Improved Particle Approximations to the Joint Smoothing Distribution Using Markov Chain Monte Carlo

Pete Bunch*, *Member, IEEE,* and Simon Godsill, *Member, IEEE*

Abstract

Particle filtering and smoothing algorithms approximate posterior state distributions with a set of samples drawn from those distributions. Conventionally, samples from the joint smoothing distribution are generated by sequentially resampling from the particle filter results. If the number of filtering particles is high, this process is limited by computational complexity. In addition, the support of the smoothing distribution is restricted to the values which appear in the filtering approximation. In this paper, a Metropolis-Hastings sampling procedure is used to improve the efficiency of the particle smoother, achieving comparable error performance but with a lower execution time. In addition, an algorithm for approximating the joint smoothing distribution without limited support is presented, which achieves simultaneous improvements in both execution time and error. These algorithms also provide a greater degree of flexibility over existing methods, allowing a trade-off between execution time and error, controlled by the length of the Markov chains.

EDICS Categories: MLR-BAYL, SSP-NGAU, SSP-TRAC, SSP-FILT

P. Bunch and S. Godsill are with the Department of Engineering, Cambridge University, UK. email: {pb404,sjg30}@cam.ac.uk

Manuscript received April 03, 2012.

I. INTRODUCTION

The objective of sequential Bayesian inference is the estimation of an unknown, time-varying quantity from incomplete or inaccurate observations. This is commonly achieved through the use of discrete-time probabilistic models for the evolution of the latent state and the measurement process, defined by transition and observation densities,

$$x_k \sim p(x_k|x_{k-1}) \tag{1}$$

$$y_k \sim p(y_k|x_k). \tag{2}$$

Here, the random variable x_k denotes the value of the latent state at the kth instant, and y_k the value of the observation. A set of random variables will be written as $x_{1:k} = \{x_1, x_2, \ldots, x_k\}$. The state evolution is assumed to be Markovian, i.e. x_k depends only on the previous value, x_{k-1}.

Commonly of interest are the problems of filtering and smoothing. Filtering is the inference of $p(x_k|y_{1:k})$, the distribution of the current state given all previous observations. Smoothing comprises two related tasks, the inference of $p(x_{1:K}|y_{1:K})$ (where K is the number of time steps), the joint distribution of the entire state sequence given all the observations, and that of $p(x_k|y_{1:K})$, the marginal distribution of each state given all the observations. In this paper, we are concerned with the estimation of the joint smoothing distribution.

In the case where the state and observation processes of (1) and (2) are linear and Gaussian, the filtering problem may be solved analytically using the Kalman filter [1]. For nonlinear or non-Gaussian models, tractable closed-form solutions are rare. Instead, numerical approximations are often employed, including the particle filter algorithm, first introduced by [2]. (See [3], [4] for a thorough introduction.)

In a particle filter, the filtering distribution is approximated by a set of weighted samples (or "particles") drawn from that distribution,

$$\hat{p}(x_k|y_{1:k}) = \sum_i w_k^{(i)} \delta_{x_k^{(i)}}(x_k). \tag{3}$$

Here, $\delta_a(x)$ indicates a discrete probability mass at the point $x = a$, and $w_k^{(i)}$ is the weight of the ith particle. Weights are non-negative and sum to 1.

Similar ideas apply to the problem of smoothing. In the linear Gaussian case, the entire state sequence, $x_{1:K}$, may be estimated in closed-form using the Rauch-Tung-Striebel (RTS) smoother [5]. This works by modifying the ordinary Kalman filter results in a backward processing pass through the state sequence. For nonlinear or non-Gaussian models, it is possible again to resort to numerical methods. Now, particles

are used to approximate the joint smoothing distribution,

$$\hat{p}(x_{1:K}|y_{1:K}) = \sum_i w_K^{(i)} \delta_{x_{1:K}^{(i)}}(x_{1:K}). \tag{4}$$

Each particle is an entire realisation of the state process, with K values corresponding to the state at each time step. Methods for generating these samples have been described in [6]–[8]. In this paper new algorithms are presented for drawing samples from the joint smoothing distribution, using Metropolis-Hastings (MH). These novel procedures have computational advantages and greater flexibility over previous methods.

In section II, we briefly review the basic particle filter and existing joint smoother algorithms. The new MH-based smoothing methods are presented in sections III and IV with supporting simulations in section V.

II. Particle Filtering and Smoothing

A. The Particle Filter

The particle filter is a recursive numerical algorithm for estimation of the filtering distribution. At the kth processing step, a particle estimate of the joint density over x_{k-1} and x_k is made using importance sampling (IS). A particle is first sampled from the factorised proposal density,

$$\{x_{k-1}^{(i)}, x_k^{(i)}\} \sim q(x_x|x_{k-1})q(x_{k-1}). \tag{5}$$

The proposal for x_k, $q(x_x|x_{k-1})$, is often called the "importance density". Common choices for this include the model transition density (1), and the "optimal importance density" [9]. The proposal density for x_{k-1}, $q(x_{k-1})$, is constructed from the particles of the filtering approximation from the previous time step. The choice of weights and sampling procedure for x_{k-1} determine what form of "resampling" is to be used. Here we use arbitrary weights for generality,

$$q(x_{k-1}) = \sum_i v_{k-1}^{(i)} \delta_{x_{k-1}^{(i)}}(x_{k-1}). \tag{6}$$

The choice $v_{k-1}^{(i)} = w_{k-1}^{(i)}$ corresponds to ordinary resampling, and other choices of $v_{k-1}^{(i)}$ represent auxiliary sampling schemes [3], [10]. For the least computational expense, the proposal density with weights $v_{k-1}^{(i)} = 1/N_F$ (where N_F is the number of filter particles) can be "sampled" by simply keeping the set of particles generated in the last step, with no resampling. This, however, leads to degeneracy of the particle weights over time. (See [3], [4] for further discussion of resampling.)

