_\omega \mathcal{L}(\theta, \phi, \omega, J) = \mathcal{L}(\theta)$ as well.

By assumption 1 the function $\mathcal{L}(\theta, \phi, \omega, J)$ is twice-differentiable at θ. The first and second partial derivatives exist at θ and are bounded by assumption 3. In conjunction with the linearity of the derivative, we can exchange the expectation and derivative to find that

$$\mathbb{E}_\omega \nabla_\theta \mathcal{L}(\theta, \phi, \omega, J) = \nabla_\theta \mathbb{E}_\omega \mathcal{L}(\theta, \phi, \omega, J) = \nabla_\theta \mathcal{L}(\theta)$$

$$\mathbb{E}_\omega \nabla^2_\theta \mathcal{L}(\theta, \phi, \omega, J) = \nabla^2_\theta \mathbb{E}_\omega \mathcal{L}(\theta, \phi, \omega, J) = \nabla^2_\theta \mathcal{L}(\theta)$$

□

Lemma 4 (Consistency of Seed-Fixing Likelihood Maximization)**.** *Let $\mathcal{M}$ be a bounded neighborhood around the optimum θ^*. Fix $\omega \in \Omega$. Let $\hat{\theta}$ be the estimate of θ that maximizes $\mathcal{L}(\theta, \omega, J)$. As $J \to \infty$, $\hat{\theta} \to \theta$ in probability.*

Proof. To show this, we require the argmax theorem, which in turn requires the uniform law of large numbers (ULLN).

To show that the ULLN holds here, with $\text{cl}(\mathcal{M})$ denoting the closure of $\mathcal{M}$,

$$\sup_{\theta \in \text{cl}(\mathcal{M})} \sum_{j=1}^J |\ell(\theta, \omega, J) - \ell(\theta)| \to 0$$

by the compactness of $\text{cl}(\mathcal{M})$ and the CLT for particle filters from [6].

Choose $\hat{\theta}$ that maximizes $\mathcal{L}(\theta, \omega, J)$. We then have $\ell(\hat{\theta}, \omega, J) \geq \sup_{\theta \in \mathcal{N}} \ell(\theta, \omega, J) - o_P(1)$. Then by the argmax theorem, $\hat{\theta} \to \theta^*$ in probability as $J \to \infty$. □

Remark: It is known that there is a neighborhood $\mathcal{M}$ around θ^* where local asymptotic normality holds. If local asymptotic normality holds within $\mathcal{M}$, then as $J \to \infty$ a single Newton step with line search can find this optimum. With the right step size, we can maximize $\mathcal{L}(\theta, \omega, J)$, and we have just shown that the resulting estimate of θ is consistent in the number of particles.

The above result shows that fixing ω and obtaining $\hat{\theta}(\omega)$ provides a consistent estimate in the number of particles J. However, since $\mathcal{L}(\theta, \phi, \omega, J)$ is only stable locally, we do not necessarily wish to maximize it for fixed ω. Indeed, our basic algorithm only lets us maintain the plug-and-play property by evaluating and differentiating only at $\theta = \phi$, we are not in a position to maximize it as a function of θ for fixed ϕ. A natural idea is to iteratively construct θ_n by one step of the maximization of $\ell(\theta, \theta_{n-1}, \omega, J)$. Unfortunately, each iteration of this procedure has a different value of ϕ and so we have lost the smoothness that made the deterministic maximization appealing.

Therefore, we suggest performing only a single step of maximizing $\ell(\theta, \phi, \omega, J)$, and drawing a new ω for the next step. This is a very natural implementation detail of iterative optimization methods that coincides with how many would implement it in practice.

5.1. Newton Steps

Within the neighborhood of the optimum where local asymptotic normality (LAN) [4] holds, second-order optimization methods become more attractive. Poyiadjis et. al [19] offer an estimator of the Hessian, which Scibior and Wood [27] show is obtainable by applying AD twice to the log-likelihood – something many software libraries are easily capable of.

We therefore suggest Algorithm 3, which performs a single Newton step for $\ell(\theta, \phi, \omega, J)$, and draws a new ω for the next step. If a quadratic approximation to $\ell(\theta)$ and $\ell(\theta, \phi, \omega, J)$ is good for values of ϕ having log-likelihood within Monte Carlo evaluation error of the maximized log-likelihood, then we might expect this algorithm to dance around this high likelihood region (with excessive movement avoided through e.g. Armijo line search). That is perhaps the best that can be hoped for, but should be entirely adequate for the Monte Carlo Adjusted Profile (MCAP) [10] and other Monte Carlo adjusted inference tools to work with.

In a later section, we provide a finite-particle analysis of the convergence of this procedure, relying on the concentration inequality from Del Moral and Rio [17] to perform a similar analysis as in Roosta-Khorasani and Mahoney [20] for subsampled Newton methods.

Algorithm 3: Newton's Method for Likelihood Maximization with Particle Filters

Input: Number of particles J, timesteps T, measurement model $p(y_t^*|x_t, \theta)$, simulator process$(x_{t+1}|x_t, \theta)$
Initialize θ_0, $n = 0$
while Procedure not converged: **do**
    Draw $\omega \in \Omega$
    Perform forward pass by running Algorithm 2 to obtain $\ell(\theta_n, \theta_n, \omega, J)$
    Obtain $g(\theta_n) = \nabla_{\theta_n}(-\ell(\theta_n, \theta_n, \omega, J))$, $H(\theta_n) = \nabla^2_{\theta_n}(-\ell(\theta_n, \theta_n, \omega, J))$ via backpropagation
    Obtain η with line search or an annealing schedule
    Update $\theta_{n+1} := \theta_n - \eta(H(\theta_n))^{-1} g(\theta_n)$, $n := n + 1$
end while
Return $\hat{\theta} := \theta_n$

5.2. Incorporating Gradient Proposals into IF2

This suggests a natural extension to the IF2 algorithm from Ionides et. al. [11]. Broadly, IF2 modifies the particle filter such that each particle has its own parameters. It performs M modified particle filters sequentially, updating the particles and their parameters in proportion to the likelihood of each particle, while perturbing the parameters of each particle with Gaussian noise at each timestep and in between each particle filtering iteration.

With the stop-gradient trick from Scibior and Wood [27] at each resampling step, each iteration of both the inner and outer loops, as well as the entire algorithm, is differentiable – the Gaussian noise perturbation is differentiable with the reparametrization trick and the simulator follows the same conditions as discussed earlier.

One can therefore replace the Gaussian noise perturbation before the start of each particle filter and/or the Gaussian noise perturbation during each timestep of the particle filter, zeroing out gradients at the end of each outer loop. Performing a gradient or Newton update in the outer loop corresponds to performing an

update after each iteration of the particle filter, while performing them at each iteration in the inner loop corresponds to an online updating procedure similar to that in Singh et. al. [23].

