Bernoulli Race Particle Filters

Sebastian M Schmon **Arnaud Doucet** **George Deligiannidis**
Department of Statistics, University of Oxford

Abstract

When the weights in a particle filter are not available analytically, standard resampling methods cannot be employed. To circumvent this problem state-of-the-art algorithms replace the true weights with non-negative unbiased estimates. This algorithm is still valid but at the cost of higher variance of the resulting filtering estimates in comparison to a particle filter using the true weights. We propose here a novel algorithm that allows for resampling according to the true intractable weights when only an unbiased estimator of the weights is available. We demonstrate our algorithm on several examples.

Keywords: Bernoulli factory, random weight particle filter, unbiased estimation, Rao-Blackwellization

1 INTRODUCTION

Over the last 25 years particle filters have become a standard tool for optimal estimation in the context of general state space hidden Markov models (HMMs) with applications in ever more complex scenarios. HMMs are described by a latent (unobserved) process $(X_t)_{t\in\mathbb{N}}$ taking values in a space X and evolving according to the state transition density

$$X_t \mid (X_{t-1} = x) \sim f(\,\cdot \mid x)$$

for $t \geq 2$ and $X_1 \sim \mu(\cdot)$. The states can be observed through an observation process $(Y_t)_{t\in\mathbb{N}}$, taking values in a space Y with observation density

$$Y_t \mid (X_t = x) \sim g(\,\cdot \mid x).$$

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

Particle filters (PFs) sequentially approximate the joint distributions

$$\begin{aligned}\pi_t\left(x_{1:t}\right) &= p(x_{1:t} \mid y_{1:t}) \\ &= \frac{g(y_1 \mid x_1)\mu(x_1)\prod_{k=2}^t g(y_k \mid x_k) f(x_k \mid x_{k-1})}{p(y_{1:t})} \\ &= \frac{g(y_1 \mid x_1)\mu(x_1)\prod_{k=2}^t \varpi\left(x_k \mid y_k, x_{k-1}\right)}{p(y_{1:t})}\end{aligned}$$

on the space X^t at time t by evolving a set of samples, termed *particles*, through time. We use here the shorthand notation $x_{j:k} = (x_j, \ldots, x_k)$. The quantity $p(y_{1:t})$ denotes the marginal likelihood of the model

$$p(y_{1:t}) = \int g(y_1 \mid x_1)\mu(x_1) \prod_{k=2}^{t} \varpi\left(x_k \mid y_k, x_{k-1}\right) dx_{1:t} \tag{1}$$

which is usually intractable. The PF algorithm proceeds as follows. Given a set of particles $\{X_{t-1}^i, i = 1, \ldots, N\}$ at time $t-1$ the particles are propagated through $q(x_t \mid x_{t-1}, y_t)$. To adjust for the discrepancy between the distribution of the proposed states and $\pi_t\left(x_{1:t}\right)$ the particles are weighted by

$$w_t(x_{t-1}, x_t) = \frac{\varpi\left(x_t \mid y_t, x_{t-1}\right)}{q(x_t \mid x_{t-1}, y_t)}, \quad t \geq 2, \tag{2}$$

$$w_1(x_1) = \frac{g(y_1 \mid x_1)\mu(x_1)}{q(x_1 \mid y_1)}.$$

A subsequent resampling step according to these weights ensures that only promising particles survive. Here we consider only multinomial resampling, where we write $I_{1:N} \sim \mathsf{Mult}\left\{N; w_1, \ldots, w_N\right\}$ to denote the vector $I_{1:N}$ consisting of N samples from a multinomial distribution with probabilities proportional to $w_1, \ldots, w_N$. The algorithm is summarized in Algorithm 1.

In most applications, interest lies not in the distribution itself, but rather in the expectations

$$\mathcal{I}(h) = \int h(x_{1:t})\pi_t(x_{1:t})dx_{1:t} \tag{3}$$

for some test function $h\colon \mathsf{X}^t \to \mathbb{R}$. Applying a PF we

Algorithm 1 Particle filter with N particles

```
    At time t = 1
1:  Sample $\widetilde{X}_1^i \sim q(\cdot \mid y_1), \quad i = 1 \dots N$
2:  Compute weights
        $w_{1,i} = w_t(\widetilde{X}_1^i), \quad i = 1, \dots, N$
3:  $I_{1:N} \sim \mathsf{Mult}\{N; w_{1,1}, \dots, w_{1,N}\}$
4:  Set $X_1^i = \widetilde{X}_1^{I_i}, \quad i = 1 \dots, N$
    At times t ≥ 2
5:  Sample $\widetilde{X}_t^i \sim q(\cdot \mid X_{t-1}^i, y_t), \quad i = 1, \dots, N$
6:  Compute weights as in (2)
        $w_{t,i} = w_t(X_{t-1}^i, \widetilde{X}_t^i), \quad i = 1, \dots, N$
7:  $I_{1:N} \sim \mathsf{Mult}\{N; w_{t,1}, \dots, w_{t,N}\}$
8:  For $i = 1, \dots, N$ set
        $X_{1:t}^i = \left(X_{1:(t-1)}^{I_i}, \widetilde{X}_t^{I_i}\right).$
```

can estimate this integral, for a set of particle genealogies $\left\{X_{1:t}^i, i = 1, \dots, N\right\}$, by taking

$$\widehat{\mathcal{I}}_{t,\mathrm{PF}}(h) = \frac{1}{N} \sum_{i=1}^{N} h\left(X_{1:t}^i\right). \tag{4}$$

When the PF weights (2) are not available analytically, standard resampling routines—steps 3 and 7 in Algorithm 1—cannot be performed. However, in many cases one might be able to construct an unbiased estimator of the resampling weights. Del Moral et al. [6], Fearnhead et al. [9], Liu and Chen [13], and Rousset and Doucet [16] show that using random but unbiased non-negative weights still yields a valid algorithm. This can be easily seen by considering a standard particle filter on an extended space. Yet, this flexibility does not come without cost. Replacing the true weights with a noisy estimate increases the variance for Monte Carlo estimates of type (4).