1: Initialise particles from prior, $x_0^{(i)} \sim p(x_0)$.
2: **for** $k = 1$ **to** K **do**
3: **for** $i = 1$ **to** N_F **do**
4: Resample last state $x_{k-1}^{(i)} \sim \sum_j v_{k-1}^{(j)} \delta_{x_{k-1}^{(j)}}(x_{k-1})$.
5: Sample new state $x_k^{(i)} \sim q(x_k|x_{k-1}^{(i)})$.
6: Weight $w_k^{(i)} \propto \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_x^{(i)}|x_{k-1}^{(i)})} \times \frac{w_{k-1}^{(i)}}{v_{k-1}^{(i)}}$.
7: Discard $x_{k-1}^{(i)}$.
8: **end for**
9: Scale weights so that $\sum_i w_k^{(i)} = 1$.
10: **end for**

Fig. 1. Particle filter algorithm

Next, the particles are assigned an importance weight according to the ratio of the target and the proposal densities,

$$\begin{aligned} w_k^{(i)} &= \frac{p(x_{k-1}^{(i)}, x_k^{(i)}|y_{1:k})}{q(x_x^{(i)}|x_{k-1}^{(i)})q(x_{k-1}^{(i)})} \\ &\propto \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})p(x_{k-1}^{(i)}|y_{1:k-1})}{q(x_x^{(i)}|x_{k-1}^{(i)})q(x_{k-1}^{(i)})} \\ &= \frac{p(y_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_x^{(i)}|x_{k-1}^{(i)})} \times \frac{w_{k-1}^{(i)}}{v_{k-1}^{(i)}}. \end{aligned} \tag{7}$$

Normalisation is enforced by scaling the weights so that they sum to 1. Finally, x_{k-1} is marginalised from the distribution by simply discarding the $x_{k-1}^{(i)}$ values from each particle. The particle filter is summarised in Fig. 1.

B. Existing Algorithms for Particle Smoothing

The particle filter generates a numerical approximation to each filtering distribution, $p(x_k|y_{1:k})$ by simulating a set of samples from each. A similar sampling-based method may be used to estimate the smoothing distribution.

The most basic particle smoother was presented in [6], and is a straightforward extension of the particle filter. Rather than marginalising the previous states from each particle, these past values are stored as well.

Initialise particles from prior, $x_0^{(i)} \sim p(x_0)$.
for $k = 1$ **to** K **do**
  for $i = 1$ **to** N_F **do**
    Resample history $x_{1:k-1}^{(i)} \sim \sum_j v_{k-1}^{(j)} \delta_{x_{1:k-1}^{(j)}}(x_{1:k-1})$.
    Sample new state $x_k^{(i)} \sim q(x_k | x_{k-1}^{(i)})$.
    Weight $w_k^{(i)} \propto \frac{p(y_k|x_k^{(i)}) p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_x^{(i)}|x_{k-1}^{(i)})} \times \frac{w_{k-1}^{(i)}}{v_{k-1}^{(i)}}$.
  end for
  Scale weights so that $\sum_i w_k^{(i)} = 1$.
end for

Fig. 2. Filter-smoother algorithm

Thus, at each step the algorithm generates an approximation to $p(x_{1:k}|y_{1:k})$. The output from the final processing step is an approximation to the desired smoothing distribution. This simple filter-smoother is summarised in Fig. 2.

The weakness of the filter-smoother is degeneracy of the particles in the path-space. Every time the particles are resampled, those with low weights do not appear in the approximation at the next step, whereas those with high weights appear multiple times. The result is that many, or even all, particles will share a common ancestor, before which they all have the same set of state values. If the objective is only to produce a single point estimate then this might be sufficient, but to characterise the complete smoothing distribution, the path-space diversity of the approximation must be increased.

An improved particle smoother was presented in [7], the forward-filtering-backward-sampling (FFBS) algorithm. This works by constructing new particles from the smoothing distribution by sampling from the filtering approximations. The sampling procedure exploits the following factorisation of the joint distribution,

$$p(x_{1:K}|y_{1:K}) = p(x_K|y_{1:K}) \prod_{k=1}^{K-1} p(x_k|x_{k+1}, y_{1:K}). \tag{8}$$

Particles are generated by sampling backwards in time from the factors of this expansion. At the kth step, a sample $\tilde{x}_{k+1:K} \sim p(x_{k+1:K}|y_{1:K})$ will already have been drawn. By sampling $x_k \sim p(x_k|\tilde{x}_{k+1}, y_{1:K})$ and appending x_k to $\tilde{x}_{k+1:K}$, the state sequence is sequentially extended back in time. The recursion is initialised with $x_K \sim p(x_K|y_{1:K})$, i.e. a sample from the final filtering approximation.

Run a particle filter to approximate $p(x_k|y_{1:k})$ for each k and store the resulting particles $\{x_k^{(i)}, w_k^{(i)}\}$.
for $i = 1$ **to** N_S **do**
  Sample $\tilde{x}_K^{(i)} \sim \sum_j w_K^{(j)} \delta_{x_K^{(j)}}(x_K)$.
  for $k = K-1$ **to** 1 **do**
    for $j = 1$ **to** N_F **do**
      Calculate weight $\tilde{w}_k^{(j)} = w_k^{(j)} p(\tilde{x}_{k+1}|x_k^{(j)})$.
    end for
    Sample $\tilde{x}_k^{(i)} \sim \sum_j \tilde{w}_k^{(j)} \delta_{x_k^{(j)}}(x_k)$.
  end for
end for

Fig. 3. Direct forward-filtering-backward-sampling algorithm

The backwards conditional density may be expanded with Bayes' rule,

$$\begin{aligned} p(x_k|\tilde{x}_{k+1}, y_{1:K}) &= p(x_k|\tilde{x}_{k+1}, y_{1:k}) \\ &\propto p(\tilde{x}_{k+1}|x_k) p(x_k|y_{1:k}). \end{aligned} \tag{9}$$

