AN EFFICIENT STOCHASTIC APPROXIMATION EM ALGORITHM USING CONDITIONAL PARTICLE FILTERS

Fredrik Lindsten

Division of Automatic Control, Linköping University, Linköping, Sweden, e-mail: lindsten@isy.liu.se.

ABSTRACT

I present a novel method for maximum likelihood parameter estimation in nonlinear/non-Gaussian state-space models. It is an expectation maximization (EM) like method, which uses sequential Monte Carlo (SMC) for the intermediate state inference problem. Contrary to existing SMC-based EM algorithms, however, it makes efficient use of the simulated particles through the use of particle Markov chain Monte Carlo (PMCMC) theory. More precisely, the proposed method combines the efficient conditional particle filter with ancestor sampling (CPF-AS) with the stochastic approximation EM (SAEM) algorithm. This results in a procedure which does not rely on asymptotics in the number of particles for convergence, meaning that the method is very computationally competitive. Indeed, the method is evaluated in a simulation study, using a small number of particles, with promising results.

1. INTRODUCTION

State-space models (SSMs) are commonly used in statistical signal processing to model dynamical systems. Methods such as sequential Monte Carlo (SMC), have emerged to allow inference beyond the linear Gaussian case [1, 2]. However, estimation of fixed model parameters remains a challenging problem. We consider here a general, discrete-time SSM with state $x_t \in \mathsf{X}$ and observation $y_t \in \mathsf{Y}$, parameterized by some unknown parameter $\theta \in \Theta$,

$$x_{t+1} \sim f_\theta(x_{t+1} \mid x_t), \qquad y_t \sim g_\theta(y_t \mid x_t).$$

We observe a batch of measurements $y_{1:T} = \{y_1, \ldots, y_T\}$ and seek to identify θ offline. Methods addressing this problem are often iterative, in the sense that they iterate between updating θ and updating/estimating the latent states $x_{1:T}$. Examples are the expectation maximization (EM) algorithm [3] for maximum likelihood (ML) inference and Gibbs sampling [4] for Bayesian inference. SMC can naturally be used within these methods to address the intermediate state inference problem at each iteration. For instance, particle smoothing (PS) has been used within the EM algorithm (PSEM) for challenging identification problems [5, 6, 2].

However, if used in a standard way, a large number of particles is typically required to obtain accurate state inference results. Since this has to be done at each iteration of the top level identification algorithm, the resulting method will be very computationally intensive. Recently, however, a framework for Bayesian inference referred to as particle Markov chain Monte Carlo (PMCMC) has been developed [7]. PMCMC uses SMC within MCMC, but do so in a way which ensures that the methods are, in some sense, exact, for any number of particles (see [7] for further discussion). Furthermore, a certain branch of PMCMC, based on so called conditional particle filters (CPFs), has been found to make very efficient use of the simulated particles [8, 9, 10]. This is achieved by propagating information from one iteration to the next, by *conditioning* the PF on previously simulated particles.

The purpose of this contribution is to illustrate that these attractive methods are not exclusive to the Bayesian. Indeed, we develop a method for ML inference, i.e. the problem of finding $\widehat{\theta}_{\mathrm{ML}} = \arg\max_\theta p_\theta(y_{1:T})$. The method is a combination of stochastic approximation EM (SAEM) [11] and the conditional PF with ancestor sampling (CPF-AS) [8]. There have been previous contributions on combining PMCMC with SAEM [12, 13]. However, these methods differ from the present contribution, in that they are based on the particle independent Metropolis-Hastings kernel, which is not able to reuse information across iterations, as is done in CPF-AS.

2. THE EM, MCEM AND SAEM ALGORITHMS

To introduce the methods that we will be working with we consider a general missing data model. The observed variable is denoted y and the latent variable is denoted z. In the state-space setting, we thus have $y = y_{1:T}$ and $z = x_{1:T}$. Let $p_\theta(y)$ be the likelihood of the data, parameterized by $\theta \in \Theta$. For each θ, the complete data likelihood is given by $p_\theta(z, y)$ and the posterior of z given y is $p_\theta(z \mid y) = p_\theta(z, y)/p_\theta(y)$.

