Statistica Sinica Preprint No: SS-2022-0256	
Title	Differentiable Particle Filters with Smoothly Jittered Resampling
Manuscript ID	SS-2022-0256
URL	http://www.stat.sinica.edu.tw/statistica/
DOI	10.5705/ss.202022.0256
Complete List of Authors	Yichao Li, Wenshuo Wang, Ke Deng and Jun S. Liu
Corresponding Authors	Yichao Li
E-mails	ycli@mail.tsinghua.edu.cn
Notice: Accepted version subject to English editing.	

Statistica Sinica

DIFFERENTIABLE PARTICLE FILTERS WITH SMOOTHLY JITTERED RESAMPLING

Yichao Li, Wenshuo Wang, Ke Deng and Jun S. Liu

Tsinghua University and Harvard University

Abstract: Particle filters, also known as sequential Monte Carlo, are a powerful computational tool for making inference with dynamical systems. In particular, it is widely used in state space models to estimate the likelihood function. However, estimating the gradient of the likelihood function is hard with sequential Monte Carlo, partially because the commonly used reparametrization trick is not applicable due to the discrete nature of the resampling step. To address this problem, we propose utilizing the smoothly jittered particle filter, which smooths the discrete resampling by adding noise to the resampled particles. We show that when the noise level is chosen correctly, no additional asymptotic error is introduced to the resampling step. We support our method with simulations.

Key words and phrases: Reparametrization trick; Resampling; Sequential Monte Carlo; State space models.

Li and Wang contributed equally to this work; Deng and Liu co-supervised this study.

1. Introduction

Computing or estimating the likelihood function and its gradient for a statistical model with hidden variables is a challenge with applications in many areas of statistics and machine learning (Andrieu et al., 2005; Kingma and Welling, 2014; Le et al., 2017; Mohamed et al., 2020). In a general form, such a likelihood function can be written as

$$L(\theta \mid y) = \int p(y, z; \theta) dz, \tag{1.1}$$

where $p(y, z; \theta)$ is the joint likelihood of the observed variables y and the unobserved latent variables z with θ as parameters. In practice, the integral is usually estimated by a Monte Carlo method, typically a form of importance sampling:

$$\hat{L}(\theta \mid y) = \frac{1}{n} \sum_{i=1}^{n} \frac{p(y, z_i; \theta)}{q(z_i; \theta)} \quad \text{with} \quad z_i \sim q, \tag{1.2}$$

where q is the sampling distribution of z_i's which could potentially depend on θ. The gradient of the likelihood function can be estimated by differentiating $\hat{L}(\theta \mid y)$ with respect to θ, provided that the sampling distribution q is differentiable (in practice, usually the target is the log-likelihood instead). However, in some high-dimensional cases, q may not be well behaved and this naive method could result in considerable instability. A notable example is the inference of state space models, where particle filters, also known

as sequential Monte Carlo (SMC), are usually used to construct q (Liu and Chen, 1998; Doucet et al., 2001).

A state space model consists of a Markovian system of hidden states and a probabilistic observation model. The q function provided by SMC is high-dimensional and often incurs a high variance in estimating the derivative of the likelihood function (Naesseth et al., 2018). Recently, there has been a line of work on differentiable particle filters (DPF) (Naesseth et al., 2018; Zhu et al., 2020; Corenflos et al., 2021) that addresses this difficulty and enables end-to-end training in state space models. Research on DPF addresses the challenge of using particle filters to estimate the gradient of an estimating equation, usually the log-likelihood function, with respect to the estimand, where directly applying backpropagation through the proposal function q is generally highly unstable due to the discreteness of the resampling step. As a result, DPF can be used to infer the parameters of a complex model using gradient-based methods. Although gradient-based methods are not theoretically guaranteed to converge, they have garnered growing attention due to their promising performance in practice and inherent regularization ability (Ali et al., 2020).