Substituting the filtering approximation into this expression yields a particle distribution which may be sampled easily,

$$\hat{p}(x_k|\tilde{x}_{k+1}, y_{1:K}) = \sum_i \tilde{w}_k^{(i)} \delta_{x_k^{(i)}}(x_k) \tag{10}$$

$$\tilde{w}_k^{(i)} \propto w_k^{(i)} p(\tilde{x}_{k+1}|x_k^{(i)}). \tag{11}$$

The procedure may be repeated to produce as many samples from the joint smoothing distribution as required. In this paper, we refer to this algorithm as the direct FFBS (D-FFBS) algorithm, in order to distinguish it from the MH variant developed in the next section. The D-FFBS algorithm is summarised in Fig. 3.

A method for drawing samples from the joint smoothing distribution, $p(x_{1:K}|y_{1:K})$, is included in [8]. The principal topic of this paper is a two-filter smoother for estimation of the marginal smoothing distributions, $p(x_k|y_{1:K})$, by combination of the results from one forward and one backward particle filter. Particles from the joint distribution may then be generated by resampling from either the forward or backward filters (or some combination) in the manner of the D-FFBS algorithm outlined above.

The complexity of both the D-FFBS smoother of [7] and the two-filter variant of [8] is $\mathcal{O}(N_F \times N_S \times K)$, where N_F is the number of filter particles, N_S the number of smoother particles, and K the number of time steps. In [11], a theoretically linear-cost scheme using rejection sampling is outlined. In practice this often appears to be slower than the standard algorithms when there are more than a few state dimensions, due to high rejection rates. Methods for reducing the computational complexity of smoothing algorithms are also discussed in [12]. However, these focus on approximating a sum of kernel values over a set of particles, such as those which arise in forward-backward-smoothing [9] and two-filter smoothing [8] for estimation of the marginal smoothing distributions. They are not applicable for the joint smoothing procedures.

In the algorithms described so far, no new states are sampled during the backwards smoothing pass; state values are only "recycled" from the particle filtering approximations. This limits the support of the smoother to the states that appear in these filtering particles. In [13], an IS-based algorithm for estimation of the marginal smoothing distributions is described in which new states are sampled from a proposal density, thus recovering the full support. In addition, this algorithm achieves $\mathcal{O}(N_S \times K)$ complexity, i.e. linear in the number of particles.

III. An MCMC Particle Smoother

The bottleneck of the D-FFBS algorithm is the calculation of the sampling weights, specified by (11). For each smoother particle and for each time step, a weight must be calculated for each of the N_F filter particles, in order to draw a sample from the conditional distribution $\hat{p}(x_k|\tilde{x}_{k+1}, y_{1:K})$. The innovation here is to use a Markov chain Monte Carlo (MCMC) procedure to produce these particles instead of sampling directly.

MCMC algorithms are a class of numerical procedures which use a Markov chain to draw dependent samples from a target probability distribution. A thorough introduction can be found in [14]. The basic Metropolis-Hastings (MH) [15] sampler uses a series of steps in which a new state is drawn from a proposal distribution and accepted with a probability determined by the ratios of the target and proposal densities.

For the D-FFBS, the objective at each time step was to draw a sample from $p(x_k|\tilde{x}_{k+1}, y_{1:K})$. To simplify the notation and initialisation of chains, the MH version will target $p(x_{1:k}|\tilde{x}_{k+1}, y_{1:K})$. Theoretically this means using the particles of the filter-smoother, $\{x_{1:k}^{(i)}\}$, rather than just the filter, $\{x_k^{(j)}\}$, which would require $\mathcal{O}(K^2)$ memory. However, as discussed later, this is not required in practice.

As before, the algorithm begins by first running a particle filter-smoother and then resampling states

backwards through time to produce particles from the joint smoothing distribution, using the factorisation of (8). At the kth time step, a Markov chain is used to sample from $p(x_{1:k}|\tilde{x}_{k+1}, y_{1:K})$. The chain is initialised with the output from the previous processing step. As this is itself a sample from the target distribution, no burn-in period is required.

Following an equivalent derivation to that for (10), the target distribution may be approximated by a set of particles,

$$\hat{p}(x_{1:k}|\tilde{x}_{k+1}, y_{1:K}) = \sum_i \tilde{w}_k^{(i)} \delta_{x_{1:k}^{(i)}}(x_{1:k}) \tag{12}$$

$$\tilde{w}_k^{(i)} \propto w_k^{(i)} p(\tilde{x}_{k+1}|x_k^{(i)}). \tag{13}$$

A valid proposal may be constructed using the particles of the filter-smoother approximation with any arbitrary weights,

$$q(x_{1:k}|\tilde{x}_{k+1}, y_{1:K}) = \sum_j \tilde{v}_k^{(j)} \delta_{x_{1:k}^{(j)}}(x_{1:k}). \tag{14}$$

If the current state of the chain is $x_{1:k}^{(m)}$, and a new sample is drawn from the proposal $x_{1:k}^* \sim q(x_{1:k}|\tilde{x}_{k+1}, y_{1:K})$, then it should be accepted with the following probability,

$$\begin{aligned} \alpha &= \min\left[1, \frac{\hat{p}(x_{1:k}^*|\tilde{x}_{k+1}, y_{1:K}) q(x_{1:k}^{(m)}|\tilde{x}_{k+1}, y_{1:K})}{\hat{p}(x_{1:k}^{(m)}|\tilde{x}_{k+1}, y_{1:K}) q(x_{1:k}^*|\tilde{x}_{k+1}, y_{1:K})}\right] \\ &= \min\left[1, \frac{w_k^* \tilde{v}_k^{(m)}}{w_k^{(m)} \tilde{v}_k^*} \times \frac{p(\tilde{x}_{k+1}|x_k^*)}{p(\tilde{x}_{k+1}|x_k^{(m)})}\right]. \end{aligned} \tag{15}$$

Note that w_k^* indicates the filtering particle weight of the proposed new state, $x_{1:k}^*$. The final state of each chain is used as a sample from the required conditional distribution. The resulting MH-FFBS algorithm is summarised in Fig. 4.

