A Nearly-Black-Box Online Algorithm for Joint Parameter and State Estimation in Temporal Models

Yusuf B. Erol[*]
EECS, UC Berkeley
yberol@eecs.berkeley.edu

Yi Wu[*]
EECS, UC Berkeley
jxwuyi@eecs.berkeley.edu

Lei Li
Toutiao Lab
lileicc@gmail.com

Stuart Russell
EECS, UC Berkeley
russell@eecs.berkeley.edu

Abstract

Online joint parameter and state estimation is a core problem for temporal models. Most existing methods are either restricted to a particular class of models (e.g., the Storvik filter) or computationally expensive (e.g., particle MCMC). We propose a novel nearly-black-box algorithm, the Assumed Parameter Filter (APF), a hybrid of particle filtering for state variables and assumed density filtering for parameter variables. It has the following advantages: (a) it is online and computationally efficient; (b) it is applicable to both discrete and continuous parameter spaces with arbitrary transition dynamics. On a variety of toy and real models, APF generates more accurate results within a fixed computation budget compared to several standard algorithms from the literature.

Introduction

Many problems in scientific studies and real-world applications involve modeling of dynamic processes, which are often modeled by temporal models, namely state space models (SSMs) (Elmohamed, Kozen, and Sheldon 2007; Arora et al. 2010). Online parameter and state estimation –computing the posterior probability for both (static) parameters and (dynamic) states, incrementally over time– is crucial for many applications such as simultaneous localization and mapping (Montemerlo et al. 2002), object tracking (Ristic, Arulampalam, and Gordon 2004) and 3D design suggestion (Ritchie et al. 2015).

Sequential Monte-Carlo (particle filter) based algorithms have been introduced for real-world applications (Gordon, Salmond, and Smith 1993; Arulampalam et al. 2002; Cappé, Godsill, and Moulines 2007). However, classical particle filter algorithms suffer from the path degeneracy problem, especially for parameters, and leave it a challenge to jointly estimate parameters and states for SSMs with complex dependencies and nonlinear dynamics. Real-world models can involve both discrete and continuous variables, arbitrary dependencies and a rich collection of nonlinearities and distributions. Existing algorithms are either restricted to a particular class of models, such as the Storvik filter (Storvik 2002) and the Liu-West filter (Liu and West 2001), or very expensive in time complexity, such as particle MCMC (Andrieu, Doucet, and Holenstein 2010), which utilizes an expensive MCMC kernel over the parameter space and typically requires a very large number of MCMC iterations to converge.

In this paper, we propose a practical algorithm for the general combined parameter and state estimation problem in SSMs. Our algorithm, called the Assumed Parameter Filter (APF), is a hybrid of particle filtering for state variables and assumed density filtering for parameter variables. It projects the posterior distribution for parameters, $p(\theta \mid x_{0:t}, y_{0:t})$, into an approximation distribution and generates samples of parameters in constant-time per update. APF streams data and is a nearly-black-box algorithm: when an appropriate approximate distribution for the parameter is chosen, APF can be applied to any SSM that one can sample from and compute evidence likelihood. We emphasize the nearly-black-box property of APF by developing it as an automatic inference engine for a probabilistic programming language. Experiment results show that APF converges much faster than existing algorithms on a variety of models.

Background

State space models: A state space model (SSM) consists of the parameters $\Theta \in \mathbb{R}^p$, latent states $\{X_t\}_{0\leq t\leq T} \in \mathbb{R}^d$ and the observations $\{Y_t\}_{0\leq t\leq T} \in \mathbb{R}^n$ defined by

$$\begin{array}{cc} \Theta \sim f_1(\theta) & X_0 \sim f_2(x_0) \\ X_t \mid x_{t-1}, \theta \sim g(x_t \mid x_{t-1}, \theta) & Y_t \mid x_t, \theta \sim h(y_t \mid x_t, \theta) \end{array}$$

where f_i, g, h denote some arbitrary probability distributions (the model). Given an SSM, the goal of the joint parameter and state estimation is to compute the posterior distribution of both the states (i.e., $\{X_t\}$) and the parameters (i.e., Θ) given the observations. In the filtering setting, we aim to compute the posterior of Θ and X_t for every time t based on evidence until time t, namely $p(x_t, \theta \mid y_{0\ldots t})$.

Inference Algorithms: Sequential Monte Carlo (SMC) is a widely adopted class of methods for inference on SSMs. Given the observed values $Y_{0:T} = y_{0:T}$, the posterior distribution $p(x_{0:t}, \theta|y_{0:t})$ is approximated by a set of K particles, with each particle k denoted by X_t^k for $1 \leq k \leq K$. X_t^k consists of a particular assignment of the states and the parameter denoted by $x_{0:t}^k$ and θ^k. Particles are propagated forward through the transition model $g(x_t^k|x_{t-1}^k, \theta^k)$ and re-

[*] The first two authors contributed equally to this work.

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

sampled at each time step t according to the weights w_t^k:

$$X_t^k | X_{t-1}^k \sim g(x_t^k | x_{t-1}^k, \theta^k), \quad w_t^k = h(y_t | x_t^k, \theta^k).$$

At the propagation step, we assume that the propagation proposal is the transition probability for conciseness. The choice of proposal is orthogonal to the focus of this paper.

The most critical issue for static parameter estimation via the classical particle filter (PF) is the *sample impoverishment problem*. Due to the resampling step, which "kills" the particles with low weights, the diversity of the Θ-particles reduces at every time step (parameter particles are only sampled once at the beginning of the inference procedure). Hence, K needs to be sufficiently large to prevent an early convergence to a degenerate particle set. Thus, PF is mostly used in the cases where only state estimation is required.

For joint parameter and state estimation, the "gold standard" approaches are particle Markov chain Monte Carlo (PMCMC) algorithms (Andrieu, Doucet, and Holenstein 2010), such as particle independent Metropolis-Hastings (PIMH), particle marginal Metropolis-Hastings (PMMH), particle Gibbs (PGibbs) (Andrieu, Doucet, and Holenstein 2010) and particle Gibbs with ancestor resampling (PGAS) (Lindsten, Jordan, and Schön 2014). PMCMC algorithms utilize an MCMC transition kernel over the parameter space and a classical particle filter for state estimation and likelihood computation. PMCMC methods are favored due to their theoretical guarantees as an unbiased estimator as well as their "black-box" property: the only requirement for PMCMC methods is that one needs to sample from the SSM and compute likelihood for the evidence, which is in most cases straightforward.