Here we introduce a new resampling scheme that allows for multinomial resampling from the true intractable weights while just requiring access to non-negative unbiased estimates. Thus, any PF estimate of type (4) will have the same variance as if the true weights were known. This algorithm relies on an extension of recent work in Dughmi et al. [8] where the authors consider an unrelated problem.

In the supplementary material we collect the proofs for Propositions 3, 4 and 5, Theorem 6 and additional simulation studies. Code to reproduce our results is available online[1].

[1]URL: `http://www.stats.ox.ac.uk/~schmon/`.

2 PARTICLE FILTERS WITH INTRACTABLE WEIGHTS

Particle filters for state space models with intractable weights rely on the observation that replacing the true weights with a non-negative unbiased estimate is equivalent to a particle filter on an extended space. In this section we introduce some examples of models where the weights are indeed not available analytically and review briefly how the random weight particle filter (RWPF) can be applied in these instances.

2.1 Locally optimal proposal

Recall that in the setting of a PF for a state space model at time $t-1$ the proposal for t which leads to the minimum one step variance, termed the *locally optimal proposal* q^*, is

$$q^*(x_t \mid x_{t-1}, y_t) = \frac{g(y_t \mid x_t) f(x_t \mid x_{t-1})}{p(y_t \mid x_{t-1})},$$

see, e.g. Doucet et al. [7, Proposition 2]. Sampling from the locally optimal proposal is usually straightforward using a rejection sampler. The weights

$$\begin{aligned} w_t(x_{t-1}, x_t) &= p(y_t \mid x_{t-1}) \\ &= \int g(y_t \mid x_t) f(x_t \mid x_{t-1}) dx_t, \end{aligned} \tag{5}$$

however, are intractable if the integral on the right-hand side does not have an analytical expression, thus prohibiting an exact implementation of this algorithm. Our algorithm will enable us to resample according to these weights.

2.2 Partially observed diffusions

Most research on particle filters with intractable weights has been carried out in the setting of partially observed diffusions, see e.g. Fearnhead et al. [9], which we will describe here briefly. For simplicity, consider the univariate diffusion[2] process of the form

$$dX_t = a(X_t)dt + dB_t, \quad t \in [0, \mathcal{T}], \tag{6}$$

where $a\colon \mathbb{R} \to \mathbb{R}$ denotes the drift function and $\mathcal{T} \in \mathbb{R}$ is the time horizon. For a general diffusion constant speed as assumed in (6) can be obtained whenever the Lamperti transform is available. The diffusion is observed at discrete times $t_i, i = 1, \dots T$ with measurement error described by the density $g(y_{t_i} \mid x_{t_i})$. In this model the resampling weights are often intractable since for most diffusion processes the transition density $f_{\Delta t}(x_{t_i} \mid x_{t_{i-1}})$ over the interval with

[2]It is not necessary to restrict oneself to univariate diffusions, see e.g. Fearnhead et al. [9].

length $\Delta t = t_i - t_{i-1}$ is not available analytically. However, as shown in Beskos and Roberts [2], Dacunha-Castelle and Florens-Zmirou [4], and Rogers [15] the transition density over an interval of length Δt can be expressed as

$$f_{\Delta t}(y \mid x) = \varphi(y; x, \Delta t) \exp\left(A(y) - A(x)\right) \times \mathbb{E}\left[\exp\left(-\int_0^{\Delta t} \phi(W_s) ds\right)\right] \quad (7)$$

where $\varphi(\,\cdot\,; x, \Delta t)$ is the density of a Normal distribution with mean x and variance Δt, $(W_s)_{s\in[0,\Delta t]}$ denotes a Brownian bridge from x to y and $\phi(x) = \left(a^2(x) + a'(x)\right)/2$ for a function $A\colon \mathbb{R} \to \mathbb{R}$ with $A'(x) = a(x)$. The expectation on the right-hand side in (7) is with respect to the Brownian bridge and is usually intractable. Thus, for a particle filter to work in this example one either needs to implement a Bootstrap PF, which can sample the diffusion (6) exactly [see, e.g. 2], or one constructs an unbiased estimator of the expectation on the right-hand side to employ the RWPF. Note that for a function g and $U \sim \text{Unif}[0, t]$ we have

$$\mathbb{E}\left[\frac{g(W_U)}{\lambda} \mid W_s, 0 \le s \le t\right] = \int_0^t \frac{g(W_s)}{\lambda t} ds.$$

Using this relationship we can use debiasing schemes such as the Poisson estimator [1] to find a non-negative unbiased estimator of the expectation of the exponential.

