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

Yusuf B. Erol[†*] **Yi Wu**[†*] **Lei Li**[‡] **Stuart Russell**[†]
[†] EECS, UC Berkeley {yberol,jxwuyi,russell}@eecs.berkeley.edu
[‡] Toutiao Lab lileicc@gmail.com

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: it is (a) online and computationally efficient; (b) applicable to both discrete and continuous parameter spaces with arbitrary transition dynamics. On a variety of models, APF generates more accurate results within a fixed computation budget compared to several standard algorithms.

1 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) [6, 2]. Online parameter and state estimation –computing the posterior probability for both (static) parameters and (dynamic) states, incrementally over time– is crucial for many applications [19, 24, 25].

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

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

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 $p(x_t, \theta \mid y_{0\ldots t})$.

Existing inference algorithms: 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.

For joint parameter and state estimation, the "gold standard" approaches are particle Markov chain Monte Carlo (PMCMC) algorithms [1, 15], which 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. However, one significant drawback of PMCMC algorithms is the computational budget. Note that to ensure the MCMC kernel adequately mixes, often an extremely large number of MCMC iterations is required, especially for real-world applications with a large number of time steps and complex dynamics. Moreover, the PMCMC algorithms are infeasible for online/streaming applications since they require multiple sweeps over all the observations. There are also online algorithms for joint parameter and state estimation problems. However, existing algorithms are either computationally inefficient [8] or only suitable for a restricted domain of models [27, 17, 7, 16].

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

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 The Assumed Parameter Filter (APF)

We propose the Assumed Parameter Filter (APF), which is a hybrid of particle filtering for state variables and assumed density filtering for parameter variables. APF aims to inherit the properties of the classical particle filter, which is online and applicable to arbitrary transitions, and better overcome the path degeneracy problem for parameter estimation without an expensive MCMC kernel. In APF, the posterior of both states and parameters are jointly represented by K particles. 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. For parameter θ_t^k at time t in particle k, we sample from a distribution $q_t^k(\theta)$ in some parametric family $\mathcal{Q}$. $q_t^k(\theta)$ is the approximate representation of the true particle posterior $p(\theta \mid x_{0:t}^k, y_{0:t})$. 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, where the Update function generates an approximate density q from $\mathcal{Q}$ via minimizing the KL-divergence between $p(\theta \mid x_{0:t}, y_{0:t})$ and q.

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 $\left\{\frac{1}{N}, \bar{x}_t^k, \bar{q}_t^k\right\} \sim \mathrm{Multinomial}\left\{w_t^k, x_t^k, q_t^k\right\}$, $\left\{x_t^k, q_t^k\right\} \leftarrow \left\{\bar{x}_t^k, \bar{q}_t^k\right\}$;

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

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

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

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 $\hat{p}(\theta \mid x_{0:i}, y_{0:i}) = \frac{s_i(\theta)q_{i-1}(\theta)}{\int_\theta s_i(\theta)q_{i-1}(\theta)d\theta}$. For most models, $\hat{p}$ will not belong to $\mathcal{Q}$. We project $\hat{p}$ into $\mathcal{Q}$ via minimizing KL-divergence:

$$q_i(\theta) = \arg\min_{q \in \mathcal{Q}} D\left(\hat{p}(\theta \mid x_{0:i}, y_{0:i}) \mid\mid q(\theta)\right) \tag{1}$$

For $\mathcal{Q}$ in the exponential family, minimizing the KL-divergence reduces to moment matching [26]. 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.

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

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 from $q_{i-1}(\theta)$: $Z \approx \frac{1}{M}\sum_{j=1}^M s_i(\theta^j)$, $g(\gamma_i) \approx \frac{1}{MZ}\sum_{j=1}^M m(\theta^j)s_i(\theta^j)$, 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. In practice, a very small M will be sufficient and the overall overhead can be negligible (more details in Appendix F).

