ADAPTIVE STOPPING FOR FAST PARTICLE SMOOTHING

Ehsan Taghavi[⋆]*, Fredrik Lindsten*[†]*, Lennart Svensson*[⋆] *and Thomas B. Schön*[†]

[⋆] Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden
[†]Division of Automatic Control, Linköping University, Linköping, Sweden

ABSTRACT

Particle smoothing is useful for offline state inference and parameter learning in nonlinear/non-Gaussian state-space models. However, many particle smoothers, such as the popular forward filter/backward simulator (FFBS), are plagued by a quadratic computational complexity in the number of particles. One approach to tackle this issue is to use rejection-sampling-based FFBS (RS-FFBS), which asymptotically reaches linear complexity. In practice, however, the constants can be quite large and the actual gain in computational time limited. In this contribution, we develop a hybrid method, governed by an adaptive stopping rule, in order to exploit the benefits, but avoid the drawbacks, of RS-FFBS. The resulting particle smoother is shown in a simulation study to be considerably more computationally efficient than both FFBS and RS-FFBS.

***Index Terms*—** Sequential Monte Carlo, particle smoothing, backward simulation.

1. INTRODUCTION

Consider a general nonlinear/non-Gaussian state-space model (SSM), with latent state $x_t \mid x_{t-1} \sim f(x_t \mid x_{t-1})$ and observation $y_t \mid x_t \sim g(y_t \mid x_t)$. We consider the problem of state smoothing, i.e. to infer x_t for $t \leq T$, given the collection up to time T of observations denoted as $y_{1:T} = (y_1, \ldots, y_T)$. More generally, we are interested in finding the joint smoothing distribution $p(x_{1:T} \mid y_{1:T})$, i.e. the posterior distribution of the latent state sequence $x_{1:T}$, conditionally on the observed data.

Two decades of work on sequential Monte Carlo (SMC) methods have enabled inference in SSMs beyond the linear Gaussian case [1, 2]. Several SMC-based smoothing algorithms, i.e., particle smoothers (PS), have been presented in the literature. These include forward filtering/backward smoothing (FFBSm) [3], two-filter smoothing [4, 5] and Markov chain Monte Carlo (MCMC) smoothing [6, 7, 8]. However, it remains a major challenge to develop accurate and computationally efficient methods for particle smoothing. In this paper, we are concerned with the forward filter/backward simulator (FFBS) [9, 10]. This method addresses the joint smoothing problem by first running a forward (in time) particle filter (PF) with N particles, followed by a backward simulation of M trajectories $\{\tilde{x}^j_{1:T}\}_{j=1}^M$ (see Section 2). These backward trajectories can be seen as approximate draws from $p(x_{1:T} \mid y_{1:T})$.

The computational complexity of FFBS is $O(MN)$, which can be prohibitive for many applications. However, [11] proposed a modified FFBS in which rejection sampling (RS) is used in an internal step of the algorithm, leading to the RS-FFBS algorithm. This modification does not introduce any additional approximations, retaining the good performance of FFBS, but it can be useful in reducing its computational complexity. Indeed, for $M = N$, [11] shows that RS-FFBS reaches $O(N)$ complexity as $N \to \infty$. In practice, however, RS-FFBS often requires more computational time than standard FFBS. This is due to the fact that in performing RS-FFBS, some of the particles take a long time to handle because they have very low acceptance probabilities.

To alleviate this, we propose a hybrid smoother, which mixes RS-FFBS with standard FFBS. The method starts out as an RS-FFBS at each time step, but switches to FFBS when the former method is perceived as being too slow. To determine when to switch, we propose two different stopping rules, one simple deterministic rule and one adaptive rule. As RS-FFBS, the proposed method does not introduce any additional approximations. Indeed, the proposed method is equivalent to FFBS and to RS-FFBS in terms of accuracy, but it has a lower computational cost than both of them.

There exist several related contributions, aiming at reducing the computational complexity of various PS. Numerical approximations are used by [12] to reduce the complexity of FFBSm from $O(N^2)$ to $O(N \log N)$. In [13], a modified FFBS is proposed, similar to RS-FFBS but with MCMC moves instead of RS. Another approach has been proposed by [4], the modified two-filter smoother, which uses a specific form of importance sampling to reach linear computational complexity. Contrary to RS-FFBS and the present work, however, these methods all introduce some additional approximations and/or modifications to the original algorithms.