2.3 Random weight particle filter

Here we briefly review the RWPF and compare its asymptotic variance to that of a PF with known weights. Assume one does not have access to the weight $\varpi\left(x_t \mid y_t, x_{t-1}\right)$ in (2) but only to some non-negative unbiased estimate $\hat{\varpi}(x_t \mid y_t, x_{t-1}, U_t)$, where U_t are auxiliary variables sampled from some density m. Then we carry out the multinomial resampling in Algorithm 1 by taking $I_{1:N} \sim \mathsf{Mult}\{N; \hat{w}_{t,1}, \ldots, \hat{w}_{t,N}\}$, where $\hat{w}_{t,k}$ is defined as in (2) with $\varpi\left(x_t \mid y_t, x_{t-1}\right)$ replaced by $\hat{\varpi}(x_t \mid y_t, x_{t-1}, U_t)$. This is equivalent to a standard particle filter on the extended space targeting at time t the distribution

$$\bar{\pi}_t\left(x_{1:t}, u_{1:t}\right) \propto \prod_{k=1}^t \hat{\varpi}(x_k \mid y_k, x_{k-1}, u_k) m\left(u_k\right)$$

which satisfies

$$\bar{\pi}_t\left(x_{1:t}, u_{1:t}\right) = \pi_t(x_{1:t}) \bar{\pi}_t(u_{1:t} \mid x_{1:t}) \quad (8)$$

with

$$\bar{\pi}_t(u_{1:t} \mid x_{1:t}) = \prod_{k=1}^t \frac{\hat{\varpi}(x_k \mid y_k, x_{k-1}, u_k) m\left(u_k\right)}{\varpi\left(x_k \mid y_k, x_{k-1}\right)}.$$

A PF targeting the sequence of distributions π_t directly can be interpreted as a Rao-Blackwellized PF [7] of the PF on the extended space. Hence, the following is a direct consequence of [3, Theorem 3].

Proposition 1. *For any sufficiently regular*[3] *real-valued test function h, the exact weight PF (EWPF) and RWPF estimators of $\mathcal{I}(h)$ defined in (3) both satisfy a $\sqrt{N}$-central limit theorem with asymptotic variances satisfying $\sigma^2_{EWPF,h} \le \sigma^2_{RWPF,h}$.*

3 BERNOULLI RACES

Assume now that the intractable weights can be written as follows

$$w_t(x_{t-1}, x_t) = \frac{g(y_t \mid x_t) f\left(x_t \mid x_{t-1}\right)}{q\left(x_t \mid x_{t-1}, y_t\right)} = c\left(x_{t-1}, x_t, y_t\right) b\left(x_{t-1}, x_t, y_t\right), \quad (9)$$

where $0 \le b(x, x', y) \le 1$ for all $x, x', y \in \mathsf{X}$, and that we are able to generate a coin flip Z with $\mathbb{P}(Z = 1) = b\left(x, x', y\right)$. We will assume that $c(x, x', y)$ in (9) is analytically available for any x, x', y. For brevity we will denote (9) as $w_i = c_i b_i$, for particles $i = 1, \ldots, N$, dropping for now the dependence on t.

The aim of this section is to develop an algorithm to perform multinomial sampling proportional to the weights w_i, that is, sample from discrete distributions of the form

$$p(i) = \frac{c_i b_i}{\sum_{k=1}^N c_k b_k}, \quad i = 1, \ldots N \quad (10)$$

where $c_1, \ldots, c_N$ denote fixed constants whereas the $b_1, \ldots, b_N$ are unknown probabilities. We assume we are able to sample coins $Z_i \sim \text{Ber}(b_i), i = 1, \ldots, N$. In practice, the $c_1, \ldots, c_N$ are selected to ensure that $b_1, \ldots, b_N$ take values between 0 and 1. The Bernoulli race algorithm for multinomial sampling proceeds by first proposing from the distribution

$$\mathbb{P}\left(I = i\right) = \frac{c_i}{\sum_{k=1}^N c_k} \quad (11)$$

and then sampling $Z_I \sim \text{Ber}(b_I)$. If $Z_I = 1$, return I, otherwise the algorithm restarts. Pseudo code describing this procedure is presented in Algorithm 2. If we take $c_j = 1$ for all $j = 1, \ldots, N$ we recover the original Bernoulli race algorithm from Dughmi et al. [8].

Proposition 2. *Algorithm 2 samples from the distribution*

$$p(i) = \mathbb{P}(I = i \mid Z_I = 1) = \frac{c_i b_i}{\sum_{k=1}^N c_k b_k}.$$

[3] We refer the reader to Chopin [3] for the mathematical details.

Algorithm 2 Bernoulli race

```
1: Draw $I \sim \mathsf{Mult}\{1; c_1, \ldots, c_N\}$
2: Draw $Z \sim b_I$
3: if $Z = 1$ then return $I = I$
4: else go back to line 1
5: end if
```

Proof. Note that the probability of sampling $I = i$ and accepting is

$$\mathbb{P}(I = i, Z_i = 1) = \frac{b_i c_i}{\sum_k c_k}.$$

It follows that observing $Z_I = 1$ has probability $\mathbb{P}(Z_I = 1) = \sum_k b_k c_k / \sum_k c_k$. Now for any $i = 1, \ldots N$

$$\begin{aligned} p(i) &= \mathbb{P}(I = i \mid Z_I = 1) \\ &= \frac{\mathbb{P}(I = i, Z_i = 1)}{\mathbb{P}(Z_I = 1)} \\ &= \frac{b_i c_i}{\sum_k c_k} \Big/ \frac{\sum_k b_k c_k}{\sum_k c_k} = \frac{b_i c_i}{\sum_k b_k c_k}. \end{aligned}$$ □

3.1 Efficient Implementation

This algorithm repeatedly proposes an a priori unknown amount of random variables from the multinomial distribution (11). Naïve implementations of multinomial sampling are of complexity $O(N)$ for one draw from the multinomial distribution. Standard resampling algorithms, however, can sample N random variables at cost $O(N)$ [see e.g. 11] but in this case the number of samples needs to be known beforehand. We show here that Bernoulli resampling can still be implemented with an average of $O(N)$ operations.