Define, $Q(\theta, \theta') = \int \log p_\theta(z, y) p_{\theta'}(z \mid y)\, dz$. The EM algorithm [3] is an iterative method, which maximizes $p_\theta(y)$ by iteratively maximizing the auxiliary quantity $Q(\theta, \theta')$. It is useful when maximization of $\theta \mapsto Q(\theta, \theta')$, for fixed θ', is simpler than direct maximization of the likelihood, $\theta \mapsto p_\theta(y)$. The procedure is initialized at some $\theta_0 \in \Theta$ and then iterates between two steps, expectation (E) and maximization (M),

(E) Compute $Q(\theta, \theta_{k-1})$.

(M) Compute $\theta_k = \arg\max_{\theta \in \Theta} Q(\theta, \theta_{k-1})$.

The resulting sequence $\{\theta_k\}_{k \geq 0}$ will, under weak assumptions, converge to a stationary point of the likelihood $p_\theta(y)$.

We shall throughout this work assume that the M-step can be carried out straightforwardly, which is the case for many models encountered in practice. For the E-step, however, we note that we have to compute an expectation under the posterior $p_{\theta'}(z \mid y)$. In many situations, this computation is complicated or even intractable. One way to address this issue is to compute the E-step using Monte Carlo integration, leading to the MCEM algorithm [14]. Assume that it is possible to simulate from the posterior $p_{\theta'}(z \mid y)$. Then, at iteration k, the E-step is replaced by the following;

(E′) Generate M_k realizations $\{z^j\}_{j=1}^{M_k}$ from $p_{\theta_{k-1}}(z \mid y)$ and compute, $\widetilde{Q}_k(\theta) = M_k^{-1} \sum_{j=1}^{M_k} \log p_\theta(z^j, y)$.

This work was supported by: the project Calibrating Nonlinear Dynamical Models (Contract number: 621-2010-5876) funded by the Swedish Research Council and CADICS, a Linneaus Center also funded by the Swedish Research Council.

The M-step is left unchanged, but now the Monte Carlo approximation $\widetilde{Q}_k(\theta)$ is maximized in place of $Q(\theta, \theta_{k-1})$.

The MCEM algorithm can be very useful in situations where the E-step of the EM algorithm is intractable. A problem with MCEM, however, is that it relies on the number of simulations M_k to increase with k to be convergent [2, 15]. That is, the method can be thought of as *doubly asymptotic*, since it requires the number of iterations to tend to infinity, $k \to \infty$, as well as the number of simulations, $M_k \to \infty$. Furthermore, a complete set of simulated values $\{z^j\}_{j=1}^{M_k}$ has to be generated at each iteration of the algorithm. After making an update of the parameter, these values are discarded and a new set has to be simulated at the next iteration.

To be able to make more efficient use of the simulated variables, a related method, referred to as stochastic approximation EM (SAEM) was proposed by [11]. This method uses a stochastic approximation update of the auxiliary quantity Q,

$$\widehat{Q}_k(\theta) = (1 - \gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k \left(\frac{1}{m_k} \sum_{j=1}^{m_k} \log p_\theta(z^j, y) \right). \quad (2)$$

The E-step is thus replaced by the following;

(E″) Generate m_k realizations $\{z^j\}_{j=1}^{m_k}$ from $p_{\theta_{k-1}}(z \mid y)$ and update $\widehat{Q}_k(\theta)$ according to (2).

In (2), $\{\gamma_k\}_{k\geq 1}$ is a decreasing sequence of positive step sizes, satisfying the usual stochastic approximation conditions, $\sum_k \gamma_k = \infty$ and $\sum_k \gamma_k^2 < \infty$. In SAEM, all simulated values contribute to $\widehat{Q}_{k-1}(\theta)$, but they are down-weighted using a forgetting factor given by the step size. Under appropriate assumptions, SAEM can be shown to converge for fixed m_k (e.g. $m_k \equiv 1$), as $k \to \infty$ [11, 2]. When the simulation step is computationally involved, there is a considerable computational advantage of SAEM over MCEM [11].

3. CONDITIONAL PARTICLE FILTER SAEM

We now turn to the new procedure; SAEM using conditional particle filters (CPF). We refer to this method as CPF-SAEM.