However, one significant drawback of PMCMC algorithms is the computational budget. Suppose there are T time steps and we perform N MCMC steps with K particles. Then the time complexity for PMCMC algorithms is $O(NKT)$. Note that for adequate mixing, it is necessary for N to be sufficiently large. For a real-world application with a large number of time steps and complex dynamics, the mixing problem becomes critical. Moreover, since PMCMC algorithms require multiple sweeps over observations, T must be fixed in advance and the full history of the particles must be stored. This "offline" property of PMCMC algorithms is infeasible for online/streaming applications, such as real-time object tracking and signal monitoring, for which constant time per update is required and storing the whole history is prohibitive.

There are also online algorithms for joint parameter and state estimation problems. However, existing algorithms are either computationally inefficient or only suitable for a restricted domain of models. The resample-move algorithm (Gilks and Berzuini 2001) utilizes kernel moves that target $p(\theta, x_{0:t} \mid y_{0:t})$ as its invariant. However, this method requires $O(t)$ computation per time step, leading Gilks and Berzuini to propose a move at a rate proportional to $1/t$ so as to have asymptotically constant-time updates. Sampling from $p(\theta \mid x_{0:t}, y_{0:t})$ at each time step for models with fixed-dimensional sufficient statistics has also been proposed by several authors (Storvik 2002; Lopes et al. 2010). However, the sufficient statistics assumption restricts the applicable models. The extended parameter filter (Erol et al. 2013), which generates approximate sufficient statistics via polynomial approximation overcomes this issue, however, polynomial approximations are often hard to derive for models with complicated transitions. Lastly, there are other related works, such as the nested particle filter (Crisan and Miguez 2013), which is a recursive implementation of the offline SMC^2 (Chopin, Jacob, and Papaspiliopoulos 2013) algorithm, and the *artificial dynamics* approach (Liu-West filter, LW) (Liu and West 2001), which diversifies the parameter particles by introducing random perturbation and applies only to continuous variables.

The Assumed Parameter Filter (APF)

The design principles of APF are to inherit the appealing properties of the classical particle filter, which is applicable to arbitrary transitions and suitable for streaming data, while better overcoming the path degeneracy problem for parameter estimation without an expensive MCMC kernel.

We propose a nearly-black-box algorithm, called the Assumed Parameter Filter (APF), for online parameter and state estimation. In APF, the posterior of both states and parameters are jointly represented by K particles. The key point is that on the contrary to the bootstrap filter which keeps a static parameter value in each particle, APF maintains an extra approximate distribution and samples from that distribution for the parameter at each time step.

In order to fight against sample impoverishment, for parameter θ_t^k at time t in particle k, we sample from a distribution $q_t^k(\theta)$ in some parametric family $\mathcal{Q}$ where $q_t^k(\theta)$ is the approximate representation of the true particle posterior $p(\theta \mid x_{0:t}^k, y_{0:t})$. In the special case where q is a delta function, APF recovers the bootstrap particle filter (proven in the supplementary material). In order to obtain the approximating distribution q_t^k from q_{t-1}^k, M additional Monte Carlo samples are utilized for each particle to perform the moment-matching operations required for assumed density approximation. The proposed method is illustrated in Alg. 1.

```
Algorithm 1: Assumed Parameter Filter
Input: $y_{0:T}$, $\mathcal{Q}$, $K$, and $M$, the model $(f_1, f_2, g, h)$
Output: Samples $\left\{x_{0:T}^k, \theta_T^k\right\}_{k=1}^K$
Initialize $\left\{x_0^k, q_0^k(\theta)\right\}_{k=1}^K$ according to $f_1$, $f_2$ and $\mathcal{Q}$;
for $t = 1, \ldots, T$ do
    for $k = 1, \ldots, K$ do
        sample $\theta_t^k \sim q_{t-1}^k(\theta) \approx p(\theta \mid x_{0:t-1}^k, y_{0:t-1})$;
        sample $x_t^k \sim g(x_t \mid x_{t-1}^k, \theta_t^k)$ ;
        $w_t^k \leftarrow h(y_t \mid x_t^k, \theta_t^k)$;
        $q_t^k(\theta) \leftarrow \mathrm{Update}(M, \mathcal{Q}; q_{t-1}^k(\theta), x_t^k, x_{t-1}^k, y_t)$;
    sample $\left\{\frac{1}{N}, \bar{x}_t^k, \bar{q}_t^k\right\} \sim$ Multinomial$\left\{w_t^k, x_t^k, q_t^k\right\}$;
    $\left\{x_t^k, q_t^k\right\} \leftarrow \left\{\bar{x}_t^k, \bar{q}_t^k\right\}$;
```

Following the assumed density filtering framework, we are approximating $p(\theta \mid x_{0:t}, y_{0:t})$ in a parametric distribution family $\mathcal{Q}$. In our algorithm this is expressed through the Update function. The Update function generates the approximating density q via minimizing the KL-divergence between the target p and the approximating distribution q.

Approximating $p(\theta \mid x_{0:t}, y_{0:t})$

At each time step with each new incoming data point we approximate the posterior distribution by a tractable and compact distribution from $\mathcal{Q}$. Our approach is inspired by assumed density filtering (ADF) for state estimation (Lauritzen 1992; Boyen and Koller 1998).