To ensure that the Bernoulli resampling is fast we need cheap samples from the multinomial distribution with weights proportional to $c_1, \ldots, c_N$. We can achieve this with the Alias method [17, 18] which requires $O(N)$ preprocessing after which we can sample with $O(1)$ complexity. Hence, the overall complexity depends on the number of calls to the Bernoulli factory per sample. Denote $C_{j,N}$ the number of coin flips that are required to accept a value for the jth sample, where $j = 1, \ldots, N$. The random variable $C_{j,N}$ follows a geometric distribution with success probability

$$\rho_N = \mathbb{P}(Z_I = 1) = \frac{\sum_{k=1}^N c_k b_k}{\sum_{k=1}^N c_k}. \tag{12}$$

The expected number of trials until a value is accepted is then

$$\mathbb{E}\left[C_{j,N}\right] = \frac{1}{\rho_N} \quad (j = 1, \ldots, N).$$

The complexity of the resampling algorithm depends on the values of the acceptance probabilities $b_1, \ldots, b_N$. For example, if all probabilities are identical, that is, $b_1 = \ldots = b_N = b$, we expect N/b coin flips, each of cost $O(1)$, which leads to overall order $O(N)$ complexity. In practice, however, the success probabilities of our Bernoulli factories are all different. Assuming all $b_j, j = 1, \ldots, N$ are non-zero, the expected algorithmic complexity per particle is bounded by the inverse of smallest and largest Bernoulli probabilities. Let $\underline{b} = \min\{b_1, \ldots, b_N\}$. Then,

$$\mathbb{E}[C_{j,N}] = \frac{\sum_k c_k}{\sum_k b_k c_k} \leq \frac{\sum_k c_k}{\underline{b} \sum_k c_k} = \frac{1}{\underline{b}}$$

and with $\bar{b} = \max\{b_1, \ldots, b_m\}$

$$\mathbb{E}[C_{j,N}] \geq \frac{1}{\bar{b}}.$$

In practice, the complexity of the Bernoulli sampling algorithm depends on the behavior of ρ_N as shown by the following central limit theorem.

Proposition 3. *Assume* $\lim_{N\to\infty} \rho_N =: \rho \in (0, 1)$. *Then we have the following central limit theorem for the average number of coin flips as* $N \to \infty$

$$\sqrt{N}\left(\frac{1}{N}\sum_{j=1}^N C_{j,N} - \frac{1}{\rho_N}\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{1-\rho}{\rho^2}\right),$$

where $\xrightarrow{d}$ *denotes convergence in distribution.*

In particular, Proposition 3 implies that the run-time concentrates around N/ρ_N with fluctuations of order $\sqrt{N}$. Thus, the order of complexity depends on the order of ρ_N and we have complexity $O(N)$ if

$$\limsup_{N\to\infty} \left| \frac{\sum_{k=1}^N c_k}{\sum_{k=1}^N c_k b_k} \right| < \infty. \tag{13}$$

As an example consider the case where $c_1 = \ldots = c_N = c$ are all identical. Then at time t

$$\rho_{t,N} = \frac{1}{N}\sum_{k=1}^N w_{t,k}. \tag{14}$$

An instance of this setting is the locally optimal proposal presented in Section 2.1. It is well known that as $N \to \infty$ (14) will converge towards $p(y_t \mid y_{1:t-1})$ and indeed we see that the algorithm's run-time will concentrate around a quantity of order N.

Remark. After the Alias table is constructed, the algorithm can be implemented in parallel. This can lead to considerable gains if the number of particles used in the particle filter is high. If $c_1 = \ldots = c_N = c$, or if the constants denote a distribution for which a table as in the Alias method is already implemented before program execution, the above algorithm can be implemented entirely in parallel. Murray et al. [14] consider the case of multinomial resampling using a rejection sampler with uniform proposal on the set $\{1, \dots, N\}$ with known weights and show that this algorithm has parallel complexity $O(\log N)$.

3.2 Estimating the Probability of Stopping

In our later applications we will be interested in evaluating the success probability

$$\rho_N = \frac{\sum_{k=1}^N c_k b_k}{\sum_{k=1}^N c_k}.$$

Assume that we sample N independent realizations from a multinomial distribution using the Bernoulli race algorithm described above and that $C_{1,N}, \dots, C_{N,N}$ are the geometric random variables that count the number of trials until the algorithm accepts a value and terminates. Then $\hat{\rho}_N^{\text{naive}} = 1/\bar{C}_N$, where $\bar{C}_N = \sum_{k=1}^N C_{k,N}/N$ is a consistent estimator of ρ since $\mathbb{E}(C_{i,N}) = 1/\rho_N$ for all $i = 1, \dots, N$ and therefore by a weak law of large numbers $\bar{C}_N \to 1/\rho$ in probability and $1/\bar{C}_N \to \rho$ in probability by the continuous mapping theorem. Unfortunately, the estimator $\hat{\rho}_N^{\text{naive}}$ is not unbiased which would be desirable. However, this can be remedied by constructing the minimum variance unbiased estimator [see 10, Definition 7.1.1] for a geometric distribution. This result is well known, but we repeat it here for convenience.