3.1. Markovian stochastic approximation

Let us return to our original problem; inference in nonlinear state-space models. The latent variable posterior is then given by $p_\theta(x_{1:T} \mid y_{1:T})$, i.e. by the joint smoothing distribution. This distribution is intractable to compute, as well as to sample from, for the models under study. We can thus not employ MCEM or SAEM directly. To address this issue, it has been suggested to use SMC, i.e. particle smoothers (PS), to compute the E-step of the EM algorithm [5, 6, 2]. This leads to an SMC-analogue of MCEM, which we refer to as PSEM. Unfortunately, PSEM inherits the drawbacks of MCEM. That is, it relies on double asymptotics for convergence and is not able to reuse the simulated values (i.e. the particles) across iterations.

For instance, assume that we employ the forward filter/backward simulator particle smoother by [16] in PSEM. At iteration k, we use N_k particles and backward trajectories, with a computational complexity of $O(N_k^2)$. For this approach to be successful, we need to take N_k large (in fact $N_k \to \infty$ as $k \to \infty$) which leads to a very computationally costly E-step. If we instead take an SAEM approach, it is sufficient to generate a single sample at each iteration ($m_k \equiv 1$), reducing the computational cost to $O(N_k)$. However, we still need to take N_k large to get an accurate particle approximation from the PF. Furthermore, it is not clear how the approximation error for finite N_k will affect the parameter estimates in the SAEM algorithm.

To avoid this, and thus be able to reduce the computational complexity, we will use a Markovian version of stochastic approximation [17, 18]. It has been recognized that it not necessary to sample exactly from the posterior distribution of the latent variables, to assess convergence of the SAEM algorithm. Instead, it is sufficient to sample from a family of Markov kernels $\{M_\theta : \theta \in \Theta\}$, leaving the family of posteriors invariant [19]. In our case, we thus seek a family of Markov kernels on X^T, such that, for each $\theta \in \Theta$, $M_\theta(dx_{1:T} \mid x'_{1:T})$ leaves the joint smoothing distribution $p_\theta(x_{1:T} \mid y_{1:T})$ invariant.

Assume for the time being that this family of kernels is available. At iteration k of the SAEM algorithm, let θ_{k-1} be the previous value of the parameter estimate and let $x_{1:T}[k-1]$ be the previous draw from the Markov kernel. Then, we proceed by sampling

$$x_{1:T}[k] \sim M_{\theta_{k-1}}(dx_{1:T} \mid x_{1:T}[k-1]), \quad (3)$$

and updating the auxiliary quantity according to

$$\widehat{Q}_k(\theta) = (1 - \gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k \log p_\theta(y_{1:T}, x_{1:T}[k]). \quad (4)$$

This quantity is then maximized w.r.t. θ in the M-step, analogously to the standard SAEM algorithm.

3.2. Conditional particle filter with ancestor sampling

To find the sought family of Markov kernels, we will make use of PMCMC theory [7]. More precisely, I suggest to run a conditional particle filter with ancestor sampling (CPF-AS). The CPF-AS has previously been used for Gibbs sampling in a PMCMC setting [8]. Other options are available, e.g. to use the original CPF by [7] or the CPF with backward simulation, originally proposed by [20]. However, we focus here on CPF-AS since ancestor sampling has been found to considerably improve the mixing over the basic CPF. Furthermore, it can be implemented in a forward only recursion, its computational cost is linear in the number of particles and it allows for a simple type of Rao-Blackwellization (as will be discussed later).

The CPF-AS procedure is an SMC sampler, akin to a standard PF but with the difference that one particle at each time step is specified *a priori*. Let these prespecified particles be denoted $x'_{1:T} = \{x'_1, \ldots, x'_T\}$. The method is most easily described as an auxiliary PF (see e.g. [1, 21, 22] for an introduction). As in a standard auxiliary PF, the sequence of distributions $p_\theta(x_{1:t} \mid y_{1:t})$, for $t = 1, \ldots, T$, is approximated sequentially by collections of weighted particles. Let $\{x^i_{1:t-1}, w^i_{t-1}\}_{i=1}^N$ be a weighted particle system targeting $p_\theta(x_{1:t-1} \mid y_{1:t-1})$. That is, the particle system defines an empirical distribution,