2.2 Asymptotic performance for APF

For simplicity, we only consider continuous parameters and Gaussian as the approximate distribution. We assume that the model is *identifiable* (posterior is asymptotically Gaussian around the true parameter) and only the transition is parametrized by 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 $p(\theta \mid x_{0:t})$ and the assumed density $q_t(\theta)$ computed as explained in the previous subsection will converge to zero as $t \to \infty$.*

$$\lim_{t\to\infty} D_{\mathrm{KL}}\left(p(\theta \mid x_{0:t}, y_{0:t}) || q_t(\theta)\right) = 0. \tag{2}$$

The proof is in Appendix B. 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. Consequences of path degeneracy are further explained in Appendix C.

2.3 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

$$\mu_i = \tfrac{1}{Z}\int \theta s_i(\theta) q_{i-1}(\theta) d\theta, \quad \Sigma_i = \tfrac{1}{Z}\int \theta\theta^T s_i(\theta) q_{i-1}(\theta) d\theta - \mu_i \mu_i^T. \tag{3}$$

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 [29, 10]. 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. The unscented transform [11] 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: $\forall 1 \le j \le d, \theta_j = \mu_{i-1} + \left(\sqrt{d\Sigma_{i-1}}\right)_j, \theta_{d+j} = \mu_{i-1} - \left(\sqrt{d\Sigma_{i-1}}\right)_j$, where $(\cdot)_j$ means the jth column of the corresponding matrix. Then, one can approximately evaluate the moment matching integrals as follows:

$$Z \approx \tfrac{1}{2d}\sum_{j=1}^{2d} s_i(\theta^j), \quad \mu_i \approx \tfrac{1}{2dZ}\sum_{j=1}^{2d} \theta^j s_i(\theta^j), \quad \Sigma_i \approx \tfrac{1}{2dZ}\sum_{j=1}^{2d} \theta^j(\theta^j)^T s_i(\theta^j) - \mu_i\mu_i^T.$$

Mixtures of Gaussians: Weighted sums of Gaussian probability density functions can be used to approximate another density function arbitrarily closely. Let us assume that at time step $i-1$ it is possible to represent $p(\theta \mid x_{0:i-1}, y_{0:i-1})$ as a mixture of Gaussians with L components: $p(\theta | x_{0:i-1}, y_{0:i-1}) = \sum_{m=1}^{L} \alpha_{i-1}^m \mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m) = q_{i-1}(\theta)$. Given x_i and y_i,

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

$\hat{p}$ will not be a Gaussian mixture for arbitrary s_i. We project $\hat{p}$ back into the Gaussian mixture family and the approximated density is $q_i(\theta) = \sum_{m=1}^{L} \alpha_i^m \mathcal{N}(\theta; \mu_i^m, \Sigma_i^m)$ (details in Appendix D).

$$\mu_i^m = \tfrac{1}{\beta^m}\int \theta s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta, \quad \Sigma_i^m = \tfrac{1}{\beta^m}\int \theta\theta^T s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta - \mu_i^m(\mu_i^m)^T$$

where $\beta^m = \int s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)d\theta$ and $\alpha_i^m = \frac{\alpha_{i-1}^m \beta^m}{\sum_\ell \alpha_{i-1}^\ell \beta^\ell}$. Either a Monte Carlo sum or a Gaussian quadrature rule can be utilized to approximately update the means and covariances.

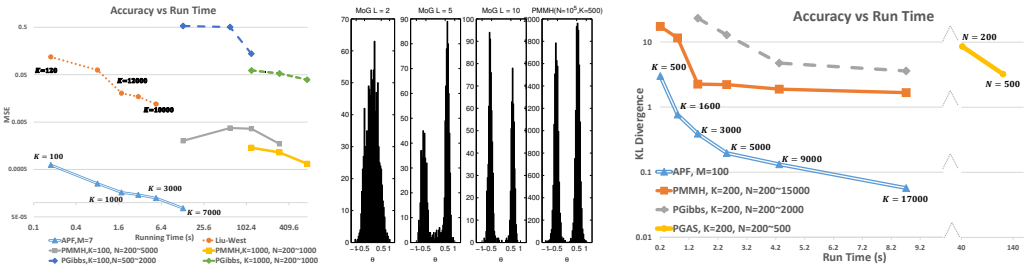
Discrete parameter spaces: Let us consider a d-dimensional parameter space where each parameter can take at most N_θ values. One can always track $p(\theta \mid x_{0:t}, y_{0:t})$ exactly with a constant-time update; the constant, however, is exponential in d [3], which is computationally intractable with increasing dimensionality. 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:

$$Z = \textstyle\sum_\theta s_i(\theta) q_{i-1}(\theta), \;\; q_{j,i}(\theta_j) = \tfrac{1}{Z}\sum_{\theta\setminus\theta_j} s_i(\theta) q_{i-1}(\theta).$$

Computing these summations is intractable. We propose using Monte Carlo summation.

3 Experiments

We experimented on three models: 1. `SIN`: a nonlinear dynamical model with 1 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 Liu-West filter (LW) and PMCMC



(a) Accuracy on `SIN` (b) multimodal `SIN` (c) Accuracy on `SLAM`

Figure 1: (a) accuracy on `SIN`; (b) histograms of samples for θ in the multimodal `SIN` by APF ($L = 2, 5, 10$ mixtures of Gaussians) and the almost-true posterior by PMMH; (c) accuracy on `SLAM`.

algorithms: particle marginal Metropolis-Hastings (PMMH), particle Gibbs (PGibbs) and particle Gibbs with ancestor sampling (PGAS). Due to space limit, results on `BIRD` are in Appendix E.

Due to the black-box property of APF, we developed APF in a probabilistic programming system, the State and Parameter Estimate Compiler (SPEC), utilizing some techniques from probabilistic programming community (details in Appendix G). SPEC also supports LW. PMMH is manually adapted from the code by SPEC. Code for PGibbs and PGAS are generated by the Anglican compiler [28].

3.1 Toy nonlinear model (`SIN`)

We consider the `SIN` model: $X_t \sim \mathcal{N}(\sin(\theta x_{t-1}), 1)$, $Y_t \sim \mathcal{N}(x_t, 0.5^2), \Theta \sim \mathcal{N}(0, 1), X_0 \sim \mathcal{N}(0, 1)$ with the true parameter $\theta^\star = 0.5$. 5000 data points are generated to ensure a sharp posterior.

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 PMCMC, we take the average of the last half of the samples and leave the first half as burn-in. For APF, we choose Gaussian as the approximate distribution with $M = 7$. For PMCMC, we use a local truncated Gaussian as the proposal distribution. The results are shown in Fig. 1(a). 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.

Bimodal Variant: Consider a multimodal variant of `SIN`: $X_t \sim \mathcal{N}(\sin(\theta^2 x_{t-1}), 1)$, $Y_t \sim \mathcal{N}(x_t, 0.5^2)$. Due to the θ^2 term, $p(\theta \mid y_{0:t})$ will be bimodal. We generate 200 data points 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 in Fig. 1(b). APF successfully approximates the multimodal posterior when $L = 5, 10$ but fails for $L = 2$. This suggests that increasing the number of mixtures used for approximation can help find different modes in the true posterior in practice.

3.2 Simultaneous localization and mapping (`SLAM`)

We consider a simultaneous localization and mapping example (`SLAM`) modified from [20]. The map is defined as a 1-dimensional grid world with N_L cells, where each cell has a static label (boolean parameter) which will be (noisily) observed by the robot. Neither the map nor the robot's location is observed. Given the action, move right or left, the robot moves in the direction with a probability of p_a and stays with a probability of $1 - p_a$ (i.e., robot's wheels slip). The robot observes the label of its current cell correctly with probability p_o. The original example has $N_L = 8$ cells and 16 actions. Here, we make the problem more difficult by setting $N_L = 20$ and deriving a sequence of 41 actions to ensure a sharp posterior.