Proposition 4. *For $N \in \mathbb{N}$ let $C_{1,N}, \dots, C_{N,N}$ denote independent samples from a geometric distribution with success probability ρ_N, then the minimum variance unbiased estimator for ρ_N is*

$$\hat{\rho}_N^{\text{mvue}} = \frac{N-1}{\sum_{k=1}^N C_{k,N} - 1}. \tag{15}$$

In order to understand the asymptotic behavior of the estimator (15) we also provide the following central limit theorem.

Proposition 5. *For $N \in \mathbb{N}$ let $C_{1,N}, C_{2,N}, \dots$ denote independent samples from a geometric distribution with success probability ρ_N, then*

$$\sqrt{N}\left(\hat{\rho}_N^{\text{mvue}} - \rho_N\right) \xrightarrow{d} \mathcal{N}\left(0, (1-\rho)\,\rho^2\right).$$

4 BERNOULLI RACE PARTICLE FILTER

4.1 Algorithm Description

We now consider the application of the Bernoulli race algorithm to particle filter methods. The Bernoulli race can be employed as a multinomial resampling algorithm, $\mathsf{Mult}\{w_1, \dots, w_N\}$, in Algorithm 1. However, the clear advantage is that this algorithm can be implemented even if the true weights are not analytically available, but we do have access to non-negative unbiased estimates. A $[0,1]$-valued unbiased estimator $\hat{b}$ for b can be converted into an unbiased coin flip by noting that $\mathbb{P}\left(V \le \hat{b}\right) = b$, where $V \sim \text{Unif}[0,1]$ [see e.g. Lemma 2.1 in 12].

We will refer to such a particle filter as a Bernoulli race particle filter (BRPF).

4.2 Likelihood estimation

Even though the Bernoulli race resampling scheme enables us to resample according to the true weights, the normalizing constant or marginal likelihood (1) remains intractable. We show here how we can still obtain an unbiased estimator for the normalizing constant. We first recall that in particle filters an estimator of the normalizing constant is obtained by

$$\hat{p}(y_{1:T}) = \prod_{t=1}^T \hat{p}(y_t \mid y_{1:t-1}) = \prod_{t=1}^T \frac{1}{N} \sum_{k=1}^N w_{t,k}. \tag{16}$$

This estimator is well-known to be unbiased [see 5, Chapter 7]. If the weights are not available this estimator cannot be employed. Fortunately, the quantity (16) comes up naturally when running the BRPF. Recall that the probability for the Bernoulli race to stop at a given iteration, i.e. to accept a value, is

$$\mathbb{P}\left(C_{j,N} = 1\right) = \frac{\sum_{k=1}^N c_{t,k} b_{t,k}}{\sum_{k=1}^N c_{t,k}} = \frac{\frac{1}{N}\sum_{k=1}^N w_{t,k}}{\frac{1}{N}\sum_{k=1}^N c_{t,k}}.$$

Thus, conditional on the weights $w_{t,k}, k = 1, \dots, N$, an unbiased estimator of (16) is given by virtue of (15):

$$\hat{\rho}_{N,T} = \prod_{t=1}^T \frac{1}{N} \sum_{k=1}^N c_{t,k} \cdot \frac{N-1}{\sum_{k=1}^N C_{k,N} - 1}. \tag{17}$$

Theorem 6. *The estimator (17) is unbiased for $p(y_{1:T})$, i.e. $\mathbb{E}\left[\hat{\rho}_{N,T}\right] = p(y_{1:T})$.*

5 APPLICATIONS

5.1 Locally optimal proposal

We start with a Gaussian state space model. In linear Gaussian state space models the Kalman Filter can be used to analytically evolve the system through time. Nevertheless, the aim here is a proof of concept of the BRPF. We assume the latent variables follow the Markov chain

$$X_t = aX_{t-1} + V_t$$

where we take $a = 0.8$ and we observe these hidden variables through the observation equation

$$Y_t = X_t + W_t$$

with initialization $X_0 \sim \mathcal{N}(0,5)$ and $V_t \sim \mathcal{N}(0,5)$, $W_t \sim \mathcal{N}(0,5)$. In this particular instance the locally optimal proposal is available analytically, but in most practical scenarios this will not be the case. For this reason sampling from the locally optimal proposal is implemented using a rejection sampler that proposes from the state equation. Coin flips for the weights (5) can be obtained by sampling from the model $\xi_t \sim f(\cdot \mid x_{t-1})$ and computing

$$Z_t = 1\left\{U \leq \exp\left(-\frac{(y_t - \xi_t)^2}{10}\right)\right\}, \quad U \sim \text{Unif}[0,1].$$

This leads to the choice $c_{t,1} = \ldots = c_{t,N} = 1/\sqrt{10\pi}$ and

$$b_{t,k} = \int \exp\left(-\frac{(y_t - x)^2}{10}\right) f(x \mid x_{t-1}) dx_{t-1}.$$

Note that $c_{t,k}$ is defined such that

$$\sup_{x,x',y} b_t(x, x', y) = 1$$

Such a choice ensures that the acceptance probability in the Bernoulli race, Algorithm 2, is as large as possible. This can lead to a considerable speedup when using the Bernoulli resampling algorithm.