$$\widehat{p}_\theta^N(dx_{1:t-1} \mid y_{1:t-1}) \triangleq \sum_{i=1}^N \frac{w^i_{t-1}}{\sum_l w^l_{t-1}} \delta_{x^i_{1:t-1}}(dx_{1:t-1}), \quad (5)$$

which approximates the target. To propagate this sample to time t, we introduce the auxiliary variables $\{a^i_t\}_{i=1}^N$, referred to as *ancestor indices*. To generate a specific particle x^i_t at time t, we first sample the ancestor index with $P(a^i_t = j) \propto w^j_{t-1}$. Then, x^i_t is sampled from some proposal kernel $q_{\theta,t}$,

$$x^i_t \sim q_{\theta,t}(x_t \mid x^{a^i_t}_{t-1}, y_t). \quad (6)$$

Hence, a^i_t is the index of the ancestor particle at time $t-1$, of particle x^i_t. The particle trajectories can then be augmented according to

$$x^i_{1:t} = \{x^{a^i_t}_{1:t-1}, x^i_t\}. \quad (7)$$

In this formulation, the resampling step is implicit and it corresponds to sampling the ancestor indices.

Now, in a standard auxiliary PF, we would repeat this procedure for each $i = 1, \ldots, N$, to generate N particles at time t. In CPF-AS, however, we condition on the event that x'_t is contained in the collection $\{x_t^i\}_{i=1}^N$. To accomplish this, we sample according to (6) only for $i = 1, \ldots, N-1$. The Nth particle is then set deterministically; $x_t^N = x'_t$.

To be able to construct the Nth particle trajectory as in (7), the conditioned particle has to be associated with an ancestor at time $t-1$. That is, we need to generate a value for the ancestor variable a_t^N. In CPF-AS, this is done in a so called ancestor sampling step, in which a_t^N is sampled conditionally on x'_t. From Bayes' rule, it follows that $p_\theta(x_{t-1} \mid x'_t, y_{1:t}) \propto f_\theta(x'_t \mid x_{t-1}) p_\theta(x_{t-1} \mid y_{1:t-1})$. By plugging (5) into the above expression, we arrive at the approximation,

$$\widehat{p}_\theta^N(dx_{t-1} \mid x'_t, y_{1:t}) = \sum_{i=1}^N \frac{w_{t-1}^i f_\theta(x'_t \mid x_{t-1}^i)}{\sum_l w_{t-1}^l f_\theta(x'_t \mid x_{t-1}^l)} \delta_{x_{t-1}^i}(dx_{t-1}).$$

To sample an ancestor particle for x'_t, we draw from this empirical distribution. That is, we sample the ancestor index with $P(a_t^N = j) \propto w_{t-1}^j f_\theta(x'_t \mid x_{t-1}^j)$.

Finally, all the particles, for $i = 1, \ldots, N$, are assigned importance weights, analogously to a standard auxiliary PF; $w_t^i = W_{\theta,t}(x_t^i, x_{t-1}^{a_t^i})$, where the weight function is given by,

$$W_{\theta,t}(x_t, x_{t-1}) \triangleq \frac{g_\theta(y_t \mid x_t) f_\theta(x_t \mid x_{t-1})}{q_{\theta,t}(x_t \mid x_{t-1}, y_t)}. \tag{8}$$

This results in a new weighted particle system $\{x_{1:t}^i, w_t^i\}_{i=1}^N$, targeting the joint smoothing distribution at time t. The method is initialised by sampling from a proposal density $x_1^i \sim q_{\theta,1}(x_1 \mid y_1)$ for $i = 1, \ldots, N-1$ and setting $x_1^N = x'_1$. The initial particles are assigned weights $w_1^i = W_{\theta,1}(x_1^i)$ where the weight function is given by $W_1^\theta(x_1) \triangleq g_\theta(y_1 \mid x_1) p_\theta(x_1) / q_{\theta,1}(x_1 \mid y_1)$. The CPF-AS is summarized in Algorithm 1.

