Sequential Monte Carlo Methods in Practice

Article · December 1999

DOI: 10.1007/978-1-4757-3437-9_11 · Source: CiteSeer

3 authors:

Jun S Liu
Harvard University
513 PUBLICATIONS **30,933** CITATIONS
SEE PROFILE

Rong Chen
Rutgers, The State University of New Jersey
93 PUBLICATIONS **5,055** CITATIONS
SEE PROFILE

Tanya Logvinenko
Boston Children's Hospital
58 PUBLICATIONS **620** CITATIONS
SEE PROFILE

A Theoretical Framework for Sequential Importance Sampling and Resampling

Jun S. Liu[1]*, Rong Chen, and Tanya Logvinenko*

Abstract

Sequential importance sampling (SIS) was first developed in 1950s for molecular simulation. Although half a century has passed by, the SIS methodology remains one of the most versatile and powerful means for the simulation and optimization of chain polymers. In 1990s, statisticians reinvented the same methodology in a more general form, brought forth a number of enhancements, and applied it to a much broader spectrum of problems. In this article, along with a historical account of the methodology, we present a theoretical framework for the SIS with resampling. We emphasize the basic concept of a *weighted sample*, the fundamental idea of *sequential build-up*, the important technique of reweighting and resampling, and various other methods, e.g., the partial rejection control and marginalization, for improving a sequential importance sampler. To illustrate, we show how to analyze a state-space model with SIS and how to treat the conditional dynamic linear model with marginalization, resampling, and rejection control techniques. We report some simulation results for two-dimensional target tracking in clutter.

1 Introduction

Monte Carlo filters (MCF) can be loosely defined as a set of methods that use Monte Carlo simulation to solve *on-line* estimation and prediction problems in a dynamic system. Compared with traditional filtering methods, simple, flexible, yet powerful MCF techniques provide effective means to overcome computational difficulties in dealing with nonlinear dynamic models. One of the key elements among all the MCF techniques is a recursive use of the importance sampling principle, which leads to a more precise name, *sequential importance sampling* (SIS), for the techniques focused on by this article.

[1]Address for correspondence and reprints: Department of Statistics, Sequoia Hall, Stanford University, Stanford, CA 94305-4065

The earliest SIS method can be dated back to 1950s (Hammersley and Morton 1954, Rosenbluth and Rosenbluth 1955) when scientists were interested in the computer simulation of a long-chain polymer. The polymer model analyzed by these pioneers was the *self-avoid (random) walk* (SAW) on a d-dimensional lattice space (Kremer and Binder 1988). Figure 1 shows the realization of a SAW of length $N = 149$ (with 150 nodes/links), on a 2-D lattice space. Each node on the chain represents the position of a monomer.

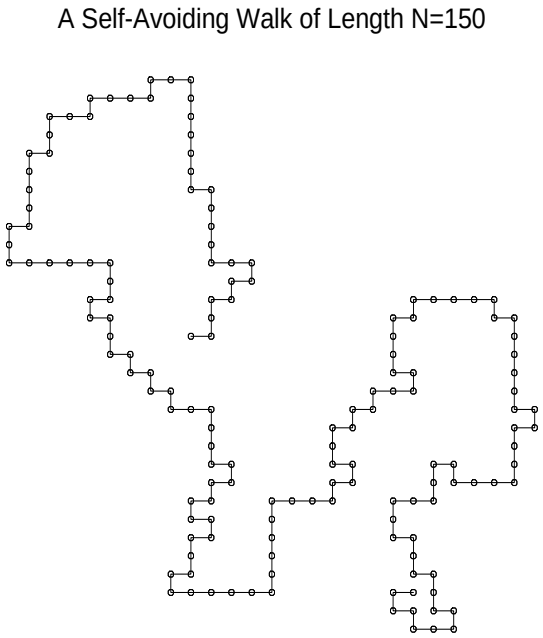

Figure 1: A SAW of length 150 on a 2-D lattice space. Without loss of generality, we always let the chain start at $(0, 0)$ and take its first step to $(0, 1)$.

For a given spatial configuration of this polymer, say, $\boldsymbol{x}_N = (x_0, \ldots, x_N)$, where x_j denotes the position of the jth monomer, one can compute the *potential energy* $U(\boldsymbol{x}_N)$ of this polymer according to natural laws in chemistry and physics. By the basic principle of statistical physics, the probability of seeing such a configuration (in nature) then follows the Boltzmann distribution

$$\pi_N(\boldsymbol{x}_N) = \frac{1}{Z_N} \exp\{-U(\boldsymbol{x}_N)/kT\},$$

where Z_N is the normalizing constant (also called the partition function). Temperature T and the Boltzmann constant k are assumed known. One is

often interested in estimating Z_N or some *averages*, e.g., the mean squared extension $E_\pi \|x_N - x_0\|^2$, regarding this polymer system.

The simple energy function used by (Hammersley and Morton 1954, Rosenbluth and Rosenbluth 1955) in their polymer studies is $U(\boldsymbol{x}_N) = 0$, i.e., $\pi_N(\boldsymbol{x}_N)$ is *uniform* on all allowable (self-avoiding) configurations. However, generating a long SAW uniformly is not as simple as it first appears. The most straightforward approach, i.e., the one that directly generates simple random walks and accepts only those that do not self-cross, becomes ineffective very rapidly (it will take about 50 million independently generated simple random walks in order to get a self-avoiding one of length 149). Another viable approach is to use the Metropolis algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller 1953), but the result was not very satisfactory for the SAW simulation because of the highly local-nature of the Metropolis move (Kremer and Binder 1988). The most effective Markov chain based Monte Carlo method is the "slithering snake" method (Kremer and Binder 1988) which treats the SAW chain as a snake moving on the lattice space. However, this approach can not be generalized to handle non-constant energy function $U(\boldsymbol{x}_N)$. The *growth Monte Carlo* method of Hammersley and Morton (1954) and Rosenbluth and Rosenbluth (1955) is rather simple. Instead of treating the whole configuration $\boldsymbol{x}_N$ directly, they propose to sequentially build up this configuration by adding one monomer a time, just like growing a molecule. To compensate for the bias induced by the growth Monte Carlo, an "importance weight" is computed and attached to the resulting polymer.

Although it has been nearly half a century, the growth Monte Carlo together with its variations and improvements (Kremer and Binder 1988, Grassberger 1997, Wall and Erpenbeck 1959a) remains as the most attractive and versatile method for polymer simulations. As we will explain in the next section, the key component of this method is the *sequential importance sampling principle*, which can be very generally and simply formulated and applied to different problems. Indeed, the MCF techniques emerged recently and independently from the fields of statistics and engineering follow exactly the same principle. The applications of these techniques include computer vision (Isard and Blake 1996), deconvolution of digital signals (Liu and Chen 1995), financial data modeling (Pitt and Shephard 1999), genetic linkage analysis (Irwing, Cox and Kong 1994), radar signal analysis (Gordon, Salmon and Smith 1993), medical diagnosis (Berzuini, Best, Gilks and Larizza 1997), and the standard Bayesian computations (Kong, Liu and Wong 1994, Liu 1996, MacEachern, Clyde and Liu 1999).

This article is organized as follows: Section 2 introduces the basic concept of *weighted samples* and the sequential importance sampling principle. Section 3 describes some major improvements of the SIS including resampling, rejection control and marginalization. Section 4 explains the Monte Carlo filters for state-space models; Section 5 shows a few applications of the techniques described in Sections 3 and 4, and Section 6 gives a brief discussion to

conclude the article.

2 Sequential importance sampling principle

2.1 Properly weighted sample

Suppose we are interested in Monte Carlo estimation of $\theta = E_\pi h(\boldsymbol{x})$ for some arbitrary function h under the target distribution $\pi(\boldsymbol{x})$.

Definition 1 *A set of weighted random samples* $\{(\boldsymbol{x}^{(j)}, w^{(j)})\}_{j=1}^m$ *is called proper with respect to* π *if for any square integrable function* $h(\cdot)$,

$$E[h(\boldsymbol{x}^{(j)})w^{(j)}] = cE_\pi h(\boldsymbol{x}), \quad \textit{for} \quad j = 1, \ldots, m,$$

where c *is a normalizing constant common to all the* m *samples.*

With this set of weighted samples, we can estimate θ as

$$\hat{\theta} = \frac{1}{W}\sum_{j=1}^{m} w^{(j)} h(\boldsymbol{x}^{(j)}), \tag{2.1}$$

where $W = \sum_{j=1}^m w^{(j)}$. For example, if the $\boldsymbol{x}^{(j)}$ are drawn from π directly, the set of $\{(\boldsymbol{x}^{(j)}, 1)\}$ is proper. In the context of importance sampling (Marshall 1956), we draw $\boldsymbol{x}^{(j)}$ from a trial distribution $q(\boldsymbol{x})$ and give it a weight $w^{(j)} = \pi(\boldsymbol{x}^{(j)})/q(\boldsymbol{x}^{(j)})$. Then $\{(\boldsymbol{x}^{(j)}, w^{(j)})\}$ is proper with respect to π. A main reason of using the renormalized estimator (2.1) in an importance sampling framework is that one does not need to know the normalizing constant for π (in Definition 1, the constant c does not need to be known).