5.1.1 Complexity and Run-time

Figure 1 shows the run-time for RWPF and the BRPF for the Gaussian state space model. The BRPF clearly has linear complexity in the number of particles. In a sequential implementation the Bernoulli race (orange) performs worse than the classical resampling scheme (blue), but the difference vanishes when implementing a parallel version of the Bernoulli race (green, with 32 cores). This demonstrates the speedup due to parallel sampling of the coin flips. Since we need many Bernoulli random variables each of which is computationally cheap this algorithm lends itself to a implementation using GPUs to further improve the performance of the Bernoulli race.

5.1.2 Efficiency

As alluded to earlier, if Bernoulli resampling is performed, the variance for any Monte Carlo estimate (4) will be the same as if the true weights were known and one applies standard multinomial resampling. From Proposition 1 it follows that the asymptotic variance of any Monte Carlo estimate of type (4) will be smaller

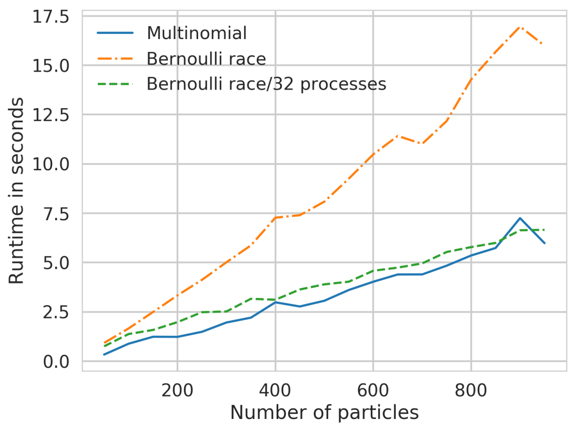

Figure 1: Comparison of the run-time for RWPF and BRPF for Gaussian state space model. For BRPF we show a sequential and a parallel implementation with 32 cores. For all algorithms we show wall-clock time for number of particles N.

when applying a BRPF over a RWPF. While the variance for functions (4) in the BRPF coincides with the standard particle filter we do not have access to the same estimator for the normalising constant and instead need to use the methods from Section 4.2.

For these reasons we will compare the performance of the BRPF with the RWPF for a set of different test functions $h \colon \mathsf{X}^T \to \mathbb{R}$ by comparing the variance of PF estimates (4). We then study the estimation of the normalizing constant separately. For the simulation we consider the functions

$$h_1(x_{1:T}) = \frac{1}{T}\sum_{t=1}^{T} x_t, \qquad h_2(x_{1:T}) = \|x_{1:T}\|_2,$$
$$h_3(x_{1:T}) = x_T, \qquad h_4(x_{1:T}) = (x_T - \bar{x}_T)^2,$$

where $\bar{x}_T = \sum_{i=1}^{N} x_T^i / N$. The PF approximations (4) using h_3 and h_4 are estimators for the quantities $\mu_T = \int x_T p(dx_T \mid y_{1:T})$ and $\int (x_T - \mu_T)^2 p(dx_T \mid y_{1:T})$.

All estimates are based on 100 runs of each particle filter using $N = 100$ particles. The results are collected in Table 1. We denote the standard deviation of the test functions under the BRPF as σ_{BRPF} and σ_{RWPF} for the RWPF. All measures indicate a reduction in the standard deviation. For the estimator of the function h_1, the standard deviation is reduced by 26% when compared to the RWPF.

We now investigate the estimates for the normalizing constant. Table 2 shows the standard deviation of the (log-)normalizing constant of a Gaussian state space

Table 1: For test functions $h_i, \dots i = 1, \dots, 4$ the standard deviation for RWPF and BRPF over 100 iterations.

Test function	σ_{RWPF}	σ_{BRPF}	$\sigma_{\text{BRPF}}/\sigma_{\text{RWPF}}$
h_1	0.17	0.12	0.74
h_2	0.93	0.78	0.84
h_3	0.19	0.19	0.96
h_4	0.42	0.40	0.94

Table 2: Comparison of the normalizing constant estimate for different implementations of the particle filter.

	$\text{sd}(\log \hat{p}(y_{1:T}))$
BRPF	0.55
RWPF	0.66

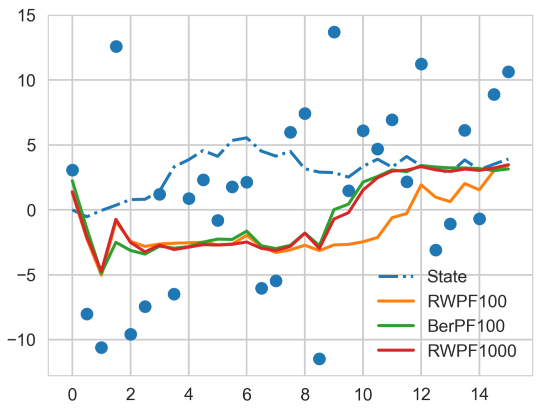

Figure 2: Tracking performance for different particle filters; RWPF with 100 particles (RWPF100) and 1000 particles (RWPF1000) as well as a particle filter using Bernoulli resampling with 100 particles (BerPF100).

model for $T = 50$ time steps. In this setting it is not obvious why the Bernoulli race resampling estimate should outperform the estimate provided by the RWPF. In our case however, we find that the BRPF performs better.