Algorithm 4: Iterated Filtering 2 with Derivative Updates

Input: Number of particles J, number of iterations M, timesteps T, cooling parameter $0 < a < 1$, covariance matrix Φ, initial parameter vectors $\{\theta_j, j = 1, ..., J\}$, measurement model $p(y_t^*|x_t, \theta)$, simulator process$(x_{t+1}|x_t, \theta)$, sequence of outer loop learning rates η_m, sequence of inner loop learning rates η_t
Initialize $H(\theta_j) = I$, $g(\theta_j) = 0$.
for m=1,...,M **do**
  Perform Newton update $\theta_{0,j}^F := \theta_j - \eta_m (H(\theta_J))^{-1} g(\theta_j)$ (Vanilla IF2: Set $\theta_{0,j}^F := \mathcal{N}(\theta_j, a^{m-1}\Phi)$)
  Initialize $x_{0,j}^F$ according to $\theta_{0,j}^F$ for $j = 1, ..., J$
  for t=1,...,T **do**
    Draw $\omega \in \Omega$, and condition the rest of the inner loop on this seed
    Optionally perform an online gradient update on the normalized likelihood weights of particle j, $\theta_{t,j}^P := \theta_{t-1,j}^F - a^{m-1}\eta_t \nabla_{\theta_{t-1,j}^F} \bar{w}_{t-1,j}^F$. Else, draw $\theta_{t,j}^P \sim \mathcal{N}(\theta_{t-1,j}^F, a^{m-1}\Phi))$ as in vanilla IF2
    Simulate $x_{t,j}^P \sim \text{process}(x_{t-1,j}^F, \theta_{t-1,j}^P)$
    Obtain weights $w_{t,j} = \frac{1}{J} p(y_t^*|x_{t,j}^P, \theta_{t,j}^P) \frac{\bar{w}_{t-1,j}}{\text{stop}(\bar{w}_{t-1,j})}$ and normalized weights $\bar{w}_{t,j} = \frac{w_{t,j}}{\sum_{j=1}^J w_{t,j}}$
    Draw indices $k_j \sim \text{Categorical}(\bar{w}_{t,1}, ..., \bar{w}_{t,j})$
    Assign filtering particles $x_{t,j}^F = x_{t,k_j}^P$, filtering parameter vectors $\theta_{t,j}^F = \theta_{t,k_j}^P$, and filtering weights $\bar{w}_{t,j}^F = \bar{w}_{t,k_j}$
  end for
  Set $\theta_j = \theta_{T,j}^F$, obtain $g(\theta_j)$ and $H(\theta_j)$ for each $j = 1, ..., J$, and zero out gradients
end for
Return $\{\theta_j, j = 1, ..., J\}$

We leave the empirical validation of Algorithm 4 to future work.

6. Finite-Particle Analysis of Newton Steps

The analysis in this section roughly follows the analysis in Roosta-Khorasani and Mahoney [20], except with the caveat that none of the matrix concentration bounds they use apply here as the particles are dependent. We instead use the concentration inequality from Del Moral and Rio [17] to bound the gradient and Hessian estimates. In this section, we fix $\omega \in \Omega$ only within each filtering iteration, evaluate Algorithm 2 at $\theta = \phi$, and analyze Algorithm 3.

These bounds are not sharp by any means. Yet, to our knowledge, there are no other finite-particle high-probability bounds of stochastic derivative estimates in the literature.

The convergence analysis in Theorem 7 is limited to the case where $-\ell$ is γ-strongly convex. Though it is true that in a neighborhood of the optimum local asymptotic normality holds and the log-likelihood is strongly convex in this neighborhood, in practice likelihood surfaces for POMPs are often highly nonconvex globally. The convergence to an optimum, local or global, must therefore be sensitive to initialization.

6.1. Bounding the Gradient

Lemma 5 (Concentration of Measure for Gradient Estimate)**.** *Assume* $osc\left(\frac{\partial}{\partial_\theta}\right) \leq 1$. *For* $||\nabla_\theta \hat{\ell}(\theta) - \nabla_\theta \ell(\theta)||$ *to be bounded by* ϵ *with probability* $1-\delta$, *we require*

$$J > \max\left\{\frac{r_t\sqrt{p}}{\epsilon}\left(1+h^{-1}\left(\log\left(\frac{6p}{\delta}\right)\right)\right),\ \frac{2p\log(6p/\delta)}{\epsilon^2}\beta_t^2,\ \frac{G(\theta)^2}{\epsilon^2}p\log\left(\frac{6p}{\delta}\right)\right\} \tag{27}$$

where $G(\theta)$ *is defined as in Assumption 3,* $h(t) = \frac{1}{2}(t - \log(1+t))$, *and* β_t *and* r_t *are two additional model-specific parameters that do not depend on* J.

Proof. We will seek to use the concentration inequality of Del Moral and Rio to bound the deviation of the gradient estimate from the gradient of the negative log-likelihood in the sup norm with a union bound. To use this, we will first need to resample the particles once more to reassign the normalized weights to $1/J$.

Perform another resampling step at time T to obtain indices $k_j \sim \text{Categorical}(\bar{w}_{T,1}, ..., \bar{w}_{T,J})$. Obtain ancestral trajectories of surviving time T filtering particles $x^A_{k_j,1:T}$. Define $g_j(\theta) = \nabla_\theta - \log L_{T,k_j}$, the gradient of the negative logarithm of the accumulated importance weights of particle k_j along its ancestral trajectory.

We can bound

$$||\nabla_\theta \hat{\ell}(\theta) - \nabla_\theta \ell(\theta)|| \leq \left\|(\nabla_\theta - \hat{\ell}(\theta)) - \frac{1}{J}\sum_{j=1}^{J} g_j(\theta)\right\| + \left\|\frac{1}{J}\sum_{j=1}^{J} g_j(\theta) - (\nabla_\theta - \ell(\theta))\right\| \tag{28}$$

We begin by examining the first term. This term bounds the error of the gradient estimate that the differentiable particle filter provides from the gradient estimate after the final resampling step.

This amounts to solving the following problem. Say you have a weighted mean $\sum w_j x_j$, where $j = 1, \ldots, J$. The x_j are fixed numbers bounded by some constant G and the w_j are non-negative and sum to 1. Resample indices $k_j, j = 1, \ldots, J$ according to the multinomial distribution to get x_{k_j} for each j, and also $J^{-1}\sum x_{k_j}$. We want to bound $|\sum w_j x_j - J^{-1}\sum x_{k_j}|$.

By a simple application of Hoeffding's inequality and a union bound, we have

$$\mathbb{P}\left(\exists\ j\ s.t.\ \left|(\nabla_\theta - \hat{\ell}(\theta)) - \frac{1}{J}\sum_{j=1}^{J} g_j(\theta)\right| \geq \epsilon\right) \leq 2p\exp\left(\frac{-2\epsilon^2 J}{G(\theta)^2}\right) \tag{29}$$

We require

$$J > \frac{G(\theta)^2}{2\epsilon^2}\log(2p/\delta) \tag{30}$$

particles to bound $\left\|(\nabla_\theta - \hat{\ell}(\theta)) - \frac{1}{J}\sum_{j=1}^{J} g_j(\theta)\right\|_\infty < \epsilon$ with probability $1-\delta$, where $G(\theta)$ is defined as in Assumption 3.

We can bound the sup norm by $||x||_\infty \leq \sqrt{p}||x||_2$. So to bound $\left\|(\nabla_\theta - \hat{\ell}(\theta)) - \frac{1}{J}\sum_{j=1}^{J} g_j(\theta)\right\| < \epsilon$ with probability $1-\delta/3$, we scale the previous ϵ to $\epsilon/\sqrt{p}$ and therefore require

$$J > \frac{G(\theta)^2}{2\epsilon^2}p\log\left(\frac{6p}{\delta}\right) \tag{31}$$

We now tackle the second term. Write g_{ij} for the i-th component of the gradient of the negative log-likelihood with respect to the j-th particle, $(-\bar{w}_{T,j}\nabla_\theta \log L_{T,j})_i$.

