Ann Inst Stat Math (2014) 66:443–471
DOI 10.1007/s10463-014-0446-0

SPECIAL ISSUE: BAYESIAN INFERENCE AND STOCHASTIC COMPUTATION

Computational aspects of sequential Monte Carlo filter and smoother

Genshiro Kitagawa

Received: 10 May 2013 / Revised: 6 January 2014 / Published online: 4 March 2014

Abstract Progress in information technologies has enabled to apply computer-intensive methods to statistical analysis. In time series modeling, sequential Monte Carlo method was developed for general nonlinear non-Gaussian state-space models and it enables to consider very complex nonlinear non-Gaussian models for real-world problems. In this paper, we consider several computational problems associated with sequential Monte Carlo filter and smoother, such as the use of a huge number of particles, two-filter formula for smoothing, and parallel computation. The posterior mean smoother and the Gaussian-sum smoother are also considered.

Keywords Nonlinear non-Gaussian state-space model · Particle filter · Gaussian-sum filter · Two-filter formula · Parallel computation · Posterior mean smoother

1 Introduction

In time series analysis, the prior knowledge of the dynamics of the phenomena and the mechanism of the observation process can usually be combined into state-space model form. As a result, many important problems in time series analysis can be solved using the linear-Gaussian state-space model, and various nonstationary time series models have been developed using state-space models (Harrison and Stevens 1976; West and Harrison 1989; Kitagawa and Gersch 1996, Doucet et al. 2001, Prado and West 2010). However, there are numerous situations in which the ordinary time

G. Kitagawa (✉)
Transdisciplinary Research Integration Center, Research Organization of Information and Systems, Hulic Kamiyacho Building 2F, 4-3-13 Toranomon, Minato-ku, Tokyo 105-0001, Japan
e-mail: kitagawa@rois.ac.jp

series model is too restrictive and a more general nonlinear model or non-Gaussian model is required (Kitagawa 1987, 2010; Kitagawa and Gersch 1984). In the 1990s, various sequential Monte Carlo methods, referred to as bootstrap filters, Monte Carlo filters, and particle filters, were developed (Gordon et al. 1993; Kitagawa 1996; Higuchi 1997; Pitt and Shephard 1999; Doucet et al. 2000, 2001; Doucet and Johansen 2011). In these methods, arbitrary distributions of the state and the system noise are expressed by numerous particles. Then, it is possible to develop a recursive filter and smoother for very complex nonlinear non-Gaussian state-space models. These methods have been successfully applied to a number of complex real-world problems (Doucet et al. 2001; Nakano et al. 2007).

Recent progress of information and communication technologies brought us both the difficult problems and various possibilities in statistical modeling. To properly handle and fully utilize the information contained in the "big data", it is necessary to develop high-dimensional very complex nonlinear models. On the other hand, emergence of massive parallel processors enables to apply computer-intensive methods to mitigate the difficulties in complex nonlinear modeling. As an example, the difficulty with sequential Monte Carlo filtering and smoothing comes from the degeneracy of the posterior distribution due to resampling of the particles, and several particle filter algorithms have been developed to address this problem (Briers et al. 2010; Doucet et al. 2001; Fearnhead et al. 2010). However, progress in information technologies has enabled the use of highly parallel processors, and even the use of a billion of particles or more for approximation of complex distributions is becoming realistic. It is interesting to see to what extent the use of a huge number of particles alleviate the difficulties in smoothing (Klaas et al. 2006).

In this paper, various computational problems are considered via simulation study using a simple one-dimensional trend model. This is because, we need a "true" posterior distribution to evaluate the accuracy of the filtering and smoothing algorithms. In Sect. 2, a brief review of state-space model and recursive filtering methods is presented. Computational efficiency and accuracy of the filter and the smoother when we use a huge number of particles are considered in Sect. 3. The improvement of the smoothing distribution by two-filter formula is presented in Sect. 4, and three algorithms for parallel computation are considered in Sect. 5. The posterior mean smoother and the Gaussian-sum filter and smoother are considered in Sects. 6 and 7, respectively. Finally, some concluding remarks are give in Sect. 8.

2 A brief review of the filtering and smoothing algorithms

2.1 The state-space model and the state estimation problems

Assume that a time series (y_n) is expressed by a linear state-space model

$$\begin{aligned} x_n &= F_n x_{n-1} + G_n v_n \\ y_n &= H_n x_n + w_n, \end{aligned} \tag{1}$$

where x_n is an k-dimensional state vector, v_n and w_n are ℓ-dimensional and 1-dimensional white noise sequences having density functions $q_n(v)$ and $r_n(w)$, respectively. The initial state vector x_0 is assumed to be distributed according to the density $p(x_0)$.

The information from the observations up to time j is denoted by Y_j, namely, $Y_j \equiv \{y_1, \ldots, y_j\}$. The problem of state estimation is to evaluate $p(x_n|Y_j)$, the conditional density of x_n given the observations Y_j and the initial density $p(x_0|Y_0) \equiv p(x_0)$. For $n > j$, $n = j$ and $n < j$, it is called the problem of prediction, filtering and smoothing, respectively.

This linear state-space model can be generalized to a nonlinear non-Gaussian state-space model,

$$\begin{aligned} x_n &= F_n(x_{n-1}, v_n) \\ y_n &= H_n(x_n) + w_n, \end{aligned} \tag{2}$$

where $F_n(x, v)$ and $H_n(x)$ are possibly nonlinear functions of the state and the noise inputs. Diverse problems in time series analysis can be treated using this nonlinear state-space model (Kitagawa and Gersch 1996; Doucet et al. 2001). Note that this nonlinear non-Gaussian state-space model can be further generalized to general state-space model which is defined using conditional distributions.

2.2 The Kalman filter and the smoother

It is well known that if all of the noise densities $q_n(v)$ and $r_n(w)$ and the initial state density $p(x_0)$ are Gaussian, then the conditional density of linear state-space model (1), the conditional density $p(x_n|Y_m)$, is also Gaussian and that the mean and the covariance can be obtained by the Kalman filter and the fixed-interval smoothing algorithms (Sage and Mersa 1971; Anderson and Moore 1979).

To be specific, if we assume $q_n(v) \sim N(0, Q_n)$, $r_n(w) \sim N(0, R_n)$, $p(x_0|Y_0) \sim N(x_{0|0}, V_{0|0})$, and $p(x_n|Y_m) \sim N(x_{n|m}, V_{n|m})$, then the Kalman filter is given as follows:

One-step-ahead prediction:

$$\begin{aligned} x_{n|n-1} &= F_n x_{n-1|n-1} \\ V_{n|n-1} &= F_n V_{n-1|n-1} F_n^T + G_n Q_n G_n^T. \end{aligned} \tag{3}$$

Filter

$$\begin{aligned} K_n &= V_{n|n-1} H_n^T (H_n V_{n|n-1} H_n^t + R_n)^{-1} \\ x_{n|n} &= x_{n|n-1} + K_n (y_n - H_n x_{n|n-1}) \\ V_{n|n} &= (I - K_n H_n) V_{n|n-1}. \end{aligned} \tag{4}$$

Using these estimates, the smoothed density is obtained by the following

Fixed-interval smoothing algorithm:

$$\begin{aligned} A_n &= V_{n|n} F_n^T V_{n+1|n}^{-1} \\ x_{n|N} &= x_{n|n} + A_n(x_{n+1|N} - x_{n+1|n}) \\ V_{n|N} &= V_{n|n} + A_n(V_{n+1|N} - V_{n+1|n})A_n^T. \end{aligned} \tag{5}$$

2.3 The non-Gaussian filter and the smoother

It is well known that for the nonlinear non-Gaussian state-space model (2), the recursive formulas for obtaining the densities of the one-step-ahead predictor, the filter and the smoother are as follows:

One-step-ahead prediction:

$$p(x_n|Y_{n-1}) = \int_{-\infty}^{\infty} p(x_n|x_{n-1})p(x_{n-1}|Y_{n-1})\mathrm{d}x_{n-1}. \tag{6}$$

Filtering:

$$p(x_n|Y_n) = \frac{p(y_n|x_n)p(x_n|Y_{n-1})}{\int p(y_n|x_n)p(x_n|Y_{n-1})\mathrm{d}x_n}. \tag{7}$$

Smoothing:

$$p(x_n|Y_N) = p(x_n|Y_n)\int_{-\infty}^{\infty} \frac{p(x_{n+1}|Y_N)p(x_{n+1}|x_n)}{p(x_{n+1}|Y_n)}\mathrm{d}x_{n+1}. \tag{8}$$

In Kitagawa (1987, 1988), an algorithm for implementing the non-Gaussian filter and smoother was developed by approximating each density function using a step-function or a continuous piecewise linear function and by performing numerical computations. This method was successfully applied to various problems such as estimation of trend or volatility, spectrum smoothing, smoothing discrete process and tracking problem (Kitagawa and Gersch 1996; Kitagawa 2010).

2.4 Sequential Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models

The non-Gaussian filter and the smoother based on numerical integration mentioned in the previous subsection have a limitation that they can be applied to only lower dimensional, such as the third- or the fourth-order, state-space model. Sequential Monte Carlo filter and smoother, hereinafter denoted as MCF, were developed to mitigate this problem. In this method, each distribution appeared in recursive filter and smoother is approximated by many "particles" that can be considered as realizations from that distribution (Gordon et al. 1993; Kitagawa 1993, 1996).

In this paper, we use the following notations, $\{p_n^{(1)}, \ldots, p_n^{(m)}\} \sim p(x_n|Y_{n-1})$, $\{f_n^{(1)}, \ldots, f_n^{(m)}\} \sim p(x_n|Y_n)$, $\{s_{n|N}^{(1)}, \ldots, s_{n|N}^{(m)}\} \sim p(x_n|Y_N)$. In effect we approximate

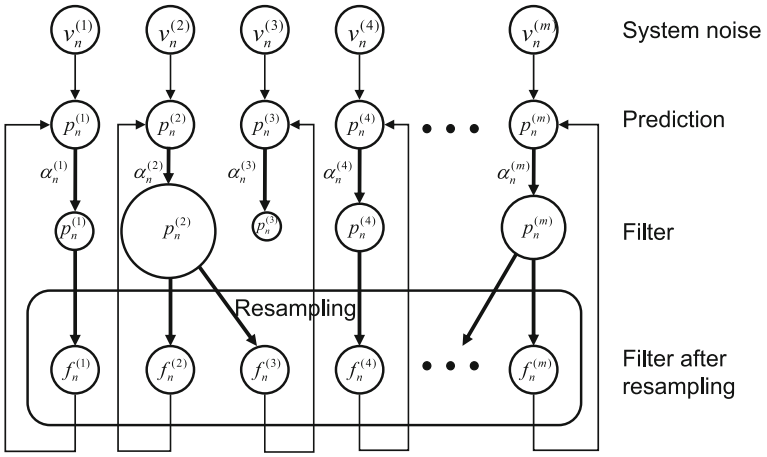

Fig. 1 Sequential Monte Carlo filter. One cycle of prediction, filter, and resampling

the cumulative distributions by the empirical distributions determined by the set of "particles".