A key concept in all Monte Carlo computations is that a probability distribution π, no matter how complicated it is, can always be *represented* by a discrete (Monte Carlo) sample from it. By "representation" we mean that any computation of expectations with respect to π can be replaced, to a acceptable degree of accuracy, by that with respect to the empirical distribution resulting from the discrete sample. This viewpoint is also central to the *multiple imputation* (Rubin 1987). When dealing with importance sampling, we see that π can be represented, at least conceptually, by any set of properly weighted samples. An importance issue in real Monte Carlo applications, however, is how to find a convenient and efficient discrete representation (i.e., the representation set can be easily generated and the estimation by using the set is accurate).

2.2 Sequential build-up

Now we go back to the problem of estimating $E_\pi h(\boldsymbol{x})$ for a target distribution π. For example, in the SAW case $\boldsymbol{x}$ is the configuration (positions of each

monomer) of the polymer of length N, π as the uniform distribution on all SAWs of length N, and $h(\boldsymbol{x})$ as the mean squared extension of the SAW. If $\boldsymbol{x}$ has a "natural decomposition,", e.g., $\boldsymbol{x} = (x_0, \ldots, x_N)$ as in the SAW case (N is suppressed from $\boldsymbol{x}$), we can imagine building up $\boldsymbol{x}$ sequentially by adding one monomer a time. According to the "telescope"-law of probability:

$$\pi(\boldsymbol{x}) = \pi(x_0)\pi(x_1 \mid x_0) \cdots \pi(x_N \mid x_0, \ldots, x_{N-1}),$$

we should draw x_0 according to its marginal distribution under π, and then add x_1 conditional on x_0 according to $\pi(x_0, x_1)$ (with other components, $x_2, \ldots, x_N$, integrated out), etc. However, this task is infeasible in most applications, although an interesting exception exists (Diaconis and Shahshahani 1987).

An effective way of sequentially building up $\boldsymbol{x}$ is to follow the basic idea of importance sampling. Instead of sampling x_0 from π, we may be able to draw it from some other distribution q_0 which is reasonably close to π, and then sample x_1 conditional on x_0 from $q_1(x_1|x_0)$, and so on. Finally, we obtain a sample of $\boldsymbol{x} = (x_0, \ldots, x_N)$ according to the sampling distribution

$$q(\boldsymbol{x}) = q_0(x_0)q_1(x_1 \mid x_0) \cdots q_N(x_N \mid x_0, \ldots, x_{N-1}).$$

To make this sample proper with respect to π, we need to give it a weight $w = \pi(\boldsymbol{x})/q(\boldsymbol{x})$. This "idea" offers only one more insight beyond the standard importance sampling (Marshall 1956): it is often fruitful to construct the trial distribution sequentially. Indeed, in many real problems we do have a hint on how to choose those q's. More formally, this "hint" can be formulated as a evolving *probabilistic dynamic system* (Liu and Chen 1998):

Definition 2 *A probabilistic dynamic system is a sequence of probability distributions defined on spaces with increasing dimensions: $\pi_t(\boldsymbol{x}_t)$ for $t = 0, 1, \ldots, N$, where $\boldsymbol{x}_t = (x_0, \ldots, x_t)$.*

Suppose this system is so defined that $\pi_t(\boldsymbol{x}_t)$ is the "best" distribution (closest to π in a certain sense) that we can come up with at time t and that the end distribution π_N is identical to the target distribution π, then we can let $q_t(x_t|\boldsymbol{x}_{t-1})$ be the same as or at least close to $\pi_t(x_t|\boldsymbol{x}_{t-1})$. In this scenario, we are able to update the importance weight recursively as

$$w_t = w_{t-1}\frac{\pi_t(\boldsymbol{x}_t)}{\pi_{t-1}(\boldsymbol{x}_{t-1})q_t(x_t \mid bx_{t-1})} = w_{t-1}\frac{\pi_t(\boldsymbol{x}_{t-1})}{\pi_{t-1}(\boldsymbol{x}_{t-1})}\frac{\pi_t(x_t \mid \boldsymbol{x}_{t-1})}{q_t(x_t \mid bx_{t-1})}. \qquad (2.2)$$

In practice, we only need to know (2.2) up to a normalizing constant, which is sufficient (and often more efficient) for us to estimate the quantity of interest (say, θ) by using (2.1). In order for the *sequential importance sampling* (SIS) method to work efficiently, we require that π_t evolves towards the target distribution π smoothly as t increases.

In simulating the SAWs, the growth Monte Carlo (Hammersley and Morton 1954, Rosenbluth and Rosenbluth 1955) chooses π_t as the uniform distribution on all the SAWs of length t and uses $q_t(x_t \mid \boldsymbol{x}_{t-1}) = \pi_t(x_t \mid \boldsymbol{x}_{t-1})$. If x_{t-1} has k_{t-1} unoccupied neighbors (i.e., those neighbors that have not been visited by $x_0, \ldots, x_{t-2}$), then $q_t(x_t \mid \boldsymbol{x}_{t-1})$ places the t-th monomer (i.e., x_t) uniformly at one of the k_{t-1} neighbors and that $\pi_t(\boldsymbol{x}_{t-1})$ differs from $\pi_{t-1}(\boldsymbol{x}_{t-1})$ by giving more probability to those $\boldsymbol{x}_{t-1}$ with more "open ends" (i.e., bigger k_{t-1}). The weight of the SAW such generated can be computed recursively as

$$w_t = w_{t-1} k_{t-1}. \tag{2.3}$$

A byproduct of the SIS is that it gives an immediate estimate, i.e., the sample mean of all the unnormalized weights, of the normalizing constant (partition function) Z_N (Rosenbluth and Rosenbluth 1955, Kong et al. 1994). As we can easily see in the SAW simulation, the unnormalized weight (2.3) satisfies the identity $Ew_N = Z_N$.

3 Operations for enhancing SIS

It has been noted early on by some researchers (Wall and Erpenbeck 1959a) that the simple application of the sequential build-up strategy is still not good enough to simulate very long SAWs. Specifically, the SAW simulated by the growth Monte Carlo can run into "cages" (i.e., the number of unoccupied neighbors of x_t is zero at a time $t < N$) more and more frequently as N increases. An improvement strategy was introduced soon after (Wall and Erpenbeck 1959a, Wall and Erpenbeck 1959b) in which one reuses those successfully simulated partial SAWs instead of restarting from scratch. Recently, one extra trick is added (Grassberger 1997) to "prune" away those partial SAWs that are associated with very small weights (i.e., are "doomed"). This set of strategies is, to a large extend, equivalent to the resampling approach employed either inexplicitly or explicitly by statisticians (Gordon et al. 1993, Liu and Chen 1995). Other possibilities including marginalization and rejection control for improving the performance of SIS has also been proposed (Doucet 1998, Liu and Chen 1998).

3.1 Reweighting, Resampling, and Reallocation

Suppose at time t we have a set of random samples $\mathcal{S}_t = \{(\boldsymbol{x}_t^{(j)}, w_t^{(j)})\}_{j=1}^m$ properly weighted with respect to π. By treating $\mathcal{S}_t$ as a discrete representation of π, we can generate another discrete representation as follows:

- For $j' = 1, \ldots, \tilde{m}$,
  - let $\tilde{\boldsymbol{x}}_t^{(j')}$ be $\boldsymbol{x}_t^{(j)}$ independently with probability proportional to $a^{(j)}$;

  - let the new weight associated with this sample be $\tilde{w}_t^{(j')} = w_t^{(j)}/a^{(j)}$

- Return the new representation $\tilde{\mathcal{S}}_t = \{(\tilde{\boldsymbol{x}}_t^{(j')}, \tilde{w}_t^{(j')})\}_{j'=1}^{\tilde{m}}$.