For our application, we are interested in approximately representing $p(\theta \mid x_{0:t}, y_{0:t})$ in a compact form that belongs to a family of distributions. Due to the Markovian structure of the SSM, the posterior can be factorized as

$$\begin{aligned} &p(\theta \mid x_{0:t}, y_{0:t}) \propto \textstyle\prod_{i=0}^{t} s_i(\theta), \\ &s_i(\theta) = \begin{cases} p(\theta)p(y_0 \mid x_0, \theta), i = 0 \\ p(x_i \mid x_{i-1}, \theta)p(y_i \mid x_i, \theta), i \geq 1 \end{cases} \end{aligned} .$$

Let us assume that at time step $i-1$ the posterior was approximated by $q_{i-1} \in \mathcal{Q}$. Then with incorporation of (x_i, y_i), the posterior $\hat{p}$ will be

$$\hat{p}(\theta \mid x_{0:i}, y_{0:i}) = \frac{s_i(\theta)q_{i-1}(\theta)}{\int_\theta s_i(\theta)q_{i-1}(\theta)d\theta}. \tag{1}$$

For most models, $\hat{p}$ will not belong to $\mathcal{Q}$. ADF projects $\hat{p}$ into $\mathcal{Q}$ via minimizing KL-divergence:

$$q_i(\theta) = \arg\min_{q \in \mathcal{Q}} D\left(\hat{p}(\theta \mid x_{0:i}, y_{0:i}) \mid\mid q(\theta)\right) \tag{2}$$

For $\mathcal{Q}$ in the exponential family, minimizing the KL-divergence reduces to moment matching (Seeger 2005). For $q_i(\theta) \propto \exp\left\{\gamma_i^T m(\theta)\right\}$, where γ_i is the canonical parameter and $m(\theta)$ is the sufficient statistic, we compute moments of the one-step ahead posterior $\hat{p}$ and update γ_i to match.

$$\begin{aligned} g(\gamma_i) &= \int m(\theta)q_i(\theta)d\theta = \int m(\theta)\hat{p}(\theta)d\theta \\ &\propto \int m(\theta)s_i(\theta)q_{i-1}(\theta)d\theta \end{aligned}$$

where $g(\cdot)$ is the unique and invertible link function. Thus, for the exponential family, the Update function computes the moment matching integrals to update the canonical parameters of $q_i(\theta)$. For the general case, where these integrals may not be tractable, we propose approximating them by a Monte Carlo sum with M samples, sampled from $q_{i-1}(\theta)$:

$$Z \approx \tfrac{1}{M}\textstyle\sum_{j=1}^{M} s_i(\theta^j), \quad g(\gamma_i) \approx \tfrac{1}{MZ}\textstyle\sum_{j=1}^{M} m(\theta^j)s_i(\theta^j)$$

where $\theta^j \sim q_{i-1}(\theta)$. In our framework, this approximation is done for all particle paths $x_{0:i}^k$ and the corresponding q_{i-1}^k, hence leading to $O(KM)$ sampling operations per time step.

Asymptotic performance for APF

In a similar spirit to (Opper and Winther 1998), we prove that assumed density filtering framework can successfully converge to the target posterior with increasing amount of data. For simplicity, we only consider continuous parameters and use Gaussian as the approximation distribution. We assume an *identifiable* model (posterior is asmptotically Gaussian around the true parameter) and also assume that in the model, only the transition is parametrized by the parameter θ while the observation model is known.

Theorem 1. *Let $(f_1, f_2, g_\theta, h_\theta)$ be an identifiable Markovian SSM, and let $\mathcal{Q}$ be multivariate Gaussian. The KL divergence between $p(\theta \mid x_{0:t}, y_{0:t})$ and the assumed density $q_t(\theta)$ computed as explained in the previous subsection will converge to zero as $t \to \infty$.*

$$\lim_{t\to\infty} D_{\mathrm{KL}}\left(p(\theta \mid x_{0:t}, y_{0:t}) || q_t(\theta)\right) = 0. \tag{3}$$

The proof is given in the supplementary material. The theorem states that the error due to the projection diminishes in the long-sequence limit. Therefore, with $K, M \to \infty$, APF would produce samples from the true posterior distribution $p(\theta, x_t \mid y_{0:t})$. For finite K, however, the method is susceptible to path degeneracy.

Similar to (Storvik 2002; Lopes et al. 2010), we are sampling from $p(\theta \mid x_{0:t}^i, y_{0:t})$ at each time step to fight against sample impoverishment. It has been discussed before that these methods suffer from *ancestral path degeneracy* (Chopin et al. 2010; Lopes et al. 2010; Poyiadjis, Doucet, and Singh 2011). For any number of particles and for a large enough n, there exists some $m < n$ such that $p(x_{0:m} \mid y_{0:n})$ is represented by a single unique particle. For dynamic models with long memory, this will lead to a poor approximation of sufficient statistics, which in turn will affect the posterior of the parameters. In (Poyiadjis, Doucet, and Singh 2011), it has been shown that even under favorable mixing assumptions, the variance of an additive path functional computed via a particle approximation grows quadratically with time. To fight against path degeneracy, one may have to resort to *fixed-lag smoothing* or *smoothing*. Olsson et al. used fixed-lag smoothing to control the variance of the estimates (Olsson et al. 2008). Del Moral et al. proposed an $O(K^2)$ per time step forward smoothing algorithm which leads to variances growing linearly with t instead of quadratically (Del Moral, Doucet, and Singh 2010). Poyiadjis et al. similarly proposed an $O(K^2)$ algorithm that leads to linearly growing variances (Poyiadjis, Doucet, and Singh 2011). These techniques can be augmented into the APF framework to overcome the path degeneracy problem for models with long memory.

Special cases: Gaussians, mixtures of Gaussians and discrete distributions

Gaussian case: For a multivariate Gaussian $\mathcal{Q}$, explicit recursions can be derived for $\hat{p}(\theta) \propto s_i(\theta)q_{i-1}(\theta)$ where $q_{i-1}(\theta) = \mathcal{N}(\theta; \mu_{i-1}, \Sigma_{i-1})$. The moment matching recursions are

$$\begin{aligned} \mu_i &= \tfrac{1}{Z}\textstyle\int \theta s_i(\theta)q_{i-1}(\theta)d\theta, \\ \Sigma_i &= \tfrac{1}{Z}\textstyle\int \theta\theta^T s_i(\theta)q_{i-1}(\theta)d\theta - \mu_i\mu_i^T. \end{aligned} \tag{4}$$

where $Z = \int \hat{p}(\theta)d\theta = \int s_i(\theta)q_{i-1}(\theta)d\theta$. These integrals can be approximated via Monte Carlo summation as previously described. One alternative is *deterministic sampling*. Since q is multivariate Gaussian, Gaussian quadrature rules can be utilized. In the context of expectation-propagation this has been proposed by (Zoeter and Heskes 2005). In the context of Gaussian filtering, similar quadrature ideas have been applied as well (Huber and Hanebeck 2008).