Apply the concentration inequality in 11 from Del Moral and Rio [17] and another union bound to find that

$$\left\| \frac{1}{J} \sum_{j=1}^{J} g_{ij} - \left(-\frac{\partial}{\partial \theta_i} \ell(\theta) \right) \right\|_\infty \leq \frac{r_t}{J}(1 + h^{-1}(t)) + \sqrt{\frac{2t}{J}} \beta_t \tag{32}$$

with probability $1 - 2p\exp(-t)$. We split the remaining $2\delta/3$ failure probability among these two terms, to find $\delta/3 = 2p\exp(-t) => t = \log(6p/\delta)$. Here, $h(t) = \frac{1}{2}(t - \log(1+t))$. The two additional model-specific parameters are β_t and r_t, which do not depend on J.

The analogous bound for the 2-norm falls out by scaling the right-hand side by $\sqrt{p}$, to require

$$\epsilon = \frac{r_t\sqrt{p}}{J}(1 + h^{-1}(\log(6p/\delta))) + \sqrt{\frac{2p\log(6p/\delta)}{J}} \beta_t \tag{33}$$

and therefore need

$$J > \max\left\{ \frac{r_t\sqrt{p}}{\epsilon} \left(1 + h^{-1}\left(\log\left(\frac{6p}{\delta} \right) \right) \right), \frac{2p\log(6p/\delta)}{\epsilon^2} \beta_t^2 \right\} \tag{34}$$

Overall, we finally require

$$J > \max\left\{ \frac{r_t\sqrt{p}}{\epsilon} \left(1 + h^{-1}\left(\log\left(\frac{6p}{\delta} \right) \right) \right), \ \frac{2p\log(6p/\delta)}{\epsilon^2} \beta_t^2, \ \frac{G(\theta)^2}{\epsilon^2} p \log\left(\frac{6p}{\delta} \right) \right\} \tag{35}$$

□

6.2. Bounding the Hessian

Lemma 6 (Minimum Eigenvalue Bound for Hessian Estimate)**.** *Assume that the Hessian of the negative log-likelihood $H = \sum_{j=1}^{J} \mathbb{E}H_j$ has a minimum eigenvalue $\gamma > 0$, that $\mathbb{E}\lambda_{\min}(H_j) = \gamma' > 0$, and finally that $osc(\lambda_{\min}) \leq 1$.*

We then require

$$J > \max\left\{ \frac{2r_t(1 + h^{-1}(t)) + 2c}{\gamma'}, \frac{2(2t\beta_t^2 + c)^2}{\gamma'^2} \right\} \geq \frac{r_t(1 + h^{-1}(t))}{\gamma'} + \sqrt{2tJ}\beta_t/\gamma' + c/\gamma' \tag{36}$$

for $\hat{H}(\theta)$ to be invertible and positive definite with minimum eigenvalue greater than or equal to $c \in (0, \sum_{j=1}^{J} \mathbb{E}\lambda_{\min}(H_j))$ and with probability at least $1 - \exp(-t)$.

Proof. Write $\hat{H}(\theta) = \hat{H} = \sum_{j=1}^{J} H_j$ for the estimate of the negative of the Hessian in equation 22, where H_j is an element of the outer sum over the J particles.

As the negative log-likelihood is convex, we want to bound the minimum eigenvalue of $\hat{H}(\theta)$ from below with high probability, so that all the eigenvalues of $\hat{H}(\theta)$ are positive with high probability. This ensures that the estimated Hessian is invertible and positive-definite.

From Boyd and Lieven [1], we know that the minimum eigenvalue of a symmetric matrix is concave. Therefore, it suffices to show that the first inequality in the below expression

$$0 < \sum_{j=1}^{J} \lambda_{\min}(H_j) \leq \lambda_{\min}\left(\sum_{j=1}^{J} H_j\right) = \lambda_{\min}(\hat{H}) \tag{37}$$

holds with high probability.

Apply the concentration inequality in 11 from Del Moral and Rio [17] to find that

$$\frac{1}{J}\sum_{j=1}^{J} \lambda_{\min}(H_j) - \mathbb{E}_{\tilde{\pi}_t}\lambda_{\min}(H_j) = \frac{1}{J}\sum_{j=1}^{J} \lambda_{\min}(H_j) - \gamma' \geq -\frac{r_t}{J}(1 + h^{-1}(t)) - \sqrt{\frac{2t}{J}}\beta_t \tag{38}$$

$$\sum_{j=1}^{J} \lambda_{\min}(H_j) \geq -r_t(1 + h^{-1}(t)) - \sqrt{2tJ}\beta_t + J\gamma' \tag{39}$$

with probability at least $1 - \exp(-t)$. Here, $h(t) = \frac{1}{2}(t - \log(1+t))$. The two additional model-specific parameters are β_t and r_t, which do not depend on J.

We additionally require, for $c \in (0, \sum_{j=1}^{J} \mathbb{E}\lambda_{\min}(H_j))$,

$$\sum_{j=1}^{J} \lambda_{\min}(H_j) \geq -r_t(1 + h^{-1}(t)) - \sqrt{2tJ}\beta_t + J\gamma' \geq c \tag{40}$$

$$J\gamma' \geq c + r_t(1 + h^{-1}(t)) + \sqrt{2tJ}\beta_t \tag{41}$$

We therefore require

$$J > \max\left\{\frac{2r_t(1 + h^{-1}(t)) + 2c}{\gamma'}, \frac{2(2t\beta_t^2 + c)^2}{\gamma'^2}\right\} \geq \frac{r_t(1 + h^{-1}(t))}{\gamma'} + \sqrt{2tJ}\beta_t/\gamma' + c/\gamma'$$

for $\hat{H}(\theta)$ to be invertible and positive definite with minimum eigenvalue greater than or equal to c with probability at least $1 - \exp(-t)$.

□

6.3. Convergence Analysis

Theorem 7 (Convergence of Newton Steps)**.** *Consider a variant of Algorithm 3 where one stops if the gradient estimate $||g(\theta_n)|| \leq \sigma\epsilon$. Write $f = -\ell$ for the objective function to minimize, the negative log-likelihood. Assume f is γ-strongly convex, $\gamma I \preceq \nabla_\theta^2 - \ell \preceq \Gamma I$. Also assume that $\sigma > 4$. Define G as in Assumption 3.*

Choose J according to the maximum of the J prescribed in Lemmas 5 and 6 with failure probability δ each. With minimum Hessian eigenvalue estimate tolerance $c \in (0, \sum_{j=1}^{J} \mathbb{E}\lambda_{\min}(H_j))$, as well as learning rate η and error tolerance ϵ such that

$$\eta \leq \frac{c(1-\beta)}{\Gamma}, \;\; \epsilon \leq \frac{c(1-\beta)}{2\Gamma}||g(\theta_n)|| \tag{42}$$

the following holds with probability at least $(1-\delta)^2$:

$$f(\theta_{n+1}) - f(\theta^*) \leq \left(1 - \eta\beta\frac{8\gamma}{9c}\right)(f(\theta_n) - f(\theta^*)) \tag{43}$$

and the algorithm eventually terminates after we obtain $||\nabla_\theta f(\theta_n)|| \leq (1+\sigma)\epsilon$.

Proof. In this analysis, we largely follow the proof of Theorem 6 in Roosta-Khorasani and Mahoney [20], with only minor details changed.