The reparametrization trick is a method used to provide low-variance gradient estimate of an expectation, such as $\nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z;\theta)]$, where θ ap-

pears in the underlying density function p (see Section 4.2 for a detailed description). However, the reparametrization trick only works for continuous distributions. As briefly mentioned above, the main challenge of DPF arises from the resampling step, which introduces discreteness and prevents the deployment of the reparametrization trick. While this issue is created by resampling, the resampling step is necessary for coping with the weight degeneracy problem that often appears in particle filters. If no resampling is performed, most of the particle weights will soon converge to zero and only a tiny number of particles are actually used for approximating the posterior. A main intuition behind resampling is that particles with small weights are less informative and thus discarded so as to save computational resources to explore regions that may be more promising for the future (Liu and Chen, 1995).

Numerous approaches have been proposed to resample from a set of weighted particles, including the bootstrap resampling or multinomial resampling (Gordon et al., 1993), residual resampling (Liu and Chen, 1998), stratified resampling (Kitagawa, 1996), optimal resampling (Fearnhead and Clifford, 2003), and so on. Most, if not all, of the commonly used resampling methods are discrete in nature, preventing a direct use of the reparametrization trick. Some researchers have investigated jittered resampling, which

involves adding a small amount of noise to the resampling step to alleviate the particle degeneracy issue(Shephard and Flury, 2009). Although jittered resampling is not popularly deployed in a regular SMC algorithm, perhaps because it does not improve the performance of SMC for simple posterior estimation, we demonstrate in this paper how to use it as a building block to facilitate the reparametrization trick for gradient estimation with SMC.

Naesseth et al. (2018) point out that in the absence of the reparametrization trick, the gradient estimate is highly unstable, and propose to discard the term that corresponds to resampling and thus cannot exploit the reparametrization trick. Clearly, this method produces non-negligible bias if resampling is frequently implemented . Zhu et al. (2020) propose to replace the traditional resampling mechanisms with a continuous transformation called particle transformer to enable the reparametrization trick. In a particle transformer, the resampling procedure is realized through a neural network (NN) with additional parameters, where the input layer is the weighted particles and the output layer is the resampled unweighted particles. Corenflos et al. (2021) apply ensemble transform (Reich, 2013) to the resampling step, which is accomplished through a linear transformation (interpolation). Specifically, the linear transformation is determined by an entropy penalized optimal transport measure. Compared to the par-

ticle transformer, the ensemble transform can avoid training a neural network. However, neither the particle transformer nor the ensemble transform necessarily preserve well the particles' empirical distribution after resampling. As will be detailed in Section 2, while the NN-based transformer lacks theoretical guarantees and is computationally expensive, the linear transformation may significantly alter the empirical particle distribution. In this article, we develop a deployment of the "smoothly jittered particle filter," which replaces standard discrete resampling mechanisms with a kernel smoothed resampling procedure, to enable the reparametrization trick. We also present that specially designed kernel functions can enable efficient computation. Our experiments show that the proposed method outperforms existing techniques in various model settings.

The rest of the paper is organized as follows. Section 2 reviews some related work. Section 3 offers a comprehensive yet concise overview of the state space model and the SMC framework under a unified notation system. Section 4 is dedicated to the gradient estimation problem, in which we first introduce the general gradient estimation problem and two main approaches to solve it, and then show the obstacles in realizing the approaches under the SMC framework. Section 5 proposes the reparametrized resampling via jittering and presents a fast computation algorithm to solve the

differentiability problem of SMC discussed in Section 4. Section 6 provides numerical simulation results to support the proposed method. Section 7 concludes the article and discusses some further extensions.

2. Related work

Pitt (2002) presents an early work on smooth likelihood estimators, suggesting a method to sample from a linearly interpolated empirical CDF of particles in one-dimensional cases. Building on this concept, Lee (2008) extends the approach by utilizing tree-based partitions to construct piece-wise continuous estimators.

Ścibior and Wood (2021) suggest that the stop-gradient operator can be used to customize which terms to ignore in gradient calculation. Maddison et al. (2016), Le et al. (2017) and Corenflos et al. (2021) discussed the gradient estimation bias when the resampling term is ignored. In particular, Corenflos et al. (2021, Proposition 4.1) showed that the bias is zero when $p((x^{(t-1)}, x^{(t)}) \mid y_{1:t}) = p((x^{(t-1)}, x^{(t)}) \mid y_{1:T})$.