For an arbitrary polynomial $f(x)$ of order up to $2M-1$, $\int f(x)e^{-x^2}dx$ can be calculated exactly via Gauss-Hermite quadrature with M quadrature points. Hence, the required moment matching integrals in Eq.(4) can be approximated arbitrarily well by using more quadrature points. The unscented transform (Julier and Uhlmann 2004) is one specific Gaussian quadrature rule that uses $M = 2d$ *deterministic* samples to approximate an integral involving a d-dimensional multivariate Gaussian. In our case these samples are: $\forall 1 \leq j \leq d$,

$$\theta_j = \mu_{i-1} + \left(\sqrt{d\Sigma_{i-1}}\right)_j, \quad \theta_{d+j} = \mu_{i-1} - \left(\sqrt{d\Sigma_{i-1}}\right)_j .$$

where $(\cdot)_j$ means the jth column of the corresponding matrix. Then, one can approximately evaluate the moment matching integrals as follows:

$$\begin{array}{c} Z \approx \frac{1}{2d}\sum_{j=1}^{2d} s_i(\theta^j), \;\; \mu_i \approx \frac{1}{2dZ}\sum_{j=1}^{2d}\theta^j s_i(\theta^j), \\ \Sigma_i \approx \frac{1}{2dZ}\sum_{j=1}^{2d}\theta^j(\theta^j)^T s_i(\theta^j) - \mu_i\mu_i^T . \end{array}$$

Mixtures of Gaussians: Weighted sums of Gaussian probability density functions can be used to approximate another density function arbitrarily closely. Gaussian mixtures have been used for state estimation since the 1970s (Alspach and Sorenson 1972).

Let us assume that at time step $i-1$ it was possible to represent $p(\theta \mid x_{0:i-1}, y_{0:i-1})$ as a mixture of Gaussians with L components.

$$\begin{aligned} p(\theta \mid x_{0:i-1}, y_{0:i-1}) &= \sum_{m=1}^{L} \alpha_{i-1}^m \mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m) \\ &= q_{i-1}(\theta) \end{aligned}$$

Given x_i and y_i;

$$\hat{p}(\theta \mid x_{0:i}, y_{0:i}) \propto \sum_{m=1}^{L} \alpha_{i-1}^m s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)$$

The above form will not be a Gaussian mixture for arbitrary s_i. We can rewrite it as:

$$\hat{p} \propto \sum_{m=1}^{L} \alpha_{i-1}^m \beta^m \frac{s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)}{\beta^m} \tag{5}$$

where the fraction is to be approximated by a Gaussian via moment matching and the weights are to be normalized. Here, each $\beta^m = \int s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta$ describes how well the m-th mixture component explains the new data. That is, a mixture component that explains the new data well will get up-weighted and vice versa. It is important to note that the proposed approximation is not exactly an ADF update. The problem of finding a mixture of fixed number of components to minimize the KL divergence to a target distribution is intractable (Hershey and Olsen 2007). The proposed update here is the one that matches the mean and covariance.

The resulting approximated density would be $q_i(\theta) = \sum_{m=1}^{K} \alpha_i^m \mathcal{N}(\theta; \mu_i^m, \Sigma_i^m)$ where the recursion for updating each term is as follows:

$$\begin{aligned} \beta^m &= \int s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta \\ \alpha_i^m &= \frac{\alpha_{i-1}^m \beta^m}{\sum_\ell \alpha_{i-1}^\ell \beta^\ell} \\ \mu_i^m &= \frac{1}{\beta^m}\int \theta s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta \\ \Sigma_i^m &= \frac{1}{\beta^m}\int \theta\theta^T s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta - \mu_i^m(\mu_i^m)^T \end{aligned}$$

Similar to the Gaussian case, the above integrals are generally intractable. Either a Monte Carlo sum or a Gaussian quadrature rule can be utilized to approximately update the means and covariances.

Discrete parameter spaces: Let us consider a d-dimensional parameter space where each parameter can take at most N_θ values. For discrete parameter spaces, one can always track $p(\theta \mid x_{0:t}, y_{0:t})$ exactly with a constant-time update; the constant, however, is exponential in d (Boyen and Koller 1998). Hence, tracking the sufficient statistics becomes computationally intractable with increasing dimensionality. For discrete parameter spaces we propose projection onto a fully factorized distribution, i.e., $q_i(\theta) = \prod_j q_{j,i}(\theta_j)$. For this choice, minimizing the KL-divergence reduces to matching marginals:

$$\begin{array}{c} Z = \sum_\theta s_i(\theta)q_{i-1}(\theta) \\ q_{j,i}(\theta_j) = \frac{1}{Z}\sum_{\theta\setminus\theta_j} s_i(\theta)q_{i-1}(\theta). \end{array}$$

Computing these summations is intractable for high-dimensional models, hence we propose using Monte Carlo summation. In the experiments, we consider a simultaneous localization and mapping problem with a discrete map.

Discussions

Applicability: APF follows the framework of particle filtering and the only requirement for updating the approximate distribution is being able to compute the likelihood of the states conditioned on the sampled parameter. Thus, APF can be applied to the same category of models as the particle filter. The critical issue for APF is the choice of the family of approximation distributions $\mathcal{Q}$. Although Gaussian mixtures can arbitrarily approximate any density, different forms of $\mathcal{Q}$ can significantly improve the practical performance. For example, when the dimensionality of the parameter space is large, one may want to use a diagonal Gaussian distribution for fast convergence; for non-negative parameters, the gamma distribution may be favored. Note that different $\mathcal{Q}$s yield different updating formulae. To this perspective, APF is nearly-black-box: a user can apply APF to arbitrary SSMs just by choosing an appropriate approximating distribution. In the next section, we further explore the nearly-black-box property of APF by adapting it to the backend inference engine of a probabilistic programming language.