Define $\theta_\eta = \theta_n + \eta p_n$, where $p_n = -(H(\theta_n))^{-1} g(\theta_n)$. As in Roosta-Khorasani and Mahoney [20], we want to show there is some iteration-independent $\tilde{\eta} > 0$ such that the Armijo condition

$$f(\theta_n + \eta p_n) \leq f(\theta_n) + \eta \beta p_n^T g(\theta_n) \tag{44}$$

for any $0 < \eta < \tilde{\eta}$ and some $\beta \in (0, 1)$.

By an argument found in the beginning of the proof of Theorem 6 in Roosta-Khorasani and Mahoney [20], we have that choosing J such that $||\nabla_\theta \hat{\ell}(\theta_n) - \nabla_\theta \ell(\theta_n)|| \leq \epsilon$ and $\lambda_{\min}(H(\theta_n)) \geq c$ with probability $(1-\delta)^2$ for each n yields

$$f(\theta_\eta) - f(\theta_n) \leq \eta p_n^T g(\theta_n) + \epsilon \eta ||p_n|| + \eta^2 \Gamma ||p_n||^2/2 \tag{45}$$

Now, for some $c > 0$, with probability $1 - \delta$

$$p_n^T g(\theta_n) = -p_n^T H(\theta_n) p_n \geq -c||p_n||^2 \tag{46}$$

and we can obtain a decrease in the objective.

Substituting this into the previous expression,

$$f(\theta_\eta) - f(\theta_n) \leq -\eta p_n^T H(\theta_n) p_n + \epsilon \eta ||p_n|| + \eta^2 \Gamma ||p_n||^2/2 \tag{47}$$

and the Armijo condition then becomes

$$-\eta p_n^T H(\theta_n) p_n + \epsilon \eta ||p_n|| + \eta^2 \Gamma ||p_n||^2/2 \leq \eta \beta p_n^T g(\theta_n) = -\eta \beta p_n^T H(\theta_n) p_n \tag{48}$$

$$\epsilon ||p_n|| + \eta \Gamma ||p_n||^2/2 \leq (1-\beta) p_n^T H(\theta_n) p_n \tag{49}$$

$$\epsilon + \eta \Gamma ||p_n||/2 \leq c(1-\beta)||p_n|| \tag{50}$$

which holds and guarantees an iteration-independent lower bound if

$$\eta \leq \frac{c(1-\beta)}{\Gamma}, \;\; \epsilon \leq \frac{c(1-\beta)}{2\Gamma} ||g(\theta_n)|| \leq \frac{c(1-\beta)}{2} ||p_n|| \tag{51}$$

Now, first note that

$$||g(\theta_n)|| - ||\nabla_\theta f(\theta_n)|| \leq ||g(\theta_n) - \nabla_\theta f(\theta_n)|| \leq \epsilon => ||\nabla_\theta f(\theta_n)|| \geq ||g(\theta_n)|| - \epsilon \tag{52}$$

and

$$||\nabla_\theta f(\theta_n)|| - ||g(\theta_n)|| \leq ||\nabla_\theta f(\theta_n) - g(\theta_n)|| \leq \epsilon => ||g(\theta_n)|| \geq ||\nabla_\theta f(\theta_n)|| - \epsilon \tag{53}$$

There are now two cases. If the algorithm terminates and $||g(\theta_n)|| \leq \sigma\epsilon$, we can derive

$$||\nabla_\theta f(\theta_n)|| \leq ||g(\theta_n)|| + \epsilon = \sigma\epsilon + \epsilon = (\sigma + 1)\epsilon \tag{54}$$

If the algorithm does not terminate, then $||g(\theta_n)|| > \sigma\epsilon$. Notice that

$$\epsilon \geq ||g(\theta_n) - \nabla_\theta f(\theta_n)|| \geq ||g(\theta_n)|| - ||\nabla_\theta f(\theta_n)|| \tag{55}$$

$$||\nabla_\theta f(\theta_n)|| + \epsilon \geq ||g(\theta_n)|| \geq \sigma\epsilon \tag{56}$$

$$||\nabla_\theta f(\theta_n)|| \geq \sigma\epsilon - \epsilon = (\sigma - 1)\epsilon \tag{57}$$

$$\frac{||\nabla_\theta f(\theta_n)||}{\sigma - 1} \geq \epsilon \tag{58}$$

and now

$$||\nabla_\theta f(\theta_n)|| - \epsilon \geq ||\nabla_\theta f(\theta_n)|| - \frac{||\nabla_\theta f(\theta_n)||}{\sigma - 1} = \left(1 - \frac{1}{\sigma - 1}\right) ||\nabla_\theta f(\theta_n|| = \frac{\sigma - 2}{\sigma - 1}||\nabla_\theta f(\theta_n)|| \geq \frac{2}{3}||\nabla_\theta f(\theta_n)|| \tag{59}$$

From $||A^{-1}|| = 1/\sigma_{\min}(A)$,

$$p_n^T H(\theta_n) p_n = (-(H(\theta_n))^{-1} g(\theta_n))^T H(\theta_n)(-(H(\theta_n))^{-1} g(\theta_n)) \tag{60}$$

$$= g(\theta_n)^T (H(\theta_n))^{-1} g(\theta_n) \tag{61}$$

$$\geq \frac{1}{c}||g(\theta_n)||^2 \tag{62}$$

$$\geq \frac{1}{c}(||\nabla_\theta f(\theta_n)|| - \epsilon)^2 \tag{63}$$

$$\geq \frac{4}{9c}||\nabla_\theta f(\theta_n)||^2 \tag{64}$$

From the assumption that f is γ-strongly convex, $\gamma I \preceq \nabla_\theta^2 - \ell \preceq \Gamma I$, by an implication of γ-strong convexity we have

$$f(\theta_n) - f(\theta^*) \leq \frac{1}{2\gamma}||\nabla_\theta f(\theta_n)||^2 \tag{65}$$

and we put together:

$$f(\theta_n) - f(\theta^*) \leq \frac{1}{2\gamma}||\nabla_\theta f(\theta_n)||^2 \leq \frac{9c}{4}\frac{1}{2\gamma} p_n^T H(\theta_n) p_n \tag{66}$$

$$\frac{8\gamma}{9c}(f(\theta_n) - f(\theta^*)) \leq \frac{4}{9c}||\nabla_\theta f(\theta_n)||^2 \leq p_n^T H(\theta_n) p_n \tag{67}$$

$$f(\theta_n) - f(\theta^*) \leq \frac{9c}{8\gamma} p_n^T H(\theta_n) p_n \tag{68}$$

$$-\frac{8\gamma}{9c}(f(\theta_n) - f(\theta^*)) \geq -\frac{4}{9c}||\nabla_\theta f(\theta_n)||^2 \geq -p_n^T H(\theta_n) p_n \tag{69}$$

But from earlier, as the Armijo condition is fulfilled with our choice of η and ϵ,

$$f(\theta_{n+1}) - f(\theta_n) \leq -\eta p_n^T H(\theta_n) p_n + \epsilon\eta||p_n|| + \eta^2\Gamma||p_n||^2/2 \tag{70}$$

$$\leq -\eta\beta p_n^T H(\theta_n) p_n \tag{71}$$

$$\leq -\eta\beta\frac{8\gamma}{9c}(f(\theta_n) - f(\theta^*)) \tag{72}$$

and therefore

$$f(\theta_{n+1}) - f(\theta^*) = f(\theta_{n+1}) - f(\theta_n) + f(\theta_n) - f(\theta^*) \tag{73}$$

$$\leq f(\theta_n) - f(\theta^*) - \eta\beta\frac{8\gamma}{9c}(f(\theta_n) - f(\theta^*)) \tag{74}$$

$$= (1 - \eta\beta\frac{8\gamma}{9c})(f(\theta_n) - f(\theta^*)) \tag{75}$$

□

7. Numerical Experiments

To validate the results, we perform a couple of numerical experiments. Both these examples feature state-space models where the process model is differentiable with the reparametrization trick. Future work will consider state-space models where one has to explicitly consider a surrogate loss as in Section 3.4. All experiments were performed with the JAX library [2].