Then a recursive filtering algorithm is realized as follows (Gordon et al. 1993; Kitagawa 1993, 1996) (Fig. 1):

1. Generate a k-dimensional random number $f_0^{(j)} \sim p_0(x)$, for $j = 1, \ldots, m$.
2. Repeat the following steps for $n = 1, \ldots, N$.
   (a) Generate an ℓ-dimensional random number $v_n^{(j)} \sim q(v)$, for $j = 1, \ldots, m$.
   (b) Generate a new particle by $p_n^{(j)} = F(f_{n-1}^{(j)}, v_n^{(j)})$, for $j = 1, \ldots, m$.
   (c) Compute the importance weight $\alpha_n^{(j)} = r(y_n - H(p_n^{(j)}))$, for $j = 1, \ldots, m$.
   (d) Generate $f_n^{(j)} \sim (\sum_{i=1}^m \alpha_n^{(i)})^{-1} \sum_{i=1}^m \alpha_n^{(i)} I(x, p_n^{(i)})$, for $j = 1, \ldots, m$ by the resampling of $p_n^{(1)}, \ldots, p_n^{(m)}$.

In Kitagawa (1993, 1996), it is shown that the particles approximating the smoothing distribution are obtained by a simple modification of the Monte Carlo filter. Assume that $(s_{1|i}^{(j)}, \ldots, s_{n|i}^{(j)})^T$ denotes the j-th realization of the conditional joint density $p(x_1, \ldots, x_n|Y_i)$. Then an algorithm for smoothing is obtained by replacing the Step 2 (d) of the algorithm for filtering;

(d-S) Generate $\{(s_{n-L|n}^{(j)}, \cdots, s_{n-1|n}^{(j)}, s_{n|n}^{(j)})^T,\ j = 1, \ldots, m\}$
by the resampling of $\{(s_{n-L|n-1}^{(j)}, \cdots, s_{n-1|n-1}^{(j)}, p_n^{(j)})^T,\ j = 1, \ldots, m\}$.

This is equivalent to applying the L-lag fixed-lag smoother rather than the fixed-interval smoother (Anderson and Moore 1979). The increase of lag, L, will improve the accuracy of the $p(x_n|Y_{n+L})$ as an approximation to $p(x_n|Y_N)$, while it is very likely to decrease the accuracy of $\{s_{n|N}^{(1)}, \cdots, s_{n|N}^{(m)}\}$ as representatives of $p(x_n|Y_{n+L})$ (Kitagawa 1996). Since $p(x_n|Y_{n+L})$ usually converges quickly to $p(x_n|Y_N)$, it is recommended to take L not so large (such as, 20, or at the largest 50).

3 MCF with a large number of particles: an empirical study

Although many refinements on the MCF algorithm have been developed, because of the development of large-scale parallel processors, it is interesting to see what happens if the number of particles becomes huge in the crude MCF algorithm. In this section, by an empirical study, we investigate the properties of the MCF from the viewpoints of accuracy and computation time when the number of particles gets very large. Here, we consider the simplest first-order trend model

$$\begin{aligned} x_n &= x_{n-1} + v_n \\ y_n &= x_n + w_n, \end{aligned} \tag{9}$$

where y_n is the observed time series, x_n is the unknown trend component, and v_n and w_n are the system noise and observation noise, respectively. Here, w_n is assumed to follow a Gaussian distribution with mean 0 and variance σ^2. As distributions of the system noise, v_n, we consider the following two cases: a Gaussian distribution $N(0, \tau^2)$ and a Cauchy distribution $C(0, \tau^2)$ with density function $f(v) = \tau\pi^{-1}(v^2 + \tau^2)^{-1}$.

The test data consisting of 500 observations are generated from the model, $y_n \sim N(t_n, 1)$ where $t_n = 0$ for $1 \le n \le 150$ and $351 \le n \le 500$, $t_n = -1$ for $151 \le n \le 250$ and $t_n = 1$ for $251 \le n \le 350$. For simplicity, the variance or dispersion parameters, σ^2 and τ^2, are assumed to be given as $\tau^2 = 1.22 \times 10^{-2}$ and $\sigma^2 = 1.043$ for the Gaussian model and as $\tau^2 = 3.48 \times 10^{-5}$ and $\sigma^2 = 1.022$ for the Cauchy model (Kitagawa 1987, 1996).

The problem here is to obtain the one-step-ahead predictive distribution, $p(t_n|Y_{n-1})$, the filter distribution, $p(t_n|Y_n)$, and the smoothing distribution, $p(t_n|Y_N)$, of the trend component $t_1, \ldots, t_{500}$ given the 500 observations, $Y_N = \{y_1, \ldots, y_{500}\}$. For the Gaussian noise case, the exact marginal posterior distributions can be obtained by the Kalman filter and the smoothing algorithm, which are used as the true distributions in the evaluation of the MCF. On the other hand, for the Cauchy noise model, the non-Gaussian filter and smoother (Kitagawa 1987) are used to obtain an approximately "true" distributions.

Computer codes were written by FORTRAN and were performed on a shared-memory parallel processor with 352 cores (SPARC64 VII (2.88 GHz), 88 CPUs, 2TB, 4.0TFLOPS), and a Mersenne twister (Matsumoto and Nishimura 1998) was used as a pseudo-random number generator.

To reduce the computation time, we use the following stratified sampling (Kitagawa 1996) in the resampling step 2-(d);

1. Set the search starting point as $j_1 = 1$.
2. For $i = 1, \ldots, m$,
   (a) Set $s_i = (i - r_i)/m$, where r_i is a common uniform random number on [0,1). Note that, r_i may be an individual uniform random number drawn for each i or a constant such as 1/2.
   (b) Find the smallest $j \in [j_i, m]$ that satisfies $\alpha_j \le s_i$.
   (c) Set the i-th particle by $f_i = p_j$ and the search starting point by $j_{i+1} = j$.

3.1 Computation time and sampling variability of log-likelihood

Table 1 shows the computation times and the log-likelihoods for MCF with various numbers of particles. The second column shows the computation time in seconds required for single-core computation. The computation time of the current stratified resampling algorithm with a single core is proportional to the number of particles, m. Note that a crude resampling algorithm requires $O(m^2)$ operations. The third and fourth columns show the mean and the standard deviation of the log-likelihoods for 100 repetitions of the MCF with different random numbers for the Gaussian model. The table shows that the standard deviation of the log-likelihood is larger than 0.1, even with $m = 10^5$ particles.

The fifth through seventh columns show the results for the Cauchy model. The standard deviations of the log-likelihood are larger than those of the Gaussian model for 10^3 particles or fewer. However, for m greater than or equal to 10^4, standard deviations of the log-likelihood are smaller than those of the Gaussian model. This reflects the fact that non-Gaussian distributions cannot be reasonably represented by a small number of particles, whereas for the case in which the number of particles is large, the posterior distribution can quickly adapt to the jumps of the mean values in the test data, which results in the reduction of the instability of the filter.

Table 2 shows the accuracy of the distributions obtained by the MCF with various numbers of particles $m = 10^2, \ldots, 10^7$. The accuracies of the predictive distribution, the filter distribution, the filter distribution after resampling, the smoothing distribution with the best lag, the smoothing distribution with the maximum lag ($k = 500$), together with the log-likelihoods and the best lags that attain the maximum accuracy among all possible lag time, $0, \ldots, N$ are shown. Note that the smoothing with the maximum lag is equivalent to the fixed-interval smoothing.

The accuracy of the marginal posterior distributions obtained by MCF is evaluated on the basis of the divergence of the true and the estimated distribution functions measured by

Table 1 Log-likelihoods and computation times in seconds for various numbers of particles, $m = 10^k$, $k = 2, \ldots, 9$

m	Gauss model			Cauchy model		
	CPU-time	Log-L	SD	CPU-time	Log-L	SD
10^2	0.02	−750.859	2.287	0.02	−752.207	6.247
10^3	0.06	−748.529	1.115	0.06	−743.244	2.055
10^4	0.58	−748.127	0.577	0.63	−742.086	0.429
10^5	5.84	−747.960	0.232	6.27	−742.024	0.124
10^6	59.41	−747.931	0.059	62.73	−742.029	0.038
10^7	591.04	−747.926	0.023	680.33	−742.026	0.013
10^8	5,906.62	−747.930	0.008	6,801.55	−742.026	0.003
10^9	59,077.35	−747.928	0.002	69,255.03	−742.026	0.001

The computation times, and the mean and standard deviation of log-likelihood for 100 runs with different random numbers are shown

Table 2 Accuracy of the MCF with different numbers of particles

Noise model	m	Log-likelihood	Pred.	Filter	Resamp. filter	Fixed-lag smoother	Max-lag smoother	Optimal lag
Gauss	10^2	−750.995	3.1811	3.2227	3.3411	8.6931	41.7225	16.4
	10^3	−748.728	0.5201	0.5385	0.5500	2.2594	16.2752	22.9
	10^4	−748.077	0.1131	0.1189	0.1201	0.7171	5.5469	27.9
	10^5	−747.951	0.0251	0.0265	0.0266	0.1848	1.4475	33.8
	10^6	−747.923	0.0042	0.0045	0.0045	0.0341	0.1794	42.9
	10^7	−747.932	0.0004	0.0004	0.0004	0.0042	0.0173	54.4
Cauchy	10^2	−751.439	19.9358	20.2267	20.2927	21.2479	47.8807	18.2
	10^3	−742.897	4.0350	4.1334	4.1409	6.0420	23.6541	30.5
	10^4	−742.141	0.3762	0.3875	0.3883	1.0009	3.6785	50.2
	10^5	−742.076	0.0431	0.0431	0.0431	0.1396	0.3800	78.0
	10^6	−742.028	0.0049	0.0046	0.0046	0.0173	0.0413	94.7
	10^7	−742.027	0.0021	0.0017	0.0017	0.0035	0.0055	114.0

Gaussian and Cauchy cases: from left to right, the number of particles, the log-likelihood, and the accuracies of prediction, filter, filter after resampling, smoothing with the best lag, smoothing with the maximum lag, and the optimal lag obtained as the average of 100 runs are shown

$$Dist(D, \hat{D}) = \sum_{n=1}^{500} \sum_{i=1}^{I} \{D(x_i, n) - \hat{D}(x_i, n)\}^2 \Delta x, \tag{10}$$

where $\Delta x = 16/I$ and $x_i = -8 + (i-1)\Delta x$. Hereinafter, we use $I = 6400$ and in actual evaluation, the $\hat{D}(x_i, n)$ is replaced with either the predictive distribution function $\hat{D}_p(x, n)$, the filter distribution $\hat{D}_f(x, n)$, or the smoother distribution $\hat{D}_s(x, n)$ obtained by the MCF and $D(x_i, n)$ by the "true" predictive distributions $D_p(x, n)$, filter distribution $D_f(x, n)$, or the fixed-interval smoother distribution $D_s(x, n)$, respectively. As mentioned above, in the case of the Gaussian model, the true distributions can be easily obtained by the Kalman filter. However, for the Cauchy model, the "true" distributions are obtained numerically by applying the non-Gaussian filter and smoother (Kitagawa 1987).

From the table, it can be seen that, for the current example of one-dimensional Gaussian model, the predictive distribution and the filter distribution can be reasonably approximated by $m = 10^3$ particles. Actually, they are closer to the true one than the numerical integration method with k =500 nodes.