Modularity: One can utilize a Storvik filter for a subset of parameters with fixed-dimensional sufficient statistics, and for the rest of the parameters, approximate sufficient statistics can be generated via the APF framework. This is similar to the extended Liu-West filter (Rios and Lopes 2013) where a Storvik filter is used in conjunction with the artificial dynamics approach.

Complexity: The time complexity of APF is $O(MKT)$ over T time steps for K particles with M extra samples to update the sufficient statistics through the moment matching integrals. Setting K and M adequately is crucial for performance. Small K prevents APF exploring the state space sufficiently whereas small M leads to inaccurate sufficient statistics updates which will in turn result in inaccurate parameter estimation.

Note that the typical complexity of PMCMC algorithms is $O(NKT)$ where N denotes the number of MCMC samples. Although in the same order of APF for time complexity, we find in practice that M is often orders of magnitude smaller than N for achieving a given level of accuracy. PMCMC algorithms often requires a large amount of MCMC iterations for mixing properly while very small M is sufficient for APF to produce accurate parameter estimation, especially for the Gaussian case as discussed above.

Moreover, the actual running time for APF is often much smaller than its theoretical upper bound $O(MKT)$. Notice that the approximation computation in APF only requires the local data in a single particle and does not influence the weight of that particle. Hence, one important optimization specialized for APF is to resample all the particles prior to the update step and only update the approximate distribution for those particles that do not disappear after resampling. It is often the case that a small fraction of particles have significantly large weights. Consequently, in our experiment, the actual running time of APF is several times faster than the theoretically required time $O(MKT)$ (see practical performances of APF in the Experiment section).

Lastly, the space complexity for APF is in the same order as the bootstrap particle filter, namely $O(K)$. Overall, APF is constant time and memory per update and hence fits into online/streaming applications.

Using APF in probabilistic programming

This section shows APF can be integrated into a probabilistic programming language (PPL), BLOG (Milch et al. 2005), from which the general research community can benefit. PPLs aim to allow users to express an arbitrary Bayesian model via a probabilistic program while the backend engine of PPL automatically performs black-box inference over the model. PPLs largely simplify the development process of AI applications involving rich domain knowledge and have led to many successes, such as the human-level concept learning (Lake, Salakhutdinov, and Tenenbaum 2015) and the 3D scene perception (Kulkarni et al. 2015).

We developed a compiled inference engine, the State and Parameter Estimation Compiler (SPEC), utilizing APF under BLOG (Milch et al. 2005) thanks to its concise syntax (Li and Russell 2013): in the BLOG language, state variables are those indexed by timestep, while all other variables are effectively parameters; thus, by identifying the static and dynamic variables, the SPEC compiler can automatically work out how to apply APF to the filtering problem. Since APF is based on the BLOG language, the general compilation process is based on the Swift compiler for BLOG (Wu et al. 2016). There are also other compiled PPL systems, such as Church (Goodman et al. 2008) and Anglican (Wood, van de Meent, and Mansinghka 2014). However, these systems do not have any language primitives distinguishing state and parameter. Potentially APF can be also applied to these systems by adding new syntax for declaring the parameter.

In order to demonstrate the online property of APF, SPEC also includes some extended syntax to allow streaming applications. The following BLOG program describes a simple SSM, the SIN model:

$$X_0 \sim \mathcal{N}(0,1) \qquad \Theta \sim \mathcal{N}(0,1)$$
$$X_t \sim \mathcal{N}(\sin(\theta x_{t-1}),1) \quad Y_t \sim \mathcal{N}(x_t, 0.5^2)$$

```
1  // parameter
2  random Real theta ~ Gaussian(0,1);
3  // state X(t)
4  random Real X(Timestep t) ~
5    // initial value, X(0)
6    if t == @0 then Gaussian(0, 1)
7    // transition
8    else Gaussian(sin(theta * X(t - @1)), 1);
9  // observed variable Y(t)
10 random Real Y(Timestep t)~Gaussian(X(t),0.25);
11 // user declared C++ function
12 extern Real loadData(Timestep t);
13 // observations
14 obs Y(t) = loadData(t) for Timestep t;
15 // query states and the parameter
16 query X(t) for Timestep t;
17 query theta;
```

The keyword `random` declares random variables in the model: those with an argument of type `Timestep` are states (dynamic variables, i.e., `X(t)` and `Y(t)`) while others are parameters (static variables, i.e., `theta`). Line 14 states that `Y(t)` is observed while line 16 and 17 query the posterior distribution of the state `X(t)` at each time step and the parameter `theta`.

In SPEC, we extend the original syntax of BLOG by (1) introducing a new keyword `extern` (Line 12) to import arbitrary customized C++ functions (e.g., input functions for streaming data at each time step) and (2) the for-loop style observation statement (Line 14) and query statement (Line 16, 17). These changes allow APF to be applied in a completely online fashion.

A user can utilize SPEC to perform inference with APF for both $\{X_t\}$ and Θ by simply coding this tiny program without knowing algorithm details. SPEC automatically analyzes the parameters, selects approximate distributions and applies APF to this model. By default, we use Gaussian distributions with Gauss-Hermite quadratures for continuous parameters and factored categorical distributions for discrete parameters. SPEC is extensible for more approximate distributions for further development. Moreover, due to the memory efficiency of APF, many other optimizations from the programming language community can be applied to further accelerate the practical performance of APF.[1]

Experiments

We evaluated APF on three benchmark models: 1. SIN: a nonlinear dynamical model with a single continuous parameter; 2. SLAM: a simultaneous localization and Bayesian map learning problem with 20 discrete parameters; 3. BIRD: a 4-parameter model to track migrating birds with real-world data. We compare the estimation accuracy of APF, as a function of run time, against the Liu-West filter (LW) and PMCMC algorithms including particle marginal Metropolis-Hastings (PMMH), particle Gibbs (PGibbs) and particle Gibbs with ancestor sampling (PGAS). For implementation, APF and LW are natively supported by our automatic engine SPEC. PMMH is manually adapted from the code compiled by SPEC. For PGibbs and PGAS, we compare both the code generated the Anglican compiler (Wood, van de Meent, and Mansinghka 2014), and our customized implementations.