7.1. Linear Gaussian State-Space Model

Consider a simple discrete-time linear dynamical system with Gaussian noise, where $\mathbf{x}_t$ is the state, A denotes the average process model transition, C is the observation matrix, $\nu_t \sim \mathcal{N}(\prime, \mathcal{Q})$ denotes the process noise, and $\epsilon_t \sim \mathcal{N}(0, R)$ denotes the observation noise. The likelihood for this model is nonconvex. This model evolves according to the following equations:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + \nu_t \tag{76}$$

$$\mathbf{y}_{t+1} = C\mathbf{x}_{t+1} + \epsilon_t \tag{77}$$

$$A = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_2) \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 0.01 & 0.000001 \\ 0.000001 & 0.01 \end{bmatrix}, R = \begin{bmatrix} 0.1 & 0.01 \\ 0.01 & 0.1 \end{bmatrix} \tag{78}$$

A trajectory for this linear-Gaussian state space model, displayed in the top-most plot in Figure 4, was simulated with $T = 40$, $\theta_1 = 0.2, \theta_2 = -0.5$. One can observe that while the states tend to oscillate around 0, the measurements are very noisy.

In the top-most plot in Figure 2, we provide a comparison of the performance of gradient descent with a fixed step size against Newton's method with Armijo line search. Both methods were initialized at $\hat{\theta}_1 = 0.5, \hat{\theta}_2 = -0.1$, and run with 100 particles. Unsurprisingly, Newton's method converges smoothly in fewer iterations than gradient descent, which oscillates wildly in the first few iterations before finding the minimum.

The top-most plot in Figure 3 showcases the near-convergence of parameter estimates with Newton's method to the ground truth, which on a single short trajectory of $T = 40$, is fairly convincing.

7.2. Stochastic Volatility Model

We perform another numerical experiment on the very same stochastic volatility model within the code repository provided by Corenflos et. al. [7], that Scibior and Wood [27] use. The model equations are:

$$x_0 \sim \mathcal{N}(0, \frac{\sigma_x^2}{1-\phi^2}) \tag{79}$$

$$x_{t+1} = \mu(1-\phi) + \phi x_t + \sigma_x \eta_t \tag{80}$$

$$y_t = \epsilon_t \exp(x_t/2) \tag{81}$$

$$\eta_t, \epsilon_t \sim \mathcal{N}(0, 1) \tag{82}$$

$$\theta = (\mu, \phi, \sigma_x) \tag{83}$$

Within this model, the state space consists of a single component, the volatility of the asset. The asset prices y_t are lognormal, The measurement model is nonlinear, and scaled by the observation noise.

Intuitively, there are three parameters. A long-run mean volatility $\mu > 0$, a mixing parameter $\phi \in (0, 1)$ determining how much of the next timestep's volatility is driven by the current timestep's volatility, and $\sigma_x > 0$, a scaling parameter for the magnitude of the process noise.

We simulate a trajectory visualized in Figure 4 with $T = 100$, $\mu = 2$, $\phi = 0.9$, and $\sigma_x = 1$. We initialize the optimization with $\hat{\mu} = 4$, $\hat{\phi} = 0.5$, $\hat{\sigma_x} = 1.5$, and just 10 particles.

In Figure 2, one can observe that Newton's method with line search once again outperforms gradient descent, converging in fewer iterations and remaining more stable. This time, gradient descent gets stuck on a local minimum and fails to escape it.

In Figure 3, we display the convergence of the parameters to the ground truth. Interestingly, like in Scibior and Wood, we initialize $\hat{\mu}$ 2 units away from μ and observe the convergence of the optimization procedure. Interestingly, every method they tested (including DPF, which should be exactly equivalent to our gradient descent implementation, and the optimal transport resampling from Corenflos et. al. [7]) took more than 500 iterations to get within one unit of μ, but both gradient descent and Newton's method in our experiments managed to pass that threshold within 10 iterations. Our learning rate for gradient descent was even slightly less aggressive, at 0.0025 v.s. their 0.01. This discrepancy could be explained by a different trajectory observed or different initialization of parameters.

8. Discussion

In this paper, we have:

1. Introduced a new framework for stochastic derivative estimates that fix a random seed
2. Derived the off-policy weighted particle filter and showcased some of its desirable properties, including the plug-and-play property
3. Proposed two methods for numerical optimization of the log-likelihood that take advantage of AD for the particle filter
4. Obtained high-probability bounds on the deviation of the gradient estimate and the smallest eigenvalue of the Hessian
5. Performed a finite-particle analysis of the convergence of Newton steps towards the optimum under a strong convexity assumption
6. Validated the methods and theory with numerical experiments

While we recognize that a lot of work still remains to be done, we are encouraged by the possibilities, theory, and empirical results presented.

8.1. Obstacles and Limitations

Equation (26) shows that the algorithmic derivative of the particle filter likelihood estimate implemented via the stop-gradient trick [27] does not have the pleasant mixing property enjoyed by the filtered likelihood estimate itself. The derivative estimate ends up depending only on the extant particles and their history: increasing the density at particles destined to go extinct does not affect the final estimate. One has to accept variance quadratic in horizon for the stochastic derivative estimates discussed here.

Alternative ways to make a particle filter differentiable, that modify the forward pass, may not have this problem. That is why Scibior and Wood [27] turn their attention to population SMC, and why Corenflos et.

FIG 2. Comparison of the convergence of gradient descent with a fixed learning rate against Newton's method with backtracking-Armijo line search, on the linear-Gaussian state-space model (top) and the stochastic volatility model (bottom).