We write $m : n$ for the set $\{m, m+1, \ldots, n\}$, $\mathrm{Cat}(\{p_i\}_{i=1}^n)$ with $\sum_i p_i = 1$ is the categorical (i.e. discrete) probability distribution on $1 : n$ with probabilities $\{p_i\}_{i=1}^n$ and $\mathcal{U}(a, b)$ is the uniform distribution on the interval $[a, b]$.

2. FORWARD FILTERING/BACKWARD SIMULATION

The FFBS algorithm [9, 10] can be used to generate approximate draws from the joint smoothing distribution. The method is a forward/backward procedure. The first step is to apply a PF to the whole dataset (see e.g. [1, 2]). Hence, for each time $t \in 1 : T$, we assume that we have access to a weighted particle system $\{x^i_t, w^i_t\}_{i=1}^N$, defining a point-mass approximation of the filtering distribution at time t,

$$\widehat{p}(dx_t \mid y_{1:t}) \triangleq \sum_{i=1}^N w^i_t \delta_{x^i_t}(dx_t), \tag{1}$$

where $\delta_z(\cdot)$ is a Dirac point-mass located at the point z. The FFBS relies on a factorization of the smoothing density according to $p(x_{1:T} \mid y_{1:T}) = p(x_T \mid y_{1:T}) \prod_{t=1}^{T-1} p(x_t \mid x_{t+1:T}, y_{1:T})$, where $p(x_t \mid x_{t+1:T}, y_{1:T}) \propto f(x_{t+1} \mid x_t) p(x_t \mid y_{1:t})$. By plugging the PF approximation (1) into this expression, we obtain a point mass

The second and the fourth authors were 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.

Algorithm 1 FFBS (at time t) [9]

Input: Partial backward trajectories $\{\tilde{x}_{t+1:T}^j\}_{j=1}^M$.
Output: Augmented backward trajectories $\{\tilde{x}_{t:T}^j\}_{j=1}^M$.

1: **for** $j = 1$ **to** M **do**
2: Compute $v_t^j = \sum_{\ell=1}^N w_t^\ell f(\tilde{x}_{t+1}^j \mid x_t^\ell)$.
3: For $i \in 1:N$, compute $\tilde{w}_{t|T}^{i,j} = w_t^i f(\tilde{x}_{t+1}^j \mid x_t^i)/v_t^j$.
4: Sample $I(j) \sim \mathrm{Cat}(\{\tilde{w}_{t|T}^{i,j}\}_{i=1}^N)$.
5: Set $\tilde{x}_t^j = x_t^{I(j)}$.
6: **end for**

approximation of the *backward kernel*,

$$\widehat{p}(dx_t \mid x_{t+1}, y_{1:T}) \triangleq \sum_{i=1}^N \frac{w_t^i f(x_{t+1} \mid x_t^i)}{\sum_l w_t^l f(x_{t+1} \mid x_t^l)} \delta_{x_t^i}(dx_t). \quad (2)$$

It follows that we can generate a backward trajectory by sampling $\tilde{x}_T \sim \widehat{p}(dx_T \mid y_{1:T})$ and then, for $t = T-1, \ldots, 1$, sample $\tilde{x}_t \sim \widehat{p}(dx_t \mid \tilde{x}_{t+1}, y_{1:T})$. The procedure is repeated M times to generate a collection of backward trajectories $\{\tilde{x}_{1:T}^j\}_{j=1}^M$ approximately distributed according to $p(x_{1:T} \mid y_{1:T})$. One time step of the FFBS is summarized in Algorithm 1.

The bottleneck of the FFBS is the computation of the smoothing weights $\{\tilde{w}_{t|T}^{i,j}\}_{i=1}^N$ (Row 3 in Algorithm 1) for $j \in 1:M$, yielding $O(MN)$ complexity. These weights are used to sample the variable $I(j)$ at Row 4 of Algorithm 1, where $I(j)$ is the index of the forward filter particle that is to be appended to the jth backward trajectory. To avoid an exhaustive evaluation of these weights, [11] suggested to instead sample $I(j)$ using rejection sampling (RS).