The ensemble transform (Corenflos et al., 2021) replaces resampling with a linear transformation of particles (with the coefficients being non-linear functions of the weights and particles). The linear transformation may be problematic, for instance, when the particles concentrate on a low-

dimensional manifold and a linear transformation could send the particles off the manifold. We refer the reader to Figure 1 for an illustration. Intuitively, this issue is less severe when the number of particles is large. Note that since the optimal transport matrix is usually close to a diagonal matrix, this issue may not be very pronounced in practice. Our proposed jittering method statistically produces particles close to existing particles. Thus, we believe it mitigates this particular issue to a certain extent, though not entirely.

Another closely related work is VMPF-UG by Lai et al. (2022), in which the resampling and proposal steps are merged into a single step, thus eliminating the discrete resampling component in SMC and enabling the reparametrization trick. The key difference between their method and ours is that by keeping the resampling step, our approach preserves the trajectory structure. The absence of a trajectory structure prohibits the use of some models including the variational recurrent neural network (VRNN) (Lai et al., 2022, Section 7). Another advantage of our method is its flexibility to accommodate any proposal distribution, whereas VMPF-UG currently only supports Gaussian or product distributions. In a more divergent approach, Aitchison (2019) introduced tensor Monte Carlo as an alternative to SMC, which no longer has a sequential structure and avoids resampling entirely.

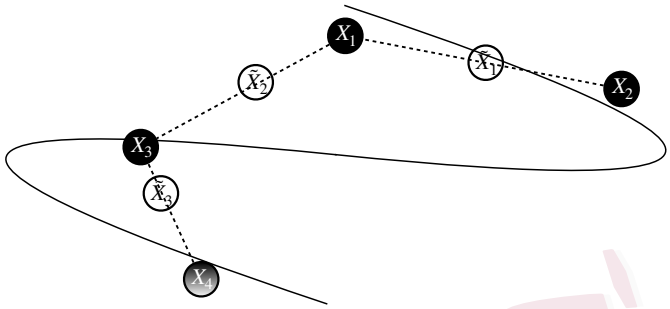


Figure 1: Cartoon illustration of a potential issue of linear-transformation-based resampling. Let $X_{1:4}$ denote the original particles, which are resampled to $\tilde{X}_{1:4}$ ($X_4 = \tilde{X}_4$). The solid curve represents a low-dimensional manifold, around which the probability distribution is concentrated. While the original particles are close to the manifold, after such linear resampling $\tilde{X}_2$ and $\tilde{X}_3$ drifted away and became bad particles.

3. Preliminaries

3.1 Notation

We use capital letters to denote random variables and lower case letters for their realizations. Superscripts and subscripts are used to denote the step/iteration and the sample index, respectively. Temporal notations are omitted for clarity whenever there is no confusion. We use $\mathbb{E}$ and $\mathbb{V}$ to represent expectation and variance, with subscripts, when used, to highlight

the underlying distribution. The gradient with respect to variable θ is denoted by ∇_θ.

3.2 State space model

State space models, also known as hidden Markov models when the hidden states take on finite discrete values, are characterized by a transition model and an observation model. In its general form, the state space model can be expressed as:

$$\begin{aligned} X^{(1)} &\sim g_1(\cdot;\theta), \\ Y^{(t)} \mid \left(X^{(1:t)} = x^{(1:t)}, Y^{(1:t-1)}\right) &\sim f_t(\cdot \mid x^{(t)};\theta), \\ X^{(t)} \mid \left(X^{(1:t-1)} = x^{(1:t-1)}, Y^{(1:t-1)}\right) &\sim g_t(\cdot \mid x^{(t-1)};\theta), t = 2, \cdots, T, \end{aligned} \tag{3.3}$$

where the $X^{(t)}$'s are unobserved hidden states while the $Y^{(t)}$'s are fully observed, and f and g represent distributions as well as density functions. A well-known problem for such models is the so-called filtering problem, which is to estimate the hidden state $X^{(t)}$ given all of the observations up to time t for fixed θ. In other words, the filtering problem focuses on the *on-line* posterior distribution of the hidden state at each time:

$$p(x^{(t)} \mid y^{(1:t)};\theta) = \int p(x^{(1:t)} \mid y^{(1:t)};\theta) dx^{(1:t-1)},$$

where

$$p(x^{(1:t)} \mid y^{(1:t)}) \propto \prod_{s=1}^{t} \left[g_s(x^{(s)} \mid x^{(s-1)}; \theta) f_s(y^{(s)} \mid x^{(s)}; \theta) \right].$$