For a practical implementation, it is not necessary to store the complete particles of the filter-smoother approximations, $\hat{p}(x_{1:k}|y_{1:k})$, which would require memory space which scaled quadratically with K. As the smoother progresses backwards in time, the earlier states in the sequence are repeatedly overwritten, and so it is not necessary ever to have known them. Thus, it is only necessary to store the last two states in each filter-smoother particle (i.e. $\{x_{k-1}^{(j)}\}$ and $\{x_k^{(j)}\}$).

The computational complexity of the MH-based algorithm is $\mathcal{O}(M \times N_S \times K)$, where M is the (mean) number of MH steps used for each time step and smoothing particle. The advantage of the new procedure is that M can often be much smaller than N_F and yet achieve almost equal error performance.

Run a particle filter to approximate $p(x_{1:k}|y_{1:k})$ for each k and store the resulting particles $\{x_{1:k}^{(i)}, w_k^{(i)}\}$.
for $i = 1$ **to** N_S **do**
  Sample $\tilde{x}_{1:K}^{(i)} \sim \sum_j w_K^{(j)} \delta_{x_{1:K}^{(j)}}(x_{1:K})$.
  for $k = K-1$ **to** 1 **do**
    $x_{1:k}^{(i)(0)} \leftarrow \tilde{x}_{1:k}^{(i)}$.
    for $m = 1$ **to** M **do**
      Propose a new state, $x_{1:k}^{(i)*} \sim \sum_j \tilde{v}_k^{(j)} \delta_{x_{1:k}^{(j)}}(x_{1:k})$.
      With probability α specified by (15),
      $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)*}$. Otherwise, $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)(m-1)}$.
    end for
    $\tilde{x}_{1:k}^{(i)} \leftarrow x_{1:k}^{(i)(M)}$
  end for
end for

Fig. 4. Metropolis-Hastings forward-filtering-backward-sampling algorithm

With the D-FFBS, the only parameter to be chosen is N_S, the number of smoother particles. With the MH variant, we also have the freedom to set M, the number of MH steps in each Markov chain. This controls a trade-off between speed and accuracy/particle diversity. If the chains are short, and the acceptance probabilities low, then this scheme will simply reproduce particles from the filter-smoother. However, when the weights of the conditional distribution approximation (13) are similar, then an independent sample may be produced with only a very short chain.

The new scheme targets the same empirical approximation to the smoothing distribution as used in the D-FFBS method. However, rather than drawing independent samples from this approximation, MH is used, resulting in dependent samples. In the limit as the number of MH steps, M, tends to infinity, the Markov chain will produce an independent sample from the target distribution, as the initial state of the chain will have been forgotten. At this point, the performance of the MH method is expected to match that of the direct method in terms of error and particle diversity. At the other extreme, with $M = 0$, the algorithm simply outputs particles from the filter-smoother approximation, which is expected to give worse performance. With intermediate values of M, the quality of the approximation is expected to fall

between those of the filter-smoother and direct backward-resampling smoother.

This new scheme bears a resemblance to the "linear cost" simulation scheme of [11] which uses rejection sampling instead of MH. Although theoretically elegant, the latter algorithm has been found to suffer from such high rejection rates as to render it consistently slower than the direct sampling implementation on problems with more than one state dimension (see section V).

IV. An MCMC particle smoother with improved support

The FFBS smoothers, both the direct sampling version of [7] and the MH version outlined in the previous section, are limited by the fact that states are only selected from those which appear in the filtering approximation. If there is a significant difference between the filtering and smoothing densities, then the particles are unlikely to be in the right locations to represent the smoothing distribution well, and the resulting approximation will be poor.

In [13], the authors described an algorithm for estimating the marginal smoothing distributions. For this method, states were proposed afresh rather than simply resampling from the filtering approximations. This yielded improvements in accuracy and particle diversity over other marginal smoothing techniques, and also the advantage of $\mathcal{O}(N_S \times K)$ complexity. Here we adapt this method for generating particles from the joint smoothing distribution. Again, an MCMC procedure is employed.

The smoother described in [13] uses importance sampling to approximate the marginal smoothing distributions. This method could be extended to estimation of the joint smoothing distribution by applying it sequentially. However, the resampling required to limit weight degeneracy would lead to a loss of path-space diversity similar to that suffered by the filter-smoother.

The proposed new method proceeds in a similar manner to the MH-FFBS algorithm. However, this time only $x_{1:k-1}$ is resampled from the particle filter-smoother approximation from time step $k-1$. The current state, x_k, is then sampled afresh from a new proposal distribution. Thus the factorised proposal may be written as follows,

$$\begin{aligned} q(x_{1:k}|\tilde{x}_{k+1}, y_{1:K}) &= q(x_{1:k-1}|y_{1:k-1})q(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k) \\ &= \sum_j \tilde{v}_{k-1}^{(j)} \delta_{x_{1:k-1}^{(j)}}(x_{1:k-1}) q(x_k|x_{k-1}^{(j)}, \tilde{x}_{k+1}, y_k). \end{aligned} \tag{16}$$

The target distribution is also factorised,

$$\begin{aligned} p(x_{1:k}|\tilde{x}_{k+1}, y_{1:K}) &= p(x_{1:k}|\tilde{x}_{k+1}, y_{1:k}) \\ &\propto p(\tilde{x}_{k+1}|x_k)p(x_k|x_{k-1})p(y_k|x_x)p(x_{1:k-1}|y_{1:k-1}). \end{aligned} \tag{17}$$