Lower half of Table 2 shows the results for the Cauchy model. Compared with the Gaussian model, for smaller m, the Cauchy model is less accurate, and in order to attain the same accuracy as the Gaussian model with $m = 10^2$, $m = 10^3$ is required. Accuracy in the smoothing is also interesting and will be discussed later in the next subsection.

Figure 2 illustrates the results listed in Table 2. The left-hand panel shows the relation between the number of particles and the accuracy for Gaussian model. From the bottom to the top, three curves show the filter distribution, the smoothing distribution

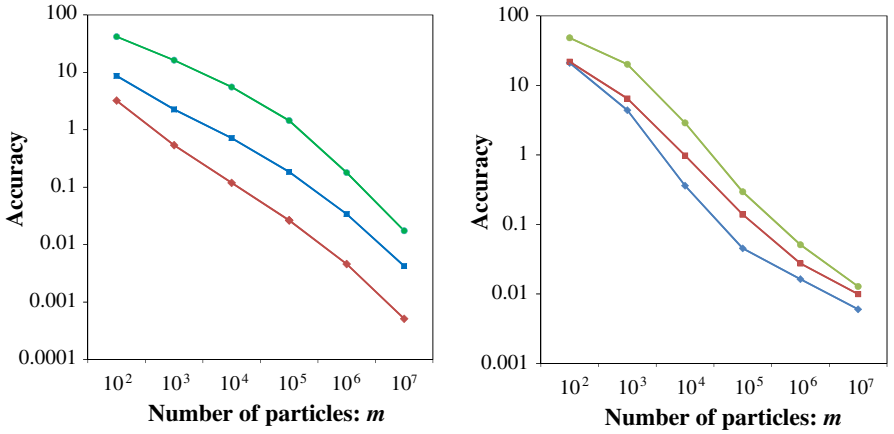

Fig. 2 Accuracy of the MCF for various numbers of particles. *Horizontal axis* number of particles; *vertical axis* accuracy as measured by $Dist(D, \hat{D})$. *Left plot* Gaussian case, *right plot* Cauchy model case. In each plot, from *bottom* to *top*, *three curves* show accuracy of the filter, smoother with the best lag, and the smoother with the maximum lag

with the best lag and the smoothing distribution with the maximum lag, i.e., $k = 500$, respectively. On a log–log scale, the relations are almost linear.

The right-hand panel shows the change in accuracy for the Cauchy model. The increase in accuracy for $m = 10^3, ..., 10^6$ is more rapid than that for the Gaussian model.

3.2 Fixed-lag smoothing with large lag

In this subsection, we shall consider the accuracy of the fixed-lag smoothing distribution as an approximation to the exact fixed-interval smoothing distribution. Figure 3 shows the accuracy of the fixed-lag smoothers up to the maximum lag of $k = 500$. Note that fixed-lag smoothing with $k = 500$ yields a fixed-interval smoother, but is not recommended to use very large lag for small m. The left-hand panel shows the results for the Gaussian model, and the right-hand panel shows the results for the Cauchy model. In each panel, from top to bottom, the six curves show the cases with $m = 10^2, \ldots, 10^7$ particles.

The smoothing distribution is less accurate than the filter and a larger number of particles, e.g., $m = 10^4$, is required in order to attain the accuracy measure less than 1. Obviously, this is due to the collapse of the distribution caused by resampling of the particles. Note that in the MCF computation, fixed-interval smoothing distribution is usually degenerated, and it is recommended to use the fixed-lag smoothing with a moderate lag, such as 20 or 50, since the best lag is known only when we know the true distribution such as the present simulation study.

For the Gaussian case, the best approximation is attained at lag=16 for $m = 100$ (see also the last column of Table 2), and for the lag larger than 100, it becomes worse than the accuracy with $k = 0$, namely that of the filter. However, as the number of particles, m, increases, the best lag length becomes large, and for $m = 10^7$, it becomes larger

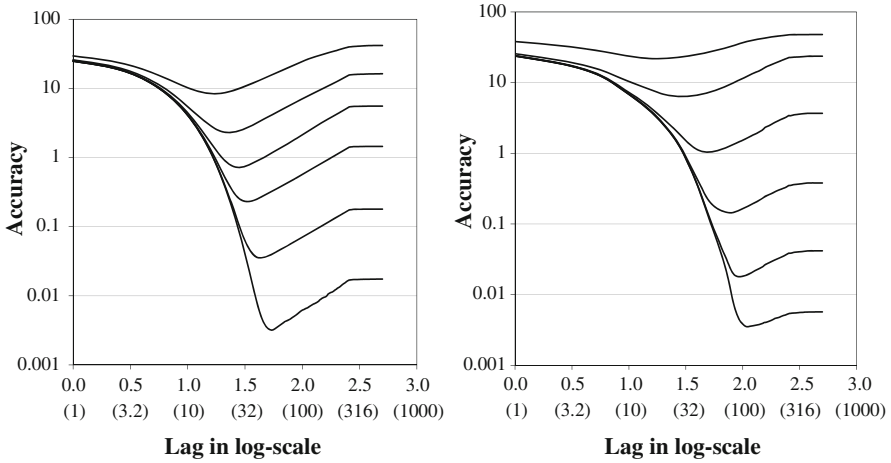

Fig. 3 Accuracy of the smoothed distributions for large lag in fixed-lag smoothing. *Left* Gaussian model, *right* Cauchy model. *Horizontal axis* lag, *vertical axis* accuracy. From *top* to *bottom*, the numbers of particle, $m = 10^2, \ldots, 10^7$

than 50. The figure also shows that for large m, a large lag will not adversely affect the accuracy of the distribution so severely, and rule of thumb to set the maximum lag to 20-50 recommended in the literature (i.e., 1996) is reasonable in this case.

The right-hand panel shows the results for Cauchy model. Compared with the case of the Gaussian model, the optimal lag length shifts to the right quickly with the increase in the number of particles, and for $m = 10^7$, the optimal lag length exceeds $k = 100$. In this case, this figure indicates that the degeneracy of the distribution becomes slight using a large number of particles, and the best lag may become larger than the one determined by the rule of thumb mentioned above. It should be noted that the curves in both plots are parallel to each other on the right-hand half of the log scale plots. This means the increase of the number of particles will ensure the increase of the accuracy of the smoother for large lag.

Figure 4 shows the marginal posterior distribution of the trend component obtained by the fixed-lag smoothing with the maximum lag length, i.e., $k = 500$. The four panels show the MCFs with $m = 10^4$, 10^5, 10^6, and 10^7, respectively. In each panel, seven curves show the 0.13, 2,27, 15.87, 50, 84.13, 97.73, and 99.87 % points that correspond to the mean and the ± 1, 2, $3\times$ (standard deviation) intervals for the Gaussian distributions. For $m = 10^4$ and 10^5, some spiky fluctuations are observed in the $\pm 3\times$ (standard deviation) curves. However, at least visually, no deterioration of the distribution is observed for $m = 10^7$, as compared to the fixed-lag smoothing with the best lag and the "exact" fixed-interval smoother obtained by the numerical integration method. This also demonstrates that the degeneracy of the smoothing distribution can be mitigated using a large number of particles for the MCF.

Figure 5 shows the details of the changes of the fixed-lag smoothing distributions for the state x_{252} just after the largest jump in the trend, as the lag length increases. The histograms with 80 bins representing the posterior distributions of the filter and the smoothers with various lags are shown for four different numbers of particles, $m = 100$, 10^4, 10^6, and 10^8.

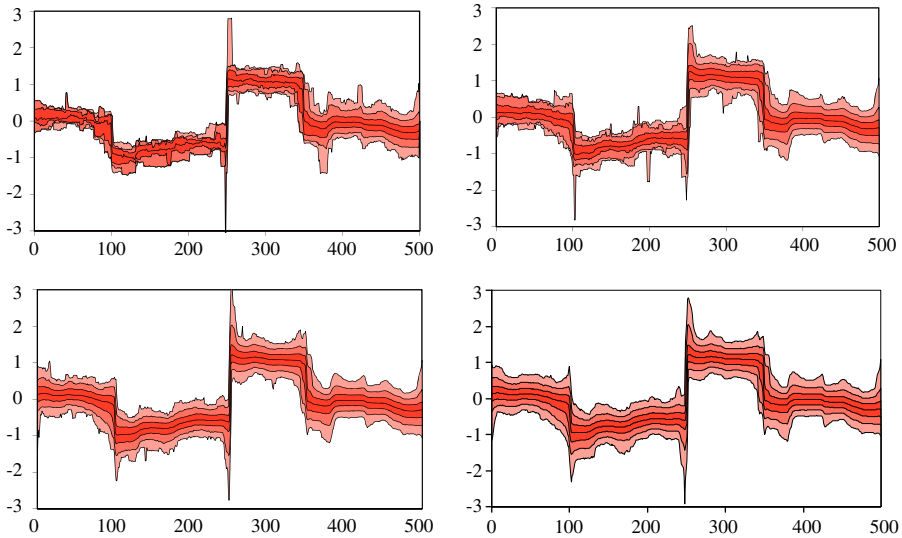

Fig. 4 Smoothed posterior distribution of the trend obtained using a Cauchy model. $m = 10^4$, 10^5, 10^6, and 10^7, Lag $= 500$. Each figure shows time changes of the 0.13, 2.27, 15.87, 50, 84.13, 97.73, and 99.87 % points of the smoothed posterior distributions

For all m, filtered distributions, $p(x_{252}|Y_{252})$, are uni-modal and do not reflect a jump in the trend, and most of the particles remain on the negative side. However, except for $m = 100$, after obtaining the observation y_{253}, the smoothed distributions became bimodal and a large part of the particles shifted to the right (positive) side. The histograms obtained with $m = 10^8$ particles are very smooth, but become rougher as the number of particles decreases. For $m = 100$, all of the particles shrunk into a single bin at $n = 256$ and failed to shift to the positive side. For $m = 1{,}000$, for which the histograms are not shown, the particles shrunk to a single bin located on the positive side.

To avoid misunderstanding that the particles always shrink to a single bin for smaller m, smoothed distributions of the state x_{450} are shown in Fig. 6 as a typical case in which no level shift occurs. In this case, the particles do not shrink to a single bin and we can obtain fairly reasonable posterior distributions, even for a very small number of particles such as $m = 100$.