We evaluate the KL-divergence between the prediction posterior and the true posterior within various time limits. In APF, we use a Bernoulli distribution as the approximate distribution for each cell. For PMMH, we use a coordinate MCMC kernel: we only sample a single grid at each MCMC iteration.

Fig. 1(c) shows that APF approximates the posterior distribution much more accurately than other methods within a shorter run time.

References

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

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

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

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

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

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

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

[8] W. R. Gilks and C. Berzuini. 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, 2001.

[9] J. R. Hershey and P. A. Olsen. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–317. IEEE, 2007.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix

A Bootstrap particle filter as a subcase of APF

Here we will show that when q is a delta function, APF recovers the bootstrap particle filter. The Dirac delta function can be considered as the limit of a Gaussian as the variance goes to zero, $\delta(\theta - \mu) = \lim_{\sigma^2 \to 0} \mathcal{N}(\theta; \mu, \sigma^2)$. Therefore, we can view q as an exponential family distribution. Specifically we are dealing with a Gaussian distribution with unknown mean and known variance (zero-variance). Then the moment matching integral required for assumed density filtering reduces to matching the means. If $q_{i-1} = \delta(\theta - \mu_{i-1})$, then

$$\begin{aligned} \mu_i = \int \theta q_i(\theta) d\theta &= \int \theta \hat{p}(\theta) d\theta \\ &= \frac{\int \theta s_i(\theta) \delta(\theta - \mu_{i-1}) d\theta}{\int s_i(\theta) \delta(\theta - \mu_{i-1}) d\theta} \\ &= \frac{\mu_{i-1} s_i(\mu_{i-1})}{s_i(\mu_{i-1})} = \mu_{i-1} \end{aligned} \tag{4}$$

where the last equality follows from the *sifting property* of the Dirac delta function. The main result here is that for $\mathcal{Q}$ Dirac delta, $\mu_i = \mu_{i-1}$; that is, the APF Update step does not propose new values. Therefore, our proposed algorithm recovers the standard bootstrap particle filter.

B Proof for Theorem 1

In this section[2] we will assume an identifiable model where the posterior distribution approaches normality and concentrates in the neighborhood of the posterior mode. Suppose $\hat{\theta}$ is the posterior mode and hence the first-order partial derivatives of $\log p(\theta \mid x_{0:T}, y_{0:T})$ vanish at $\hat{\theta}$. Define

$$\hat{I} = -\frac{\partial^2 \log p(\theta \mid x_{0:T}, y_{0:T})}{\partial \theta \partial \theta^T}\bigg|_{\theta = \hat{\theta}} \tag{5}$$

Applying a second-order Taylor approximation around $\hat{\theta}$ to the posterior density results in

$$\log p(\theta \mid x_{0:T}, y_{0:T}) \approx \log p(\hat{\theta} \mid x_{0:T}) - \frac{1}{2}(\theta - \hat{\theta})^T \hat{I} (\theta - \hat{\theta})$$

Hence;

$$p(\theta \mid x_{0:T}, y_{0:T}) \propto \exp\left\{ -\frac{1}{2}(\theta - \hat{\theta})^T \hat{I} (\theta - \hat{\theta}) \right\} \tag{6}$$

which is a p ($\theta \in \mathbb{R}^p$) dimensional Gaussian density with mean $\hat{\theta}$ and covariance $\hat{I}^{-1}$. As the posterior becomes highly concentrated in a neighborhood of the posterior mode, the effect of the prior on the posterior diminishes, which is the Bernstein-von Mises theorem. Then we can rewrite Eq. 6 as

$$p(\theta \mid x_{0:T}, y_{0:T}) \propto \exp\left\{ -\frac{T}{2} \sum_{ij} (\theta_i - \hat{\theta}_i) \hat{J}_{ij} (\theta_j - \hat{\theta}_j) \right\}$$

where $\hat{J}_{ij} = -\partial_i \partial_j \frac{1}{T} \sum_{t=1}^{T} \log s_t(\hat{\theta})$ and $s_t(\theta) = p(x_t \mid x_{t-1}, \theta) p(y_t \mid x_t, \theta)$.