Algorithm 1 CPF with ancestor sampling, conditioned on $\{x'_{1:T}\}$

1: Draw $x_1^i \sim q_{\theta,1}(x_1 \mid y_1)$ for $i = 1, \ldots, N-1$.
2: Set $x_1^N = x'_1$.
3: Set $w_1^i = W_{\theta,1}(x_1^i)$ for $i = 1, \ldots, N$.
4: **for** $t = 2$ **to** T **do**
5: Draw a_t^i with $P(a_t^i = j) \propto w_{t-1}^j$ for $i = 1, \ldots, N-1$.
6: Draw $x_t^i \sim q_{\theta,t}(x_t \mid x_{t-1}^{a_t^i}, y_t)$ for $i = 1, \ldots, N-1$.
7: Draw a_t^N with $P(a_t^N = j) \propto w_{t-1}^j f_\theta(x'_t \mid x_{t-1}^j)$.
8: Set $x_t^N = x'_t$.
9: Set $x_{1:t}^i = \{x_{1:t-1}^{a_t^i}, x_t^i\}$ for $i = 1, \ldots, N$.
10: Set $w_t^i = W_{\theta,t}(x_{t-1}^{a_t^i}, x_t^i)$ for $i = 1, \ldots, N$.
11: **end for**

It might not be obvious why it is attractive to condition the PF on a prespecified set of particles. The reason for why this is useful is that it implies an invariance property which is key to our development. To state this more formally, we first make a standard assumption on the support of the proposal kernels used in the PF.

(A1) For any $\theta \in \Theta$ and any $t \in \{1, \ldots, T\}$, $\mathcal{S}_t^\theta \subset \mathcal{Q}_t^\theta$ where,

$$\mathcal{S}_t^\theta = \{x_{1:t} \in \mathsf{X}^t : p_\theta(x_{1:t} \mid y_{1:t}) > 0\},$$
$$\mathcal{Q}_t^\theta = \{x_{1:t} \in \mathsf{X}^t : q_{\theta,t}(x_t \mid x_{t-1}, y_t) p_\theta(x_{1:t-1} \mid y_{1:t-1}) > 0\}.$$

The key property of CPF-AS can now be stated as follows.

Proposition 1. *Assume (A1). Then, for any $\theta \in \Theta$ and any $N \geq 2$, the procedure;*

(i) Run Algorithm 1 conditionally on $x'_{1:T}$;
(ii) Sample $x_{1:T}^\star$ with $P(x_{1:T}^\star = x_{1:T}^i) \propto w_T^i$;

defines a p-irreducible and aperiodic Markov kernel on X^T, with invariant distribution $p_\theta(x_{1:T} \mid y_{1:T})$.

Proof. The invariance property follows by the construction of the CPF-AS in [8], and the fact that the law of $x_{1:T}^\star$ is independent of permutations of the particle indices. This allows us to always place the conditioned particles at the Nth position. Irreducibility and aperiodicity follows from [7, Theorem 5]. □

In other words, if $x'_{1:T} \sim p_\theta(x_{1:T} \mid y_{1:T})$ and we sample $x_{1:T}^\star$ according to the procedure in Proposition 1, then, for any number of particles N, it holds that $x_{1:T}^\star \sim p_\theta(x_{1:T} \mid y_{1:T})$. To understand this result, it can be instructive to think about the extreme cases, $N = 1$ and $N = \infty$, respectively. For $N = 1$, since we condition on $x'_{1:T}$, the sampling procedure will simply return $x_{1:T}^\star = x'_{1:T}$. Since $x'_{1:T}$ is distributed according to the exact smoothing distribution, then so is $x_{1:T}^\star$ (though, with correlation 1 between $x_{1:T}^\star$ and $x'_{1:T}$). For $N = \infty$, on the other hand, the conditioning will have a negligible effect on Algorithm 1. That is, the CPF-AS reduces to a standard PF, using an infinite number of particles. Since the PF in this case exactly recovers the joint smoothing distribution, it again holds that $x_{1:T}^\star \sim p_\theta(x_{1:T} \mid y_{1:T})$, but now $x_{1:T}^\star$ is independent of $x'_{1:T}$.

Intuitively, using a fixed N can be though of as an interpolation between these two results. The invariance property will hold for any N, but the larger we take N, the smaller the correlation will be between $x_{1:T}^\star$ and $x'_{1:T}$. However, it has been experienced in practice that the correlation drops of very quickly as N increases [8, 9], and for many models a moderate N (e.g. in the range 5–20) is enough to get a rapidly mixing kernel.