4 The two-filter formula for smoothing

As mentioned in the previous section, the main reason that the smoothing distribution loses the accuracy is the reduction of the number of different particles by repeating the resampling step. Various resampling algorithms have been developed to mitigate this difficulty (Doucet et al. 2000). One way to address this problem, however, is to use two-filter formula based on the decomposition of $p(x_n|Y_N)$ (Solo 1982; Kitagawa 1989, 1994, 1996; Briers et al. 2010);

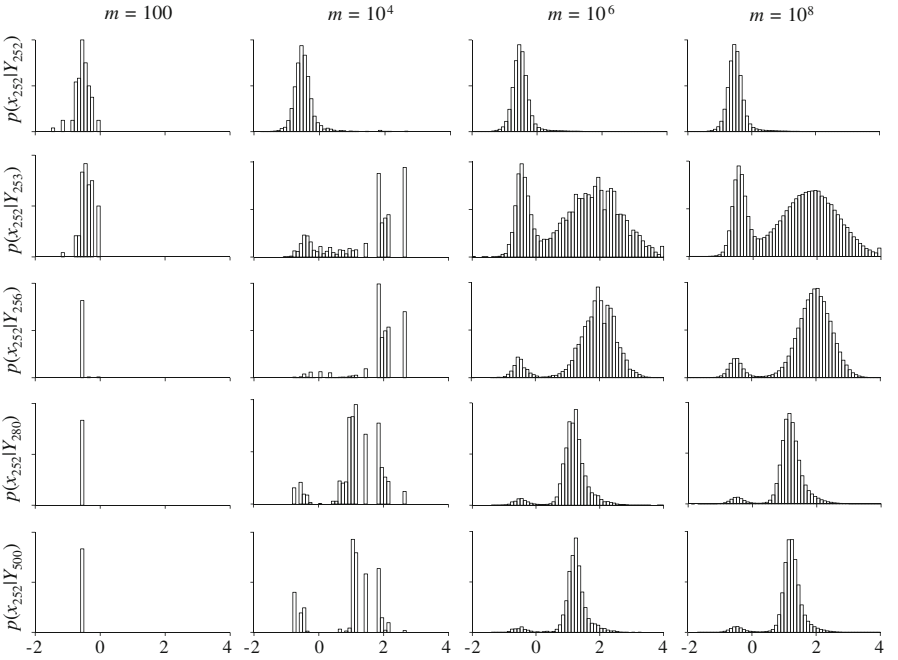

Fig. 5 Filter and fixed-lag smoothing distributions of the state x_{252} obtained by a Cauchy model, for $m = 10^2, 10^4, 10^6$, and 10^8

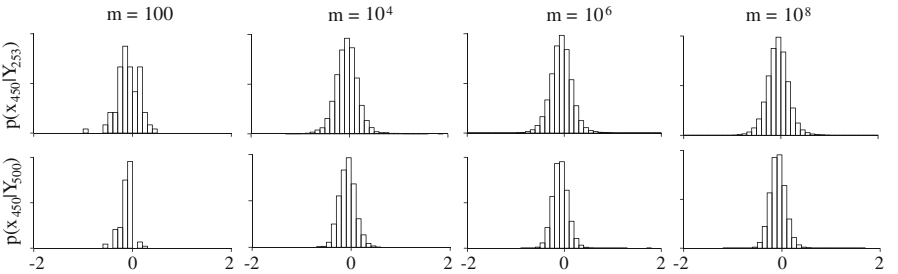

Fig. 6 Filter and fixed-lag smoothing distributions of the state x_{450} obtained using a Cauchy model for $m = 10^2, 10^4, 10^6$, and 10^8

$$p(x_n|Y_N) = p(x_n|Y_{n-1})p(Y_{n:N}|x_n)p(Y_{n:N}|Y_{n-1})^{-1}, \tag{11}$$

where $Y_{n:N} \equiv Y_N \ominus Y_{n-1} = \{y_n, ..., y_N\}$. Here $p(Y_{n:N}|x_n)$ can be evaluated by the following filtering for the properly defined backward state-space model.

To obtain smoothed marginal posterior distribution $p(x_n|Y_N)$ exactly, it is necessary to evaluate the importance weight of each particle of the predictive distribution $p_n^{(j)}$ by

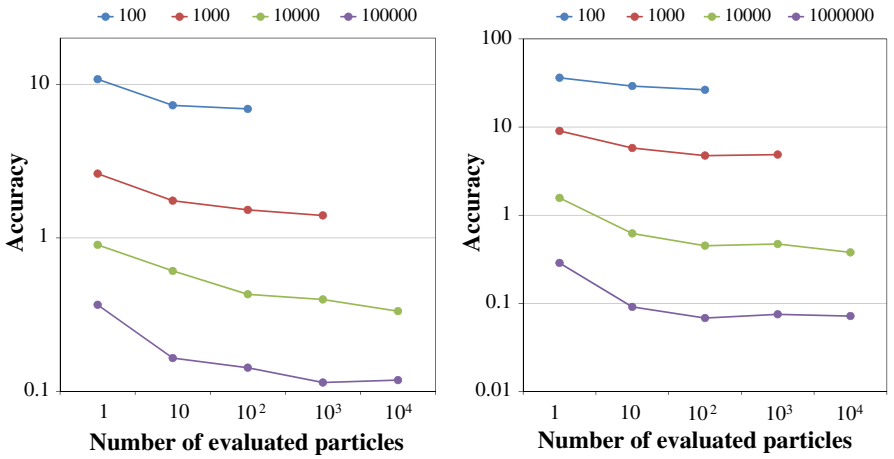

Fig. 7 Change of the accuracy of two-filter smoothing algorithm for various r. *Left plot* Gaussian model, *right plot* Cauchy model. *Horizontal axis* the number of evaluated particles, r, *vertical axis* accuracy

$$\beta_n^{(j)} = \frac{1}{m}\sum_{i=1}^{m} H_n(\tilde{f}_n^{(i)} - p_n^{(j)}), \tag{12}$$

where $\tilde{f}_n^{(i)}$ is a particle generated by the backward filtering. The evaluation of the likelihood for all m^2 combinations of particles requires a huge amount of computations for large m.

In actual computations, it can be approximated by sampling r particles from m particles, $\tilde{p}_n^{(1)}, \ldots, \tilde{p}_n^{(m)}$,

$$\beta'^{(j)}_n = \frac{1}{r}\sum_{\alpha=1}^{r} H_n(\tilde{f}_n^{(i_\alpha)} - p_n^{(j)}). \tag{13}$$

As shown in Fig. 7, it is possible to achieve good approximation of the exact distribution by evaluating for only $r = 10$ to 10^2 particles of $p(Y_{n:N}|x_n)$.

Table 3 shows the comparison of the accuracy among the fixed-lag smoother with the best lag, the fixed-interval smoother and the smoother obtained by the two-filter formula. It can be seen that the smoothing by the two-filter formula outperforms not only the fixed-interval smoother but also that of the fixed-lag smoother with the best lag. Note that the best lag for the fixed-lag smoothing is actually unknown unless we know the true distribution, thus it is difficult to attain the accuracy of the best fixed-lag smoother in practice.

Figure 8 shows the marginal posterior distributions of the fixed-interval smoother and the smoother based on two-filter formula, for the number of particles $m = 10^3$, 10^4, and 10^5. From the figure, it can be seen that very good approximation of marginal posterior distribution is obtained even with $m = 10^4$ by the two-filter smoothing formula.

Table 3 Comparison of the accuracy of the fixed-lag smoother, fixed-interval smoother and the two-filter formula for various numbers of particles, $m = 10^k$, $k = 2, \ldots, 5$

m	Gauss model			Cauchy model		
	Fixed lag	Fixed interval	Two filter	Fixed lag	Fixed interval	Two filter
10^2	8.693	41.723	6.913	21.248	47.881	26.440
10^3	2.259	16.275	1.399	6.042	23.654	4.870
10^4	0.717	5.547	0.333	1.001	3.679	0.378
10^5	0.185	1.448	0.118	0.140	0.380	0.072

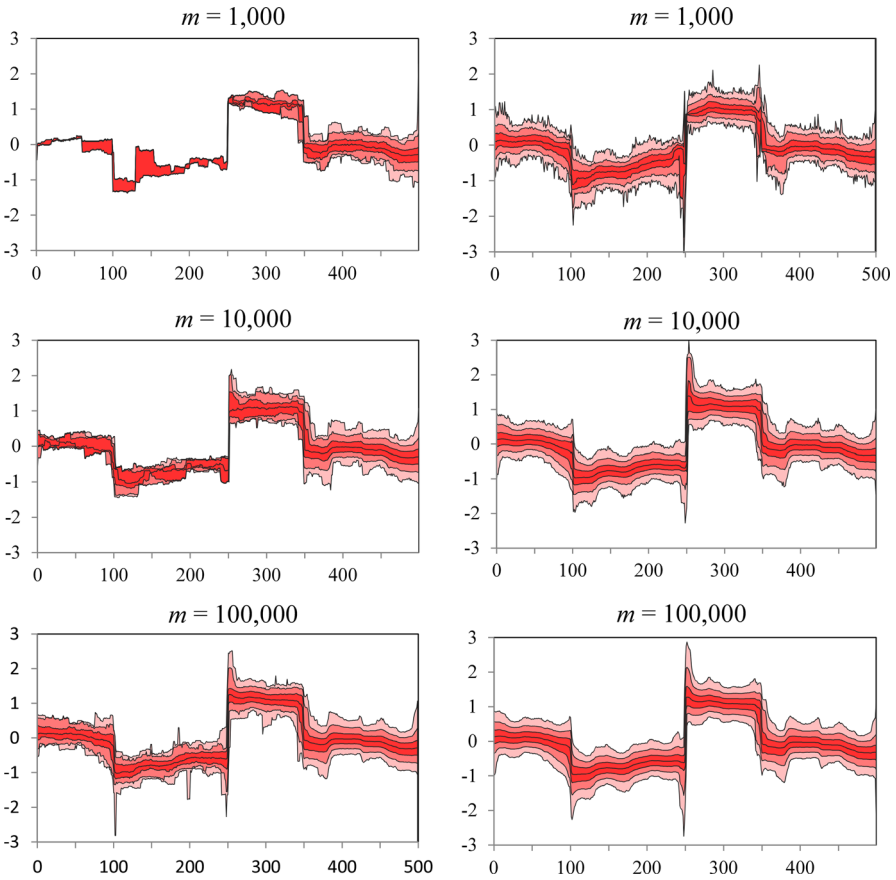

Fig. 8 Comparison of fixed-interval smoothing and two-filter formula. *Left plots* fixed-interval smoother; *right plots* two-filter formula

5 Parallel computation

5.1 Direct parallelization of the Monte Carlo filter

As shown in Table 1, even with the stratified resampling algorithm, the computation time of the MCF increases almost linearly with the number of particles. Therefore, the computation time becomes enormous as the number of particles increases.

The use of parallel computer for the MCF is considered in this section. In the parallel computation for the MCF, attention must be paid to the random number generation and resampling steps. In the random number generation, we should be careful so that the random numbers generated in parallel to be independent and computationally efficient.

In the resampling step, without careful design, numerous communication between cores occurs, which counteracts the efficiency of parallel computation. In direct parallelization of the MCF, in order to minimize communication between cores, we used the following procedure:

1. Compute the likelihood of the j-th particle in the i-th MCF, $\alpha_n^{(j,i)}$, for $j = 1, \ldots, m$ and $i = 1, \ldots, k$.
2. Define the weight of the i-th MCF, $\beta_n^{(i)} = \sum_{j=1}^{m} \alpha_n^{(j,i)}$, for $i = 1, \ldots, k$, and cumulative weight $\gamma_n^{(i)} = \sum_{q=1}^{i} \beta_n^{(q)} / \sum_{q=1}^{k} \beta_n^{(q)}$, for $i = 1, \ldots, k$.
3. Obtain the starting point $I_1^{(i)}$ and $J_1^{(i)}$ for the resampling of the i-th MCF, as follows:
   (a) $I_1^{(i)}$ is the smallest p that satisfies $\gamma_n^{(p)} \geq i/k$.
   (b) Then, $J_1^{(i)}$ is the smallest q that satisfies $\gamma_n^{(I_1^{(i)}-1)} + \sum_{p=1}^{q} \beta_n^{I_1^{(p)}} \geq i/k$.