Using the filter-smoother approximation for $p(x_{1:k-1}|y_{1:k-1})$, the MH acceptance probabilities may then be calculated,

$$\begin{aligned}\alpha &= \min\left[1, \frac{\hat{p}(x^*_{1:k}|\tilde{x}_{k+1}, y_{1:K})q(x^{(m)}_{1:k}|\tilde{x}_{k+1}, y_{1:K})}{\hat{p}(x^{(m)}_{1:k}|\tilde{x}_{k+1}, y_{1:K})q(x^*_{1:k}|\tilde{x}_{k+1}, y_{1:K})}\right] \\ &= \min\left[1, \frac{w^*_{k-1}p(\tilde{x}_{k+1}|x^*_k)p(x^*_k|x^*_{k-1})p(y_k|x^*_k)}{w^{(m)}_{k-1}p(\tilde{x}_{k+1}|x^{(m)}_k)p(x^{(m)}_k|x^{(m)}_{k-1})p(y_k|x^{(m)}_k)}\right. \\ &\quad \left.\times \frac{\tilde{v}^{(m)}_{k-1}q(x^{(m)}_k|x^{(m)}_{k-1}, \tilde{x}_{k+1}, y_k)}{\tilde{v}^*_{k-1}q(x^*_k|x^*_{k-1}, \tilde{x}_{k+1}, y_k)}\right]. \end{aligned} \tag{18}$$

We refer to this algorithm as the Metropolis-Hastings forward-filtering-backward-proposing (MH-FFBP) smoother, emphasising the fact that samples are proposed afresh rather than recycling those from the filtering approximations. The complete algorithm is summarised in Fig. 5. The computational complexity of this algorithm is $\mathcal{O}(M \times N_S \times K)$, the same as the MH-FFBS smoother, although it is likely take more time to run because of the additional sampling operation and probability calculations. In contrast with the MH-FFBS algorithm, this smoother targets the exact posterior distribution for the current state at time t_n, rather than a particle approximation. Consequently, the performance is expected to be improved relative to the FFBS methods in terms of both error and particle diversity.

We note that the MH-FFBP algorithm shares similar features with the procedure published recently in [16]. The latter iteratively applies MH steps to a particle approximation of the joint smoothing algorithm in order to improve particle diversity. In this sense, it is akin to a resample-move [17] scheme for the entire smoothing distribution. In contrast, the MH-FFBP algorithm integrates MH steps into a single backward smoothing pass. It allows changes to the entire state history of a particle by proposing $x_{1:k-1}$ from the filtering approximation.

A. Proposal Densities

When using MH, if samples can be proposed directly from the target distribution then the acceptance probability is always 1 (In combination with a block-sampling scheme, this becomes the Gibbs sampler of [18]). This is desirable as it maximises the acceptance rate. Clearly, for the basic MH-FFBS smoother of section III, such a "target density proposal" simply yields the D-FFBS smoother of [7], as described in section II-B. For the MH-FFBP smoother, a target density proposal can be made by sequentially sampling

Run a particle filter to approximate $p(x_{1:k}|y_{1:k})$ for each k.
for $i = 1$ **to** N_S **do**
  Sample $\tilde{x}_{1:K}^{(i)} \sim \sum_j w_K^{(j)} \delta_{x_{1:K}^{(j)}}(x_{1:K})$.
  for $k = K-1$ **to** 1 **do**
    $x_{1:k}^{(i)(0)} \leftarrow \tilde{x}_{1:k}^{(i)}$.
    for $m = 1$ **to** M **do**
      Propose a new state history,
      $x_{1:k-1}^{(i)*} \sim \sum_j \tilde{v}_{k-1}^{(j)} \delta_{x_{1:k-1}^{(j)}}(x_{1:k-1})$.
      Propose a new state, $x_k^{(i)*} \sim q(x_k|x_{k-1}^*, \tilde{x}_{k+1}, y_k)$.
      With probability α specified by (18),
      $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)*}$. Otherwise, $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)(m-1)}$.
    end for
    $\tilde{x}_{1:k}^{(i)} \leftarrow x_{1:k}^{(i)(M)}$
  end for
end for

Fig. 5. Metropolis-Hastings forward-filtering-backward-proposing algorithm

$x_{1:k-1}$ followed by x_k using the following factorisation,

$$\begin{aligned} p(x_{1:k}|\tilde{x}_{k+1}, y_{1:K}) &= p(x_{1:k}|\tilde{x}_{k+1}, y_{1:k}) \\ &\propto p(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k) p(x_{1:k-1}|y_{1:k}, \tilde{x}_{k+1}). \end{aligned} \tag{19}$$

A good choice for the x_k proposal is thus given by,

$$q(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k) = p(x_k|x_{k-1}, \tilde{x}_{k+1}, y_k). \tag{20}$$

The matching $x_{1:k-1}$ proposal can be approximated using the filter-smoother particles,

$$\begin{aligned} q(x_{1:k-1}|y_{1:k}, \tilde{x}_{k+1}) &= p(x_{1:k-1}|y_{1:k}, \tilde{x}_{k+1}) \\ &\propto p(y_k, \tilde{x}_{k+1}|x_{1:k-1}) p(x_{1:k-1}|y_{1:k-1}) \\ &\approx \sum_j \tilde{v}_{k-1}^{(j)} \delta_{x_{1:k-1}^{(j)}}(x_{1:k-1}) \end{aligned} \tag{21}$$

$$\tilde{v}_{k-1}^{(j)} = w_{k-1}^{(j)} p(y_k, \tilde{x}_{k+1} | x_{1:k-1}^{(j)}). \tag{22}$$

The x_k proposal density may be approximated with linear-Gaussian dynamics, e.g. using an extended [19] or unscented [20] Kalman filter. The weights for the $x_{1:k-1}$ proposal may be approximated using similar methods, but this will often be undesirable. If it is necessary to calculate proposal weights for all of the filter-smoother particles, then the computational advantages of using MH are lost. With this in mind, an appropriate choice for the proposal weights might be simply $\tilde{v}_{k-1}^{(j)} = w_{k-1}^{(j)}$, which avoids calculation of $p(y_k, \tilde{x}_{k+1} | x_{1:k-1}^{(j)})$ for all j.