Aside from inferring the hidden states, we are also interested in estimating the model parameter θ when it is unknown. There are two general inference strategies. From the perspective of maximizing the likelihood function, we aim to compute

$$\arg\max_{\theta} \int p(x^{(1:t)}, y^{(1:t)}; \theta) dx^{(1:t)}.$$

From the perspective of Bayesian inference, we assign a prior density $p(\theta)$ and aim to infer the following joint posterior density of $(\theta, x^{(1:t)})$:

$$p(\theta, x^{(1:t)} \mid y^{(1:t)}) \propto p(x^{(1:t)}, y^{(1:t)}; \theta) p(\theta).$$

The standard method of maximizing the likelihood function or computing the posterior mean is difficult because $p(y^{(1:t)}; \theta)$ or $p(\theta, x^{(1:t)} | y^{(1:t)})$ is generally intractable unless the model is linear and Gaussian. Therefore, it is necessary to use Monte Carlo approximations to carry out the inference.

3.3 Particle filters

For state space models, particle filters (PF) refer to a class of algorithms to approximate the sequence of posterior distributions (of the hidden states,

say) with weighted particles. A generic particle filter algorithm is outlined in Algorithm 1. In a sequential manner, particles and their weights are updated according to proposal distributions.

Consider a scenario where θ is the unknown parameter of interest. In a Bayesian framework, we can utilize particle filters for posterior inference. Although particle filters do not directly target the posterior distribution of θ, they can estimate the values of $p(y^{(1:t)} \mid \theta)$ and approximate the simulation from $p(x^{(1;t)} \mid y^{(1:t)}; \theta)$. Thus, particle filters can be used for making a Metropolis–Hastings type proposal and computing the acceptance ratio. This class of algorithms is widely recognized as particle MCMC methods (Andrieu et al., 2010).

4. Differentiability of Sequential Monte Carlo

4.1 Gradient-based parameter estimation

Although the particle MCMC approach and its variants have been shown effective in estimating the parameters in state space models, it may take a large number of iterations for the algorithm to converge. To avoid costly computation, one intuitive way is to employ the gradient of the log-likelihood with respect to the parameters to guide the search in the parameter space. An accurate estimation of the gradient can guide us to search in the space

Algorithm 1: Sequential importance sampling with resampling.

Input: $\{g_t, f_t, y^{(t)}\}_{t=1}^{T}$, θ

Output: weighted particles $(X_i^{(1:T)}, W_i^{(T)})_{1\le i\le n}$, estimation of log-likelihood $\ell(\hat{\theta})$

Initialization: $\hat{\ell}(\theta) \leftarrow 0$.

for $t = 1$ **to** T **do**

Draw $X_j^{(t)}$ from $g_t(X^{(t)} \mid X_j^{(1:t-1)}; \theta)$ ($g_1(X^{(1)}; \theta)$ if $t = 1$) for $j = 1, 2, \ldots, n$ conditionally independently.

Calculate the importance weight ($\pi_0 \equiv 1$):

$$W_j^{(t)} = f_t(y^{(t)} \mid X_j^{(t)}; \theta)$$

Update the estimate of the log-likelihood:

$$\hat{\ell}(\theta) \leftarrow \hat{\ell}(\theta) + \log\left(\frac{1}{n}\sum_j W_j^{(t)}\right).$$

Normalize the importance weight.

if $t < T$ **then**

Sample $a_1^{(t)}, a_2^{(t)}, \ldots, a_n^{(t)}$ from $\{1, 2, \ldots, n\}$ with probabilities $W_1^{(t)}, \cdots, W_n^{(t)}$, let $\tilde{X}_j^{(1:t)} = X_{a_j^{(t)}}^{(1:t)}$, and reweight the samples $\tilde{X}_1^{(1:t)}, \tilde{X}_2^{(1:t)}, \cdots, \tilde{X}_n^{(1:t)}$ equally with $1/n$.