Assume that the transition density function is bounded from above, $f(x_{t+1} \mid x_t) \leq \rho$, which is true in many situations. We can then let the filter weights define a proposal distribution, i.e. we sample $I(j) \sim \mathrm{Cat}(\{w_t^i\}_{i=1}^N)$. The target distribution is categorical with probabilities $\{\tilde{w}_{t|T}^{i,j}\}_{i=1}^N$. The RS acceptance probability is given by the ratio between the target and the proposal probabilities,

$$\frac{\tilde{w}_{t|T}^{I(j),j}}{w_t^{I(j)}} \propto \frac{f(\tilde{x}_{t+1}^j \mid x_t^{I(j)})}{\rho} \leq 1. \quad (3)$$

It follows that the proposed sample should be accepted with probability $f(\tilde{x}_{t+1}^j \mid x_t^{I(j)})/\rho$, otherwise it is rejected and a new sample is generated. One time step of RS-FFBS is given in Algorithm 2.

Algorithm 2 Rejection sampling FFBS (at time t) [11]

Input: Partial backward trajectories $\{\tilde{x}_{t+1:T}^j\}_{j=1}^M$.
Output: Augmented backward trajectories $\{\tilde{x}_{t:T}^j\}_{j=1}^M$.

1: $k \leftarrow 0$
2: $L_0 \leftarrow \{1:M\}$.
3: **while** L_k is not empty **do**
4: $m_k \leftarrow \mathrm{cardinality}(L_k)$.
5: $\delta \leftarrow \emptyset$.
6: Sample independently $\{C(\ell)\}_{\ell=1}^{m_k} \sim \mathrm{Cat}(\{w_t^i\}_{i=1}^N)$.
7: Sample independently $\{U(\ell)\}_{\ell=1}^{m_k} \sim \mathcal{U}(0,1)$.
8: **for** $\ell = 1$ **to** m_k **do**
9: **if** $U(\ell) \leq f(\tilde{x}_{t+1}^{L_k(\ell)} \mid x_t^{C(\ell)})/\rho$ **then**
10: $I(L_k(\ell)) \leftarrow C(\ell)$.
11: $\delta \leftarrow \delta \cup \{L_k(\ell)\}$.
12: **end if**
13: **end for**
14: $L_{k+1} \leftarrow L_k \setminus \delta$.
15: $k \leftarrow k+1$.
16: **end while**
17: For $j \in 1:M$, set $\tilde{x}_t^j = x_t^{I(j)}$.

Algorithm 3 RS-FFBS with early stopping

Input: Forward filter particle system $\{x_t^i, w_t^i\}_{i=1}^N$ for $t \in 1:T$.
Output: Backward trajectories $\{\tilde{x}_{1:T}^j\}_{j=1}^M$.

1: Sample independently $\{I(j)\}_{j=1}^M \sim \mathrm{Cat}\left(\{w_T^i\}_{i=1}^N\right)$.
2: Set $\tilde{x}_T^j = x_T^{I(j)}$ for $j \in 1:M$.
3: **for** $t = T-1$ **to** 1 **do**
4: $k \leftarrow 0$.
5: $L_0 \leftarrow \{1:M\}$.
6: **while** Stopping criterion is not met **and** L_k is not empty **do**
7: Run one iteration of RS (line 4 to 14 in Algorithm 2).
8: $k \leftarrow k+1$.
9: **end while**
10: **if** L_k is not empty **then**
11: Sample $I(j)$ for $j \in L_k$ using FFBS (Algorithm 1).
12: **end if**
13: Set $\tilde{x}_t^j = x_t^{I(j)}$ for $j \in 1:M$.
14: **end for**

3. AN ADAPTIVE STOPPING RULE FOR RS-FFBS

3.1. RS-FFBS with early stopping

The rationale for using RS within the FFBS is that the RS-FFBS sampler can be shown to reach linear computational complexity as the number of particles tend to infinity [11]. However, we note that there is no upper bound on the number of times that the while-loop in Algorithm 2 may be executed. It has been observed in practice that the computational time of Algorithm 2 in fact can be larger than that of Algorithm 1 [14, 13].

