Faster particle smoothing using Markov chain Monte Carlo sampling

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

***Abstract*—The abstract goes here.**

***Index Terms*—**

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.

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.

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 smoothing distribution, using Markov chain Monte Carlo (MCMC). These novel procedures have computational advantages and greater flexibility over previous methods.

In section II, we review 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.

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

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 in algorithm 1.

Algorithm 1 Particle filter algorithm

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.

Algorithm 2 Filter-smoother algorithm

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

The filter-smoother is summarised in algorithm 2.

An improved particle smoothing algorithm was presented in [7]. This works by constructing new particles from the smoothing distribution by sampling from the filtering distributions. This has the limitation that the particle support for the smoother is limited to that of the filter. 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 approximation.

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 smoother distribution as required. This backward-resampling smoother is summarised in algorithm 3.

A third 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 in the manner of algorithm 3.

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

Algorithm 3 Direct backward-resampling smoother algorithm

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

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!

In section III, we introduce an MCMC approach for drawing samples from the joint smoothing distribution with an improved computational complexity over the direct backward-resampling smoother. In section IV, we combine this procedure with the new-proposal method used in [11] to generate joint smoothing distribution particles with full support.

III. New MCMC Particle Smoother

The bottleneck of the backward-resampling smoother (algorithm 3) 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 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 densities.

For the new MCMC smoother, the target distribution will be the conditional $\hat{p}(x_k|\tilde{x}_{k+1}, y_{1:K})$. If the current state of the chain is $x_k^{(m)}$, and a new sample is drawn from the proposal $x_k^* \sim q(x_k|\tilde{x}_{k+1}, y_{1:K})$, then it should be accepted with the following probability.

$$\alpha = \frac{\hat{p}(x_k^*|\tilde{x}_{k+1}, y_{1:K}) q(x_k^{(m)}|\tilde{x}_{k+1}, y_{1:K})}{\hat{p}(x_k^{(m)}|\tilde{x}_{k+1}, y_{1:K}) q(x_k^*|\tilde{x}_{k+1}, y_{1:K})} \tag{12}$$

A valid proposal distribution may be constructed using the particles of the filter approximation with any arbitrary weights.

$$q(x_k|\tilde{x}_{k+1}, y_{1:K}) = \sum_j \tilde{v}_k \delta_{x_k^{(j)}}(x_k) \tag{13}$$

The resulting acceptance probability simplifies to the following:

$$\alpha = \frac{w_k^* \tilde{v}_k^{(m)}}{w_k^{(m)} \tilde{v}_k^*} \times \frac{p(\tilde{x}_{k+1}|x_k^*)}{p(\tilde{x}_{k+1}|x_k^{(m)})} \tag{14}$$

Note that w_k^* indicates the particle weight of the proposed new state, x_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 in algorithm 4. Here we target the entire history conditional distribution, $\hat{p}(x_{1:k}|\tilde{x}_{k+1}, y_{1:K})$, which simpifies the initialisation of chains. This makes no practical difference to the algorithm.

Algorithm 4 MCMC backward-resampling smoother algorithm

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 history, $x_{1:k}^{(i)*} \sim \sum_j \tilde{v}_k^{(j)} \delta_{x_{1:k}^{(j)}}(x_{1:k})$.
      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
    $\tilde{x}_{1:k}^{(i)} \leftarrow x_{1:k}^{(i)(M)}$
  end for
end for

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 at each time step and smoothing particle. The advantage of the new procedure is that M can often be much smaller N_F and yet achieve the same error performance.

- Motivate - high complexity, low speed of other smoothers

- MCMC avoids need to calculate all weights at each step
- maths
- algorithm
- complexity
- See direct simulation smoother as a single step of Gibbs Sampler

A. *Performance Control*

- The effect of varying M, the number of MH steps
- Situations where the new smoother works well/poorly.
- Backward sampling weight ESS.

IV. Improving the Smoother Support

- Outline Fearnhead smoother
- Discuss extension to joint estimation (possibility of IS)
- MCMC method

V. Simulations

VI. Conclusion

The conclusion goes here.

Appendix A
Appendix Title

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The authors would like to thank...

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.

PLACE PHOTO HERE

Pete Bunch Biography text here.

Simon Godsill Biography text here.