Let $X_j^{(1:t)} = \tilde{X}_j^{(1:t)}$ for $j = 1, 2, \ldots, n$.

end

end

Return $(X_i^{(1:T)}, W_i^{(T)})_{1\le i\le n}$, $\hat{\ell}(\theta)$.

more efficiently. However, the gradient of the marginal log-likelihood $\ell(\theta)$ with respect to θ is intractable analytically because the distribution itself is generally not in closed form. Instead, letting

$$\hat{\ell}(\theta) = \log \hat{L}(\theta) = \log \left(\text{SMC estimate of } p(y^{(1:t)} \mid \theta)\right), \tag{4.4}$$

where $\hat{L}(\theta)$ is an unbiased estimator for the likelihood function $L(\theta)$, we aim to compute the gradient of $\mathbb{E}[\hat{\ell}(\theta)]$, where the expectation is taken with respect to the randomness in the SMC procedure, including both the particle generation and resampling. It is not difficult to derive the joint distribution of the particles and the resampling indices, conditional on the observations $y^{(1:t)}$, which is shown in equation (4.5) below:

$$p(x_{1:n}^{(1:t)}, a_{1:n}^{(1:t)}; \theta) = \left[\prod_{i=1}^{n} g(x_i^{(1)}; \theta)\right] \cdot \prod_{s=2}^{t} \prod_{i=1}^{n} \left[\frac{w_{a_i^{(s-1)}}^{(s-1)}}{\sum_l w_l^{(s-1)}} g(x_i^{(s)} \mid x_{a_i^{(s-1)}}^{(s-1)}; \theta)\right], \tag{4.5}$$

where $a_{1:n}^{(1:t)}$ represents the resampling indices as in Algorithm 1 and w represents the importance weights, which depend on y. From the perspective of variational inference, the objective function $\mathbb{E}[\hat{\ell}(\theta)]$ can be regarded as a surrogate evidence lower bound (see Theorem 1 in Naesseth et al. (2018)).

Note that parameter θ not only appears in the term $\hat{\ell}(\theta)$, but also affects the underlying joint density $p(x_{1:n}^{(1:t)}, a_{1:n}^{(1:t)}; \theta)$. Therefore, it is imprecise to simply exchange the differentiation and the expectation signs for the com-

putation of the gradient. To propose an efficient and accurate estimation of $\nabla_\theta \mathbb{E}[\hat{\ell}(\theta)]$ is the major focus of this article.

4.2 Score estimate and the reparametrization trick

Computing the gradient of the expectation term is a ubiquitous problem in statistics and machine learning (Mohamed et al., 2020). Two most used gradient estimators are the score function gradient (Williams, 1992) and the reparametrization gradient (Kingma and Welling, 2014). Specifically for our goal, the objective function is $\mathbb{E}_{p(z;\theta)}[\hat{\ell}(\theta, z)]$, where $z = (x_{1:n}^{(1:t)}, a_{1:n}^{(1:t)})$, and we have made clear the dependency of the log-likelihood estimate $\hat{\ell}$ on z. Assuming that we can swap the derivative and integration signs, we have

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p(z;\theta)}[\hat{\ell}(\theta, z)] =& \nabla_\theta \left[\int_z p(z;\theta)\hat{\ell}(\theta, z)dz\right] \\ =& \int_z \nabla_\theta \left[p(z;\theta)\hat{\ell}(\theta, z)\right] dz \\ =& \int_z p(z;\theta)\nabla_\theta \hat{\ell}(\theta, z)dz + \int_z \hat{\ell}(\theta, z)\nabla_\theta p(z;\theta)dz.\end{aligned} \tag{4.6}$$

