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

Pete Bunch, 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 Markov chain Monte Carlo sampling procedure is used to improve the efficiency of the particle smoother. In addition, an algorithm for approximating the joint smoothing distribution without limited support is presented.**

***Index Terms*—state space model, particle filter, smoothing, MCMC, Bayesian inference**

I. Introduction

THE objective of sequential Bayesian inference is the estimation of an unknown, time-varying quanitity from incomplete or inaccurate observations. This is commonly achieved through the use of probabilistic models for the evolution of the latent state and the measurement process.

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

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

Here, the random variable x_k denotes the value of the latent state at the k^{th} instant, and y_k the value of the observation. A set of random variables will be written as $x_{1:k} = \{x_1, x_2, \dots, x_k\}$. 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 consists of 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 equations 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. In such cases, 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 $x = a$. $\hat{p}$ means an approximation to p.

A similar story applies 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 a Rauch-Tung-Striebel (RTS) smoother [5]. This works by modifying the ordinary Kalman filter results in a backward processing pass through the observations. For nonlinear or non-Gaussian models, it is possible again to resort to numerical methods. Now, the particles of the approximation are drawn from 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 methods are presented for drawing samples from the joint smoothing distribution, using Markov chain Monte Carlo (MCMC). 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 MCMC-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 k^{th} processing step, a particle estimate of the joint distribution over x_{k-1} and x_k is made by drawing samples from a proposal distribution.

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

Manuscript received January 01, 1901; revised January 02, 1901.

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

The $k-1$ state proposal distribution, $q(x_{k-1})$, is constructed from the particles of the filtering distribution from the previous time step. The choice of weights and sampling procedure here determines what from of "resampling" is to be used. Here we use arbitrary weights for generality.

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

Note that 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 [9]. For the least computational expense, the proposal distribution 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. (See [3], [4] for further discussion of resampling.)

Next, the particles are assigned an importance weight according to the ratio of the targeted joint distribution and the proposal.

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

Normalisation is enforced by scaling the weights so that they sum to 0. Finally, x_{k-1} is marginalised from the distribution by simply discarding the values from each particles.

The particle filter is summarised below.

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

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 trivial extension of the particle filter. Rather than marginalising the previous states from each particle, these past values are stored as well. 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.

The weakness of this simple, filter-smoother (FS) is particle degeneracy. Every time the particles are resampled, those with low weights do not appear in the approximation at the next step, while those with high weights appear multiple times. The result is that many, or even all, particles will have the same values at early time steps. This effect is illustrated in figure **??** in section **??**. If the objective is only to produce a single point estimate then this might be sufficient, but to characterise the complete smoothing distribution the approximation must be rejeuvenated.

The filter-smoother is summarised below.

Initialise particles from prior, $x_0^{(i)} \sim p(x_0)$.
for $k = 1 \dots K$ **do**
  for $i = 1 \dots N_F$ **do**
    Sample $\{x_{1:k-1}^{(i)}, x_k^{(i)}\} \sim \sum_j v_{k-1}^{(j)} q(x_k|x_{k-1}^{(j)})$.
    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

An improved particle smoothing algorithm was presented in [7]. This works by constructing new particles from the smoothing distribution by resampling 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 k^{th} 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 distribution.

The backwards conditional distributions 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. This backward-resampling smoother is summarised below.

Run a particle filter to approximate $p(x_k|y_{1:k})$ for each k.

for $i = 1 \dots N_S$ **do**
  Sample $\tilde{x}_K^{(i)} \sim \sum_j w_K^{(j)} \delta_{x_K^{(j)}}(x_K)$.
  for $k = K-1 \dots 1$ **do**
    for $j = 1 \dots N_F$ **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

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.

The complexity of both the backward-resampling 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. Methods for approximation of such algorithms with lower complexity are discussed in [10]. However, computational cost is often prohibitive, requiring either N_S or N_F to be held very low.

In the algorithms described so far, no new states are sampled during the smoothing stage; there is only a resampling of the filtering distributions. This limits the support of the smoother to the states that appear in the filtering particles. In [11], an algorithm for estimation of the marginal smoothing distributions is described in which new states are sampled from a proposal, allowing full support. In addition, this algorithm achieves $\mathcal{O}(N_S \times K)$ complexity, linear in the number of particles!