3.3. Final identification algorithm

From Proposition 1, if follows that CPF-AS defines a Markov kernel with the invariance properties needed for the SAEM algorithm. Before stating the final identification algorithm, however, we note that it is possible to reuse all N particle trajectories when updating the auxiliary quantity (4). That is, we update $\widehat{Q}_k$ according to,

$$\widehat{Q}_k(\theta) = (1 - \gamma_k)\widehat{Q}_{k-1}(\theta) + \gamma_k \sum_{i=1}^N \frac{w_T^i}{\sum_l w_T^l} \log p_\theta(y_{1:T}, x_{1:T}^i). \tag{9}$$

Let J be the random index of the extracted particle trajectory in Proposition 1, i.e. $x_{1:T}^\star = x_{1:T}^J$. Then, (9) is simply a Rao-Blackwellization over J. Consequently, the variance of (9) will be lower than that of (4). It should be noted, however, that the variance reduction in general will be quite small, due to path degeneracy of the CPF-AS algorithm.

We summarize the proposed method for maximum likelihood inference in nonlinear state-space models, CPF-SAEM, in Algorithm 2. The method is run until convergence, as checked by some standard convergence criterion.

Algorithm 2 CPF-SAEM

1: Set θ_0 and $x_{1:T}[0]$ arbitrarily. Set $\widehat{Q}_0(\theta) \equiv 0$.
2: **for** $k \geq 1$ **do**
3: Generate $\{x^i_{1:T}, w^i_T\}_{i=1}^N$ by running Algorithm 1, conditioned on $x_{1:T}[k-1]$ and targeting $p_{\theta_{k-1}}(x_{1:T} \mid y_{1:T})$.
4: Compute $\widehat{Q}_k(\theta)$ according to (9).
5: Compute $\theta_k = \arg\max_{\theta\in\Theta} \widehat{Q}_k(\theta)$.
6: Sample J with $P(J = i) \propto w^i_T$ and set $x_{1:T}[k] = x^J_{1:T}$.
7: **end for**

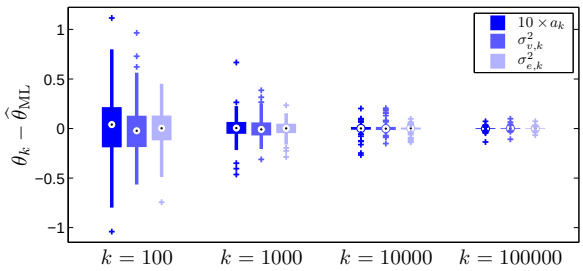

Fig. 1. Box plots of $\theta_k - \widehat{\theta}_{\mathrm{ML}}$ for different values of k, illustrating the convergence to the true MLE. The difference $(a_k - \widehat{a}_{\mathrm{ML}})$ is multiplied by a factor 10 for clarity.

4. NUMERICAL ILLUSTRATION

To start with, we evaluate CPF-SAEM on a 1st order linear system, for which the exact ML estimator (MLE) is readily available. The system is given by $x_{t+1} = ax_t + v_t$ and $y_t = x_t + e_t$ with $a = 0.9$ and where e_t and v_t are independent white Gaussian sequences, with variances $\sigma_v^2 = \sigma_e^2 = 1$. We simulate 100 batches of data from the system, each with $T = 100$. We then run Algorithm 2 for each batch to identify $\theta = (a, \sigma_v^2, \sigma_e^2)$. We use $N_{\mathrm{CPF}} = 15$ particles and a bootstrap proposal kernel in the CPF-AS. We let $\gamma_k \equiv 1$ for $k \leq 100$, and $\gamma_k \sim k^{-0.7}$ for $k > 100$. This allows for a rapid change in the parameter estimates during the initial iterations, followed by a convergent phase. We are interested in comparing the estimates with the true MLE, $\widehat{\theta}_{\mathrm{ML}}$ (this is computed by running an exact EM algorithm for 10 000 iterations). We thus compute the differences $\theta_k - \widehat{\theta}_{\mathrm{ML}}$ for $k \in \{10^2, 10^3, 10^4, 10^5\}$. Box plots of these differences, over the 100 data batches, are given in Figure 1. Despite the fact that we use a fixed (and small) number of particles, CPF-SAEM converges to the true MLE as k increases.