Since the RS acceptance probability depends on x_{t+1}, different backward trajectories get different probabilities of acceptance. The trajectories with high acceptance probabilities are typically accepted early in the process, whereas the trajectories with low probabilities can remain for many iterations. That is, most of the time required by RS-FFBS is spent on just a few particles. This has the effect that the cardinality of L_k decreases fast in the beginning, but it can linger for a long time close to zero.

To speed up the algorithm, we propose a hybrid strategy in which we run the RS-loop for a certain number of iterations, and then switch to standard FFBS for the remaining particles (i.e. by making an exhaustive evaluation of the remaining weights). The general method based on this idea, RS-FFBS with early stopping, is given in Algorithm 3.

It remains to determine an appropriate stopping criterion. The family of methods given by Algorithm 3 can be characterized by a sequence $K_{1:T}$, where K_t is the maximum number of iterations of the RS-loop, allowed at time t. This sequence of stopping times can either be specified *a priori* or computed online as the sampler progresses.

The first stopping rule that we propose is a very simple one, namely to set $K_t \equiv \bar{K}$ for some constant $\bar{K}$. That is, when running the RS-FFBS, we allow for a maximum number of $\bar{K}$ RS-iterations, before the RS-loop is stopped and FFBS is employed for the remaining particles. We refer to this deterministic stopping rule as $\mathcal{S}_{\text{det}}$. It is

instructive to note that our hybrid strategy (naturally) contains both FFBS and RS-FFBS as special cases, by taking $\bar{K} = 0$ and $\bar{K} = \infty$, respectively. Hence, it is reasonable to assume that the additional degree of freedom provided by the choice of $K_{1:T}$ can be used to find an, in some sense, optimal trade-off between the two basic methods.

Due to its simplicity, we believe that $\mathcal{S}_{\text{det}}$ can be of particular practical interest. In a numerical evaluation in Section 4, it is shown that this stopping rule indeed can reduce the computation time quite considerably, compared to both FFBS and RS-FFBS. However, an issue with $\mathcal{S}_{\text{det}}$ is how to choose $\bar{K}$, which is likely to be problem dependent. Furthermore, since we run the RS-loop for each time $t \in 1 : T$, it might be the case that we want to use different values for K_t for different t, due to variations in the data. To address these issues, we now turn to the derivation of an adaptive stopping rule, which tunes itself by monitoring the number of accepted samples in the RS-loop.

3.2. Optimal stopping

To find an adaptive stopping rule, we start by investigating the problem of minimizing the computational time of Algorithm 3 w.r.t. the sequence $K_{1:T}$. We make the following assumption.

(A1) Let N be the number of forward filter particles. Assume that there exist positive constants d_0 and d_1, such that; *(i)* the computational cost of running one iteration of the RS-loop in Algorithm 2 with m backward trajectories is $d_0 m$; *(ii)* the computational cost of running FFBS (Algorithm 1) with m backward trajectories is $N d_1 m$.

The constants d_0 and d_1 are implementation dependent, but can be computed from initial test runs. After k iterations of the RS-loop, we wish to deduce whether it is beneficial to run RS for one more iteration or not. Hence, let us now define two competing actions,

$$\begin{cases} D_0 : \text{Run RS for one more iteration, then stop,} \\ D_1 : \text{Stop RS now.} \end{cases}$$

We then decide to stop the RS-loop whenever the expected cost of D_0 is greater than the cost of D_1. Otherwise, we run the $k+1$'th iteration of the RS-loop and then check the stopping criteria again.

Let $m_k = \text{cardinality}(L_k)$ be the number of remaining particles after k iterations of the RS-loop. Hence, m_k is a non-increasing sequence with $m_0 = M$. Under (A1), the cost of D_1 is given by

$$C_k(D_1) = N d_1 m_k. \tag{4a}$$

Similarly, the expected cost of D_0 is

$$C_k(D_0) = d_0 m_k + N d_1 (m_k - \bar{a}_k), \tag{4b}$$

where $\bar{a}_k = \mathrm{E}[a_k]$ and a_k is the number of accepted draws in the $k+1$'th RS-iteration. With $\mathbf{1}(\cdot)$ being an indicator function, it follows that

$$\begin{aligned} \bar{a}_k &= \mathrm{E}\left[\sum_{\ell=1}^{m_k} \mathbf{1}(\text{the } L_k(\ell)\text{'th backward trajectory is accepted})\right] \\ &= \frac{1}{\rho}\sum_{\ell=1}^{m_k} \mathrm{E}[f(\tilde{x}_{t+1}^{L_k(\ell)} \mid x_t^{C(\ell)})] = \frac{1}{\rho}\sum_{\ell=1}^{m_k}\sum_{i=1}^{N} w_t^i f(\tilde{x}_{t+1}^{L_k(\ell)} \mid x_t^i). \end{aligned} \tag{5}$$

Here, the expectation is implicitly conditioned on the forward filter particle system and the backward trajectories down to time $t+1$.