III. New MCMC Particle Smoother

The bottleneck of the backward-resampling smoother is the calculation of the sampling weights (equation 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 Markov chain Monte Carlo (MCMC) to produce these samples 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 [12]. The basic Metropolis-Hastings (MH) [13] 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 probabilities.

For the direct backward sampling smoother, 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 MCMC 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 equation 8. At the k^{th} 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 first k states 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 equation 10, the target distribution may be approximated by a particle distribution.

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

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

A valid proposal distribution 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:

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

Note that w_k^* indicates the 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 complete MCMC backward-resampling smoother is summarised below.

Run a particle filter to approximate $p(x_{1:k}|y_{1:k})$ for each k.
for $i = 1 \dots N_S$ **do**
  Sample $\tilde{x}_{1:K}^{(i)} \sim \sum_j w_K^{(j)} \delta_{x_{1:K}^{(j)}}(x_{1:K})$.
  for $k = K-1 \dots 1$ **do**
    $x_{1:k}^{(i)(0)} \leftarrow \tilde{x}_{1:k}^{(i)}$.
    for $m = 1 \dots M$ **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 $\alpha = \frac{w_k^* \tilde{v}_k^{(m)}}{w_k^{(m)} \tilde{v}_k^*} \times \frac{p(\tilde{x}_{k+1}|x_k^*)}{p(\tilde{x}_{k+1}|x_k^{(m)})}$, $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)*}$. Otherwise, $x_{1:k}^{(i)(m)} \leftarrow x_{1:k}^{(i)(m-1)}$.
    end for
    $\tilde{x}_{1:k}^{(i)} \leftarrow x_k^{(i)(M)}$
  end for
end for

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 as K increased. 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 MCMC-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.

With the direct backward-resampling smoother, the only parameter to be chosen is N_S, the number of smoother particles. With the MCMC 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 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 (equation 13) are similar, then an independent sample may be produced with only a very short chain.

IV. Improving the smoother support

In [11], 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 the method of [11] for generating particles from the joint smoothing distribution. Again, an MCMC procedure is employed.

The method described in [11] uses a forward and a backward particle filter. To develop a marginal smoothing approximation at time step k, particles are resampled from the forward filter at step $k-1$, and from the backward filter and step $k+1$. A new state for step k is then sampled from a continuous proposal distribution conditional on the previous and next states, and an importance weight is assigned. It is undesirable to use importance sampling for generating particles from the joint sampling distribution, as the required resampling is liable to lead to a loss of particle diversity similar to that suffered by the filter-smoother.

The proposed new method proceeds in a similar manner to the MCMC backward-resampling algorithm. However, this time $x_{1:k-1}$, 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:

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

The target distribution is also factorised.

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

Using the filter-smoother approximation, the MH acceptance probabilities may then be calculated.

$$\begin{aligned} \alpha &= \frac{\hat{p}(x_{1:k}^*|\tilde{x}_{k+1}, y_{1:K})q(x_{1:k}^{(m)}|\tilde{x}_{k+1}, y_{1:K})}{\hat{p}(x_{1:k}^{(m)}|\tilde{x}_{k+1}, y_{1:K})q(x_{1:k}^*|\tilde{x}_{k+1}, y_{1:K})} \\ &= \frac{w_{k-1}^* \tilde{v}_{k-1}^{(m)}}{w_{k-1}^{(m)} \tilde{v}_{k-1}^*} \times \\ &\frac{p(\tilde{x}_{k+1}|x_k^*)p(x_k^*|x_{k-1}^*)p(y_k|x_k^*)q(x_k^{(m)}|x_{k-1}^{(m)}, \tilde{x}_{k+1}, y_k)}{p(\tilde{x}_{k+1}|x_k^{(m)})p(x_k^{(m)}|x_{k-1}^{(m)})p(y_k|x_k^{(m)})q(x_k^*|x_{k-1}^*, \tilde{x}_{k+1}, y_k)} \end{aligned} \quad (18)$$

The complete MCMC backward-sampling smoother is summarised below.

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

The computational complexity of this algorithm is $\mathcal{O}(M \times N_S \times K)$, the same as the MCMC resampling smoother. Because the support of the smoother particles is not restricted to the states which appear in the filter approximation, the error performance is expected to be improved.

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.