The assumed density filter updates for the Gaussian case has been derived in earlier sections. We will reorganize them in a more convenient form.

$$\begin{aligned} \mu_i(t) &= \frac{\int \theta_i s_t(\theta) q_{t-1}(\theta) d\theta}{\int s_t(\theta) q_{t-1}(\theta) d\theta} \\ \Sigma_{ij}(t) &= \frac{\int \theta_i \theta_j s_t(\theta) q_{t-1}(\theta) d\theta}{\int s_t(\theta) q_{t-1}(\theta) d\theta} - \mu_i(t) \mu_j(t) \end{aligned}$$

[2] Our discussion follows [22] which considers the asymptotic performance of assumed density filtering for independent identically distributed data.

We will use a simple property of centered Gaussian random variables z, $E[zf(z)] = E(f^{'}(z)).E(z^2)$ which can be proven by applying integration by parts. Then the explicit updates can be written as follows:

$$\begin{aligned} \mu_i(t) &= \mu_i(t-1) + \sum_j \Sigma_{ij}(t-1) \times \partial_j \log E_u[s_t(\mu(t-1)+u)] \\ \Sigma_{ij}(t) &= \Sigma_{ij}(t-1) + \sum_{kl} \Sigma_{ik}(t-1)\Sigma_{lj}(t-1)\partial_k\partial_l \log E_u[s_t(\mu(t-1)+u)] \end{aligned} \tag{7}$$

where u is a zero-mean Gaussian random vector with covariance $\Sigma(t-1)$. We define $V_{kl} = \partial_k\partial_l \log E_u[s_t(\theta+u)]$ and assume that for large times we can replace the difference equation for $\Sigma(t)$ with a differential equation. Then we can rewrite Eq. 7 as

$$\frac{d\Sigma}{dt} = \Sigma V \Sigma \tag{8}$$

which is solved by

$$\frac{d\Sigma^{-1}}{dt} = -V \tag{9}$$

Integrating both sides

$$\Sigma^{-1}(t) - \Sigma^{-1}(t_0) = -\int_{t_0}^{t} V(\tau)d\tau \tag{10}$$

For large t ; we expect the covariance Σ to be small such that $\log E_u[s_t(\mu+u)] = \log s_t(\mu)$. Assuming that the online dynamics is close to θ^* and dividing both sides of Eq. 10 by t and taking the limit $t \to \infty$, we get

$$\lim_{t\to\infty} \frac{(\Sigma^{-1}(t))_{ij}}{t} = \lim_{t\to\infty} \frac{-\int_{t_0}^{t} \partial_i\partial_j s(\theta^*)}{t} \tag{11}$$

Further assuming ergodicity (i.e., markov process converging to some stationary distribution π), we can replace the time average with the probabilistic average.

$$\lim_{t\to\infty} \frac{(\Sigma^{-1}(t))_{ij}}{t} = -\int \pi(x)p(x' \mid x,\theta^*)\partial_i\partial_j \log s(\theta^*)dxdx'$$

If we define the right hand side as $A_{ij} = -\int \pi(x)p(x' \mid x,\theta^*)\partial_i\partial_j \log p(x' \mid x,\theta^*)p(y \mid x',\theta^*)dxdx'$ we have;

$$\lim_{t\to\infty} \Sigma(t) = \frac{A^{-1}}{t} \tag{12}$$

We will also analyze the asymptotic scaling of the estimation error, defined as the deviation between θ^* and $\mu(t)$. Assuming that the estimate μ is close to θ^* and the posterior is sharply concentrated we can neglect the expectation with respect to u in Eq.7. Defining $\mu_i(t) = \theta_i^* + \epsilon_i(t)$, and applying a first order Taylor approximation to Eq. 7 around θ^* we get;

$$\epsilon_i(t+1) - \epsilon_i(t) = \sum_\ell \Sigma_{i\ell}\partial_\ell \log P + \sum_{k\ell} \Sigma_{i\ell}\epsilon_k(t)\partial_k\partial_l \log P$$

where $P \equiv p(x' \mid x,\theta^*)p(y \mid x',\theta^*)$. Taking the expectation with respect to the stationary distribution (denoted by an overbar) and using the relationship in Eq. 12 and replacing the difference equation with a differential equation we get an equation of motion for the expected error $e_i = \bar{\epsilon_i}$.