The new set $\tilde{\mathcal{S}}_t$ thus formed is also (approximately) proper with respect to π (Rubin 1987). It is not obvious, however, why resampling is useful. In fact, it does not help at all in a plain importance sampling framework. A few huristic supports (Liu and Chen 1995) are as follows: (a) resampling can prune away those hopelessly bad samples (by giving them a small $a^{(j)}$) and (b) resampling can produce multiple copies of those good samples (by giving them a big $a^{(j)}$) to help generating better future samples in the SIS setting. Consequently, in a probabilistic dynamic system with SIS, resampling helps one *steer* towards the right direction. In light of these arguments, one should choose $a^{(j)}$ as a monotone function of $w_t^{(j)}$.

If we let $a^{(j)} = w_t^{(j)}$, then the foregoing scheme is exactly the same as the one described earlier in the literature (Gordon et al. 1993, Liu and Chen 1995). But having an additional flexibility in choosing the sampling weights $a^{(j)}$ is rather intriguing and can be potentially very useful. For example, the $a^{(j)}$ can be chosen to reflect certain "future trend" (Pitt and Shephard 1999) or be chosen to balance between the need of diversity (i.e., having multiple distinct samples) and the need of focus (i.e., giving more presence to those samples with big weights). A generic choice is

$$a^{(j)} = \sqrt{w_t^{(j)}}. \tag{3.1}$$

as suggested by Professor W.H. Wong. More generally, we can let $a^{(j)}$ be $[w_t^{(j)}]^\alpha$, where α can vary according to the coefficient of variation of the w_t.

Another important point regarding the generation of $\tilde{\mathcal{S}}_t$ is that the *extra variation* due to resampling is unnecessary and unwanted (Liu and Chen 1995). Instead of resampling, a more efficient approach is the partial deterministic *reallocation*. For example, the following scheme can be implemented for generating $\tilde{\mathcal{S}}_t$ from $\mathcal{S}_t$ (we assume that $\sum_{j=1}^m a^{(j)} = m$):

- For $j = 1, \cdots, m$,
  1. For $a^{(j)} \geq 1$,
     - Retain $k_j = \lfloor a^{(j)} \rfloor$ copies of the sample $\boldsymbol{x}_t^{(j)}$;
     - Assign weight $\tilde{w}_t^{(j')} = w_t^{(j)}/k_j$ for each copy;
  2. For $a^{(j)} < 1$,
     - Remove the sample with probability $1 - a^{(j)}$;
     - Assign weight $\tilde{w}_t^{(j')} = w_t^{(j)}/a^{(j)}$ to the survived sample;
- Return the set $\tilde{\mathcal{S}}_t$ consisting of the new set of $\boldsymbol{x}_t$'s and $\tilde{w}_t$'s produced in the foregoing procedures.

The above scheme tends to slightly decrease the total sample size when applied. Alternatively, one can choose $k_j = \lfloor a^{(j)} \rfloor + 1$, which will tend to slightly increase the sample size. In order to maintain a fixed sample size, a residual-resampling strategy (Liu and Chen 1995) can be implemented.

3.2 Rejection control and partial rejection control

Another useful technique for rejuvenating a sequential importance sampler is the rejection control (RC) method (Liu, Chen and Wong 1998), which can be understood as a combination of the rejection method (von Neumann 1951) and importance sampling.

In the RC, one monitors the *coefficient of variation* of the importance weights for $\boldsymbol{x}_t^{(j)}$, defined as $cv_t^2 = \mathrm{var}(w_t^{(j)})/E^2(w_t^{(j)})$. This cv^2 can be used to derive a heuristic criterion, i.e., the *effective sample size*,

$$\mathrm{ESS}_t = \frac{m}{1 + cv_t^2},$$

which is heuristically understood as the equivalent number of iid samples at time t (Liu, 1996). Once ESS_t drops below a threshold, say $\mathrm{ESS}_t \leq \alpha_0 m$ $(0 < \alpha_0 < 1)$, we name this time a *(dynamic) "check-point."* The check-point sequence can also be prescribed in advance, which will be called static check-points.

When encountering a new check-point t_k (where k indicates that it is the k-th check-point), we compute a *control threshold* c_k, which may be a quantity given in advance, or the median or a quantile of the $w_{t_k}^{(j)}$. Then we check through every sample and decide whether to accept it according to probability $\min\{1, w_{t_k}^{(j)}/c_k\}$. In other words, those samples with weights greater than or equal to c_k are automatically accepted, whereas those with weights $< c_k$ are accepted with a probability. All accepted samples are given a new weight $w_{t_k}^{(j*)} = \max\{c_k, w_{t_k}^{(j)}\}$. All those rejected ones are restarted from $t = 0$ and re-checked at all previous check-points. It has been shown (Liu et al. 1998) that the RC operation is *proper* in the sense of Monte Carlo estimation and it always increases the ESS. A problem with the RC is that its computation cost increases rapidly as t increases, although the threshold c_k can be adjusted by the user to compromise between computation cost and the ESS.

To overcome computational difficulties associated with the RC, we can implement the *partial rejection control* (PRC), which combines the RC with resampling and can be understood as a *delayed resampling*. More precisely, if t is the k-th check-point (record as $t_k = t$), we do the following:

Partial Rejection Control

A. Compute the *control threshold* c_k. It can be either the median or a quantile (say, upper quartile) of the weights $w_t^{(1)}, \ldots, w_t^{(m)}$.

B. For $j = 1, \ldots, m$: accept the j-th sample with probability

$$p_j = \min\left\{1,\ \frac{w_t^{(j)}}{c_k}\right\}.$$

If accepted, its weight is updated to $\max\{w_t^{(j)}, c_k\}$.

C. For any rejected sample at time t_k, we go back to the previous check-point t_{k-1} to draw a previous *partial sample* $\boldsymbol{x}_{t_{k-1}}^{(j)}$. This partial sample is drawn from the set $\mathcal{S}_{t_{k-1}}$ with probability proportional to its weight at time t_{k-1} (i.e., $w_{t_{k-1}}^{(j)}$) and given an initial weight of $W_{t_{k-1}}/m$, where $W_{t_{k-1}} = \sum_{j=1}^{m} w_{t_{k-1}}^{(j)}$. Apply the SIS recursion to this partial sample till the current time t_k; conduct the same rejection control steps A and B as described above.

D. Repeat Step C until acceptance.

Remark 1: There are two major differences between the PRC and RC: (a) when rejection occurs, the PRC goes back to the previous check point, whereas the RC goes all the way back to time 0; (b) the PRC uses resampling whereas the RC does not.

Remark 2: One of the benefits of using RC is that it simultaneously generates almost-independent samples (some slight dependence can be caused by using an estimated threshold value c_k) and controls the coefficient of variation in the weight distribution. The PRC does not produce independent samples since it uses resampling to meet computational need. Both the RC and PRC can be seen as methods to bring future information into the generation of the current state variables. By giving a chance to go back (in time) to regenerate samples, PRC is especially advantageous in dealing with sudden changes in the dynamic system (such as outliers in the observation). In contrast, a simple resampling may destroy most of the "good" samples if an outlier presents, for the reason that it is not able to take advantage of future information.

3.3 Marginalization

By *marginalization* we mean that, whenever possible, one should analytically integrate out as many components from the system as possible, although this may sometimes introduce global dependency among the remaining variables. For example, by conditioning on a latent discrete vector I_t that indicates mixture component, one can easily integrate out the state variable x_t in a conditional Gaussian state-space model, but the remaining indicator vector I_t no longer has the Markovian structure (Chen and Liu 1999, Doucet 1998, Liu and Chen 1998). In the context of Markov chain Monte Carlo simulations,

this operation has also been shown to improve a Gibbs sampling algorithm (Carter and Kohn 1994, Liu, Wong and Kong 1994).

A general formulation of *marginalization* is as follows. Suppose each component x_s of $\boldsymbol{x}_t$ can also be decomposed "horizontally" as $x_s = (\xi_s, \nu_s)$, then we write $\boldsymbol{x}_t = (\boldsymbol{\xi}_t, \boldsymbol{\nu}_t)$. If we can obtain the marginal distributions

$$\pi_t(\boldsymbol{\xi}_t) = \int \pi_t(\boldsymbol{x}_t) d\boldsymbol{\nu}_t,$$

for the dynamic system, and have the corresponding *marginalized* sampling distribution,

$$q_t(\xi_t \mid \boldsymbol{\xi}_{t-1}) = \pi_t(\xi_t \mid \boldsymbol{\xi}_{t-1})$$

for the sequential build-up. Then the marginalized SIS performs better than the original one (MacEachern et al. 1999). This strategy is particularly useful for handling the linear state-space model with a mixture Gaussian error distribution and target tracking. Specifically, one can let ξ_t be the latent indicator for the Gaussian component from which the error term comes from. Thus, conditional on $\boldsymbol{\xi}_t$, the process is linear and Gaussian which can be dealt with by the Kalman filter (Kalman 1960).