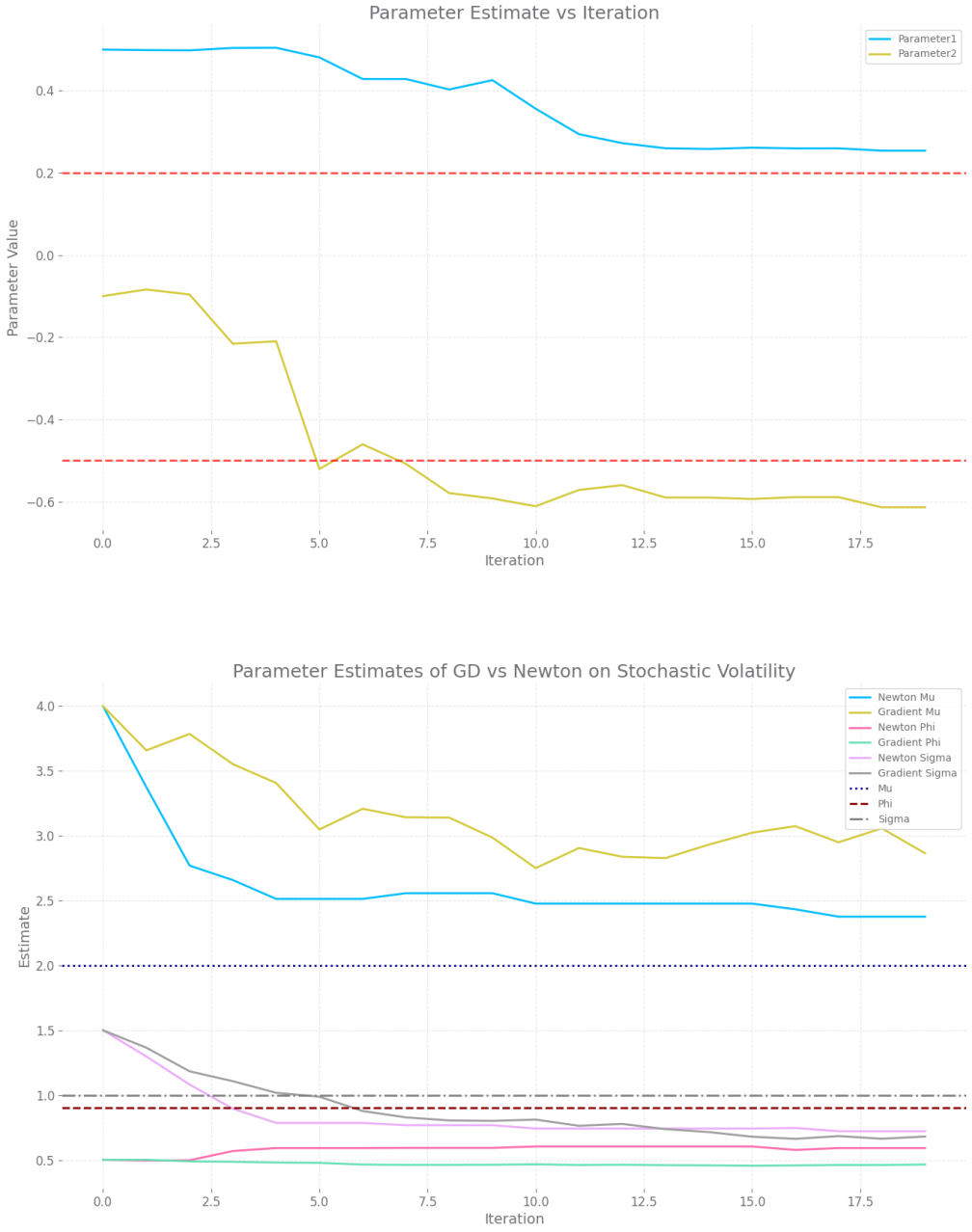

FIG 3. Near-convergence of parameter estimates on the linear-Gaussian state space model to the ground truth on a relatively short trajectory, $T = 40$, using Newton's method with backtracking-Armijo line search (top). The same, but with a comparison against gradient descent with $T = 100$ on the stochastic volatility model (bottom)

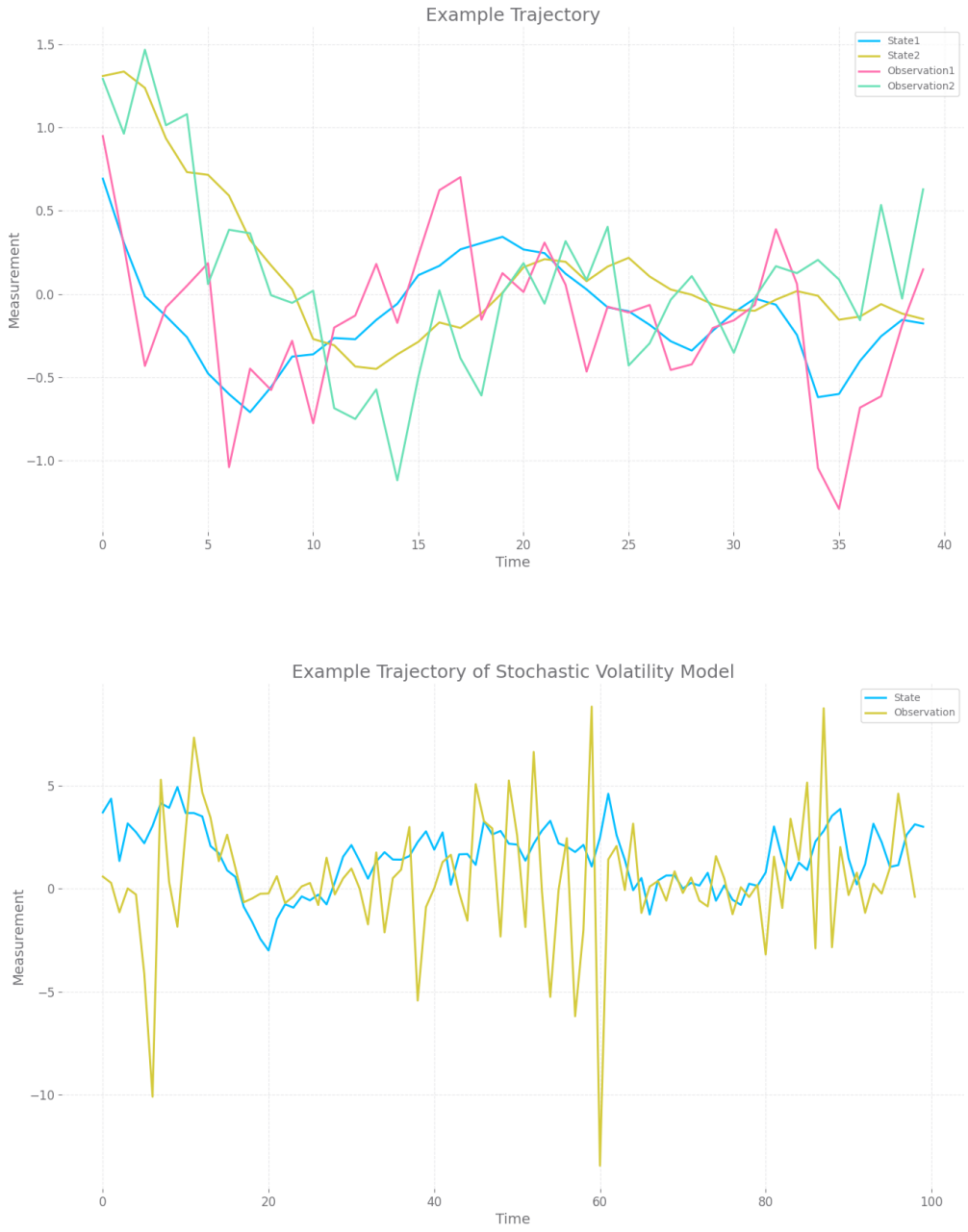

FIG 4. Sample trajectories for the linear-Gaussian state space model (top), and for the stochastic volatility model (bottom).

al. [7] resort to resampling via optimal transport. However, none of these are perfect. Both of these methods accept computation on the order of J^2, and the former loses the plug-and-play property. Alternatively, Malik and Pitt [16] propose smoothing the cumulative distribution function of the particles for the case where the latent state is continuous and one-dimensional – though this may not generalize suitably to discrete state spaces and higher dimensions. In special cases, such algorithms may exist [21, 22, 23], but their applicability may be limited.

We continue to search for an algorithmic derivative of the particle filter likelihood estimate that has (1) variance linear or sub-linear in horizon, (2) is computable in runtime linear in the number of particles, (3) does not modify the forward pass and the resulting likelihood estimate, and finally (4) satisfies the plug-and-play property.