V. Simulations

The performance of the new smoothers was tested on a simple 2D tracking model. The trajectory of a single target was simulated using linear-Gaussian dynamics and observed via a nonlinear radar-type measurement model (see [21] for details),

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}}_{\mathbf{x}_k} = \begin{bmatrix} I_2 & \Delta t I_2 \\ 0_2 & I_2 \end{bmatrix} \underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + \underbrace{\begin{bmatrix} w_{1,k} \\ w_{2,k} \\ w_{3,k} \\ w_{4,k} \end{bmatrix}}_{\mathbf{w}_k} \tag{23}$$

$$\underbrace{\begin{bmatrix} b_k \\ r_x \end{bmatrix}}_{\mathbf{y}_k} = \begin{bmatrix} \tan^{-1}(y_k / x_k) \\ \sqrt{x_k^2 + y_k^2} \end{bmatrix} + \underbrace{\begin{bmatrix} v_{1,k} \\ v_{2,k} \end{bmatrix}}_{\mathbf{v}_k}. \tag{24}$$

The time between successive time steps is Δt. I_2 and 0_2 denote the 2×2 identity and zero matrices respectively. The random variables $\mathbf{w}_k$ and $\mathbf{v}_k$ are independent and are zero-mean Gaussian distributed with covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ respectively,

$$\mathbf{Q} = \sigma_P^2 \begin{bmatrix} \frac{\Delta t^3}{3} I_2 & \frac{\Delta t^2}{2} I_2 \\ \frac{\Delta t^2}{2} I_2 & \Delta t I_2 \end{bmatrix} \tag{25}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_B^2 & 0 \\ 0 & \sigma_R^2 \end{bmatrix}. \tag{26}$$

The following algorithms were all implemented for comparison in MATLAB:

- The simple filter-smoother (FS) of [6]

- The direct forward-filtering-backward-sampling smoother (D-FFBS) of [7]
- The new Metropolis-Hastings forward-filtering-backward-sampling smoother (MH-FFBS) of section III, with several values of M
- The new Metropolis Hastings forward-filtering-backward-proposing smoother (MH-FFBP) of section IV, with several values of M

The rejection-sampling version of the FFBS described by [11] was also implemented, but high rejection rates meant that it was impractically slow so it was omitted from further testing.

The algorithms were all used to generate 100 particles from the joint smoothing distribution, using particle filters with 100 particles. Where appropriate, proposal weights were chosen to be equal to filtering weights, i.e. $v_k^{(j)} = w_k^{(j)}$. The particle filter uses the "optimal importance density" and the MH-FFBP a "target density proposal" with linearisations of the observation model where appropriate.

100 realisations of 500 time steps were simulated from the model and the algorithms tested on each one. Results are shown here for three noise parameter sets. For case 1, the observation variances were set to $\sigma_B^2 = (\pi/720)^2$ and $\sigma_R^2 = 0.1$. For case 2, $\sigma_B^2 = (\pi/36)^2$ and $\sigma_R^2 = 0.1$, resulting in a highly skewed likelihood function. For case 3, $\sigma_B = (\pi/36)^2$ and $\sigma_R^2 = 100$. Other model settings are $\Delta t = 1$, $\sigma_P = 1$, and $x_0 = [-100, 50, 10, 0]^T$. Example trajectories and observations from the two cases are shown in Fig. 6.

The performances of the various smoothing algorithms are compared using a number of measures. At each time point, a single state estimate is calculated by taking the mean of the values from each particle. These point estimates are then used to calculate a root-mean-square error (RMSE). In addition, an empirical normalised estimation error squared (ENEES) is calculated using,

$$\begin{aligned} \text{ENEES}_k &= (\hat{\mathbf{x}}_k - \mathbf{x}_k^*)^T \hat{P}_k^{-1} (\hat{\mathbf{x}}_k - \mathbf{x}_k^*) \\ \hat{\mathbf{x}}_k &= \frac{1}{N_S} \sum_i \mathbf{x}_k^{(i)} \\ \hat{P}_k &= \frac{1}{N_S} \sum_i (\mathbf{x}_k^{(i)} - \mathbf{x}_k^*)(\mathbf{x}_k^{(i)} - \mathbf{x}_k^*)^T \end{aligned} \tag{27}$$

and where $\mathbf{x}_k^*$ is the true value of the state. This statistic has a value of 1 when there are fewer linearly independent state values in the particle approximation than state dimensions. It decreases as the number of independent particles increases (improving the covariance estimate) or the estimation error decreases. Thus, it provides a simple relative measure of the quality of different particle approximations, taking into account both estimation error and particle diversity. Finally, the mean number of unique particles at each time instant and the running times (excluding the time for filtering) are also recorded. These performance

TABLE I

SMOOTHER PERFORMANCE MEASURES FOR NOISE COVARIANCE CASE 1. ALL RESULTS ARE AVERAGED OVER 100 RUNS, WITH 500 TIME STEPS IN EACH RUN.