4 Monte Carlo filter for state-space models

Monte Carlo methods for non-linear non-Gaussian filtering for the state-space models have received considerable attention recently. A key observation in this article is the *mathematical equivalence* between the structure of these MCF methods and the general SIS framework described in Section 2.

4.1 The general state-space model

Consider the *generalized state-space model*

$$\begin{aligned} x_t &\sim f_t(\cdot \mid \boldsymbol{x}_{t-1}), \\ y_t &\sim g_t(\cdot \mid x_t), \end{aligned}$$

where $\boldsymbol{x}_{t-1} = (x_0, \ldots, x_{t-1})$ is a *latent process*. When this process is a Markov chain, the model reduces to the ordinary state-space model (or hidden Markov Model). Let $\boldsymbol{y}_{t-1} = (y_1, \ldots, y_{t-1})$ be the observations available at time $t-1$. Suppose at time $t-1$ we have the posterior distribution of $\boldsymbol{x}_{t-1}$ as $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{t-1})$, then the predictive distribution for x_t is

$$p(x_t \mid \boldsymbol{y}_{t-1}) = \int f_t(x_t \mid \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{t-1}) d\boldsymbol{x}_{t-1}.$$

If we have a further observation y_t, the new posterior at time t becomes

$$p(\boldsymbol{x}_t \mid \boldsymbol{y}_t) \propto g_t(y_t \mid x_t) f_t(x_t \mid \boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{t-1}).$$

In many problems, of interest to researchers is the estimation of the "true signal characteristics," say $h(x_t)$, *on-line*. Clearly, the optimal solution (under the MSE criterion) of this estimation problem is the *Bayes estimator*, which has the form

$$\hat{h}_t = E[h(x_t) \mid \boldsymbol{y}_t] = \int h(x_t) p(\boldsymbol{x}_t \mid \boldsymbol{y}_t) d\boldsymbol{x}_t.$$

When the system is linear and Gaussian, this Bayes solution can be obtained recursively through the Kalman filter; when x_t only takes on a few discrete values, the solution can also be obtained by dynamic programming. Otherwise, the Bayes estimator can only be obtained via costly numerical approximations (often impractical for many real problems).

By denoting $\pi_t(\boldsymbol{x}_t) = p(\boldsymbol{x}_t \mid \boldsymbol{y}_t)$, we notice that the *posterior probability system* induced by the state-space model is just like the one we introduced in Section 2. If we forget about the special features of the state-space model or long-chain polymers, the system of interest here is identical in form to the polymer model. Therefore, the general SIS strategy, together with its improvements, can be applied. More interestingly, we do not have to take the dynamic system $\{\pi_t\}$ as the one directly suggested by the problem. Instead, we may choose to form a dynamic system with those $\pi_t(\boldsymbol{x}_t)$ reflecting more on the future trend. For example, we can do delayed estimation or some other forms of forward-looking when we simulate and estimate x_t.

Under the SIS framework, we proceed from a discrete representation $\mathcal{S}_{t-1} = \{(\boldsymbol{x}_{t-1}^{(j)}, w_{t-1}^{(j)})\}_{j=1}^m$ of the posterior distribution $p(\boldsymbol{x}_{t-1} \mid \boldsymbol{y}_{t-1})$ to the discrete representation $\mathcal{S}_t$ of $p(\boldsymbol{x}_t \mid \boldsymbol{y}_t)$ as follows: draw a sample $x_t^{(j)}$ from a $q_t(x_t \mid \boldsymbol{x}_{t-1}^{(j)})$ to form $\boldsymbol{x}_t^{(j)} = (\boldsymbol{x}_{t-1}^{(j)}, x_t^{(j)})$; attach to it a weight computed recursively according to (2.2):

$$w_t^{(j)} = w_{t-1}^{(j)} \times \frac{f_t(x_t^{(j)} \mid \boldsymbol{x}_{t-1}^{(j)})\ g_t(y_t \mid x_t^{(j)})}{q_t(x_t^{(j)} \mid \boldsymbol{x}_{t-1}^{(j)})}.$$

Then $\mathcal{S}_t = \{(\boldsymbol{x}_t^{(j)}, w_t^{(j)})\}_{j=1}^m$ is proper with respect to π_t. If the weights are too skewed (as measured by a low ESS), we can do resampling or rejection control or both as described in Section 2. This defines the recursive procedure of SIS with resampling for the state-space model.

4.2 Conditional dynamic linear model and the mixture Kalman filter

As a special state-space model, the conditional dynamic linear model (CDLM) can be defined as:

$$\text{State equation:} \qquad x_t = G_{t,i_1} x_{t-1} + \epsilon_{t,i_1}, \quad \text{if } I_{t,1} = i_1, \tag{4.1}$$

$$\text{Observation equation:} \qquad y_t = H_{t,i_2} x_t + e_{t,i_2}, \quad \text{if } I_{t,2} = i_2; \tag{4.2}$$

where $\epsilon_{t,j_1} \sim \mathcal{N}(\mathbf{0}, A_{i_1})$, $e_{t,i_2} \sim \mathcal{N}(\mathbf{0}, B_{i_2})$, and the indicator vector $I_t = (I_{t,1}, I_{t,2})$ is a discrete latent variable with a prior distribution $\pi_t(I_t)$. It is straightforward to generalize this prior to allow for Markov dependency among the I_t. The CDLM is a direct generalization of the dynamic linear model (DLM) (see West and Harrison, 1989). It possesses the simplicity of the DLM and is more flexible in dealing with outliers, sudden jumps, clutters, and other nonlinear features.

The SIS can be easily applied to directly treat the x_t's (who form a latent Markov chain). In fact, many previously available Monte Carlo filters for dealing with the CDLM model have been designed in this way (Avitzour 1995, Gordon et al. 1993, Kitagawa 1996). However, a more sophisticated algorithm can be derived by making use of the conditional Gaussian structure. That is, when conditioning on the values of $I_1, \ldots, I_t$, the CDLM defined by (4.1) and (4.2) becomes a DLM, and all the x_s, $s = 1, \ldots, t$, can be integrated out recursively by using a standard Kalman filter. When the SIS is applied to treat the marginal dynamic system for the indicators, we obtain the *mixture Kalman filter* (MKF) which is always more accurate than those Monte Carlo filters dealing with $\boldsymbol{x}_t$ directly (Chen and Liu 1999). Detailed formulas are given in the appendix.

Remark 3: Several approaches on designing efficient Markov chain Monte Carlo algorithms for this type of models have been proposed (Carlin, Polson and Stoffer 1992, Carter and Kohn 1994). Two key ideas employed in the design are the "collapsing" and "grouping" ((Liu et al. 1994)). In particular, Carter and Kohn (1994) used a discrete variable to indicate the mixing component for each observation and integrated out the state variable x_t in their MCMC iterations. Here we advocate a similar strategy, *marginalization,* for conducting sequential importance sampling.

Remark 4: Compared with those Monte Carlo filters applied to $\boldsymbol{x}_t$, which uses a discrete sample to approximate the posterior distribution of $\boldsymbol{x}_t$, MKF uses a mixture Gaussian distribution instead. Note that, under CDLM, the "true" target distribution is indeed a mixture Gaussian, although the number of the mixture components increases exponentially along with t. The advantage of the MKF can be easily seen as follows. Suppose the target distribution π is a mixture Gaussian with k components, with known μ_i, σ_i^2 and mixing probabilities p_i. The particle filter obtains m samples $x^{(j)}$ by first drawing an $I = i$ with probability p_i, then draw $x^{(j)}$ from $N(\mu_i, \sigma_i^2)$. In contrast, the MKF directly draws m samples of the indicator I. Using a Rao-Blackwellization argument, we can easily see that the latter is more efficient (MacEachern et al. 1999). In simulations, we have seen results in which an MKF with $m = 50$ provides a better estimation accuracy than a particle filter with $m > 1,000$. The downside of the MKF is that its implementation usually involves more analytical manipulations and more careful programming.