Then, by starting from $p_n^{(J_1^{(i)}, I_1^{(i)})}$, the resampling of the i-th MCF can be performed efficiently, without any conflict between different cores.

Table 4 shows the elapsed time (in seconds) required for the ordinary MCF with m particles by a parallel processor with k cores, where $k = 4$, 8, 16, 32, 64, and 128. Compilation was performed by OpenMP. The left-hand panel of Fig. 9 shows these results, and the right-hand panel shows the relative efficiency of the parallel computation compared with the single-core computation defined by

$$\mathrm{RE}(k, m) = \frac{T(1, m)}{k \times T(k, m)}, \tag{14}$$

Table 4 Elapsed time in direct parallel computation

m	1-CPU	4-CPU	8-CPU	16-CPU	32-CPU	64-CPU	128-CPU
10^2	0.02	0.03	0.03	0.06	0.11	0.19	0.45
10^3	0.06	0.05	0.04	0.06	0.12	0.20	0.48
10^4	0.63	0.21	0.14	0.11	0.14	0.27	0.46
10^5	6.27	1.76	1.01	0.59	0.39	0.40	0.52
10^6	62.73	18.44	10.03	5.33	2.94	1.75	1.34
10^7	628.02	183.44	99.97	54.09	34.52	19.68	12.07
10^8	6,280.90	1,827.65	998.87	534.53	343.66	209.29	173.13
10^9	62,854.89	18,243.37	9,963.29	5,471.30	3,422.24	2,096.21	1,730.12
10^{10}	–	–	–	51,435.37	29,980.80	19,846.13	15,762.55

Here, m is the number of particles and the elapsed time is measured in seconds

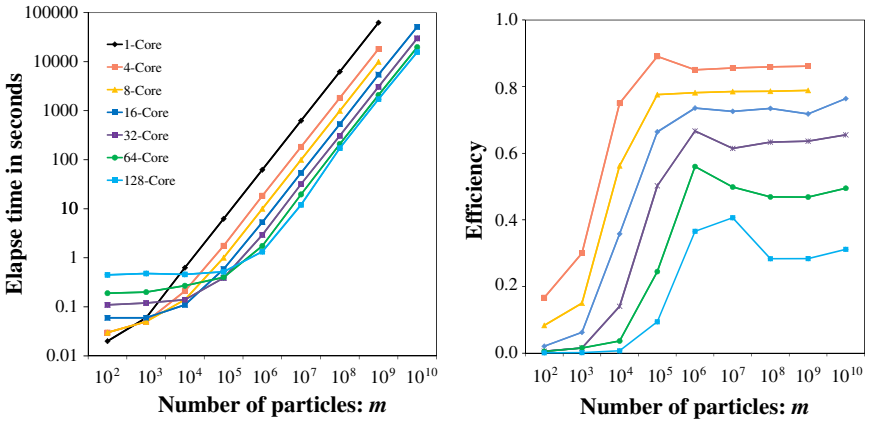

Fig. 9 Efficiency of direct parallel computation. The *left-hand panel* shows the elapsed time versus the number of particles. The *right-hand panel* shows the relative efficiency

where m is the number of particles, k is the number of cores used for parallel computation, and $T(k, m)$ is the elapsed time required for the MCF with m particles by parallel computation with k cores. From top to bottom, six curves show the change of relative efficiency in parallel computation with 4, 8, 16, 32, 64, and 128 cores. For a number of particles larger than or equal to 10^5 and a number of cores less than or equal to 16, the relative efficiency is higher than 0.7. However, low relative efficiencies are observed for $k = 64$ and 128. This is because of the overhead of the parallel computation. For larger m, the efficiency for 128 cores is 0.41–0.28. According to Amdahl's law, the parallel portion of the algorithm is evaluated to be 0.980–0.989.

5.2 Simple parallel Monte Carlo filter and smoother

One way to increase the computational efficiency in parallel computation is to compute many MCFs in parallel and to take the average of the filtered or smoothed distributions. By this method, filtering and smoothing can be performed independently on each core, and communication between cores occurs only during averaging of the posterior distributions. Therefore for large m, the elapsed time is almost inversely proportional to the number of cores.

Table 5 shows the average of the accuracies of the filter distribution in 100 runs for various values of m and k. The second through fifth columns show the results for the Gaussian model. For smaller m such as $m = 10^2$, 10^3 or 10^4, the accuracy does not increase significantly with increase of k. On the other hand, for large m, the accuracy increases by one digit as the number of parallel MCFs increases by a factor of 10. The sixth through ninth columns show the same results for the Cauchy model. In this case, to see the goodness of the parallel MCF as an approximation to the single MCF, the average of 10 distributions obtained by the MCF with $m = 10^8$ particles is used as the "true" distribution.

Table 5 Accuracy of the simple parallel Monte Carlo filter obtained as the average of 100 runs

m	Gauss model Number of parallel filters, k				Cauchy model Number of parallel filters, k			
	1	10	100	1,000	1	10	100	1,000
10^2	3.22273	1.11807	0.90401	0.90421	21.72463	11.32517	10.44000	10.26888
10^3	0.53848	0.23308	0.18685	0.18304	4.01454	1.01333	0.72036	0.72802
10^4	0.11893	0.02979	0.02150	0.01916	0.37586	0.03984	0.00677	0.00479
10^5	0.02650	0.00377	0.00141	0.00081	0.03416	0.00327	0.00034	0.00008
10^6	0.00396	0.00042	0.00004		0.00334	0.00032	0.00004	
10^7	0.00039	0.00004			0.00031	0.00003		
10^8	0.00017				0.00003			

m is the number of particles, and k is the number of parallel filters

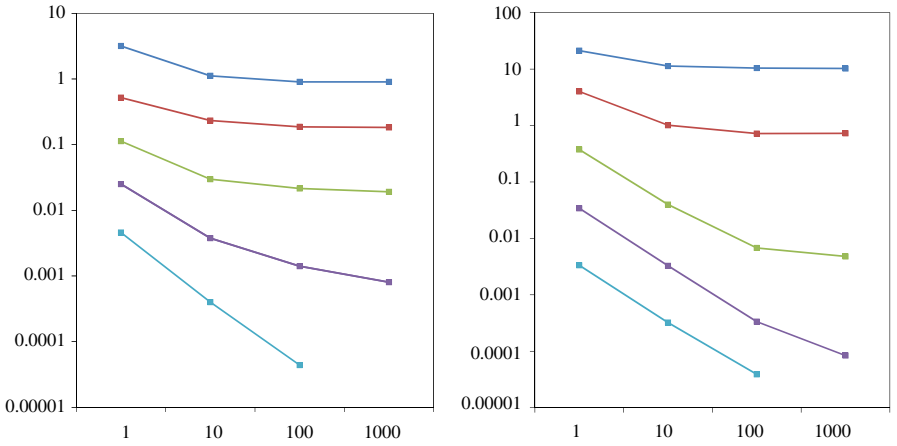

Fig. 10 Accuracy of simple parallel computation for Gauss model (*left*) and Cauchy model (*right*). From *top* to *bottom*, $m = 10^2, 10^3, 10^4, 10^5$, and 10^6. *Horizontal axis* number of parallel MCFS, *vertical axis* accuracy

Figure 10 illustrates these results. For Gaussian model, the simple parallel MCF is effective only up to $k = 10$, if the number of particles of the single MCF is less than or equal to 10^4. However, for larger m, such as 10^5 and 10^6, the single MCF is effective up to at least k=100. In the case of Cauchy model, for $m = 10^4$ or larger, the simple parallel MCF is very effective and the accuracy of the simple parallel MCF with m particles and k MCFs is almost equivalent to that of a simple MCF with $m \times k$ particles.

Summarizing, although this simple parallel MCF is efficient with respect to computation time, it has a limitation in accuracy for small m. This is because the simple parallel MCF can reduce only the variance of the filter and smoother. On the other hand, the accuracy of the posterior distributions depends on both the bias and variance,

Table 6 Biases and variances for various values of m

	m	100	1,000	10^4	10^5
Gauss	Bias	0.9109	0.1883	0.0200	0.0011
	Variance	2.2741	0.3329	0.0931	0.0240
Cauchy	Bias	10.2735	0.6985	0.0034	0.0000
	Variance	10.9688	3.3143	0.3724	0.0342

and only the variance of the estimates is reduced in inverse proportion to the number of parallel MCFs.

This suggests that the accuracy of k parallel MCFs with m particles can be decomposed as

$$\text{bias}(m) + \frac{1}{k}\text{variance}(m). \tag{15}$$

The least squares estimates of the bias and variance are shown in Table 6. Bias terms show the lower bound of the accuracy measure attained by the simple parallel MCF with m particles. For Cauchy model, the bias is almost negligible for large m, and the variance is inverse proportional to the number of particles.

5.3 Weighted parallel MCF with transplantation

The results presented in the previous subsection show that the simple parallel MCF is not efficient in accuracy, for small numbers of particles because of the presence of the bias of the estimates. To develop an efficient method with respect to both computation and accuracy, we consider a weighted parallel MCF. In this method, k MCFs are basically performed independently like the simple parallel MCF. We assume that $p_n^{(j,i)}$ and $f_n^{(j,i)}$ are the j-th particles in the predictive and filtered distributions of the i-th MCF at time step n, respectively. The likelihood of the i-th MCF at time n is obtained by

$$p^{(i)}(y_n|Y_{n-1},\theta) = \frac{1}{m}\sum_{j=1}^{m}\alpha_n^{(j,i)}, \tag{16}$$

where $\alpha_n^{(j,i)}$ is the importance weight of the particle $p_n^{(j,i)}$ defined by $\alpha_n^{(j,i)} = p(y_n|p_n^{(j,i)})$. Then, the relative importance of the i-th MCF is defined by

$$w_i = \frac{\sum_{j=1}^{m}\alpha_n^{(j,i)}}{\sum_{i=1}^{k}\sum_{j=1}^{m}\alpha_n^{(j,i)}}, \tag{17}$$

and the likelihood and the posterior distribution of the state of the weighted parallel MCF are evaluated as

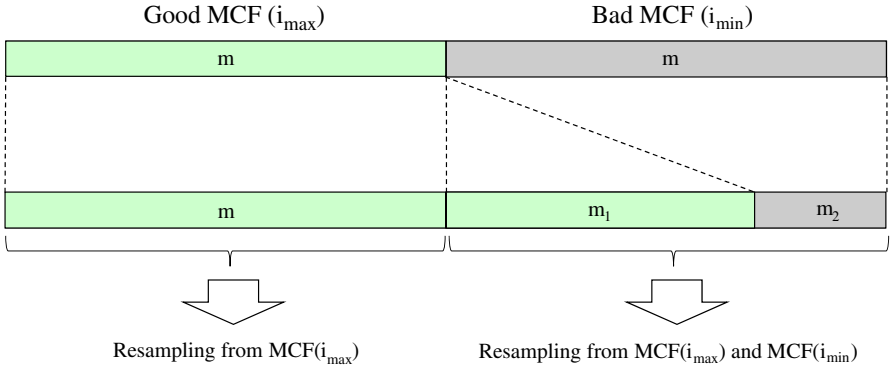

Fig. 11 Resampling in the weighted parallel MCFs with transplantation

$$p(y_n|Y_{n-1},\theta) = \sum_{i=1}^{k} w_i\, p^{(i)}(y_n|Y_{n-1},\theta) = \frac{1}{m}\sum_{i=1}^{k} w_i \sum_{j=1}^{m} \alpha_n^{(j,i)}, \tag{18}$$

$$p(x_n|Y_n,\theta) = \sum_{i=1}^{k} w_i\, p^{(i)}(x_n|Y_n,\theta) \propto \frac{1}{m}\sum_{i=1}^{k} w_i \sum_{j=1}^{m} \alpha_n^{(j,i)} I(x_n, f_n^{(j,i)}). \tag{19}$$

This weighted parallel MCF surrogates the resampling in the sense that the cumulative distribution defined by the weighted particles can yield a reasonable approximation to the distribution obtained using a single MCF with $k \times m$ particles without any communication between cores in the resampling step. However, it is likely that the weights of some MCFs may become very small as time progresses and may eventually deteriorate the efficiency of the total filter.