The first term in equation (4.6) can be rewritten as $\mathbb{E}_p[\nabla_\theta \hat{\ell}(\theta, z)]$ and thus can be easily estimated by a Monte Carlo method that draws samples $z \sim p(z;\theta)$. We can rewrite the second term as

$$\int_z \hat{\ell}(\theta, z)p(z;\theta)\nabla_\theta \log p(z;\theta)dz = \mathbb{E}_p[\hat{\ell}(\theta, z)\nabla_\theta \log p(z;\theta)],$$

and then apply the same Monte Carlo method. This Monte Carlo estimate is usually termed as the score estimate or reinforce estimate (Williams, 1992; Poyiadjis et al., 2011). However, this estimate usually has a huge variance, and it may even be better to simply estimate it as zero (Le et al., 2017; Naesseth et al., 2018). The induced bias is provided explicitly in Corenflos et al. (2021, Proposition 4.1).

The reparametrization trick is another way to solve the gradient estimation problem. Using this trick, Kingma and Welling (2014) proposed an unbiased, differentiable and scalable estimator for the variational bound (see their Section 2.2) in auto-encoding variational Bayes. The "tricky" part of the reparametrization trick is that we can make the randomness an input to the model so as to backpropagate through a random node. Consider the general question of estimating $\nabla_\theta \left(\mathbb{E}_{p(z;\theta)}\left[f(z;\theta)\right]\right)$, of which equation (4.6) is a special case. If we can reparametrize Z as $Z = \phi(\epsilon;\theta)$, such that $\phi(\epsilon;\theta)$ has density $p(\cdot;\theta)$, where ϕ is differentiable with respect to θ, and $\epsilon \sim p(\epsilon)$ independent of θ, then

$$\mathbb{E}_{p(z;\theta)}\left[f(z;\theta)\right] = \mathbb{E}_{p(\epsilon)}\left[f(\phi(\epsilon;\theta);\theta)\right], \tag{4.7}$$

and consequently,

$$\begin{aligned}\nabla_\theta \mathbb{E}_{p(z;\theta)}\left[f(z;\theta)\right] &= \nabla_\theta \mathbb{E}_{p(\epsilon)}\left[f(\phi(\epsilon;\theta);\theta)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\nabla_\theta f(\phi(\epsilon;\theta);\theta)\right],\end{aligned} \tag{4.8}$$

which can be approximated by a Monte Carlo method that draws $\epsilon \sim p(\epsilon)$.

4.3 Resampling in SMC and non-differentiability

Although the reparametrization trick has been shown to be efficient in various applications, it is not applicable to discrete distributions. For example, if we employ step functions as our ϕ, the gradient will be zero almost everywhere and undefined at some points. This means that we cannot directly apply the reparametrization trick to equation (4.6), since the resampling indices $a_{1:n}^{(1:t)}$ are discrete.

To specifically understand the subtlety of the problem in the context of SMC, let us consider the state-space model in more details. Following notations from Section 3.2 and the chain-rule of probability, we have

$$\nabla \ell(\theta) = \nabla \log p(y^{(1)};\theta) + \sum_{t=2}^{T} \nabla \log p(y^{(t)} \mid y^{(1:(t-1))};\theta). \tag{4.9}$$

Let us start from the first term. Suppose we draw $x_{1:n}^{(1)}$ from a distribution $q_1(\cdot;\theta)$, such that we can use the reparameterization trick to write $x_i^{(1)} =$

$\phi_1(\epsilon_i^{(1)}, \theta)$, $\epsilon_i^{(1)} \sim^{iid} \mathcal{N}(0,1)$. Then we have

$$p(y^{(1)};\theta) = \mathbb{E}_{q_1(\cdot;\theta)}\left[\frac{p(y^{(1)}, x^{(1)};\theta)}{q_1(x^{(1)};\theta)}\right] = \mathbb{E}\left[\frac{p(y^{(1)}, \phi_1(\epsilon^{(1)},\theta);\theta)}{q_1(\phi_1(\epsilon^{(1)},\theta);\theta)}\right], \epsilon^{(1)} \sim \mathcal{N}(0,1).$$

Since the distribution of $\epsilon^{(1)}$ is free of θ, we can directly differentiate under the integral sign and obtain an expectation form of $\nabla_\theta p(y^{(1)};\theta)$ (and hence also $\nabla_\theta \log p(y^{(1)};\theta)$), then use $x_{1:n}^{(1)}$ as Monte Carlo samples for estimation.