Algorithm	Position RMSE	Velocity RMSE	ENEES	Running Time (s)	Unique Particles
FS	0.56	0.96	0.99	0.00	2.13
D-FFBS	0.45	0.76	0.84	65.88	20.54
MH-FFBS (1)	0.48	0.81	0.90	1.80	13.95
MH-FFBS (3)	0.46	0.78	0.87	3.89	17.63
MH-FFBS (10)	0.46	0.77	0.85	11.20	19.83
MH-FFBS (30)	0.45	0.76	0.84	32.05	20.40
MH-FFBS (100)	0.45	0.76	0.84	105.12	20.55
MH-FFBP (1)	0.45	0.75	0.82	9.20	44.32
MH-FFBP (3)	0.43	0.73	0.79	19.23	70.12
MH-FFBP (10)	0.43	0.72	0.78	54.33	90.10
MH-FFBP (30)	0.42	0.71	0.79	154.47	96.89
MH-FFBP (100)	0.42	0.70	0.79	504.58	98.53

indicators are all shown in tables I–III for the two noise scenarios. The trade-off between speed and error is shown in Fig. 7.

The MH-FFBS smoother error approaches that of the D-FFBS as the number of MH steps, M, increases. Similarly, the mean number of unique particles in the MH-FFBS approximation approaches that of the D-FFBS. This behaviour is expected as a longer Markov chain is more likely to deliver an independent sample from the target distribution, whereas independent samples are guaranteed in the D-FFBS approximation due to the direct sampling procedure.

When the number of MH steps, M, is small, the error of the MH-FFBS is often still significantly reduced relative to the D-FFBS but with a shorter running time. For a particular application, it is possible to adjust M to control the trade-off between error and speed of computation. Thus, the MH-FFBS provides increased flexibility when compared to the D-FFBS scheme.

The MH-FFBP outperforms all of the other smoothers, achieving the lowest RMSE and ENEES for any given running time. Furthermore, because the support is not limited to using states from the filter distributions, the smoothing particle approximation has many more unique particles, giving a less

TABLE II

SMOOTHER PERFORMANCE MEASURES FOR NOISE COVARIANCE CASE 2. ALL RESULTS ARE AVERAGED OVER 100 RUNS, WITH 500 TIME STEPS IN EACH RUN.

Algorithm	Position RMSE	Velocity RMSE	ENEES	Running Time (s)	Unique Parti-cles
FS	8.01	1.90	0.99	0.00	2.75
D-FFBS	7.71	1.70	0.92	66.40	14.06
MH-FFBS (1)	7.91	1.83	0.98	1.79	6.51
MH-FFBS (3)	7.83	1.78	0.96	3.86	8.63
MH-FFBS (10)	7.77	1.73	0.93	11.09	11.39
MH-FFBS (30)	7.75	1.71	0.92	31.77	13.14
MH-FFBS (100)	7.70	1.70	0.92	104.14	13.88
MH-FFBP (1)	7.75	1.72	0.92	9.25	22.97
MH-FFBP (3)	7.67	1.67	0.88	19.34	42.22
MH-FFBP (10)	7.62	1.63	0.85	54.61	70.06
MH-FFBP (30)	7.60	1.61	0.84	155.38	88.81
MH-FFBP (100)	7.59	1.60	0.84	507.58	96.82

degenerate representation.

VI. CONCLUSION

Two new algorithms have been presented for drawing particles from the joint smoothing distribution using Metropolis-Hastings. These MCMC schemes have a degree of flexibility in the choice of chain length, which controls a trade-off between accuracy and running time.

The first algorithm samples particles from the filtering approximations to produce particles from the smoothing distribution. The error performance of this scheme approaches that of the standard forward-filtering-backward-sampling of [7] as the chain length increases. However, comparable error performance may be achieved with very short chains, leading to a significant saving in processing time.

The second algorithm samples new values of the state from a continuous proposal density. Although slower than the previous scheme for a given length of Markov chain, the error performance is reduced relative to the other smoothers.

TABLE III

SMOOTHER PERFORMANCE MEASURES FOR NOISE COVARIANCE CASE 3. ALL RESULTS ARE AVERAGED OVER 100 RUNS, WITH 500 TIME STEPS IN EACH RUN.

Algorithm	Position RMSE	Velocity RMSE	ENEES	Running Time (s)	Unique Particles
FS	7.41	2.15	0.98	0.00	3.97
D-FFBS	7.19	2.03	0.97	66.27	7.48
MH-FFBS (1)	7.39	2.14	0.98	1.75	4.79
MH-FFBS (3)	7.36	2.12	0.98	3.75	5.33
MH-FFBS (10)	7.30	2.08	0.98	10.78	6.16
MH-FFBS (30)	7.24	2.05	0.97	30.83	6.88
MH-FFBS (100)	7.20	2.04	0.97	101.02	7.38
MH-FFBP (1)	7.22	2.05	0.97	9.12	12.57
MH-FFBP (3)	7.08	1.98	0.94	19.01	22.78
MH-FFBP (10)	6.95	1.92	0.92	53.64	44.15
MH-FFBP (30)	6.87	1.88	0.90	152.42	70.71
MH-FFBP (100)	6.84	1.86	0.88	498.38	91.35

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal Of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[2] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, pp. 107–113, 1993.

[3] O. Cappé, S. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, pp. 899–924, 2007.

[4] A. Doucet and A. M. 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, 2009.

[5] H. Rauch, F. Tung, and C. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA journal*, vol. 3, pp. 1445–1450, 1965.

[6] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, pp. 1–25, 1996.

[7] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, pp. 156–168, 2004.

[8] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for statespace models," *Annals of the Institute of Statistical Mathematics*, vol. 62, pp. 61–89, 2010.

[9] A. Doucet, S. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, pp. 197–208, 2000.

[10] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, pp. 590–599, 1999.

[11] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "On the forward filtering backward smoothing particle approximations of the smoothing distribution in general state spaces models," 2009, arxiv preprint arXiv:0904.0316v1.

[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 *Proc. 23rd Int. Conf. Machine learning*, 2006, pp. 481–488.

[13] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, pp. 447–464, 2010.