Toy nonlinear model (SIN)

We consider the SIN model in the previous section with the true parameter $\theta^\star = 0.5$. 5000 data points are generated to ensure a sharp posterior. Notice that it is not possible to use the Storvik filter (Storvik 2002) or the particle learning algorithm (Lopes et al. 2010) for this model as sufficient statistics do not exist for Θ.

We evaluate the mean squared error over 10 trials between the estimation results and $\theta^\star$ within a fixed amount of time. For APF and LW, we consider the mean of the samples for Θ at the last time step for parameter estimation while for PMCMC algorithms, we take the average of the last half of the samples and leave the first half as burn-in. We omit PGAS results by Anglican here since Anglican takes more than 10 minutes to produce a sample with 100 particles.

For APF, we choose Gaussian as the approximate distribution with $M = 7$. For PMMH, we use a local truncated Gaussian as the proposal distribution for Θ.

Note that for PGAS and PGibbs, we need to sample from $\Pr[\Theta | X_{1:T}, Y_{1:T}]$ while this cannot be efficiently computed in SIN. As a black-box inference system, the Anglican compiler avoids this by treating every variable as state variable. In our customized implementations, we use a piecewise linear function to approximate $\Pr[\Theta | X_{1:T}, Y_{1:T}]$[2].

The results for black-box algorithms, including APF, Liu-West filter (supported by SPEC), PMMH (adapted from the code by SPEC) and PMCMC algorithms (PGibbs and PGAS in Anglican) are shown in Fig. 1(a). We also compare APF against out customized implementation of PGibbs and PGAS in Fig. 1(b). APF produced a result of orders of magnitude smaller error within a much smaller amount of run time: an estimation for θ with $1.6 * 10^{-4}$ square error with only 1000 particles in 1.5 seconds.

Note that, as mentioned in the discussion section, in theory APF with $M = 7$ should be 7 times slower than the plain particle filter. But in practice, thanks to the trick – only updating the approximate distributions for the retained particles, APF is just 2 times slower.

Density Estimation: We also show the kernel density estimates of the posterior of θ in Fig. 1(c), where we ran APF with 10^4 particles and $M = 7$ as well as our customized version of PGibbs and PGAS with 5000 particles for 6 hours (around 5000 MCMC iterations). For PGibbs and PGAS, we left the first 1000 samples as burn-in. APF produced a reasonable mode centered exactly at the true parameter value 0.5 using merely 40 seconds while PGibbs and PMMH mixed slowly.

Bimodal Variant: Consider a multimodal variant of SIN as follows: $X_t \sim \mathcal{N}(\sin(\theta^2 x_{t-1}), 1)$, $Y_t \sim \mathcal{N}(x_t, 0.5^2)$. Due to the θ^2 term, $p(\theta \mid y_{0:t})$ will be bimodal. We generate 200 data points in this case and execute APF with $K = 10^3$ particles and $M = 7$ using mixtures of $L = 2, 5, 10$ Gaussians as the approximate distribution. To illustrate the true posterior, we ran PMMH with $K = 500$ for 20 minutes (much longer than APF) to ensure it mixes properly.

The histograms of the samples for θ are demonstrated in Fig. 1(d). APF successfully approximates the multimodal posterior when $L = 5, 10$ and the weights are more accurate for $L = 10$. For $L = 2$, APF only found a single mode with a large bias. This suggests that increasing the number of mixtures used for approximation can help find different modes in the true posterior in practice.

Simultaneous localization and mapping (SLAM)

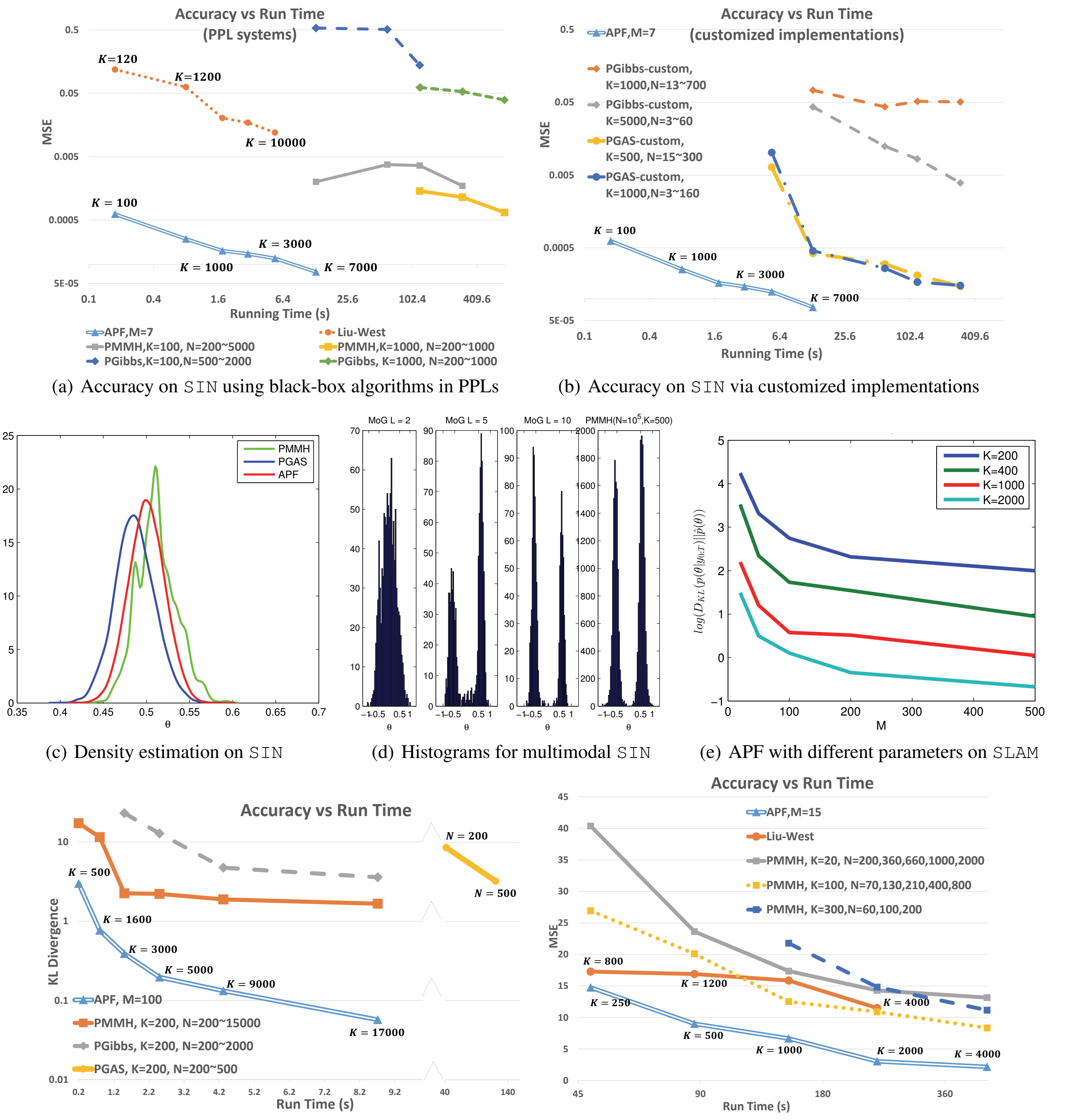
We consider a simultaneous localization and mapping example (SLAM) modified from (Murphy and others 1999). The map is defined as a 1-dimensional grid, where each cell has a static label (parameter to be estimated) which will be (noisily) observed by the robot. More formally, the map is a vector of boolean random variables $M(i) \in \{1, \ldots, N_O\}$, where $1 \le i \le N_L$. Neither the map nor the robot's location $L_t \in \{1, \ldots, N_L\}$ is observed.