To alleviate this problem, we adopted the following transplantation of particles from "good" MCF to "bad" MCF. Namely, at each time step, we find the largest and smallest w_i's, and if their ratio is larger than a certain threshold, c, then the particles of both MCFs are used in the resampling of the "bad" MCF, as shown in the following algorithm. This is equivalent to merge two MCFs. Assume that the indices of "good" and "bad" MCFs are denoted by $i_{\max}$ and $i_{\min}$, respectively.

In the resampling of $\mathrm{MCF}_{i_{\min}}$, generate m_1 particles by the resampling of $p_n^{(j,i_{\max})}$ with importance weights $\alpha_n^{(j,i_{\max})}$ and generate m_2 particles by the resampling of $p_n^{(j,i_{\min})}$ with importance weights $\alpha_n^{(j,i_{\min})}$ (Fig. 11) where

$$m_1 = \frac{2m w_{i_{\max}}}{(w_{i_{\max}} + w_{i_{\min}})} - m \tag{20}$$

and $m_2 = m - m_1$. Set the importance weights of $\mathrm{MCF}_{i_{\min}}$ as

$$\frac{m_1 w_{i_{\max}} + m_2 w_{i_{\min}}}{(m_1 + m_2)}. \tag{21}$$

Repeat this transplantation of particles until the maximum ratio becomes less than the predetermined threshold. According to the experience of the author, the accuracy of the weighted parallel MCF is not sensitive to the selection of the threshold and we may set a large value such as $c = 10$.

Figure 12 shows the results of the weighted parallel MCF with transplantation for the number of particles $m = 10^2, 10^3, 10^4$ and $k = 10$, 100, and 1,000. It shows that the k parallel MCF with number of particles m yields, at least visually, the same posterior distribution as the one by $k/10$ parallel MCF with $10m$ particles. Table 7 and Fig. 13 show the accuracy of the weighted parallel MCF. Compared with Table 5, a significant improvement of the accuracy is seen, even for smaller m. Also from the figure, it can be seen that the accuracy of the weighted parallel MCF with transplantation increases almost linearly in the log–log plot and yields almost the same accuracy as the single MCF with $m \times k$ particles for large m.

6 Monte Carlo posterior mean smoother

As stated in the previous section, the Monte Carlo smoothing algorithm is occasionally unstable and the smoothed distribution of the state collapses to a small number of particles. If only the posterior mean of the marginal distribution of the state is necessary, relatively more precise approximations to the posterior mean value function can be obtained with a smaller number of particles by the following algorithm:

1. Rearrange $p_n^{(1)}, \ldots, p_n^{(m)}$ in order of magnitude. The results are expressed as $\tilde{p}_n^{(1)}, \ldots, \tilde{p}_n^{(m)}$, and the associated normalized importance weights are given by $\tilde{\alpha}_n^{(1)}, \ldots, \tilde{\alpha}_n^{(m)}$.
2. For $j = 1, \ldots, m$,
   (a) Generate a uniform random number $u_n^{(j)} \in \left[\frac{j-1}{m}, \frac{j}{m}\right]$.
   (b) Find an integer i such that

$$\sum_{\ell=1}^{i-1} \tilde{\alpha}_n^{(\ell)} < u_n^{(j)} \leq \sum_{\ell=1}^{i} \tilde{\alpha}_n^{(\ell)}.$$

   (c) Take weighted average of two particles

$$\begin{aligned} f_n^{(j)} &= (1-\theta)\tilde{p}_n^{(i-1)} + \theta \tilde{p}_n^{(i)}, \\ s_{n|n}^{(j)} &= (1-\theta) f_n^{(i-1)} + \theta f_n^{(i)} \\ s_{t|n}^{(j)} &= (1-\theta)\tilde{s}_{t|n}^{(i-1)} + \theta \tilde{s}_{t|n}^{(i)}, \quad for\ t = 1, \ldots, n-1 \end{aligned} \tag{22}$$

   where θ is given by $(u_n^{(j)} - \sum_{\ell=1}^{i-1} \tilde{\alpha}_n^{(\ell)})/\tilde{\alpha}_n^{(i)}$.

By this weighted average operation, it seems that the generated particles converged to the mean of the distribution rather than approximating the posterior distribution.

In Fig. 14, top plots shows the marginal posterior distributions obtained by the ordinary Monte Carlo smoother with a large number of particles. The middle plots

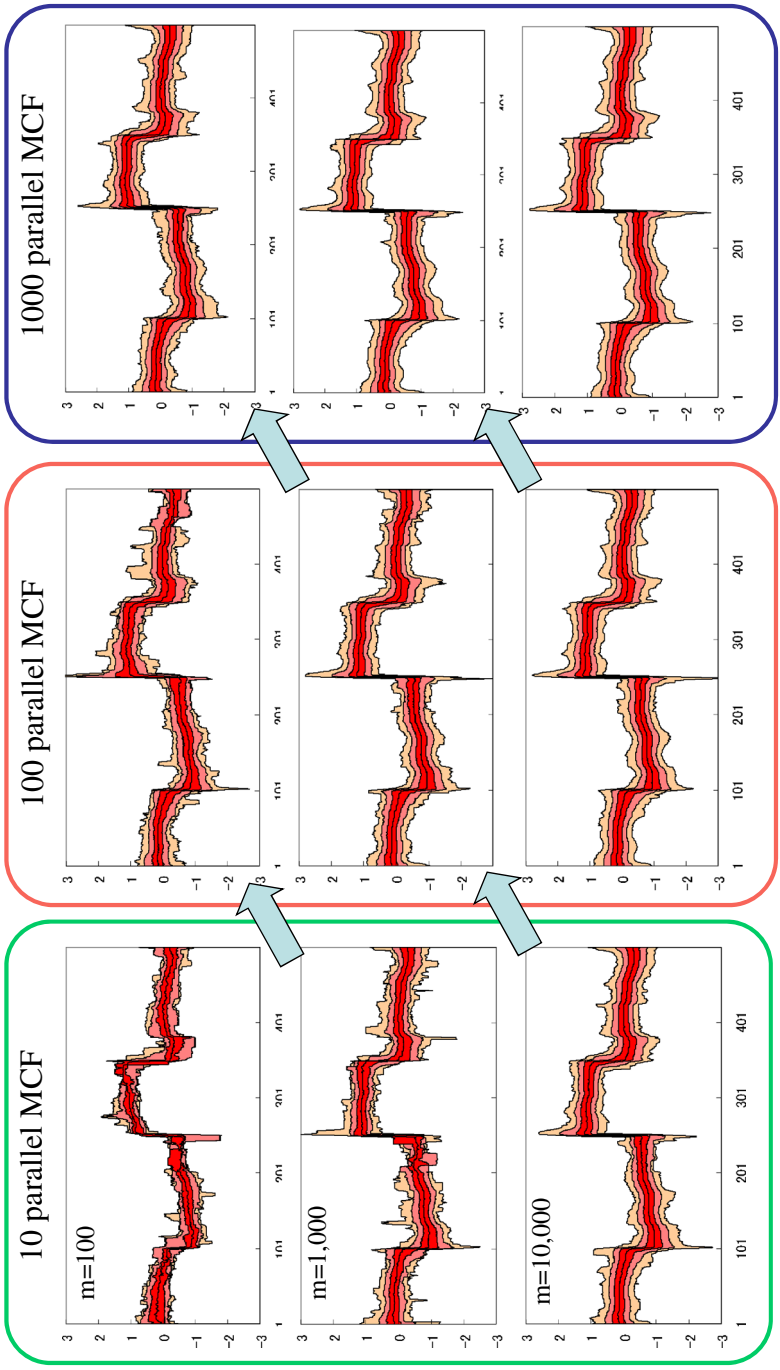

Fig. 12 Weighted parallel MCFs with transplantation for m=100, 1,000, and 10,000 and k = 10, 100, and 1,000

Table 7 Accuracy of the weighted parallel MCF with crossover obtained as the average of 100 runs

m	Number of parallel filters, k			
	1	10	100	1,000
10^2	21.24629	4.57516	0.70721	0.08344
10^3	4.01454	0.43215	0.05022	0.01415
10^4	0.37586	0.03175	0.00354	0.00037
10^5	0.03416	0.00358	0.00033	0.00004
10^6	0.00334	0.00034	0.00003	
10^7	0.00034	0.00003		

m is the number of particles, and k is the number of parallel filters

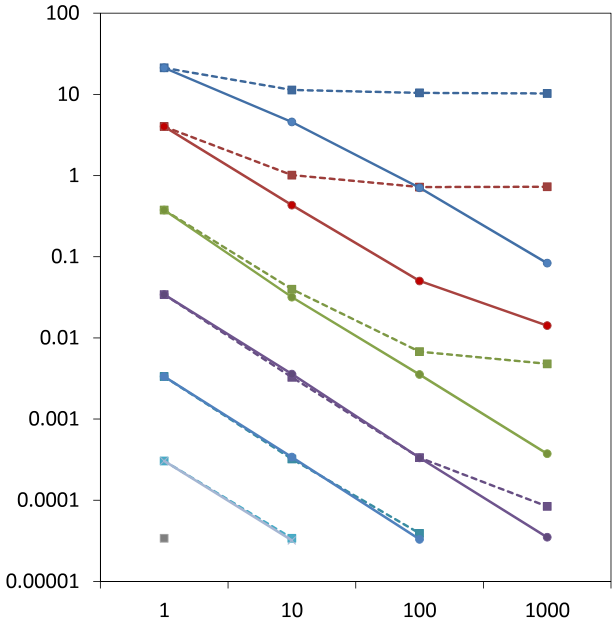

Fig. 13 Comparison of accuracy of simple parallel MCF and weighted parallel MCF for Cauchy model. From *top* to *bottom*, $m = 10^2$, 10^3, 10^4, 10^5, and 10^6. *Horizontal axis* number of parallel MCFs, *vertical axis* accuracy

show the paths of 100 particles of the Monte Carlo posterior mean smoother and the bottom plots focus on the end portion. For both Gaussian and Cauchy models, all particles of the Monte Carlo posterior mean smoother converged within 50 steps to a single point which is close to the mean of the marginal posterior distribution obtained by the Monte Carlo smoother.