[14] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. Chapman and Hall, 1996.

[15] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, pp. 97–109, 1970.

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

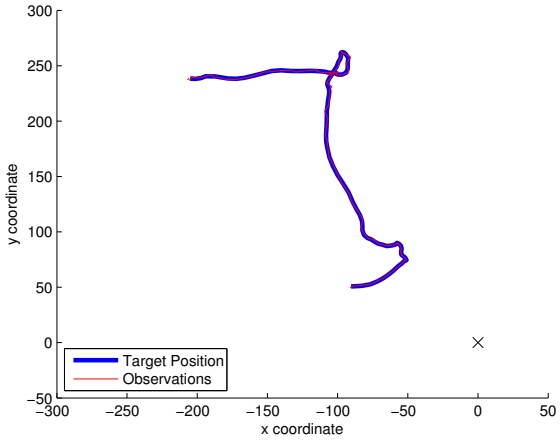
[17] W. R. Gilks and C. Berzuini, "Following a moving target – Monte Carlo inference for dynamic Bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, pp. 127–146, 2001.

[18] S. Geman and D. Geman, "Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

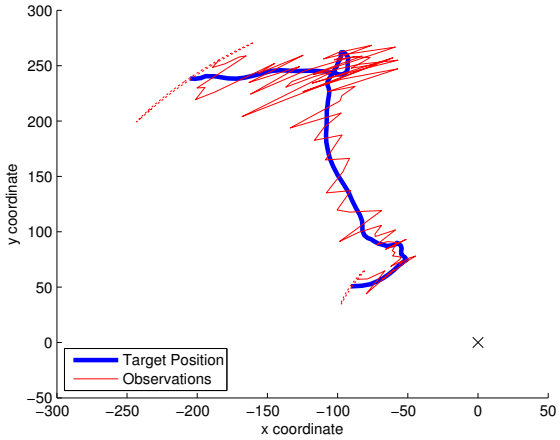
[19] M. S. Grewal and A. P. Andrews, *Kalman filtering: Theory and practice using MATLAB*. John Wiley & Sons, Inc., 2002.

[20] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, pp. 401–422, 2004.

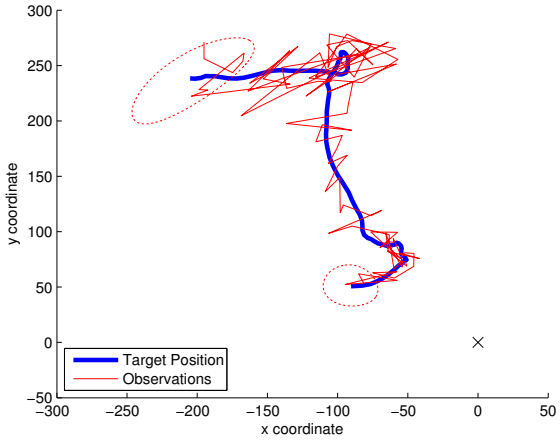
[21] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. Wiley Online Library, 2002.



(a) Case 1

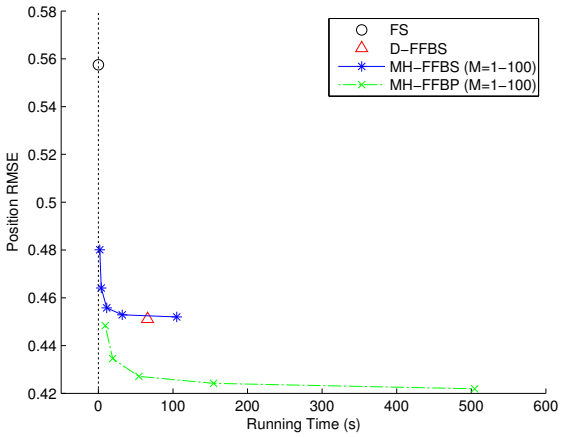


(b) Case 2

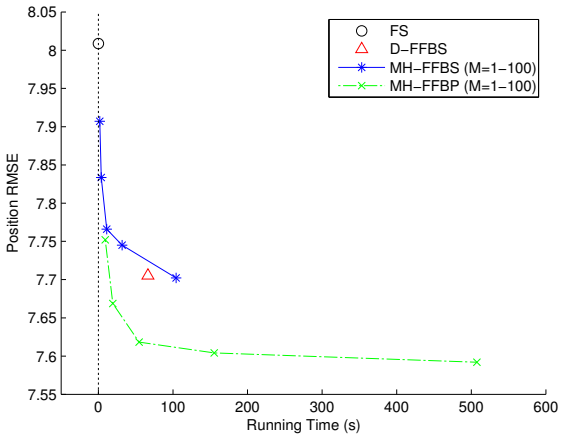


(c) Case 3

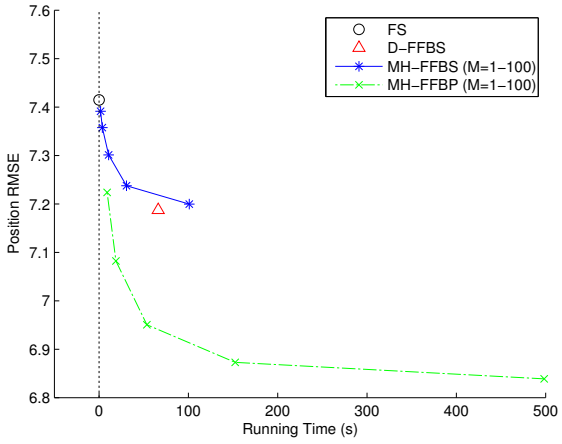
Fig. 6. Example trajectory with observations from the two noise covariance cases. Constant likelihood contours are shown for the first and last position (dashed)



(a) Case 1



(b) Case 2



(c) Case 3

Fig. 7. Position RMSEs and running times of various smoothing algorithms with different noise covariance cases.