5.2 Partially Observed Diffusion

We use the sine diffusion, a commonly used example in the context of partially observed diffusions, see e.g. Fearnhead et al. [9], given by the stochastic differential equation (SDE)

$$dX_s = \sin(X_s)dt + dB_s, \quad s \in [0, 15].$$

Here, $(B_s)_{s\in[0,15]}$ denotes a Brownian motion and the drift function in (6) is $a(x) = \sin(x)$. Consequently, $A(x) = -\cos(x)$ and with $\phi(x) = \left(\sin(x)^2 + \cos(x)\right)/2$, the transition density is

$$f_{\Delta t}(x, y) = \varphi(y; x, \Delta t) \exp\left(-\cos(y) + \cos(x)\right) \times \mathbb{E}\left[\exp\left(-\int_0^{\Delta t} \phi(W_s)ds\right)\right]. \quad (18)$$

We observe the state of the SDE through zero mean Gaussian noise with standard deviation 5 yielding weights

$$w(x_{t-1}, x_t, y_t) = \frac{\varphi(y_t; x_t, 5^2) f_{\Delta t}(x_t \mid x_{t-1})}{q(x_t \mid x_{t-1}, y_t)},$$

As the proposal $q(\cdot \mid x_{t-1}, y_t)$ we take one step of the Euler–Maruyama scheme.

The weights are intractable because of the expectation on the right-hand side in (18). The construction of unbiased coin flips can often rely on similar techniques as the construction of unbiased estimators. We can construct an unbiased estimator for the transition density in the following way. Fix a Brownian bridge $(W_s)_{s\in[0,\Delta t]}$ starting at x and finishing at y. Sample $\kappa \sim \text{Pois}(\lambda \Delta t)$, $U_1 \dots, U_\kappa \sim U[0, \Delta t]$ then the Poisson estimator [1] is given by

$$\hat{P}(\kappa, U_1 \dots, U_\kappa) = \exp\left\{(\lambda - c)\, \Delta t\right\} \prod_{i=1}^{\kappa} \frac{\left\{c - \phi\left(W_{U_i}\right)\right\}}{\lambda},$$

where c is chosen such that the estimator is non-negative. The Poisson estimator is unbiased

$$\mathbb{E}\left[\hat{P}(\kappa, U_1 \dots, U_\kappa)\right] = \mathbb{E}\left[\exp\left(-\int_0^{\Delta t} \phi(W_s)ds\right)\right]. \quad (19)$$

For the purposes of implementing the BRPF, we need to construct a coin flip with success probability proportional to (19). We use the probability generating function approach which works analogously to the Poisson estimator. Sample $\kappa \sim \text{Pois}(\lambda \Delta t)$, $V_i \sim \text{Unif}[0, 1]$, $i = 1, \dots \kappa$, then

$$Z = \prod_{i=1}^{\kappa} 1\left\{V_i \le \frac{\left\{c - \phi\left(W_{U_i}\right)\right\}}{\lambda}\right\}$$

has success probability

$$\exp\left\{(c - \lambda)\Delta t\right\} \mathbb{E}\left[\exp\left(-\int_0^{\Delta t} \phi(W_s)ds\right)\right].$$

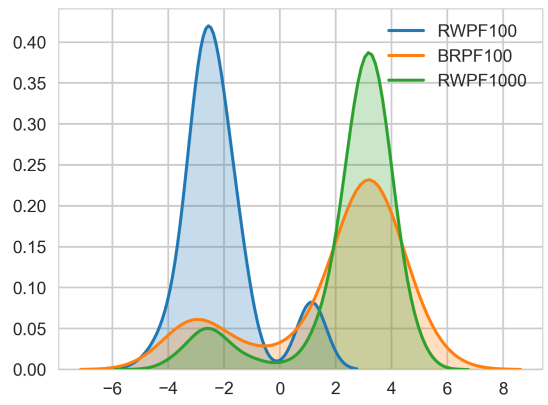

Figure 3: Kernel density estimate of $p(x_{10} \mid y_{1:10})$ based on particle approximations from a RWPF with 100 particles (RWPF) a BRPF with 100 particles (Bernoulli) and a RWPF with 1000 particles.

Hence, to implement the BRPF we can choose

$$c_{t,k} = \frac{\varphi(y_t; x_t, 5^2)\varphi(x_t; x_{t-1}, \Delta t)}{\varphi(y_t; x_{t-1} + \Delta t \sin(x_{t-1}), \Delta t)} \times \exp\left(-\cos(x_t) + \cos(x_{t-1}) - (c-\lambda)\Delta t\right)$$

$$b_{t,k} = \exp\{(c-\lambda)\Delta t\}\, \mathbb{E}\left[\exp\left(-\int_0^{\Delta t} \phi(W_s^k) ds\right)\right],$$

where $(W_s^k)_{s\in[0,\Delta t]}$ denotes a Brownian bridge from x_{t-1}^k to x_t^k.

5.2.1 Complexity and Run-time

As in the previous example, we observe the BRPF to be of order $O(N)$ and slower than the RWPF. However, implementing the Bernoulli race in parallel yields almost the same performance in terms of run-time. The details can be found in the supplementary material.