Figure 15 shows the accuracy as an estimate of the posterior mean measured by the mean squared error. The dotted lines show the least squares measure of the Monte Carlo posterior mean smoother with $m = 10^2, \ldots, 10^6$, and the solid curves show the ones by the ordinary Monte Carlo fixed-lag smoother with various lags. It can be seen that except for $m = 10^2$, the posterior mean smoother outperforms the ordinary Monte Carlo smoother with any lag length.

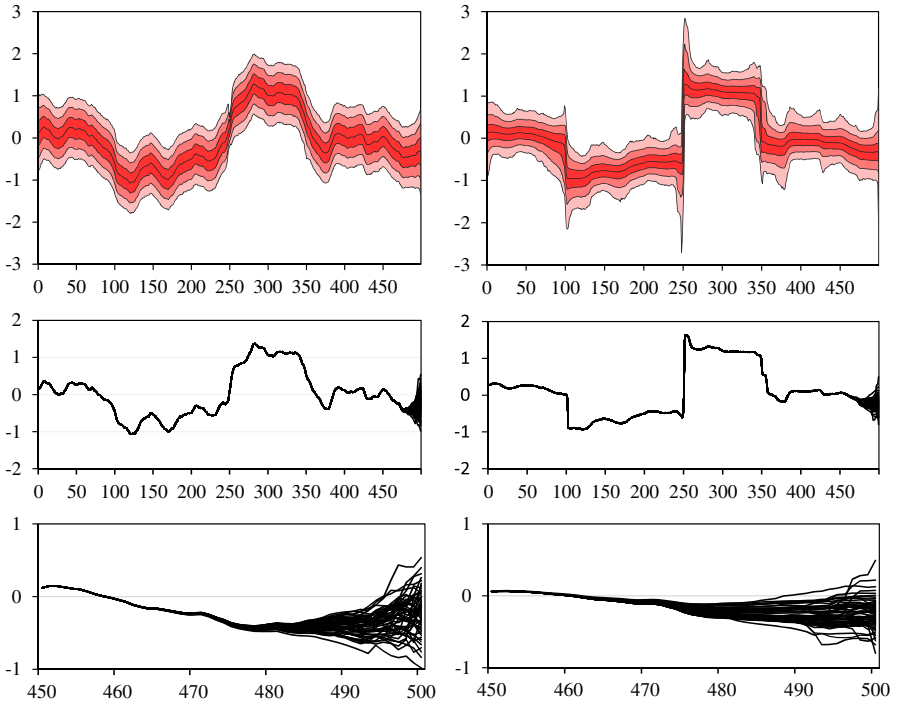

Fig. 14 Comparison of Monte Carlo Smoother and Monte Carlo Posterior Mean Smoother. *Top plots* marginal posterior distributions obtained by the Monte Carlo smoother with large number of particles, *middle plots* the posterior means obtained by the Monte Carlo posterior mean smoother, *bottom plots* enlarged end potion of the posterior means. *Left plots* Gaussian model, *right plots* Cauchy model

It is difficult to analyze the property of this smoothing algorithm, and we consider here the simplest situation where the new particles are generated independent on the observations by

$$Z_n^{(j)} = (1-\alpha)Z_{n-1}^{(\ell_i)} + \alpha Z_{n-1}^{(\ell_j)}, \quad \alpha \sim U(0,1). \tag{23}$$

Assume that the mean, the variance and the covariance of the generated particles are defined by $E[Z_n^{(j)}] = \mu$, $\mathrm{Var}[Z_n^{(j)}] = v_n$, $\mathrm{Cov}\{Z_n^{(j)}, Z_n^{(k)}\} = c_n$ for $j \neq k$. Then v_n and c_n satisfy the difference equations

$$\begin{aligned} v_n &= \frac{2}{3}v_{n-1} + \frac{1}{3}c_{n-1} \\ c_n &= \frac{3}{4m}v_{n-1} + \frac{4m-3}{4m}c_{n-1}. \end{aligned} \tag{24}$$

Therefore, under the initial conditions that $v_0 = 1$ and $c_0 = 0$, the variance and the covariance of the particles $Z_n^{(j)}$, $j = 1, \ldots,$ are, respectively, given by

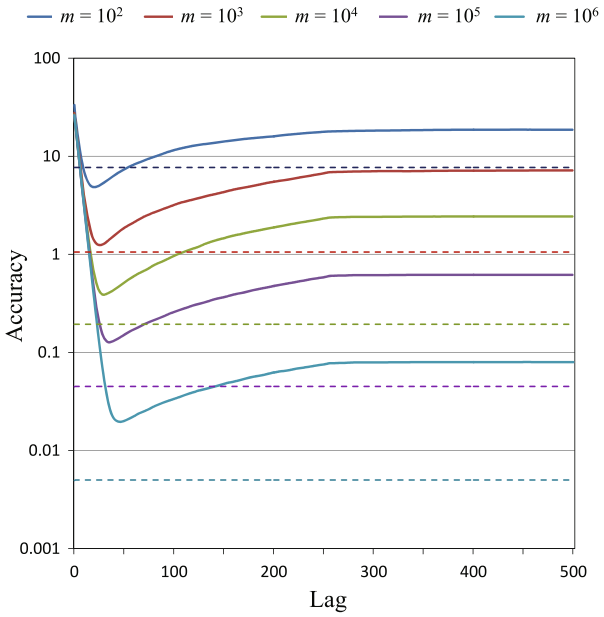

Fig. 15 Mean square errors of the Monte Carlo Posterior Mean Smoother and the Fixed-lag Monte Carlo Smoother. *Horizontal axis* lag, *vertical axis* accuracy. *Solid curve* fixed-lag smoother, *dotted line* Monte Carlo posterior mean smoother

$$v_n = (4m+9)^{-1}\left\{9 + 4m\left(\frac{2}{3} - \frac{3}{4m}\right)^n\right\} \tag{25}$$

$$c_n = (4m+9)^{-1}\left\{9 - 3\left(\frac{2}{3} - \frac{3}{4m}\right)^n\right\}. \tag{26}$$

Therefore, as n increases, both $v_n^{(j)}$ and $c_n^{(j)}$ converge to $9/(4m+9)$. This means that the correlation of the particles approaches to 1 as n increases, and that all of the particles converges to one point, namely to the estimate of the mean of the distribution, and its variance is the 9/4 of the sample mean.

Similarly, if the new particles are obtained by the average of two particles, i.e., $Z_n^{(j)} = (Z_{n-1}^{(\ell_i)} + Z_{n-1}^{(\ell_j)})/2$, v_n and c_n are given by

$$v_n = (2m+1)^{-1}\left\{3 + 2(m-1)\left(\frac{2m-1}{4m}\right)^n\right\} \tag{27}$$

$$c_n = (2m+1)^{-1}\left\{3 - 3\left(\frac{2m-1}{4m}\right)^n\right\}. \tag{28}$$

In this case, for large n, both $v_n^{(j)}$ and $c_n^{(j)}$ converge to $3/(2m+1)$ and the asymptotic variance is 3/2 of the sample mean.

7 The Gaussian-sum filter and smoother

For higher order state-space models, the application of the Monte Carlo filter and smoother may become impractical due to huge amount of computations. One practical way to mitigate this computational burden is to use of the Gaussian-sum filter that approximates arbitrary distribution by the mixture of several or many Gaussian distributions (Alspach and Sorenson 1972; Harrison and Stevens 1976; Anderson and Moore 1979; Kitagawa 1989, 1994) extended this to the Gaussian-sum smoother based on the two-filter formula.

Assume that the predictive distribution, $p(x_n|Y_{n-1})$, and the filter distribution of the backward filter $p(Y^n|x_n)$ are expressed by the mixture of L_n and M_m Gaussian components, respectively as

$$\begin{aligned} p(x_n|Y_{n-1}) &= \sum_{k=1}^{L_n} \gamma_{kn}\varphi_k(x_n|Y_{n-1}) \\ p(Y^n|x_n) &= \sum_{\ell=1}^{M_n} \delta_{\ell n}\varphi_\ell(Y^n|x_n), \end{aligned} \tag{29}$$

where $\varphi_k(x_n|Y_{n-1}) \sim N(x^k_{n|n-1}, V^k_{n|n-1})$, $\varphi_\ell(Y_n|x_n) \sim N(z^\ell_{n|n}, U^\ell_{n|n})$ and γ_{kn} and $\delta_{\ell n}$ are the weights of the Gaussian component, respectively. Here, $x^k_{n|n-1}$, $V^k_{n|n-1}$ and γ_{kn} are obtained by the Gaussian-sum filter. Similarly, $x^\ell_{n|n}$, $U^\ell_{n|n}$ and δ_ℓ are obtained by the backward Kalman filter. (For details see Kitagawa 1994.)

Then, the Gaussian-sum smoother is obtained by

$$\begin{aligned} p(x_n|Y_N) &\propto \sum_{\ell=1}^{M_n}\sum_{k=1}^{L_n} \delta_{\ell n}\gamma_{kn}\varphi_\ell(Y^n|x_n)\varphi_k(x_n|Y^{n-1}) \\ &\equiv \sum_{\ell=1}^{M_n}\sum_{k=1}^{L_n} \delta_{\ell n}\gamma_{kn}\varphi_{\ell k}(x_n|Y_N). \end{aligned} \tag{30}$$

Here $\varphi_{\ell k}(x_n|Y_N)$ is the Gaussian density whose mean and the covariance are obtained by the Kalman filter algorithm for an appropriately defined model,

$$\begin{aligned} J^{\ell k}_n &= V^k_{n|n-1}(V^k_{n|n-1} + U^\ell_{n|n})^{-1} \\ x^{\ell k}_{n|N} &= x^k_{n|n-1} + J^{\ell k}_n(z^\ell_{n|n} - x^k_{n|n-1}) \\ V^{\ell k}_{n|N} &= (I - J^{\ell k}_n)V^k_{n|n-1}. \end{aligned} \tag{31}$$

The difficulty with this Gaussian-sum filtering and smoothing is that if the system noise or observation noise are non-Gaussian and are expressed by or approximated by a mixture of several Gaussian components, then the numbers of Gaussian components, L_n and M_n, increase very rapidly as time step proceeds. Actually, even in the simplest situation where system noise is the mix-

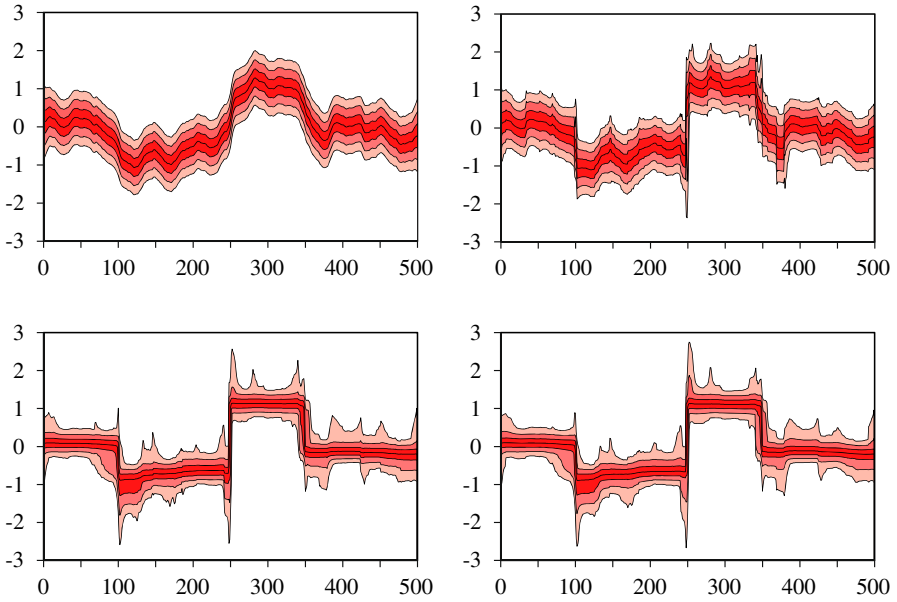

Fig. 16 Gaussian-sum smoother. *Top left plot* Gaussian smoother, *top right* Gaussian-sum smoother with $m = 1$, *bottom left* $m = 4$, *bottom right* $m = 16$

ture of two Gaussian components and the initial state distribution and observation noise are Gaussian, the number of Gaussian components at time n becomes $L_n = 2^n$.