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} I_2 & \Delta t I_2 \\ 0_2 & I_2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + \underbrace{\begin{bmatrix} w_{1,k} \\ w_{2,k} \\ w_{3,k} \\ w_{4,k} \end{bmatrix}}_{\mathbf{w}_k} \tag{19}$$

$$\underbrace{\begin{bmatrix} b_k \\ r_x \end{bmatrix}}_{\mathbf{y}_k} = \begin{bmatrix} \tan^{-1}(y_k/x_k) \\ \sqrt{x_k^2 + y_k^2} \end{bmatrix} + \underbrace{\begin{bmatrix} v_{1,k} \\ v_{2,k} \end{bmatrix}}_{\mathbf{v}_k} \tag{20}$$

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

$$\mathbf{Q} = \sigma_P^2 \begin{bmatrix} \frac{\Delta t^3}{3} I_2 & \frac{\Delta t^2}{2} I_2 \\ \frac{\Delta t^2}{2} I_2 & \Delta t I_2 \end{bmatrix} \tag{21}$$

$$\mathbf{R} = \begin{bmatrix} \sigma_B^2 & 0 \\ 0 & \sigma_R^2 \end{bmatrix} \tag{22}$$

The following algorithms were all implemented for comparison in MATLAB:

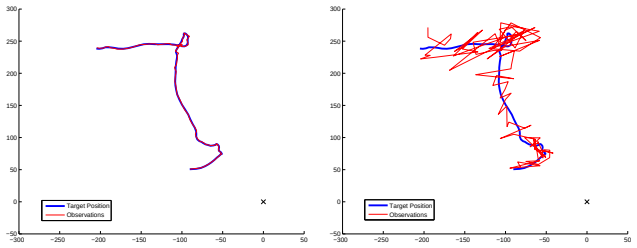
- The simple filter-smoother (FS) of [6]
- The direct backward-resampling smoother (DBRS) of [7]
- The new MCMC backward resampling smoother (MCMC-BRS) of section III, with several values of M
- The new MCMC backward sampling smoother (MCMC-BSS) of section IV, with several values of M

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 and MCMC-BSS both use "optimal importance distributions" (see e.g. [3]) with linearisations of the observation model where appropriate.

10 realisations of 500 time steps were simulated from the model and the algorithms tested on each one. Results are shown here for high and low observation noise cases. For case 1, the observation variances were set to $\sigma_B = \pi/720$ and $\sigma_R = 0.1$. For case 2, $\sigma_B = \pi/36$ and $\sigma_R = 100$. Other model settings are $\Delta t = 1$, $\sigma_P = 1$, and $x_0 = [-100, 50, 10, 0]^T$. Example tracjectories and observations from the two cases are shown in figure 1.

Tables I and I show the root-mean-square errors (RMSEs) in position and velocity obtained by the different algorithms with the two parameter cases, along with running times (excluding the time for filtering) and the mean number of unique particles at each time instant. The trade-off between speed and error is shown in figures 2 and 3.

The MCMC-BRS smoother error approaches that of the DBRS as the number of MCMC steps, M, increases. However, for small M, the error is only very slightly larger than that of the DBRS for very low running times. For a particular application, it is possible to adjust M to control the trade-off between error and speed.



(a) Case 1 (b) Case 2

Fig. 1. Example trajectory with observations from the two parameter cases.

TABLE I
SMOOTHER PERFORMANCE MEASURES FOR PARAMETER CASE 1. ALL RESULTS ARE AVERAGED OVER 10 RUNS, WITH 500 TIME STEPS IN EACH RUN.

Algorithm	Postion RMSE	Velocity RMSE	Running Time (s)	Unique Particles
FS	0.578	0.968	0	2.25
DBRS	0.475	0.762	122	20.3
MCMC-BRS (1)	0.509	0.823	3	13.7
MCMC-BRS (3)	0.493	0.790	7	17.4
MCMC-BRS (10)	0.482	0.768	21	19.6
MCMC-BRS (30)	0.480	0.761	62	20.2
MCMC-BRS (100)	0.476	0.764	202	20.4
MCMC-BSS (1)	0.473	0.758	14	44.3
MCMC-BSS (3)	0.461	0.734	29	69.7
MCMC-BSS (10)	0.451	0.720	82	89.8
MCMC-BSS (30)	0.444	0.713	232	96.7
MCMC-BSS (100)	0.443	0.709	755	98.6

TABLE II
SMOOTHER PERFORMANCE MEASURES FOR PARAMETER CASE 2. ALL RESULTS ARE AVERAGED OVER 10 RUNS, WITH 500 TIME STEPS IN EACH RUN.