$$\frac{de_i}{dt} + \frac{e_i}{t} = \sum_j \frac{(A^{-1})_{ij}}{t}\overline{\partial_j \log P} \tag{13}$$

As $t \to \infty$ right hand side vanishes and hence the error term decays like $e_i \propto \frac{1}{t}$.

Revisiting Eq. 6, the true posterior covariance matrix is given by $C^{-1} = T\hat{J}$. Due to our ergodicity assumption, $\lim_t \hat{J} = A$. Hence the true posterior density covariance asymptotically converges to A^{-1}/T which is the same limit as the assumed density filter covariance $\Sigma(t)$.

KL divergence between two d-dimensional multivariate Gaussians, $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ is given by

$$\frac{1}{2}\left[\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + \text{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T\Sigma_2^{-1}(\mu_2 - \mu_1)\right]$$

We have shown that $\lim_{t\to\infty} C, \Sigma = A^{-1}/t$ and $\mu(t) \to \theta^*$. Due to the identifiability assumption, the posterior mode is also assumed to converge to the parameter θ^*. Applying these findings to the earlier KL-divergence formula, we can see that;

$$\lim_{t\to\infty} D_{KL}(p(\theta \mid x_{0:t}, y_{0:t})||q_t(\theta)) = 0. \tag{14}$$

For the `SIN` model discussed in the experiments section, the true posterior $p(\theta \mid x_{0:t})$ is computed for a grid of parameter values in $O(t)$ time per parameter value. Assumed density filtering is also applied with $\mathcal{Q}$ Gaussian and the true density (solid) vs. assumed density (dashed) is illustrated in Fig. 2. Notice that, ADF is slightly off at earlier stages, however, does indeed catch up with the ground truth with more data.

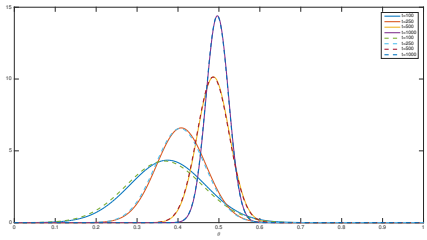
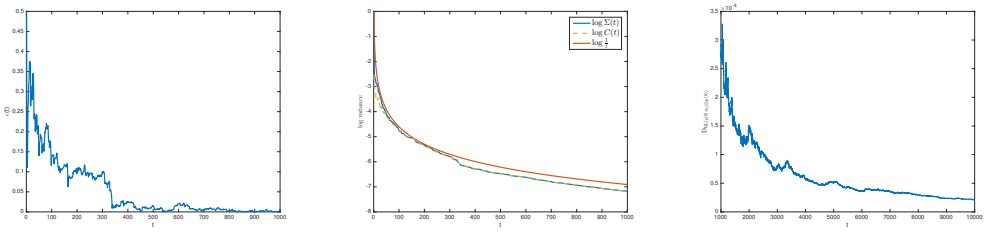


Figure 2: True posterior $p(\theta \mid x_{0:t})$ vs assumed density filter estimate $q_t(\theta)$ (solid vs dashed line respectively). for the `SIN` model.

As predicted by Eq. 13, the error term converges to zero as shown in Fig. 3(a). Figure 3(b) illustrates the asymptotic behavior of the true posterior covariance $C(t)$ and assumed density covariance $\Sigma(t)$. Assumed density filter quickly approximates the covariance. Most importantly, as can be seen from the plot, $\log C(t)$ and $\log \Sigma(t)$ is $log(1/t) + \text{constant}$ asymptotically, and this agrees with our derivations in Eq. 12. Figure 3(c) confirms our theoretical result of KL divergence converging to zero in the long-sequence limit.