As a second, more challenging, example we consider the standard nonlinear time series model, used among others by [5, 23],

$$x_{t+1} = 0.5x_t + 25\frac{x_t}{1+x_t^2} + 8\cos(1.2t) + v_t, \tag{10a}$$

$$y_t = 0.05x_t^2 + e_t, \tag{10b}$$

where v_t and e_t are independent white Gaussian sequences, with variances $\sigma_v^2 = 1$ and $\sigma_e^2 = 0.1$, respectively. We use CPF-SAEM with $N_{\mathrm{CPF}} = 15$ particles to identify $\theta = (\sigma_v^2, \sigma_e^2)$. The step size is set as above. As a comparison, we use the PSEM algorithm [5], based on a forward filtering/backward simulation (FFBS) smoother [16], with $N_{\mathrm{PS}} = 1500$ forward filter particles and $M_{\mathrm{PS}} = 300$ backward trajectories.

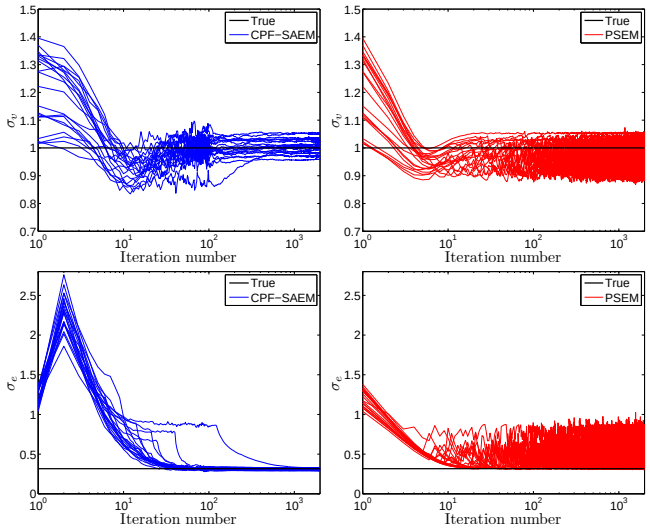

Fig. 2. Parameter estimates over 20 iterations for CPF-SAEM (left) and PSEM (right). Each line corresponds to one realization of data. The true values are $\sigma_v = 1$ and $\sigma_e = \sqrt{0.1}$, respectively.

It is interesting to note that the computational complexity of CPF-SAEM scales like $O(N_{\mathrm{CPF}}T)$ per iteration. Similarly, the complexity of PSEM is $O(N_{\mathrm{PS}}M_{\mathrm{PS}}T)$[1]. There is a striking difference where, if we neglect all computational overhead, CPF-SAEM is a factor $M_{\mathrm{PS}}N_{\mathrm{PS}}/N_{\mathrm{CPF}} = 30\,000$ less costly than PSEM!

Despite this difference, we compare the methods over equally many iterations. We generate 20 independent batches of data, each consisting of $T = 1500$ observations. For each data set, we run CPF-SAEM and PSEM for 2 000 iterations. The methods are initialized uniformly at random, $\theta_0 \in [1, 2]^2$. The results are given in Figure 2. There is a clear difference between the methods, where CPF-SAEM outperforms PSEM in both variance and bias (and, most notably, in computational time). Despite the fact that we use a fixed number of particles $N_{\mathrm{CPF}} = 15$ at each iteration, CPF-SAEM converges as we increase the number of iterations. This is not the case for PSEM, as this would require $N_{\mathrm{PS}} \to \infty$ and $M_{\mathrm{PS}} \to \infty$.

5. CONCLUSIONS

Conditional particle filters (CPFs) provide an elegant way of using SMC to construct Markov kernels which leave the exact joint smoothing distribution invariant. This holds true for any number of particles. With ancestor sampling, it is also possible to obtain a rapidly mixing kernel with a very modest number of particles. This is a key observation, meaning that we have to revise the common notion that SMC necessarily implies a high computational cost. In this contribution, the CPF with ancestor sampling has been combined with a stochastic approximation EM (SAEM) algorithm. The resulting method, CPF-SAEM, was shown to be an efficient method for maximum likelihood parameter estimation in nonlinear/non-Gaussian state-space models. Indeed, CPF-SAEM outperformed a state of the art particle-based EM algorithm in a simulation study, both in terms of bias, variance and computation time.