8.2. Future Work

The actual integration of these methods into existing software packages for partially observed Markov processes like the 'pomp' package of King et. al. [14] remains an open problem. It is likely that the integration of auto-differentiation into these packages will be a difficult and far-from-trivial task. The question remains open as to whether an existing auto-differentiation package should be adopted as the backend, or whether one should implement one's own auto-differentiation library like the probabilistic programming Stan [5] has successfully managed to do.

The analysis and numerical validation of the proposed improvements to the improved iterated filtering algorithm of Ionides et. al. [11] Algorithm 4 has been left for future work. We remain optimistic about the proposed improvements, as intuitively replacing the random walk of the particle parameters with a more targeted proposal could certainly improve performance.

Many situations of interest, like the Haiti cholera models discussed in [26], have discrete state spaces and have simulators that are not differentiable with the reparametrization trick. This remains an important concern. Future work will have to consider state-space models where one has to explicitly consider a surrogate loss as in Section 3.4.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[3] Carles Bretó, Daihai He, Edward L. Ionides, and Aaron A. King. Time series analysis via mechanistic models. *Annals of Applied Statistics*, 3(1):319–348, 2009.

[4] Lucien Le Cam, Lucien Marie LeCam, and Grace Lo Yang. *Asymptotics in Statistics: Some Basic Concepts*. Springer Science Business Media, jul 2000.

[5] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of statistical software*, 76(1), 2017.

[6] Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6), dec 2004.

[7] Adrien Corenflos, James Thornton, George Deligiannidis, and Arnaud Doucet. Differentiable particle filtering via entropy-regularized optimal transport. 139:2100–2111, 18–24 Jul 2021.

[8] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *CoRR*, abs/1711.00123, 2017.

[9] Alex Graves. Stochastic backpropagation through mixture density distributions. *CoRR*, abs/1607.05690, 2016.

[10] Bretó C. Park J. Smith R. A. Ionides, Edward L. and Aaron A. King. Monte carlo profile confidence intervals for dynamic systems. *Journal of the Royal Society Interface*, 2017.

[11] Edward L. Ionides, Dao Nguyen, Yves Atchadé, Stilian Stoev, and Aaron A. King. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. 112(3):719 LP – 724, jan 2015.

[12] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. *CoRR*, abs/1805.11122, 2018.

[13] Aaron A. King, Edward L. Ionides, and Kidus Asfaw. Lesson 3: Likelihood for POMPs: theory and practice, 2021.

[14] Aaron A. King, Dao Nguyen, and Edward L. Ionides. Statistical inference for partially observed markov processes via the R package pomp. *Journal of Statistical Software*, 69:1–43, 2016.

[15] Jinlin Lai, Justin Domke, and Daniel Sheldon. Variational marginal particle filters, 2021.

[16] Sheheryar Malik and Michael K Pitt. Particle filters for continuous likelihood evaluation and maximisation. *Journal of Econometrics*, 165(2):190–209, 2011.

[17] Pierre Del Moral and Emmanuel Rio. Concentration inequalities for mean field particle models. *The Annals of Applied Probability*, 21(3), jun 2011.

[18] Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 968–977. PMLR, 09–11 Apr 2018.

[19] George Poyiadjis, Arnaud Doucet, and Sumeetpal S. Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 02 2011.

[20] Farbod Roosta-Khorasani and Michael W. Mahoney. Sub-sampled newton methods i: Globally convergent algorithms, 2016.

[21] Conor Rosato, Lee Devlin, Vincent Beraud, Paul Horridge, Thomas B Schön, and Simon Maskell. Efficient learning of the parameters of non-linear models using differentiable resampling in particle filters. *IEEE Transactions on Signal Processing*, 70:3676–3692, 2022.

[22] Conor Rosato, John Harris, Jasmina Panovska-Griffiths, and Simon Maskell. Inference of stochastic disease transmission models using particle-mcmc and a gradient based proposal. In *25th International Conference on Information Fusion (FUSION 2022)*, pages 1–8. IEEE, 2022.

[23] Angad Singh, Omar Makhlouf, Maximilian Igl, Joao Messias, Arnaud Doucet, and Shimon Whiteson. Particle-based score estimation for state space model learning in autonomous driving, 2022.

[24] Michalis K. Titsias and Jiaxin Shi. Double control variates for gradient estimation in discrete latent variable models, 2021.

[25] George Tucker, Andriy Mnih, Chris J. Maddison, and Jascha Sohl-Dickstein. REBAR: low-variance,

unbiased gradient estimates for discrete latent variable models. *CoRR*, abs/1703.07370, 2017.

[26] Jesse Wheeler, AnnaElaine L Rosengart, Zhuoxun Jiang, Kevin Tan, Noah Treutle, and Edward L Ionides. Informing policy via dynamic models: Cholera in Haiti. *arXiv:2301.08979*, 2023.

[27] Adam Ścibior and Frank Wood. Differentiable particle filtering without modifying the forward pass. 2021.

Appendix A: Checking equation (26) on a toy example

Consider a discrete, binary state space $\mathbb{X} = \{1, 2\}$, with matching observation space $\mathbb{Y} = \mathbb{X}$. The initial latent state probabilities are $\pi_i = \mathbb{P}(X_0 = i) = 1/2$ for $i = 1, 2$. All transitions have probability zero. The measurement model is defined by a matrix,

$$R_{ik} = f_{Y_n|X_n}(k \,|\, i\,; \theta) = \begin{pmatrix} 1 & 0 \\ 1/2+\theta & 1/2-\theta \end{pmatrix} \tag{84}$$

The data are $y_1^* = y_2^* = 1$. We have $J = 2$ and the seed is chosen such that $X_{1,1}^P = X_{2,1}^P = X_{2,2}^P = 1$ and $X_{1,2}^P = 2$. Note that there is just one parameter, θ, and it affects only the measurement model for state 2.

Both extant particles at the final time $N = 2$ are only ever in state 1. Thus, $f_{X_{1:N},Y_{1:N}}(X_{j,1:N}^A, y_{1:N}^*\,; \theta)$ has no dependence on θ for any $j = 1, 2$, and so the right hand side of (26) is zero.

The likelihood estimated a regular particle filter at $\theta = 0$ is

$$\begin{aligned} \mathcal{L}_{PF}(\theta) &= (1/2)[1 + (1/2 + \theta)] \times 1, & (85) \\ \nabla\mathcal{L}_{PF}(\theta) &= 3/4. & (86) \end{aligned}$$

To reason about the ADPF value computed by the DPF algorithm of **(author?)** [27], we interpet the stop as follows. First, we run the algorithm ignoring the stop directive at an initial value $\theta = \theta_0$. This corresponds to the forward pass. Then, the program is re-run at a different parameter, $\theta = \theta_0 + \delta$, with the stop values substituted in from the $\theta = \theta_0$ run. The derivative estimate at θ_0 is then

$$\lim_{\delta \to 0} \frac{\mathcal{L}_{DFP}(\theta_0 + \delta, \theta_0) - \mathcal{L}_{DFP}(\theta_0, \theta_0)}{\delta}. \tag{87}$$

Suppose for simplicity that DPF is implemented with resampling at each timestep. Set $\theta_0 = 0$. Then,

$$\begin{aligned} \text{stop}(\bar{v}_1^1, \bar{v}_2^1) &= (1, 1/2)/(1 + 1/2) = (2/3, 1/3) & (88) \\ (\bar{v}_1^1, \bar{v}_2^1) &= (1, 1/2 + \delta)/(1 + 1/2 + \delta) & (89) \\ \tilde{v}_2^1 = \tilde{v}_2^2 &= \frac{3}{4(1 + 1/2 + \delta)} & (90) \end{aligned}$$

Then, the likelihood estimate is

$$\mathcal{L}_{DFP}(0, \delta) = (1/2)[1 + 1/2 + \delta] \times \frac{2 \times 3}{4(1 + 1/2 + \delta)} = 3/4. \tag{91}$$

So, we have validated the identity in (26) for this example.