Algorithm	Postion RMSE	Velocity RMSE	Running Time (s)	Unique Particles
FS	6.17	2.03	0	3.96
DBRS	5.79	1.89	63	7.34
MCMC-BRS (1)	6.13	2.01	2	4.77
MCMC-BRS (3)	6.07	1.99	4	5.29
MCMC-BRS (10)	5.94	1.94	10	6.1
MCMC-BRS (30)	5.82	1.91	29	6.73
MCMC-BRS (100)	5.80	1.89	95	7.38
MCMC-BSS (1)	5.90	1.91	9	12.6
MCMC-BSS (3)	5.67	1.81	18	22.7
MCMC-BSS (10)	5.48	1.73	50	44.1
MCMC-BSS (30)	5.43	1.69	144	70.9
MCMC-BSS (100)	5.34	1.68	471	91.8

The MCMC-BSS outperforms all the other smoothers, achieveing a lower RMSE even with $M = 1$. As M is increased further, the accuracy increases.

VI. CONCLUSION

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

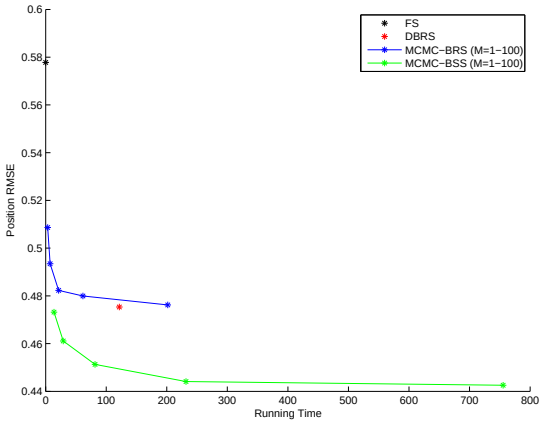


Fig. 2. RMSEs and running times of various smoothing algorithms

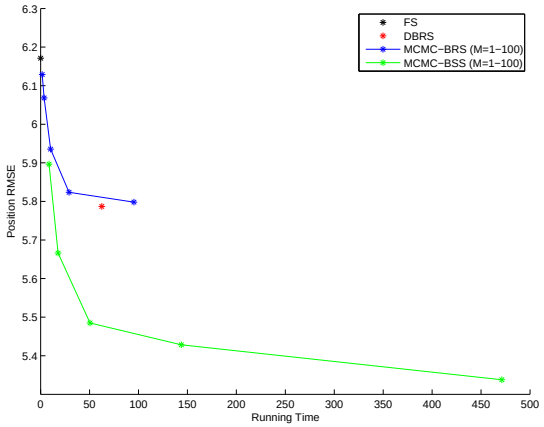


Fig. 3. RMSEs and running times of various smoothing algorithms

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

The second algorithm samples new values of the state from a proposal distribution. Although slower than the resampling version for a fixed length of Markov chain, the error performance is reduced relative to the other smoothers.

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal Of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.6247&rep=rep1&type=pdf

[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, no. 2, pp. 107 –113, apr 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, no. 5, 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. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.772&rep=rep1&type=pdf

[5] H. Rauch, F. Tung, and C. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA journal*, vol. 3, no. 8, 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, no. 1, pp. pp. 1–25, 1996. [Online]. Available: http://www.jstor.org/stable/1390750

[7] 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, 2004. [Online]. Available: http://pubs.amstat.org/doi/abs/10.1198/016214504000000151

[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, 10.1007/s10463-009-0236-2. [Online]. Available: http://dx.doi.org/10.1007/s10463-009-0236-2

[9] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. pp. 590–599, 1999. [Online]. Available: http://www.jstor.org/stable/2670179

[10] 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*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 481–488. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143905

[11] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, no. 2, pp. 447–464, 2010. [Online]. Available: http://biomet.oxfordjournals.org/content/97/2/447.abstract

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

[13] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, p. 1087, 1953.

Pete Bunch Biography text here.

Simon Godsill Biography text here.