Therefore to avoid the explosion of the number of Gaussian terms, it is necessary to reduce the number of Gaussian components at least once every several time step. In principle, the reduction of the number of Gaussian terms can be realized by minimizing the Kullback–Leibler divergence. However, especially for large L_n, searching for the parameters of the Gaussian mixture distribution with reduced number of components by the numerical optimization is very time consuming and an algorithm based on successive pooling of the most adjacent two Gaussian distributions has been developed. The details of the algorithm and criterion and several successful applications are shown in Kitagawa (1994).

Figure 16 shows the results by the Gaussian-sum smoother when the system noise is the mixture of two Gaussian components, $p(v) \sim (1-\alpha)N(0, \tau^2) + \alpha N(0, \zeta^2)$. Top left plot shows the ordinary Kalman smoothing. Top right plot shows the case when we set $L_n = 1$. Even in this case, the marginal posterior distribution is significantly different from the Gaussian model and the sudden jump of the trend is detected. Bottom left and bottom right plots show the cases when we set $L_n = 4$ and 16, respectively. It can be seen that even with $L_n = 4$, reasonable approximations were obtained by the Gaussian-sum smoother.

As shown in the example, the Gaussian-sum filter and smoother work well if the state dimension is low and the non-Gaussian state distribution is reasonably approximated by relatively small number of Gaussian components. However, if the necessary Gaussian terms becomes large, the step for reducing the number of Gaussian terms

Table 8 Comparison of computing time of the original Gaussian-sum filter/smoother and the resampling method

m	KL minimization		Resampling	
	Filter	Smoother	Filter	Smoother
1	0.00	0.00	–	–
2	0.00	0.00	0.00	0.01
4	0.01	0.01	0.00	0.00
8	0.04	0.07	0.00	0.01
16	0.25	0.34	0.00	0.03
32	1.17	1.50	0.01	0.11
64	4.60	5.90	0.02	0.41
128	21.20	25.05	0.06	1.60
256	612.93	555.91	0.19	6.29
512	–	–	0.66	24.97
1,024	–	–	1.80	98.84

dominates in the computation time and that eventually makes the Gaussian-sum filter impractical.

To mitigate this problem, we consider here the reduction of number of Gaussian components by resampling. Namely, by the analogy to the Monte Carlo filtering and smoothing, it may be considered that many Gaussian components are replaced with a smaller number of Gaussian distributions by resampling. Then it is possible to reduce the number of Gaussian components using the likelihood of each Gaussian distribution as the importance weight of the distribution.

An important remark is in order here. Namely, in the simple resampling algorithm, the Gaussian components with very small importance weight will vanish, whereas many similar distributions with relatively large importance weight will remain. Obviously, this will cause the loss of diversity of the components and will eventually spoil the ability to adjust to the sudden changes of the state which is the most significant merit of non-Gaussian distribution.

Therefore, in the present paper, we use the following ad hoc strategy in resampling. If the system noise is a mixture of two Gaussian distribution, in the resampling stem, we have to reduce the number of Gaussian components by 50 %. In doing so, 25 % of the components with high importance weights are resampled with 100 %, rest of the components except for the last one are resampled with a certain probability (approximately 25 %). The final components are defined by the weighted average of the un-selected components. This will approximately reserve the first two moments of the distribution.

Table 8 compares computing time for the Gaussian-sum filter and smoother with successive pooling and those of the filter and smoother based on the resampling. In the original Gaussian-sum filter the computation time explodes at 256. On the other hand, the resampling method is applicable even with $m = 1{,}024$.

Table 9 shows the accuracy of the Gaussian-sum filter and smoother based on the resampling algorithm. Compared with Table 2, by this ad hoc resampling method, the Gaussian-sum filter and smoother with 8 and 32 Gaussian components yield similar

Table 9 Accuracy of the original Gaussian-sum filter/smoother: original and the resampling methods

m	KL pooling		Resampling	
	Filter	Smoother	Filter	Smoother
1	20.6270	21.4308	20.6270	21.4308
2	3.8016	2.8118	5.3989	3.6996
4	0.0751	0.1008	3.5662	2.0117
8	0.0351	0.0384	1.5314	1.2371
16	0.0294	0.0346	0.6438	0.5655
32	0.0291	0.0346	0.3241	0.3402
64	0.0300	0.0345	0.1907	0.2176
128	0.0386	0.0360	0.1167	0.1713
256	0.0432	0.0415	0.0807	0.0963
512	–	–	0.0703	0.0847
1,024	–	–	0.0672	0.0781

accuracy as the MCF with 10^4 and 10^5 particles, respectively. However, unfortunately, the accuracy of this ad hoc method cannot match for the original pooling method.

8 Conclusion

In this paper, we present empirical study on the computational aspects of the Monte Carlo filtering and smoothing by applying the method to simple state-space model for which exact posterior distributions can be easily obtained by the Kalman filter or by the non-Gaussian smoother. Some of the important findings are as follows:

(i) Using a huge number of particles, we can obtain a very accurate smoothing distribution of the state using a fixed-lag smoothing algorithm with a very large lag.
(ii) By the two-filter formula for smoothing, we may obtain better smoothing distribution than the fixed-lag smoother with the best lag. The computation time of the two-filter formula can be reduced without significantly losing the accuracy by sampling in smoothing algorithm with a very large lag.
(iii) The direct parallel MCF compiled by OpenMP is effective for m larger than or equal to 10^5.
(iv) The simple parallel MCF is computationally very efficient but can only reduce the variance of the posterior, and thus is only effective when the bias of the MCF is small, i.e., the number of particles for each MCF is large.
(v) The weighted parallel MCF with transplantation can mitigate this problem of simple parallel MCF.
(vi) At least for the current example, the posterior mean smoother can yield better estimate of the mean value function than the fixed-lag smoother with any lags except for very small m.
(vii) By the resampling of Gaussian component, we can develop a fairly accurate ad hoc filter and smoother. However, these results are not as good as the one by the pooling method.

References

Alspach, D. L., Sorenson, H. W. (1972). Nonlinear Bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, *AC–17*, 439–448.

Anderson, B. D. O., Moore, J. B. (1979). *Optimal Filtering*. New Jersey: Prentice-Hall.

Briers, M., Doucet, A., Maskell, A. (2010). Smoothing algorithms for state-space models. *Annals of the Institute of Statistical Mathematics*, *62*, 61–89.

Doucet, A., Johansen, A. M. (2011). A tutorial on particle filtering and smoothing: fifteen years later. In D. Crisan, B. Rozovsky (Eds.), *Oxford Handbook of Nonlinear Filtering*. Oxford: Oxford University Press.

Doucet, A., Godsill, S., Andrieu, C. (2000). On sequential Monte Carlo ampling methods for Bayesian filtering. *Statistics and Computing*, *10*, 197–208.

Doucet, A., de Freitas, N., Gordon, N. (2001). *Sequential Monte Carlo Methods in Practice*. New York: Springer-Verlag.

Fearnhead, P., Wyncoll, D., Tawn, J. (2010). A sequential smoothing algorithm with linear computational cost. *Biometrika*, *97*, 447–464.

Gordon, N. J., Salmond, D. J., Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, *140*, 107–113.

Harrison, P. J., Stevens, C. F. (1976). Bayesian Forecasting (with discussion). *Journal of the Royal Statistical Society Series B*, *34*, 1–41.

Higuchi, T. (1997). Monte Carlo filter using the genetic algorithm operators. *Journal of Statistical Computation and Simulation*, *59*, 1–23.

Kitagawa, G. (1987). Non-Gaussian state space modeling of nonstationary time series. *Journal of American Statistical Association*, *76*(400), 1032–1064.

Kitagawa, G. (1988). Numerical Approach to Non-Gaussian Smoothing and its Applications, Computing Science and Statistics; Proceedings of the 20th Symposium on the Interface, eds. E.J. Wegman, D.T. Gantz, J.J. Miller, 379–388.

Kitagawa, G. (1989). Non-Gaussian seasonal adjustment. *Computers & Mathematics with Applications*, *18*(6/7), 503–514.

Kitagawa, G., (1993). A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state space models. *Proceedings of the 2nd U.S.-Japan Joint Seminar on Statistical Time Series, Analysis*, pp. 110–131.

Kitagawa, G. (1994). The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother. *Annals of the Institute of Statistical Mathematics*, *46*(4), 605–623.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space model. *Journal of Computational and Graphical Statistics*, *5*, 1–25.

Kitagawa, G. (2010). *Introduction to Time Series Modeling*. New York: Chapman & Hall/CRC Press.

Kitagawa, G., Gersch, W. (1984). A smoothness priors-state space approach to the modeling of time series with trend and seasonality. *Journal of the American Statistical Association*, *79*(386), 378–389.

Kitagawa, G., Gersch, W. (1996). *Smoothness Priors Analysis of Time Series*. New York: Springer-Verlag.

Klaas, M., Briers, M., de Freitas, N., Doucet, A., Maskell, S., Lang, D. (2006). Fast Particle Smoothing: If I Had a Million Particles, International Conference on Machine Learning.

Matsumoto, M., Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, *8*, 3–30.

Nakano, S., Ueno, G., Higuchi, T. (2007). Merging particle filter for sequential data assimilation. *Nonlinear Processes in Geophysics*, *14*, 395–408.

Pitt, M., Shephard, N. (1999). Fitting via simulation: auxiliary particle filters. *Journal of American Statistical Association*, *94*(446), 590–599.

Sage, A. P., Mersa, J. L. (1971). *Estimation Theory with Applications to Communications and Control, McGraw-Hill Series in System Science*. New York: McGraw-Hill.

Solo, V. (1982). Smoothing estimation of stochastic process: Two-filter formulas. *IEEE Transactions on Automatic Control*, *27*(2), 473–476.

Prado, R., West, M. (2010). *Time Series Modeling, Computation, and Inference*. Florida: Chapman & Hall, CRC Press.

West, M., Harrison, J. (1989). *Bayesian Forecasting and Dynamic Models, Springer Series in statistics*. New York: Springer-Verlag.