With $\bar{p}_k = \bar{a}_k / m_k$ being the average acceptance probability, it follows from (4) that

$$C_k(D_1) < C_k(D_0) \Leftrightarrow \bar{p}_k < \frac{d_0}{N d_1}. \tag{6}$$

Interestingly, (6) is an inequality related to only one unknown variable, namely $\bar{p}_k$. Under (A1), the stopping rule defined by (6), i.e. to stop the rejection sampling whenever the average acceptance probability falls below a given threshold, is optimal w.r.t. the expected computational time. We refer to this stopping rule as $\mathcal{S}_{\text{opt}}$.

However, so far we have neglected the overhead in computational time caused by the evaluation of the stopping criteria. For $\mathcal{S}_{\text{opt}}$, this cost is clearly not negligible, since the computation of (5) scales as $O(MN)$, i.e. of the same order as the FFBS. To overcome this problem, we seek an approximation of the optimal stopping rule $\mathcal{S}_{\text{opt}}$, with a much smaller computational overhead. In particular, we seek a method in which the computational complexity of evaluating the stopping criteria is independent of N and M. To accomplish this, we consider a filtering approach to estimate $\bar{p}_k$ at each iteration of the RS-loop.

3.3. Adaptive stopping

As pointed out above, $\bar{p}_k$ is the only unknown variable which we are interested in tracking at each iteration of the RS-loop, as d_0 and d_1 can be calculated before running the algorithm.

To obtain an as simple tracker as possible, we choose a linear Gaussian model for $\bar{p}_k$, so that it can be estimated by running a conventional Kalman filter (KF). First, we note that $\bar{p}_k$ is a variable which is normally decreasing with k. The reason is that the backward trajectories with highest acceptance probabilities are typically accepted early, leaving trajectories with lower probabilities. The measurement is taken as a_k, since this is the only variable which is readily available. We thus specify a state-space model for the average acceptance probability according to,

$$\begin{cases} \bar{p}_k = \left(1 - \frac{a_{k-1}}{m_{k-1}}\right)\bar{p}_{k-1} + v_k & \text{state model,} \\ a_k = m_k \bar{p}_k + w_k & \text{observation model,} \end{cases} \tag{7}$$

The noises v_k and w_k are modeled as mutually independent zero-mean Gaussian processes with $v_k \sim \mathcal{N}(0, \sigma_v^2)$ and $w_k \sim \mathcal{N}(0, \sigma_w^2)$, respectively. The initial state is modeled as $\bar{p}_0 \sim \mathcal{N}(x_0, P_0)$. Since a_k is not available until after RS-iteration $k+1$, we make one step ahead predictions $\widehat{p}_{k|k-1}$, which are used in place of $\bar{p}_k$ in the stopping rule (6). The additional steps for Algorithm 3, required by the adaptive stopping rule, are given in Algorithm 4. We refer to this stopping rule as $\mathcal{S}_{\text{adpt}}$.

Algorithm 4 Adaptive stopping rule

After running Step 8 of Algorithm 3, do:

1: Compute $a_{k-1} = m_{k-1} - m_k$.
2: Make a KF measurement update to compute $\widehat{p}_{k-1|k-1}$ and $P_{k-1|k-1}$, using the model (7).
3: Make a KF time update to compute $\widehat{p}_{k|k-1}$ and $P_{k|k-1}$, using the model (7).
4: **if** $\widehat{p}_{k|k-1} < \frac{d_0}{N d_1}$ **then**
5: Stop rejection sampling.
6: **end if**