Given the action, move right or left, the robot moves in the direction of action with a probability of p_a and stays at its current location with a probability of $1 - p_a$ (i.e., robot's wheels slip). The prior for the map is a product of individual cell priors, which are all uniform. The robot observes the label of its current cell correctly with probability p_o and incorrectly with probability of $1 - p_o$. In the original example, $N_L = 8$, $p_a = 0.8$, $p_o = 0.9$, and 16 actions were taken. In our experiment, we make the problem more difficult by setting $N_L = 20$ and deriving a sequence of 41 actions to ensure the posterior converge to a sharp mode.

We evaluate the KL-divergence between the prediction posterior and the exact true posterior within various time limits. We omit Liu-West filter since is not applicable for discrete parameters. In APF, we use a Bernoulli distribution as the approximate distribution for every grid cell. For PMMH,

[1] The details of the compilation optimizations in SPEC can be found in the supplementary materials.

[2] We discretize $[-1, 1]$ uniformly into 500 intervals. 500 is smaller than the number of particles used by the PMCMC algorithms, so this process does not influence the total running time significantly.



(a) Accuracy on SIN using black-box algorithms in PPLs

(b) Accuracy on SIN via customized implementations

(c) Density estimation on SIN

(d) Histograms for multimodal SIN

(e) APF with different parameters on SLAM

(f) Accuracy on SLAM

(g) Accuracy on BIRD

Figure 1: Performance plots in SIN, SLAM, BIRD experiments: (a) accuracy on SIN comparing APF with other black-box algorithms for PPLs (Liu-West, PMMH from SPEC and PGibbs by Anglican); (b) accuracy on SIN comparing APF with our customized implementations of PGibbs and PGAS in C++; (c) density estimation of the posterior distribution of θ by APF, PMMH and PGAS; (d) the histograms of samples for θ in the multimodal SIN model by APF using $L = 2, 5, 10$ mixtures of Gaussians and the almost-true posterior by PMMH; (e) APF with different configurations for K and M on SLAM; (f) accuracy on SLAM; (g) accuracy on BIRD.

we use a coordinate MCMC kernel: we only sample a single grid at each MCMC iteration. For PGibbs and PGAS, since it is hard to efficiently sample from $\Pr[\theta|X_{1:T}, Y_{1:T}]$, we only show results by Anglican.

Fig. 1(f) shows that APF approximates the posterior distribution much more accurately than other methods within a shorter run time. For PGAS by Anglican, due to its system overhead, the overall run time is significantly longer.

Similar to SIN, although APF performs $20M = 2000$ extra samples per time step in SLAM, in practice, APF is merely 60x slower than the plain particle filter due to the resampling trick.

Choices of Parameters: We experiment APF with various settings (number of particles K and number of samples M) and evaluate the average log KL-divergence over 100 trials. The results in Fig. 1(e) agree with the theory. As K increases the KL-divergence decreases whereas after a certain point, not much gain is obtained by increasing M. When M is large enough, the moment matching integrals are more or less exactly computed and the error is not due to the Monte Carlo sum but due to the error induced by the assumed-density projection step, which cannot be avoided.

Tracking bird migration (BIRD)

The bird migration problem (BIRD) is originally investigated in (Elmohamed, Kozen, and Sheldon 2007), which proposes a hidden Markov model to infer bird migration paths from a large database of observations[3].

In the BIRD model, there are 4 continuous parameters with 60 dynamic states where each time step contains 100 observed variables and more than 10^4 hidden variables.

We again measure the mean squared estimation error over 10 trials between the average of the samples for the parameters and the ground truth within different time limits. For APF, we use a diagonal Gaussian approximation with $M = 15$. For PMMH we use a truncated Gaussian proposal with diagonal covariance and leave the first half of the samples as burn-in. We did not compare against PGAS and PGibbs since these algorithms require storing the full history, which consumes too much memory (60x larger) to run enough particles. The results illustrated in Fig. 1(g) again show that APF achieves much better convergence within a much tighter computational budget.

Conclusion

We proposed the assumed parameter filter (APF), an online algorithm for joint parameter and state estimation in general state-space models, which provably converges in the long-sequence limit under standard conditons.