5 Some examples

5.1 A simple illustration

To illustrate how the method works, we consider the following simple state space model

$$\begin{aligned} x_t &= a_i x_{t-1} + \epsilon_t, \quad \text{with } \epsilon_t \sim N(0, \sigma_i^2) \text{ for } I_t = i; \\ y_t &= x_t + \eta_t \quad \text{with} \quad \eta_t \sim N(0, \sigma_\eta^2) \end{aligned}$$

Finally, we assume that $P(I_t = i \mid \boldsymbol{I}_{t-1}, \boldsymbol{x}_{t-1}) = P(I_t = i) = \pi_i$, which can be easily extended to deal with the Markovian case. For this example, we can carry out the computation easily (see appendix).

We first simulated a system with a_i=.9 (i=1,2), σ_1=.5, σ_2=1.5, π_1=.7, and σ_η=.3, and applied the MKF for on-line estimation. The numerical result was very satisfactory: the MKF estimated the state variable x_t very accurately even though several sudden jumps occurred in the system. With m=50 and without using PRC, we already produced a result better than that in Kitagawa (1996), who routinely used m=10,000 "particles." We also applied a PRC with $\alpha = 0.8$ (i.e., we conduct PRC whenever ESS_t drops below $0.8M$) together with the MKF. The MKF-PRC algorithm outperformed the plain MKF in terms of total mean-squared errors (MSE) (i.e., $\sum_{t=1}^{T} \|\hat{x}_t - x_t\|^2$, where $\hat{x}_t \approx E(x_t|\boldsymbol{y}_t)$) of on-line estimation. Figure 2 (a) presents a plot of the total MSE of the MKF estimates(small circles) versus those of the MKF-PRC estimates (small dots) in 100 repeated experiments. As we expected, such a difference disappeared as m increased to 1,000. For a total of 100 repeated experiments, the plain MKF only took 36 seconds of CPU time on a Sun Ultra 2 workstation, and the MKF-PRC only took 90 seconds.

To confirm the above observations, we simulated another dataset from the system with a different setting: a_j=.9 (j=1,2), σ_1=.2, σ_2=1.2, $\pi_1 \equiv P(J = 1)$=.9, and σ_η=.8. In this system, the noise level in the observation equation increased significantly. We found a similar improvement (more significant) of the MKF-PRC over the MKF (detail omitted). Figure 3 shows the comparison of the histograms of the logarithm of the importance weights under two schemes. As expected, the MKF-PRC had less variable importance weights.

5.2 Target tracking with MKF

Target tracking on a 2-D plane can be modeled as follows: let $s_t = (s_{t,1}, s_{t,2})^T$ be position vector, and let $v_t = (v_{t,1}, v_{t,2})^T$ be velocity vector. They are supposed to evolve in the following way

$$\begin{pmatrix} s_{t,1} \\ s_{t,2} \\ v_{t,1} \\ v_{t,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \nu & 0 \\ 0 & 0 & 0 & \nu^{-1} \end{pmatrix} \begin{pmatrix} s_{t-1,1} \\ s_{t-1,2} \\ v_{t-1,1} \\ v_{t-1,2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}$$

(a) (b)

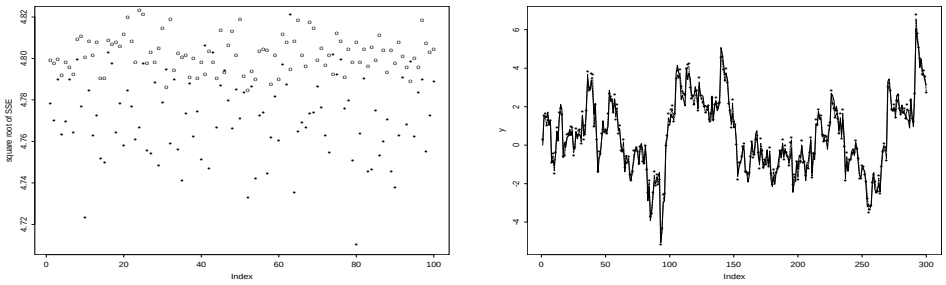

Figure 2: (a) The mean-squared-error of the estimates of x_t for using the MKF (circles) and the MKF-PRC (dots) in 100 repeated experiments. (b) The true x_t versus their on-line estimation from MKF-PRC: dots — observations (y_t); line — true values (x_t); dotted lines — estimates (i.e., $E(x_t \mid \boldsymbol{y}_t)$).

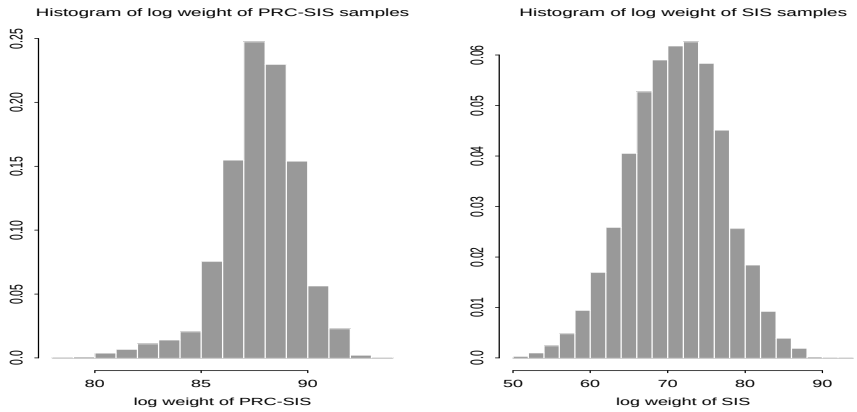

Figure 3: Histogram of the log-importance weights for MKF-PRC and the simple MKF respectively.

$$\begin{pmatrix} z_{t,1} \\ z_{t,2} \end{pmatrix} = \begin{pmatrix} s_{t,1} \\ s_{t,2} \end{pmatrix} + \begin{pmatrix} e_{t,1} \\ e_{t,2} \end{pmatrix}$$

The state equation innovation $\epsilon_t = (\epsilon_{t,1}, \epsilon_{t,2})^T$ is distributed as $\mathcal{N}(\mathbf{0}, \sigma_a^2 \boldsymbol{I})$. and the observation noise $e_t = (e_{t,1}, e_{t,2})^T$ follows $\mathcal{N}(\mathbf{0}, \sigma_b^2 \boldsymbol{I})$. The maneuvering variable ν can change with time which is assumed unknown to the observer. Thus, the state equation for this tracking system is assumed, instead, to follow

$$\begin{aligned} s_t &= s_{t-1} + v_{t-1} + \frac{1}{2}\epsilon_t, \\ v_t &= v_{t-1} + \epsilon_t, \end{aligned}$$

with $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 \boldsymbol{I})$. More generally, by writing $x_t = (s_{t,1}, s_{t,2}, v_{t,1}, v_{t,2})^T$, we use the relationship

$$x_t = Gx_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 A)$$

for tracking, where

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad A = \begin{pmatrix} \frac{1}{4} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{pmatrix}.$$

At a discrete time t, we observe a clutter of points, $y_t = \{y_{t,1}, \ldots, y_{t,k_t}\}$ in a 2-D detection region with area Δ, in which the number of false signals follow a spatial Poisson process with rate λ. The set y_t includes the true measurement $z_t = (z_{t,1}, z_{t,2})^T$ with probability p_d. Other y's are treated as uniform within the detection range. A latent indicator variable I_t can be introduced, where $I_t = i > 0$ indicates that $y_{t,i}$ corresponds to the true measurement z_t, and $I_t = 0$ means that z_t is not included in y_t.

Thus, we can write down the the joint distribution (approximately)

$$p(y_t \mid x_t, I_t = i) \propto \begin{cases} k_t(2\pi\sigma_b^2)^{-1} \exp\{-\frac{\|y_{t,i} - Ox_t)\|^2}{2\sigma_b^2}\}, & \text{if } i > 0, \\ \lambda, & \text{if } i = 0, \end{cases} \tag{5.1}$$

where the observation matrix is

$$O = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Clearly, this tracking model has the form of a CDLM and the procedure outlined in Section 4.2 can be applied to marginalize the state variable. Thus, the resulting SIS only needs to operate on the indicator variable I_t (see the Appendix for detailed formulas) and takes the form of a MKF.

We simulated this tracking system with the observation noise level σ_a=.1, state innovation level σ_b=.5, and the maneuvering parameter $\nu = 1$. The clutter of false signals is generated by a Poisson point process with rate λ=0.08. The total tracking time was set as $T = 150$. One hundred independent replications of the above simulation were carried out and the MKF formulation described in the Appendix was applied to track the target. Figure 4 shows one of the simulated data (the line corresponds to the trace of the target being tracked).