Table 1: CPU times in seconds for 1D linear model

q	FFBS	RS-FFBS	$\bar{K}=\frac{M}{5}$	$\bar{K}=\frac{M}{10}$	$\bar{K}=\frac{M}{20}$	$\mathcal{S}_{\text{adpt}}$
10	24.86	0.72	0.35	0.34	0.37	0.36
1	26.27	5.30	0.69	0.60	0.61	0.64
0.1	25.21	18.29	2.86	1.87	1.11	1.18
0.01	27.32	17.24	2.84	2.33	1.75	1.92

Table 2: CPU times in seconds for 2D linear model

τ	FFBS	RS-FFBS	$\bar{K}=\frac{M}{5}$	$\bar{K}=\frac{M}{10}$	$\bar{K}=\frac{M}{20}$	$\mathcal{S}_{\text{adpt}}$
0.1	44.65	19.5	2.03	1.94	2.36	1.92
1	45.28	77.71	3.49	3.65	5.33	3.79
10	48.63	355.7	14.83	20.21	25.86	15.25

Table 3: CPU times in seconds for a standard nonlinear model

	FFBS	RS-FFBS	$\bar{K}=\frac{M}{5}$	$\bar{K}=\frac{M}{10}$	$\bar{K}=\frac{M}{20}$	$\mathcal{S}_{\text{adpt}}$
NL	21.57	2.48	1.04	0.84	0.78	0.80

4. NUMERICAL ILLUSTRATION

To show how well the proposed stopping rules, $\mathcal{S}_{\text{det}}$ and $\mathcal{S}_{\text{adpt}}$, perform in comparison with FFBS and RS-FFBS, we evaluate them on three different models. We emphasize that all the considered particle smoothers are equivalent in terms of accuracy. Indeed, all the methods will generate samples from the same distribution, and it is only the computational time for the simulation that differs. Consequently, we only consider the computational times of the different methods in the evaluation. For an illustration of the performance of FFBS, see e.g. [9]. All the simulations are done on a standard laptop Intel(R) Core(TM) i7-3720Qm 2.60GHz platform with 8GB of RAM.

First, we consider a linear-Gaussian model defined by:

$$x_{t+1} = 0.9x_t + n_{x,t} \tag{8a}$$
$$y_t = x_t + n_{y,t}, \tag{8b}$$

where $n_{x,t} \sim \mathcal{N}(0, q)$ and $n_{y,t} \sim \mathcal{N}(0, r)$. Furthermore, we choose $r = 1$ and $q \in \{0.01, 0.1, 1, 10\}$. The reason to choose different values for q is to simulate different acceptance probabilities in the rejection sampler, where $q = 0.01$, corresponds to a very low acceptance probability and $q = 10$, correspond to a fairly high acceptance probability. Moreover, we choose $N = 5000$ forward filter particles and $M = 1000$ backward trajectories. For the deterministic stopping rule, we consider three different thresholds: $\bar{K} \in \{\frac{M}{5}, \frac{M}{10}, \frac{M}{20}\}$. The setting for the KF-based (adaptive) stopping rule is as follows: $\bar{p}_0 \sim \mathcal{N}(0.5, 0.001)$, $\sigma_v^2 = m_k^{-1}$ and $\sigma_w^2 = 1$.

We run all the algorithms for five different datasets, each consisting of $T = 100$ samples, and averaging ten times for each dataset to cancel the effects of randomness. We clock the CPU times using the `tic` and `toc` commands in Matlab. The results for different choices of q are shown in Table 1. For high acceptance probabilities ($q = 10$), there is a very big gain in using RS-FFBS instead of FFBS. However, as the acceptance probability is decreased (smaller q), this gain is diminished. By using the proposed stopping rules, we maintain a large improvement even for small acceptance probabilities. All methods with early stopping provide a significant improvement over both FFBS and RS-FFBS. Furthermore, the adaptive stopping rule results in CPU times which appear to be close to a fairly good tuning of the deterministic stopping rule (corresponding to $\bar{K} = \frac{M}{20}$).