5.2.2 Efficiency

One run for the RWPF and the BRPF is shown in Figure 2, where we show the true state of the SDE as well as the noisy observations and the particle filter approximations. In this scenario precise state estimation is hampered by the high noise and resulting partial multimodality of the filtering distribution as shown in Figure 3. For $N = 100$, Figure 2 shows that the BRPF performs much better as it finds the true state earlier. The RWPF finds this trajectory only when the number of particles is increased (here we show also the case $N = 1000$). With the same test functions as above we compare both algorithms in Table 3. We observe gains for all functions, with the most significant gain for the conditional mean, h_3. Again, the Bernoulli race will ordinarily be slower, but most of the difference in run-time vanishes when the Bernoulli race is implemented in parallel. Further details are provided in the supplementary material.

Table 3: Comparison of the normalizing constant estimate for different implementations of the particle filter.

Test function	σ_{RWBF}	σ_{BRPF}	$\sigma_{\text{BRPF}}/\sigma_{\text{RWPF}}$
h_1	1.41	1.27	0.91
h_2	1.61	1.58	0.98
h_3	1.26	0.64	0.50
h_4	1.09	0.87	0.80

Table 4: Comparison of the normalizing constant estimate for different implementations of the particle filter.

	$\text{sd}(\log \hat{p}(y_{1:T}))$
RWPF	3.83
BRPF	3.11
Bootstrap	3.13

As a final test we use both particle filters for estimating the (log-)normalizing constant. The results are listed in Table 4. For comparison we also employ a bootstrap particle filter using the exact algorithm [2] to propose from the model. We observed this implementation to be slower than the other two. The BRPF estimate (17) outperforms the RWPF and the bootstrap PF.

6 Conclusion

We have introduced the idea of Bernoulli races to resampling in particle filtering, utilizing the equivalence of unbiased estimation and the construction of unbiased coin-flips. This algorithm provides the first resampling scheme to allow for exact implementation of multinomial resampling when the weights are intractable, but unbiased estimates are available. We have shown that our algorithm has a complexity of order N, like standard multinomial resampling, and demonstrated its advantages over alternative methods in a variety of settings. In doing so we have focused our attention to the resampling step. Further gains using our algorithm could be obtained by considering auxiliary particle filters and to resample only if the effective sample size drops beyond a certain threshold. We expect both to positively impact the performance of the BRPF.

References

[1] A. Beskos, P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts. "Exact and Efficient Likelihood inference for Discretely Observed Diffusion Processes (with discussion)." *Journal of the Royal Statistical Society Series B (Statistical Methodology)* 68.3 (2006), pp. 333–382.

[2] A. Beskos and G. O. Roberts. "Exact simulation of diffusions". *The Annals of Applied Probability* 15.4 (2005), pp. 2422–2444.

[3] N. Chopin. "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference". *The Annals of Statistics* 32.6 (2004), pp. 2385–2411.

[4] D. Dacunha-Castelle and D. Florens-Zmirou. "Estimation of the coefficients of a diffusion from discrete observations". *Stochastics: An International Journal of Probability and Stochastic Processes* 19.4 (1986), pp. 263–284.

[5] P. Del Moral. *Feynman-Kac Formulae.* Springer, 2004, pp. 47–93.

[6] P. Del Moral, A. Doucet, and A. Jasra. "Sequential Monte Carlo for Bayesian Computation". *Bayesian Statistics.* Ed. by J. M. Bernardo et al. 8. 2007, pp. 1–34.

[7] A. Doucet, S. Godsill, and C. Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". *Statistics and computing* 10.3 (2000), pp. 197–208.

[8] S. Dughmi, J. D. Hartline, R. Kleinberg, and R. Niazadeh. "Bernoulli factories and black-box reductions in mechanism design". *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing.* ACM. 2017, pp. 158–169.

[9] P. Fearnhead, O. Papaspiliopoulos, and G. O. Roberts. "Particle filters for partially observed diffusions". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70.4 (2008), pp. 755–777.

[10] R. V. Hogg, J. W. McVean, and A. T. Craig. *Introduction to Mathematical Statistics.* 6th edition. Pearson Education, Upper Saddle River, New Jersey, 2005.

[11] J. D. Hol, T. B. Schon, and F. Gustafsson. "On resampling algorithms for particle filters". *IEEE Nonlinear Statistical Signal Processing Workshop.* 2006, pp. 79–82.

[12] K. Łatuszyński, I. Kosmidis, O. Papaspiliopoulos, and G. O. Roberts. "Simulating events of unknown probabilities via reverse time martingales". *Random Structures & Algorithms* 38.4 (2011), pp. 441–452.

[13] J. S. Liu and R. Chen. "Sequential Monte Carlo methods for dynamic systems". *Journal of the American Statistical Association* 93.443 (1998), pp. 1032–1044.

[14] L. M. Murray, A. Lee, and P. E. Jacob. "Parallel resampling in the particle filter". *Journal of Computational and Graphical Statistics* 25.3 (2016), pp. 789–805.

[15] L. Rogers. "Smooth Transition Densities for One-Dimensional Diffusions". *Bulletin of the London Mathematical Society* 17.2 (1985), pp. 157–161.

[16] M. Rousset and A. Doucet. "Discussion of Beskos et al.: 'Exact and Efficient Likelihood inference for Discretely Observed Diffusion Processes'". *Journal of the Royal Statistical Society Series B (Statistical Methodology)* 68 (2006), pp. 374–375.

[17] A. J. Walker. "An efficient method for generating discrete random variables with general distributions". *ACM Transactions on Mathematical Software (TOMS)* 3.3 (1977), pp. 253–256.

[18] A. J. Walker. "New fast method for generating discrete random numbers with arbitrary frequency distributions". *Electronics Letters* 10.8 (1974), pp. 127–128.