Now, suppose we resample the particles $x_{1:n}^{(1)}$, and equivalently $\epsilon_{1:n}^{(1)}$, to get a new set $\tilde{\epsilon}_{1:n}^{(1)}$ and, correspondingly $\tilde{x}_{1:n}^{(1)}$ with $\tilde{x}_i^{(1)} = \phi_1(\tilde{\epsilon}_i^{(1)}, \theta)$, which can be viewed as approximate samples from $p(x^{(1)} \mid y^{(1)};\theta)$. Let us similarly sample $x_{1:n}^{(2)}$ from $q_2(\cdot;\theta)$, reparametrize $x_i^{(2)} = \phi_2(\epsilon_i^{(2)}, \theta)$, $\epsilon_i^{(2)} \sim^{iid} \mathcal{N}(0,1)$, and write

$$\begin{aligned}
p(y^{(2)} \mid y^{(1)};\theta) &= \int p(y^{(2)}, x^{(2)} \mid y^{(1)}, x^{(1)};\theta) p(x^{(1)} \mid y^{(1)};\theta) dx^{(1)} dx^{(2)} \\
&= \mathbb{E}_{x^{(1)} \sim p(x^{(1)} \mid y^{(1)};\theta), x^{(2)} \sim q_2(\cdot;\theta)}\left[\frac{p(y^{(2)}, x^{(2)} \mid y^{(1)}, x^{(1)};\theta)}{q_2(x^{(2)};\theta)}\right] \\
&\approx \mathbb{E}_{\tilde{x}^{(1)} \sim \phi_1(\tilde{\epsilon}^{(1)},\theta), x^{(2)} \sim q_2(\cdot;\theta)}\left[\frac{p(y^{(2)}, x^{(2)} \mid y^{(1)}, \tilde{x}^{(1)};\theta)}{q_2(x^{(2)};\theta)}\right] \\
&= \mathbb{E}_{\tilde{\epsilon}^{(1)}, \epsilon^{(2)}}\left[\frac{p(y^{(2)}, \phi_2(\epsilon^{(2)},\theta) \mid y^{(1)}, \phi_1(\tilde{\epsilon}^{(1)},\theta);\theta)}{q_2(\phi_2(\epsilon^{(2)},\theta);\theta)}\right].
\end{aligned}$$

However, this is not a successful reparametrization attempt, as the $\tilde{\epsilon}^{(1)}$'s are no longer i.i.d. Gaussian and their distribution implicitly depends on θ due to resampling. Thus, we are not able to directly differentiate under the integral sign. If we ignore this fact, pretending that the $\tilde{\epsilon}$'s are i.i.d. Gaussian and differentiating under the integral sign anyway, we recover the

algorithm proposed by Naesseth et al. (2018).

Despite the non-differentiability, resampling is necessary in SMC because the repeated forward sampling procedure would eventually lead to weight degeneracy, where the weights concentrate on only a few particles (Liu and Chen, 1995). There are various means to resample from a set of weighted particles. Aside from the bootstrap resampling or multinomial resampling shown in Algorithm 1, residual resampling (Liu and Chen, 1998) and stratified resampling (Kitagawa, 1996) are two more popular resampling schemes because they tend to introduce less unnecessary randomness. Reich (2013) proposed the optimal transport resampling, borrowing ideas from transportation theory. More recently, Gerber et al. (2019) introduced the Hilbert curve resampling in a multi-dimensional space, which is shown to have optimal rate under some conditions (Li et al., 2022). Unfortunately, all these resampling schemes are discrete, in that the resampled particles can only take values in a finite set (conditional on particles from the previous step), and thus incompatible with the reparametrization trick.

To circumvent the non-differentiability, Corenflos et al. (2021) proposed a framework where the traditional resampling step is replaced by a linear transformation with parameters defined by the optimal transport resampling (Reich, 2013). However, the linear transformation could give rise to

bad particles when the posterior distribution concentrates on a non-linear manifold, or when a linear combination of two modes ends up in an area with low density.