[1] Asymptotically (as $N_{\mathrm{PS}} \to \infty$), this can be reduced to $O(N_{\mathrm{PS}}T)$ by using a rejection-sampling-based FFBS [24]. In practice, however, the constants can be quite large and the actual gain limited [25, 26].

6. REFERENCES

[1] A. Doucet and A. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *The Oxford Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds. Oxford University Press, 2011.

[2] O. Cappé, E. Moulines, and T. Rydén, *Inference in Hidden Markov Models*, Springer, 2005.

[3] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[4] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741, 1984.

[5] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[6] J. Olsson, R. Douc, O. Cappé, and E. Moulines, "Sequential Monte Carlo smoothing with application to parameter estimation in nonlinear state-space models," *Bernoulli*, vol. 14, no. 1, pp. 155–179, 2008.

[7] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B*, vol. 72, no. 3, pp. 269–342, 2010.

[8] F. Lindsten, M. I. Jordan, and T. B. Schön, "Ancestor sampling for particle Gibbs," in *Proceedings of the 2012 Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, Dec. 2012.

[9] F. Lindsten and T. B. Schön, "On the use of backward simulation in the particle Gibbs sampler," in *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012.

[10] N. Whiteley, C. Andrieu, and A. Doucet, "Efficient Bayesian inference for switching state-space models using discrete particle Markov chain Monte Carlo methods," Tech. Rep., Bristol Statistics Research Report 10:04, 2010.

[11] B. Delyon, M. Lavielle, and E. Moulines, "Convergence of a stochastic approximation version of the EM algorithm," *The Annals of Statistics*, vol. 27, no. 1, pp. 94–128, 1999.

[12] C. Andrieu and M. Vihola, "Markovian stochastic approximation with expanding projections," arXiv.org, arXiv:1111.5421, Nov. 2011.

[13] S. Donnet and A. Samson, "EM algorithm coupled with particle filter for maximum likelihood parameter estimation of stochastic differential mixed-effects models," Tech. Rep. hal-00519576, v2, Université Paris Descartes, MAP5, 2011.

[14] G. C. G. Wei and M. A. Tanner, "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 699–704, 1990.

[15] G. Fort and E. Moulines, "Convergence of the Monte Carlo expectation maximization for curved exponential families," *The Annals of Statistics*, vol. 31, no. 4, pp. 1220–1259, 2003.

[16] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, Mar. 2004.

[17] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*, Springer-Verlag, New York, USA, 1990.

[18] C. Andrieu, E. Moulines, and P. Priouret, "Stability of stochastic approximation under verifiable conditions," *SIAM Journal on Control and Optimization*, vol. 44, no. 1, pp. 283–312, 2005.

[19] E. Kuhn and M. Lavielle, "Coupling a stochastic approximation version of EM with an MCMC procedure," *ESAIM: Probability and Statistics*, vol. 8, pp. 115–131, 2004.

[20] N. Whiteley, "Discussion on Particle Markov chain Monte Carlo methods," Journal of the Royal Statistical Society: Series B, 72(3), p 306–307, 2010.

[21] F. Gustafsson, "Particle filter theory and practice with positioning applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 7, pp. 53–82, 2010.

[22] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999.

[23] O. Cappé, "Online sequential Monte Carlo EM algorithm," in *Proceedings of the IEEE Workshop Statististical Signal Process (SSP)*, Cardiff, Wales, UK, Sept. 2009.

[24] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "Sequential Monte Carlo smoothing for general state space hidden Markov models," *Annals of Applied Probability*, vol. 21, no. 6, pp. 2109–2145, 2011.

[25] E. Taghavi, F. Lindsten, L. Svensson, and T. B. Schön, "Adaptive stopping for fast particle smoothing," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013.

[26] P. Bunch and S. Godsill, "Improved particle approximations to the joint smoothing distribution using Markov chain Monte Carlo," *IEEE Transactions on Signal Processing*, vol. 61, no. 4, pp. 956–963, 2013.