We applied four different SIS algorithms to each of the 100 simulated system. For ease of notations, we name the plain SIS without resampling as MKF; the one with systematic resampling every two steps as MKF-R2; the one with resampling every five steps as MKF-R5; and the MKF with

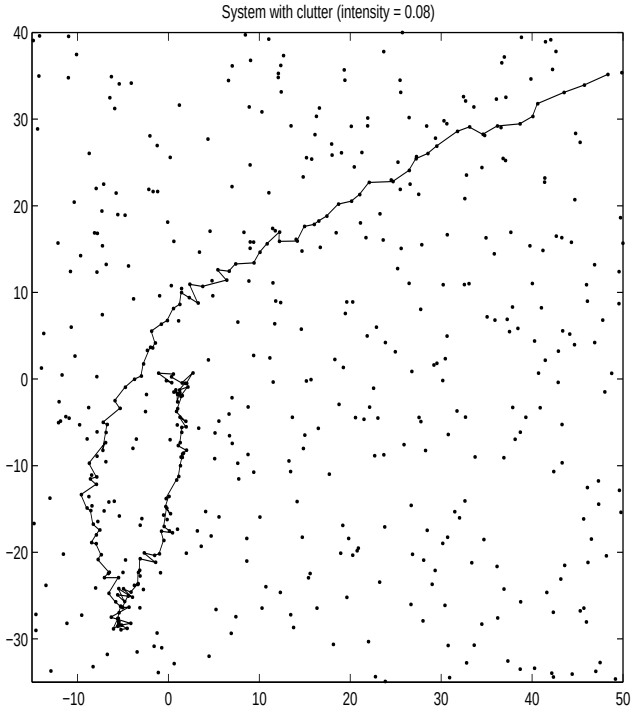

Figure 4: A simulation of the target tracking system. The dots connected by the line represent the true positions of the target at each time step; the dots elsewhere represent confusing objects.

partial rejection control a MKF-PRC. The MKF-PRC was operated with a sequence of dynamic check points corresponding to $\alpha_0 = 0.8$.

Our main interest in this simulation study was to compare the performances, in terms of both computing efficiency and tracking accuracy, of the four SIS strategies. For the first three methods, we let the number of "superparticles" (so named because each particle corresponds to one Kalman filter) m=50, and for MKF-PRC, we used $m = 30$. Since our algorithms were implemented in MATLAB for illustration purposes, substantial improvements in terms of computational efficiency are expected if these methods were programmed in more basic languages such as Fortran or C. In Table 1, we record the total number of lost targets (defined as the ones for which $\max \|\hat{s}_t - s_t\| > C$) in 100 replications and the total amount of computing time (on SGI Challenge IRIX workstation) for each method. One of our tracking results (the estimates of target positions) is shown in Figure 5.

The MKF-PRC and MKF-R out-performed the plain MKF not only in terms of number of "lost" targets but also in the mean-squared errors (MSE)

Methods	# Particles (m)	# Lost Targets	Computing Time
MKF	50	69 out of 100	39 min
MKF-R5	50	22 out of 100	74 min
MKF-R2	50	15 out of 100	87 min
MKF-PRC	30	10 out of 100	65 min

Table 1: A comparison between different sequential importance sampling strategies for the target tracking problem.

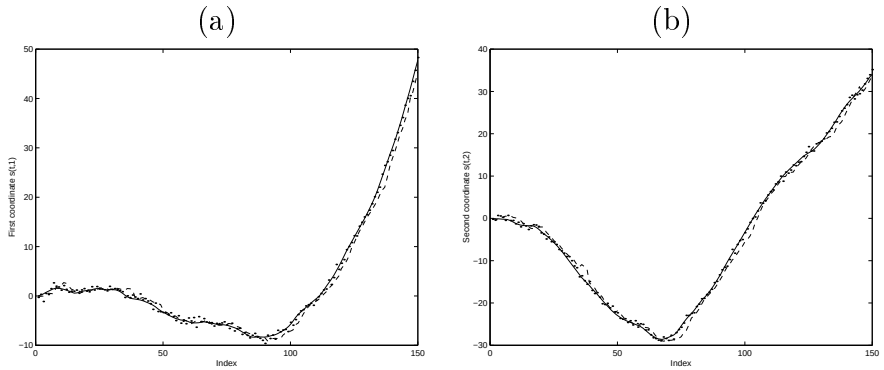

Figure 5: The true s_t versus its estimates from MKF-PRC. Dots — true signals (z_t) (not necessarily observed); line — true value of s_t; dashed lines — estimates (i.e., $E(s_t \mid \boldsymbol{y}_t)$). The first coordinate is plotted in (a) and the second in (b).

for those being correctly tracked (i.e., $\sum_{t=1}^{T} \|\hat{s}_t - s_t\|^2$, where $\hat{s}_t \approx E(s_t|\boldsymbol{y}_t)$) of on-line estimation. Figure 6 presents the total MSEs of the three MKF estimates, i.e., the plain MKF, MKF-R2, and MKF-PRC, in 100 repeated experiments (m=50 for the plain MKF, and m=30 for the MKF-PRC). It is seen that MKF-R and MSE-PRC performed better than the plain MKF even among those tracked targets.

6 Discussion

In this article, we provide a theoretical framework for the sequential importance sampling with resampling. We emphasize on the general applicability, which ranges from molecular simulation to signal processing, of SIS and its enhancements. We have also demonstrated by examples how the two strategies, namely, partial rejection control and marginalization, can be used to improve an SIS algorithm for on-line estimation with conditional dynamic linear models. Although the *marginalization* operation may not be applicable to all nonlinear dynamic models, it usually results in dramatic im-

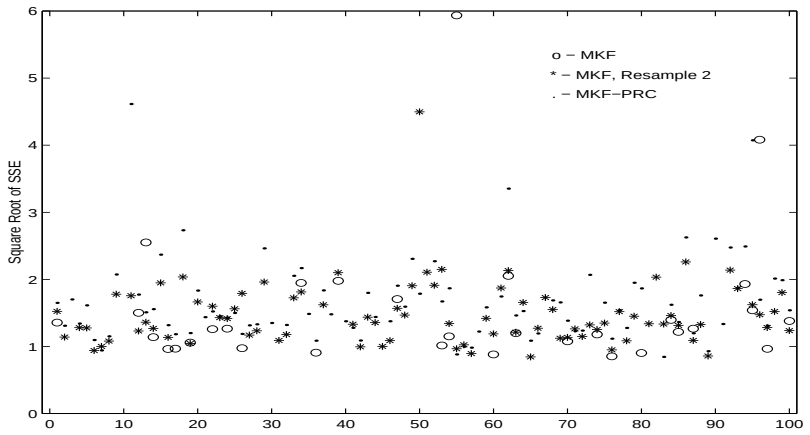

Figure 6: Comparisons between the MSEs of the MKF, MKF-R2, and MKF-PRC for the estimation of s_t in 100 repeated simulations.

provements over the plain SIS or particle filter whenever applicable. Various resampling/reallocation schemes and the *partial rejection control* method can be broadly applied to all dynamic systems under the SIS setting. They are essential for designing good Monte Carlo filters and are also useful for general Monte Carlo computation (not necessarily for on-line estimation problems). The main implication of our developments in this article is as follows: the SIS is a general and powerful platform for researchers to design various efficient sequential Monte Carlo algorithms for a large class of optimization, estimation, and prediction problems. We hope that our systematic account of the SIS is of interest to researchers in different fields.

Appendix

6.1 Detailed formulas for the MKF recursion in Section 4.2

Suppose the initial value x_0, the coefficients G_{t,i_1} and H_{t,i_2}, and the variances σ_i^2 and δ_i^2 are given. After observing y_1 at time $t = 1$, we have a posterior distribution

$$p(I_1 = (i_1, i_2), x_1 \mid y_1, x_0) \propto p(y_1 \mid x_1, I_{1,1} = i_1) p(x_1 \mid x_0, I_{1,2} = i_2).$$

which gives us a marginal distribution

$$p(I_1 = (i_1, i_2) \mid y_1, x_0) \propto \int p(y_1 \mid x_1, I_{1,1} = i_1) p(x_1 \mid x_0, I_{1,2} = i_2) dx_1. \quad (6.1)$$

Distribution (6.1) can be computed exactly since it only involves manipulating Gaussian density functions. Thus, we can draw m iid or correlated (if a

Markov chain sampler is used) samples, $I_1^{(1)}, \ldots, I_1^{(m)}$ from (6.1). For each $I_1^{(j)}$, we record the predictive mean $\mu_1^{(j)}$ and variance $V_1^{(j)}$ corresponding to the distribution $p(x_1 \mid y_1, x_0, I_1^{(m)})$, which is clearly Gaussian. Consequently, we have obtained a collection of m *Kalman filters*, $KF_1^{(1)}, \ldots, KF_1^{(m)}$, where $KF_1^{(j)} = (\mu_1^{(j)}, V_1^{(j)}, I_1^{(j)})$, at time $t = 1$. The weights $w_1^{(j)}$ associated with $KF_1^{(j)}$ are equal to one.