(a) Error term $\epsilon(t)$ (b) Covariance term $\Sigma(t)$ (c) KL divergence

Figure 3: `SIN` model with $\theta^* = 0.5$

C Path degeneracy

Similar to [27, 17], we are sampling from $p(\theta \mid x^i_{0:t}, y_{0:t})$ at each time step to fight against sample impoverishment. It has been discussed before that these methods suffer from *ancestral path degeneracy* [4, 17, 23]. 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 [23], 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 [21]. 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 [5]. Poyiadjis et al. similarly proposed an $O(K^2)$ algorithm that leads to linearly growing variances [23]. These techniques can easily be augmented into the APF framework to overcome the path degeneracy problem for models with long memory.

D Mixture of Gaussians Derivation

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:

$$\hat{p} \propto \sum_{m=1}^{L} \alpha_{i-1}^m \beta^m \frac{s_i(\theta)\mathcal{N}(\theta; \mu_{i-1}^m, \Sigma_{i-1}^m)}{\beta^m} \tag{15}$$

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 [9]. 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.

E Experiments on the `BIRD` model

The bird migration problem (`BIRD`) is originally investigated in [6], which proposes a hidden Markov model to infer bird migration paths from a large database of observations.

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. 4 again show that APF achieves much better convergence within a much tighter computational budget.

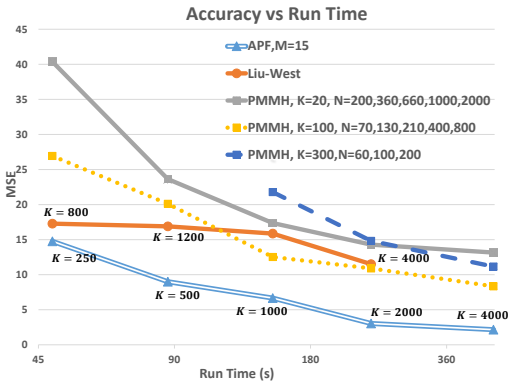


Figure 4: Estimation accuracy of APF against Liu-West filter and PMMH on the BIRD model.

F Practical Performances of APF

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: in our experiments, $M = 7$ with Gaussian quadrature rules is sufficient for continuous parameters while $M = 100$ is sufficient for discrete case.

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 $O(MKT)$.

For $M = 7$ in SIN, in theory APF should be 7 times slower than the plain particle filter. But in practice, since we resample all the particles prior to the update step and only update the approximate distribution for those particles that do not disappear, APF is just 2 times slower. Similarly, in SLAM, APF requires $20M = 2000$ extra samples per time step than plain particle filter, which implies that theoretically APF should be 2000x slower. However, due to the resampling trick, APF is merely 60x slower in practice. Likewise,

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.

G Using APF within a probabilistic programming system

This section shows APF can be integrated into a probabilistic programming language (PPL), BLOG [18], 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 [13] and the 3D scene perception [12].

We developed a compiled inference engine, the State and Parameter Estimation Compiler (SPEC), utilizing APF under BLOG [18] thanks to its concise syntax [14]: 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.

```
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 BLOG program above describes the `SIN` model: $X_t \sim \mathcal{N}(\sin(\theta x_{t-1}), 1)$, $Y_t \sim \mathcal{N}(x_t, 0.5^2), \Theta \sim \mathcal{N}(0,1), X_0 \sim \mathcal{N}(0,1)$.

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`. We also extend the original syntax of BLOG by introducing a new keyword `extern` (Line 12) to import arbitrary customized C++ functions (e.g., input functions for streaming data at each time step).

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. Due to the memory efficiency of APF, many optimizations from the programming language community can be applied to even accelerate the practical performance of APF.