Appendix B: Directly Differentiating the Resampling

B.1. Outline

To propagate gradients back to model parameters, we first instantiate a set of particles, or copies of the model. At each step, we need (1) $\nabla_\theta \mathbb{E}_x[f(x)]$, or the gradient of 'dmeasure' of particle x with regard to model parameters. At each resampling step, we need (2) $\nabla_w \mathbb{E}_x[f(x)]$, or the gradient of 'dmeasure' of particle x with regard to particle weights. At each call to 'rprocess', we need (3) a differentiable simulator (or an estimate of its gradients). We can backpropagate gradients from the final loss calculation through each resampling step with regard to particle weights at time t, $w^t = w(\hat{\ell}_t(\theta)) = w^t(\theta)$, and through each simulator call in 'rprocess'.

B.2. Score Function Estimator for Multinomial Resampling of Particles

In an abuse of notation, let $f(x) := f(y|x)$ (or 'dmeasure'). Let $x^F_{n,j}$ be a sample from the categorical distribution over the particles with weights $w_{n,j}(\theta) = f(y^*_n|x^P_{n,j}, \theta)$ summing to 1 (or the filtering distribution at timestep n). Write $w^F_{n,j}(\theta) = f(y^*_n|x^F_{n,j}, \theta)$ As $x^F_{n,j} \sim \text{Categorical}(w_{n,j}(\theta))$, it can be thought of as taking values in $\{1, ..., J\}$. The score function estimator for the gradient (with regard to θ of the 'dmeasure' of resampled particle x) is given by

$$\nabla_\theta \mathbb{E}_{x^F_{n,j} \sim \text{Categorical}(w_{n,j}(\theta))}[f(x^F_{n,j})] \approx f(x^F_{n,j}) \nabla_\theta \log p(x^F_{n,j}; \theta) = f(x^F_{n,j}) \nabla_\theta \log w^F_{n,j}(\theta)$$

As the log-likelihood deals with the prediction particles,

$$\begin{aligned}
&\nabla_\theta \mathbb{E}_{x^P_{t,j} \sim \text{rprocess}(\text{Categorical}(w_{t-1,j}(\theta)))}[f(x^P_{t,j})] \\
&\approx \frac{1}{J} \sum_{j=1}^{J} \nabla_\theta f(x^P_{t,j}) \\
&\approx \frac{1}{J} \sum_{j=1}^{J} f(x^P_{t,j}) \nabla_\theta \log p(x^P_{t,j}; \theta) \\
&\approx \frac{1}{J} \sum_{j=1}^{J} f(x^P_{t,j}) \nabla_\theta \log w_{t,j}(\theta)
\end{aligned}$$

This is obtained by differentiating the surrogate loss

$$\frac{1}{J} \sum_{j=1}^{J} f(x^A_{t,j}) \log w(x^A_{t,j}; \theta) \tag{92}$$

The variance of this estimator is commonly thought to be high. In its vanilla form, like this, we can compute

$$\begin{aligned}
\text{Var}(f(x_j) \nabla_\theta \log p(x_j; \theta)) &= \text{Var}(f(x_j) \nabla_\theta \log p(x_j; \theta)) \\
&= \text{Var}(f(x_1) \nabla_\theta \log p(x_1; \theta)) \\
&= \mathbb{E}[(f(x_1) \nabla_\theta \log p(x_1; \theta))^2] - \mathbb{E}[f(x_1) \nabla_\theta \log p(x_1; \theta)]^2
\end{aligned}$$

as the (re)samples are drawn i.i.d. Fortunately the $f(x_j)$ are bounded between 0 and 1 (being probabilities), and so it seems most of the trouble will come from $\nabla_\theta \log w_{n,j}(\theta)$.

B.3. Variance Reduction Methods

Using standard control variate techniques is an option. What's highly correlated with 'dmeasure'? We could use a learned baseline and take advantage of the various advances made in stochastic gradient estimates for discrete latent variable models ([25], [8], [24]), but that comes with added complexity. Scibior and Wood [27] suggest using the mean of the forward pass over several iterations, which in our case would be the mean likelihood or log-likelihood from several iterations of the particle filter.

B.4. The Path Derivative with regard to Particle Weights

It is possible, even with a simulator that does not use the reparametrization trick, to convolve each particle with a normal distribution, therefore enabling one to differentiate the process model with the reparametrization trick.

Cconsider a Gaussian mixture, $p(x|\theta) = \sum_{j=1}^{N_p} w_j(\mu_j(\theta_j) + \varphi_j \cdot \sigma_j) = w^T(\mu(\theta) + \vec{\varphi} * \sigma)$, $\epsilon_j \sim \mathcal{N}(0, I)$. Let $f(x) := f(y|x)$ (or 'dmeasure'), and $x(\vec{\epsilon}, \theta)$ is a sample from the weighted Gaussian mixture around the particle locations $\mu_j(\theta)$ with weights w_j summing to 1. That is, $x_j \sim p(x|\theta)$, and unlike the earlier case with the SF estimator, takes values in $\mathbb{R}^{|\mathcal{S}|}$ instead of $\{1, ..., J\}$. The path derivative estimator for the gradient (with regard to $w(\theta)$ of the 'dmeasure' of resampled particle x) is given by

$$\begin{aligned}\frac{1}{J}\sum_{j=1}^{J} \nabla_w f(x(\vec{\epsilon}, w(\theta))) &= \frac{1}{J}\sum_{j=1}^{J} \nabla_x f(x)\ \nabla_w x(\vec{\epsilon}, w(\theta)) \\ &= \frac{1}{J}\sum_{j=1}^{J} [\sum_{d=1}^{D} \frac{\partial f(x)}{\partial x_d}\frac{\partial x_d}{\partial w_j}]_{1\le j\le J}\end{aligned}$$

where D is the dimensionality of the state space, and the second line is given by a result from Graves [9].

The coefficient in the above is fixed, except for the x dependent on $\epsilon_j \sim \mathcal{N}(0, I)$, each with variance I. Therefore, the only random variable (or collection thereof) driving the variance of the path derivative estimator is the ϵ_j.

For the variance of the path derivative estimator of the gradient with regard to the 'dmeasure' of resampled particle x not to be large, we require the variance of the gradient of 'dmeasure' with regard to the location of the resampled particle x, or $\nabla_x f(x)$, to not be large. As such, in our case, the variance when $J = 1$ is $\mathrm{Var}(\nabla_x f(x(\vec{\epsilon}, \theta))\ \nabla_w x(\vec{\epsilon}, w(\theta)))$.

Empirically, having large-dimensional latent states in VAEs has not presented as much of a problem with reparametrization trick backpropagation variance as it has with overfitting, which is encouraging.

The above idea is somewhat reminiscent of another paper in the literature, Lai et. al. [15] on variational marginal particle filters. Their method on unbiased gradients with implicit reparametrization in the context of the variational marginal particle filter also uses the techniques of Graves [9], drawing each (prediction) particle for the next timestep from a mixture distribution of the process densities of the (filtering) particles at the current timestep, with weights relative to the particle likelihood. Doing so, however, (1) relies on the process density, losing the plug-and-play property, and (2) is in the context of the marginal particle filter.