Suppose we have a collection of m Kalman filters, $KF_{t-1}^{(1)}, \ldots, KF_{t-1}^{(m)}$ at time $t-1$. Each $KF_{t-1}^{(j)}$ corresponds to $(\mu_{t-1}^{(j)}, V_{t-1}^{(j)}, \boldsymbol{I}_{t-1}^{(j)})$, where $\boldsymbol{I}_{t-1}^{(j)} = (I_1^{(j)}, \ldots, I_{t-1}^{(j)})$ is the imputed latent indicator vector up to time $t-1$. The $\mu_{t-1}^{(j)}$ and $V_{t-1}^{(j)}$ are the mean vector and covariance matrix for the j-th filter. Because the CDLM is reduced to a DLM when conditioning on $\boldsymbol{I}_{t-1}^{(j)}$, vector $(\mu_{t-1}^{(j)}, V_{t-1}^{(j)})$ is simply its sufficient statistics at time $t-1$, an end-product of a standard Kalman filter. Each filter is associated with a weight $w_{t-1}^{(j)}$. The following algorithm gives rules on how to update these KF's and their associated weights at time t.

The MKF Recursion: For $j = 1, \ldots, m$,

1. Obtain the predictive density for the new observation:

$$u_t^{(j)} = \sum_i \pi_t(I_t) \int p(y_t \mid x_t, I_t, KF_{t-1}^{(j)}) p(x_t \mid KF_{t-1}^{(j)}, I_t) dx_t,$$

   and update the weight for the jth filter as

$$w_t^{(j)} = w_{t-1}^{(j)} \times u_t^{(j)}, \quad j = 1, \ldots, m.$$

2. Impute the new indicator $I_t^{(j)}$ from its *a posteriori* distribution

$$p(I_t \mid \boldsymbol{y}_t, KF_{t-1}^{(j)}) \propto \pi_t(I_t) \int p(y_t \mid x_t, I_t, KF_{t-1}^{(j)}) p(x_t \mid KF_{t-1}^{(j)}, I_t) dx_t;$$

3. Update $KF_{t-1}^{(j)}$ to $KF_t^{(j)}$ by letting $\boldsymbol{I}_t^{(j)} = (\boldsymbol{I}_{t-1}^{(j)}, I_t^{(j)})$ and computing

$$\begin{aligned} \boldsymbol{\mu}_t^{(j)} &= E(x_t \mid KF_{t-1}^{(j)}, y_t, I_t^{(j)}) \\ V_t^{(j)} &= \mathrm{var}(x_t \mid KF_{t-1}^{(j)}, y_t, I_t^{(j)}). \end{aligned}$$

At this time, an on-line estimate of the state-variable x_t is

$$\hat{x}_t = \frac{1}{W_t} \sum_{m=1}^{M} w_t^{(m)} \boldsymbol{\mu}_t^{(m)},$$

where $W_t = w_t^{(1)} + \cdots + w_t^{(M)}$. Alternatively, we can also conduct a lag-k delayed estimate as

$$\hat{x}_{t-k} = \frac{1}{W_t} \sum_{m=1}^{M} w_t^{(m)} \boldsymbol{\mu}_{t-k}^{(m)}$$

A theory is given (Liu and Chen 1998) for this procedure is proper.

Detailed computations involved in a MKF approach is very closely related to those in a standard KF method. Take any filter at time $t-1$, say $KF_{t-1}^{(j)} = (\boldsymbol{\mu}_{t-1}^{(j)}, V_{t-1}^{(m)}, \boldsymbol{I}_{t-1}^{(j)})$. For convenience, we omit the superscript (j) in the following derivation. At time t, we define

$$\boldsymbol{\mu}_{+i_1} = G_{t,i_1}\boldsymbol{\mu}_t, \quad V_{+i_1} = G_{t,i_1}A_{i_1}G_{t,i_1}^T + V_{t-1}$$

Then by standard calculation, we have

$$\begin{aligned} p(x_t \mid I_{t,1} = i_1, KF_{t-1}) &= \mathcal{N}(x_t \mid \boldsymbol{\mu}_{+i_1}, V_{+i_1}) \\ p(y_t, x_t, I_t = (i_1,i_2) \mid KF_{t-1}) &= \mathcal{N}(y_t \mid H_{t,i_2}x_t, B_{i_2}) \times \mathcal{N}(x_t \mid \boldsymbol{\mu}_{+i_1}, V_{+i_1}) \times \pi_t(i_1,i_2), \end{aligned}$$

where $\mathcal{N}(x \mid \boldsymbol{\mu}, V)$ denotes a multivariate-Gaussian density. Hence, we can work out an explicit form of the required posterior distribution

$$p(x_t, I_t = (i_1,i_2) \mid y_t, KF_{t-1}) \propto C_t(i_1,i_2)\mathcal{N}(x_t \mid \boldsymbol{\mu}_t(i_1,i_2), V_t(i_1,i_2))$$

where

$$\begin{aligned} \boldsymbol{\mu}_t(i_1,i_2) &= V_t(i_1,i_2)(H_{t,i_2}^T B_{i_2}^{-1} y_t + V_{+i_1}^{-1}\boldsymbol{\mu}_{+i_1}) \\ V_t(i_1,i_2) &= (H_{t,i_2}B_{i_2}^{-1}H_{t,i_2}^T + V_{+i_1}^{-1})^{-1} \\ C_t(i_1,i_2) &= \pi_t(i_1,i_2)\exp\left\{-\frac{y_t^T B_{i_2}^{-1} y_t + \boldsymbol{\mu}_{+i_1}^T V_{+i_1}^{-1}\boldsymbol{\mu}_{+i_1} - (\boldsymbol{\mu}_t^T V_t^{-1}\boldsymbol{\mu}_t)|_{(i_1,i_2)}}{2}\right\} \end{aligned}$$

Thus, to update the filter, we first sample $I_t{=}(i_1,i_2)$ with probability proportional to $C_t(i_1,i_2)$, then update $\boldsymbol{\mu}_t^{(m)}$ as $\boldsymbol{\mu}_t(i_1,i_2)$ and $V_t^{(m)}$ as $V_t(i_1,i_2)$. The weight for this updated filter is $w_t = w_{t-1} \times \sum_{i_1,i_2} C_t(i_1,i_2)$.

6.2 Computational details for the example in Section 5.1

Take any filter $KF_{t-1} = (\mu_{t-1}, V_{t-1}, \boldsymbol{I}_{t-1})$ at time $t-1$. We define

$$\mu_{+i} = a_i\mu_{t-1}; \quad V_{+i} = a_i^2 V_{t-1} + \sigma_i^2.$$

Then before we observe y_t, the distribution of x_t, conditional on $J_t = i$, is $N(\mu_{+i}, V_{+i})$. Thus, after seeing y_t, the posterior distribution of (J_t, x_t) is

$$\begin{aligned} p(J_t = i, X_t \mid KF_{t-1}, y_t) &\propto \pi_i \times N(x_t \mid \mu_{+i}, V_{+i}) \times N(y_t \mid x_t\sigma_\eta^2) \\ &= C_t(i) \times N(x_t \mid \mu_t(i), V_t(i)) \end{aligned}$$

where

$$\begin{aligned} \mu_t(i) &= \left(\frac{\mu_{+i}}{V_{+i}} + \frac{y_t}{\sigma_\eta^2}\right)\left(\frac{1}{V_{+i}} + \frac{1}{\sigma_\eta^2}\right)^{-1} \\ V_t(i) &= \left(\frac{1}{V_{+i}} + \frac{1}{\sigma_\eta^2}\right)^{-1} \\ C_t(i) &= \frac{\pi_i}{\sqrt{V_{+i}}}\exp\left\{-\frac{(y_t - \mu_{+i})^2}{2(V_{+i} + \sigma_\eta^2)}\right\} \end{aligned}$$

Thus, when y_t is observed the filter KF_{t-1}'s weight is adjusted to $w_t = w_{t-1} \times u_t$, where $u_t = \sum_i C_t(i)$. The indicator $I_t = i$ is drawn with probability proportional to $C_t(i)$, which, together with the new mean and variance $\mu_t(i)$ and $V_t(i)$, forms the updated filter KF_t.