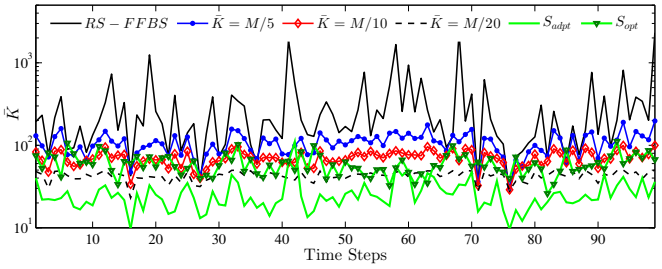

Fig. 1: Value of K_t for different stopping rules at each time step for a standard nonlinear example.

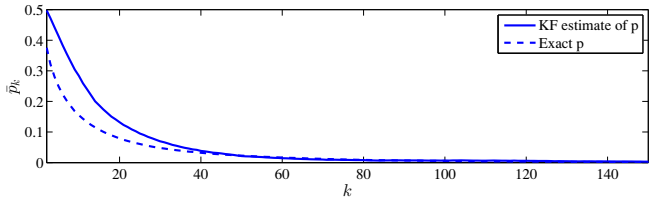

Fig. 2: Value of $\bar{p}_k$ for $\mathcal{S}_{\text{adpt}}$ and $\mathcal{S}_{\text{opt}}$ at each iteration of RS scheme averaged over all time steps for a standard nonlinear example.

As a second example, we consider a second order linear Gaussian model, previously used in [4]. We use different values for the parameter τ (see [4] for details on the model) to represent different acceptance probabilities. In all other respect, we use the same settings as above. The results are reported in Table 2. This is a more challenging model for RS-FFBS, which is evident as its CPU time increases considerably as τ increases. Again, there is a considerable gain in using early stopping. The best tuning for the deterministic rule (among the considered values) is for this model $\bar{K} = \frac{M}{5}$, which is different from the previously considered model. Still, the adaptive stopping rule results in CPU time close to this value for the deterministic rule. This highlights the ability of the adaptive rule to automatically tune itself to the properties of the model.

Finally, we consider a standard nonlinear time-series model previously used by, among others, [15, 16, 9]. The CPU times are reported in Table 3. We also plot the resulting sequences $K_{1:T}$ for the different stopping rules, for one specific batch of data, in Figure 1. In this plot, we also include the optimal stopping rule $\mathcal{S}_{\text{opt}}$, computed using (5) and (6). In Figure 2 we also plot the KF filter estimate of $\bar{p}_k$ used in $\mathcal{S}_{\text{adpt}}$, together with the actual value of $\bar{p}_k$.

5. CONCLUSIONS

We have developed an extension to the rejection-sampling-based forward filter/backward simulator (RS-FFBS) particle smoother, denoted RS-FFBS with early stopping. The proposed method is a hybrid between RS-FFBS and standard FFBS. A specific stopping rule determines when to abort the rejection sampling loop and make an exhaustive evaluation of the backward sampling probabilities. We have proposed two different stopping rules. First, a deterministic rule which is attractive due to its simplicity. Second, an adaptive rule which is attractive due to its ability to automatically tune itself to the model under study. Both stopping rules were found to significantly reduce the computational cost over both FFBS and RS-FFBS in a simulation study.

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

[3] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[4] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, no. 2, pp. 447–464, 2010.

[5] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state-space models," *Annals of the Institute of Statistical Mathematics*, vol. 62, no. 1, pp. 61–89, Feb. 2010.

[6] C. Dubarry and R. Douc, "Particle approximation improvement of the joint smoothing distribution with on-the-fly variance estimation," arXiv.org, arXiv:1107.5524, July 2011.

[7] 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.

[8] 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.

[9] 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.

[10] A. Doucet, S. J. Godsill, and M. West, "Monte Carlo filtering and smoothing with application to time-varying spectral estimation," in *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Istanbul, Turkey, June 2000.

[11] 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.

[12] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell, and D. Lang, "Fast particle smoothing: if I had a million particles," in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, USA, June 2006.

[13] 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.

[14] F. Lindsten, *Rao-Blackwellised particle methods for inference and identification*, Licentiate thesis no. 1480, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, June 2011.

[15] M. L. Andrade Netto, L. Gimeno, and M. J. Mendes, "A new spline algorithm for non-linear filtering of discrete time systems," in *Proceedings of the 7th Triennial World Congress*, Helsinki, Finland, 1979, pp. 2123–2130.

[16] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, no. 2, pp. 107 –113, Apr. 1993.