It is a "nearly-black-box" algorithm in the sense that its default assumed-density models can handle a large range of cases, while the algorithm can be extended easily to new cases by supplying a well-defined set of functions. APF is not a drop-in replacement for unbiased algorithms with theoretical guarantees, e.g., PMCMC and SMC^2, but an efficient alternative in practice. Experiments exhibit that APF has better estimation performance using much less computation time compared to several standard algorithms on a variety of applications.

Acknowledgement

We would like to thank our anonymous reviewers for valuable discussions. This work is supported by the DARPA PPAML program, contract FA8750-14-C-0011.

References

Alspach, D. L., and Sorenson, H. W. 1972. Nonlinear Bayesian estimation using Gaussian sum approximations. *Automatic Control, IEEE Transactions on* 17(4):439–448.

Andrieu, C.; Doucet, A.; and Holenstein, R. 2010. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B* 72(3):269–342.

Arora, N. S.; Russell, S. J.; Kidwell, P.; and Sudderth, E. B. 2010. Global seismic monitoring as probabilistic inference. In *NIPS*, 73–81.

Arulampalam, S.; Maskell, S.; Gordon, N.; and Clapp, T. 2002. A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing* 50(2):174–188.

Boyen, X., and Koller, D. 1998. Tractable inference for complex stochastic processes. In *UAI*, 33–42. Morgan Kaufmann Publishers Inc.

Cappé, O.; Godsill, S. J.; and Moulines, E. 2007. An overview of existing methods and recent advances in sequential Monte Carlo. *Proceedings of the IEEE* 95(5):899–924.

Chopin, N.; Iacobucci, A.; Marin, J.-M.; Mengersen, K.; Robert, C. P.; Ryder, R.; and Schäfer, C. 2010. On particle learning. *arXiv preprint arXiv:1006.0554*.

Chopin, N.; Jacob, P. E.; and Papaspiliopoulos, O. 2013. SMC^2: An efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B* 75(3):397–426.

Crisan, D., and Miguez, J. 2013. Nested particle filters for online parameter estimation in discrete-time state-space markov models. *arXiv preprint arXiv:1308.1883*.

Del Moral, P.; Doucet, A.; and Singh, S. 2010. Forward smoothing using sequential Monte Carlo. *arXiv preprint arXiv:1012.5390*.

Elmohamed, M.; Kozen, D.; and Sheldon, D. R. 2007. Collective inference on Markov models for modeling bird migration. In *Advances in Neural Information Processing Systems*, 1321–1328.

Erol, Y. B.; Li, L.; Ramsundar, B.; and Stuart, R. 2013. The extended parameter filter. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 1103–1111.

Gilks, W. R., and Berzuini, C. 2001. Following a moving target – Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 63(1):127–146.

[3] http://ppaml.galois.com/wiki/wiki/CP2BirdMigration

Goodman, N. D.; Mansinghka, V. K.; Roy, D. M.; Bonawitz, K.; and Tenenbaum, J. B. 2008. Church: A language for generative models. In *UAI*, 220–229.

Gordon, N. J.; Salmond, D. J.; and Smith, A. F. M. 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F Radar and Signal Processing* 140(2):107–113.

Hershey, J. R., and Olsen, P. A. 2007. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, IV–317. IEEE.

Huber, M. F., and Hanebeck, U. D. 2008. Gaussian filter based on deterministic sampling for high quality nonlinear estimation. In *Proceedings of the 17th IFAC World Congress (IFAC 2008)*, volume 17.

Julier, S. J., and Uhlmann, J. K. 2004. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* 92(3):401–422.

Kulkarni, T. D.; Kohli, P.; Tenenbaum, J. B.; and Mansinghka, V. 2015. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4390–4399.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

Lauritzen, S. L. 1992. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association* 87(420):1098–1108.

Li, L., and Russell, S. J. 2013. The BLOG language reference. Technical Report UCB/EECS-2013-51, EECS Department, University of California, Berkeley.

Lindsten, F.; Jordan, M. I.; and Schön, T. B. 2014. Particle Gibbs with ancestor sampling. *The Journal of Machine Learning Research* 15(1):2145–2184.

Liu, J., and West, M. 2001. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice*.

Lopes, H. F.; Carvalho, C. M.; Johannes, M.; and Polson, N. G. 2010. Particle learning for sequential Bayesian computation. *Bayesian Statistics* 9:175–96.

Milch, B.; Marthi, B.; Russell, S.; Sontag, D.; Ong, D. L.; and Kolobov, A. 2005. Blog: Probabilistic models with unknown objects. In *IJCAI*, 1352–1359.

Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B.; et al. 2002. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *AAAI*, 593–598.

Murphy, K. P., et al. 1999. Bayesian map learning in dynamic environments. In *NIPS*, 1015–1021.

Olsson, J.; Cappé, O.; Douc, R.; Moulines, E.; et al. 2008. Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state space models. *Bernoulli* 14(1):155–179.

Opper, M., and Winther, O. 1998. A Bayesian approach to on-line learning. *On-line Learning in Neural Networks, ed. D. Saad* 363–378.

Poyiadjis, G.; Doucet, A.; and Singh, S. S. 2011. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika* 98(1):pp. 65–80.

Rios, M. P., and Lopes, H. F. 2013. The extended Liu and West filter: Parameter learning in markov switching stochastic volatility models. In *State-Space Models*. Springer. 23–61.

Ristic, B.; Arulampalam, S.; and Gordon, N. J. 2004. *Beyond the Kalman filter: Particle filters for tracking applications*. Artech House.

Ritchie, D.; Mildenhall, B.; Goodman, N. D.; and Hanrahan, P. 2015. Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo. In *SIGGRAPH*.

Seeger, M. 2005. Expectation propagation for exponential families. Technical report.

Storvik, G. 2002. Particle filters for state-space models with the presence of unknown static parameters. *IEEE Transactions on Signal Processing* 50(2):281–289.

Wood, F.; van de Meent, J. W.; and Mansinghka, V. 2014. A new approach to probabilistic programming inference. In *AISTATS*, 1024–1032.

Wu, Y.; Li, L.; Russell, S.; and Bodik, R. 2016. Swift: Compiled inference for probabilistic programming languages. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*.

Zoeter, O., and Heskes, T. 2005. Gaussian quadrature based expectation propagation. In *Proceedings of AISTATS*.