6.3 Computational detail for target tracking

Before observing y_t, however, the distribution of x_t can be written as

$$p(x_t \mid KF_{t-1}^{(m)}) = \mathcal{N}(x_t \mid G\boldsymbol{\mu}_{t-1}^{(m)},\ \sigma_a^2 A + GV_{t-1}^{(m)}G^T) \equiv \mathcal{N}(x_t \mid \boldsymbol{\mu}_+, V_+). \quad (6.2)$$

For simplicity, we omit the superscript (m) with an understanding that all the following computations are conditioned on the m-th filter $KF^{(m)}$.

Multiplying (6.2) with (5.1), we obtain the joint posterior of (x_t, I_t):

$$p(x_t, I_t = i \mid KF_{t-1}^{(m)}, y_t) \propto \begin{cases} \frac{p_d}{2\pi\sigma_b^2} \exp\left\{ -\frac{\|y_{t,i} - Ox_t\|^2}{2\sigma_b^2} - \frac{(x_t - \boldsymbol{\mu}_+)^T V_+^{-1}(x_t - \boldsymbol{\mu}_+)}{2} \right\} & \text{if } i > 0 \\ \lambda(1 - p_d) \exp\left\{ -\frac{(x_t - \boldsymbol{\mu}_+)^T V_+^{-1}(x_t - \boldsymbol{\mu}_+)}{2} \right\} & \text{if } i = 0. \end{cases} \quad (6.3)$$

By further simplifying (6.3), we obtain that

$$p(x_t, I_t = i \mid KF_{t-1}, y_t) \propto C_t(i)\mathcal{N}(x_t \mid \boldsymbol{\mu}^*(i), V^*(i))$$

where $C_t(0) = \lambda(1-p_d) \times (2\pi)^2 |V_+|^{1/2}$, $\boldsymbol{\mu}^*(0) = \boldsymbol{\mu}_+$, and $V^*(0) = V_+$; whereas for $i > 0$,

$$\begin{aligned} C_t(i) &= (2\pi)^2 |V^*(i)|^{1/2} \times \frac{p_d}{2\pi\sigma_b^2} \\ &\times \exp\left[-\frac{1}{2} \left\{ \frac{\|y_{t,i}\|^2}{\sigma_b^2} + \boldsymbol{\mu}_+^T V_+^{-1} \boldsymbol{\mu}_+ - \boldsymbol{\mu}^*(i)^T V^*(i)^{-1} \boldsymbol{\mu}^*(i) \right\} \right] \\ \boldsymbol{\mu}^*(i) &= V^*(i) \left(\frac{O^T y_{t,i}}{\sigma_b^2} + V_+^{-1} \boldsymbol{\mu}_+ \right) \\ V^*(i) &= \left(V_+^{-1} + \frac{O^T O}{\sigma_b^2} \right)^{-1} \end{aligned}$$

Therefore, the incremental weight of each $KF^{(m)}$ in the MKF is computed as $\sum_{i=0}^{k_t} C_t(i)$, and the imputation step can be achieved by drawing $p(I_t^{(m)} = i) \propto C_t(i)$. The m-th filter is updated to $KF_t^{(m)} = (\boldsymbol{I}_t^{(m)}, \boldsymbol{\mu}_t^{(m)}, V_t^{(m)})$ with $\boldsymbol{I}_t^{(m)} = (\boldsymbol{I}_{t-1}^{(m)}, I_t^{(m)})$, where $\boldsymbol{\mu}_t^{(m)} = \boldsymbol{\mu}^*(i)$ and $V_t^{(m)} = V^*(i)$ for $I_t^{(m)} = i$. This completes the MKF.

References

Avitzour, D. (1995). Stochastic simulation bayesian approach to multitarget tracking, *IEE Proceedings-Radar Sonar and Navigation* **142**(2): 41–44.

Berzuini, C., Best, N., Gilks, W. and Larizza, C. (1997). Dynamic conditional independence models and markov chain monte carlo methods, *Journal of the American Statistical Association* **92**: 1403–1412.

Carlin, B. P., Polson, N. G. and Stoffer, D. S. (1992). A monte-carlo approach to nonnormal and nonlinear state-space modeling, *Journal of the American Statistical Association* **87**(418): 493–500.

Carter, C. K. and Kohn, R. (1994). On gibbs sampling for state-space models, *Biometrika* **81**(3): 541–553.

Chen, R. and Liu, J. S. (1999). The mixture kalman filter, *Technical report*, Department of Statistics, Stanford University.

Diaconis, P. and Shahshahani, M. (1987). The subgroup algorithm for generating uniform random variables, *Probability in the Engineering and Informational Sciences* **1**: 15–32.

Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering, *Technical Report TR 310*, Department of Engineering, University of Cambridge.

Gordon, N., Salmon, D. and Smith, A. (1993). A novel approach to nonlinear/non gaussian bayesian state estimation., *IEE Proceedings on Radar and Signal Processing* **140**: 107–113.

Grassberger, P. (1997). Pruned-enriched rosenbluth method: simulations of q polymers of chain length up to 1000000., *Physical Review E* **56**: 3682–93.

Hammersley, J. and Morton, K. (1954). Poor man's monte carlo, *Journal of the Royal Statistical Society B* **16**: 23–38.

Irwing, M., Cox, N. and Kong, A. (1994). Sequential imputation for multilocus linkage analysis, *Proceedings of the National Academy of Science, USA* **91**: 11684–11688.

Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density, *European Conference on Computer Vision*, Cambridge, HK, pp. 343–356.

Kalman, R. (1960). A new approach to linear filtering and prediction problems, *Journal of Basic Engineering* **82**: 35–45.

Kitagawa, G. (1996). Monte carlo filter and smoother for non-gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics* **5**: 1–25.

Kong, A., Liu, J. S. and Wong, W. H. (1994). Sequential imputations and bayesian missing data problems, *Journal of the American Statistical Association* **89**(425): 278–288.

Kremer, K. and Binder, K. (1988). Monte carlo simulation of lattice models for macromolecules, *Computer Physics Reports*.

Liu, J. S. (1996). Nonparametric hierarchical bayes via sequential imputations, *Annals of Statistics* **24**(3): 911–930.

Liu, J. S. and Chen, R. (1995). Blind deconvolution via sequential imputations, *Journal of the American Statistical Association* **90**(430): 567–576.

Liu, J. S. and Chen, R. (1998). Sequential monte-carlo methods for dynamic-systems, *Journal of the American Statistical Association* **93**(443): 1032–1044.

Liu, J. S., Chen, R. and Wong, W. H. (1998). Rejection control and sequential importance sampling, *Journal of the American Statistical Association* **93**(443): 1022–1031.

Liu, J. S., Wong, W. H. and Kong, A. (1994). Covariance structure of the gibbs sampler with applications to the comparisons of estimators and augmentation schemes, *Biometrika* **81**(1): 27–40.

MacEachern, S. N., Clyde, M. and Liu, J. S. (1999). Sequential importance sampling for nonparametric bayes models: the next generation, *Canadian Journal of Statistics* **27**(2): 251–267.

Marshall, A. (1956). The use of multi-stage sampling schemes in monte carlo computations, *in* M. Meyer (ed.), *Symposium on Monte Carlo Methods*, Wiley, New York, pp. 123–140.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953). Equations of state calculations by fast computing machines, *Journal of Chemical Physics* **21**: 1087–1091.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters, *Journal of the American Statistical Association* **94**(446): 590–599.

Rosenbluth, M. and Rosenbluth, A. (1955). Monte carlo calculation of the average extension of molecular chains, *Journal of Chemical Physics* **23**: 356–359.

Rubin, D. (1987). A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the sir algorithm, *Journal of the American Statistical Association*.

von Neumann, J. (1951). Various techniques used in connection with random digits, *National Bureau of Standards Applied Mathematics Series* **12**: 36–38.

Wall, F. and Erpenbeck, J. (1959a). New method for the statistical computation of polymer dimensions., *Journal of Chemical Physics* **30**(3): 634–37.

Wall, F. and Erpenbeck, J. (1959b). Statistical computation of radii of gyration and mean internal dimensions of polymer molecules, *Journal of Chemical Physics* **30**: 637–640.