5. Reparametrized Resampling via Jittering

5.1 Algorithm description

The vanilla multinomial resampling strategy in SMC samples independently from the following multinomial distribution:

$$\tilde{X}_i \mid X, W \sim \text{Multinomial}(1, X, (W_1, W_2, \ldots, W_n)),$$

which is equivalent to sampling from the empirical distribution:

$$\tilde{X}_i \mid X, W \sim \sum_{i=1}^{n} W_i \delta_{X_i}(x).$$

The discrete nature of multinomial distribution makes the reparametrization trick fail. To overcome this issue, we consider a kernel smoothed version of the form

$$\tilde{X}_i \mid X, W \sim \sum_{i=1}^{n} W_i \kappa_r \left(x - X_i\right), \tag{5.10}$$

where $\kappa_r(\cdot)$ is a non-negative differentiable kernel density, and r is a predetermined bandwidth parameter that balances smoothness and bias. Here, we require that $\int_{\mathbb{R}^d} \kappa_r\left(x\right) dx = 1$, where d is the dimension of x. In other

words, κ_r is the derivative of a d-dimensional cumulative distribution function (CDF).

If we resample independently according to (5.10), everything is differentiable and we can utilize the reparametrization trick. If $\kappa_r = \prod_{j=1}^{d} \kappa_{r,j} = \nabla_x K_r$ is the derivative of the CDF $K_r = \prod_{j=1}^{d} K_{r,j}$, which corresponds to independent random components. Then we can express each $\tilde{X}_i$ as $F^{-1}(U_i; W, X, K_r)$, where U_i is a uniform random vector on the d-dimensional unit cube $[0,1]^d$, and $F^{-1}(\cdot; W, X, K_r)$ is the generalized inverse function of $F^{W,X,K_r}(x)$, which is defined recursively as follows:

$$\begin{aligned} F_1^{W,X,K_r}(x_1) &= \sum_i W_i K_{r,1}(x_1 - X_{i1}), \\ F_j^{W,X,K_r}(x_j; x_{1:j-1}) &= \frac{\sum_i W_i \prod_{k=1}^{j-1} \kappa_{r,k}(x_k - X_{ik}) K_{r,j}(x_j - X_{ij})}{\sum_i W_i \prod_{k=1}^{j-1} \kappa_{r,k}(x_k - X_{ik})}. \end{aligned} \tag{5.11}$$

Here, x_j denotes the j-th entry of the d dimensional vector x.

Let $\tilde{X}_i = F^{-1}(U_i; W, X, K_r)$, it suffices to calculate the gradient of $F^{-1}(\cdot; W, X, K_r)$. The details are deferred to Section S1 in the supplemental material. Our method, named reparametrized resampling via jittering (RRJ), is detailed in Algorithm 2.

It has been pointed out to us by a reviewer that an arXiv preprint Graves (2016) proposed a recursive method that can also be used as an alternative way to calculate this gradient of F^{-1}. The main ideas behind their method and ours are very similar. In the remainder of this paper, we

present theoretical insights, practical considerations, and empirical studies.

Algorithm 2: Reparametrized Resampling via Jittering (RRJ)

Input: weighted particles $(X_i^{(1:t)}, W_i^{(t)})_{1\le i\le n}$, kernel distribution K_r with a pre-chosen bandwidth r.

Output: resampled particles. $(\tilde{X}_i^{(1:t)})_{1\le i\le n}$.

Sample $\tilde{X}_1^{(1:t)}, \tilde{X}_2^{(1:t)}, \cdots, \tilde{X}_n^{(1:t)}$ from $X_1^{(1:t)}, \cdots, X_n^{(1:t)}$ with probabilities $W_1^{(t)}, \cdots, W_n^{(t)}$, and reweight the samples $\tilde{X}_1^{(1:t)}, \tilde{X}_2^{(1:t)}, \cdots, \tilde{X}_n^{(1:t)}$ equally with $1/n$.

Let $\tilde{X}_i^{(1:t)} = \tilde{X}_i^{(1:t)} + \epsilon_i$, where $\epsilon \overset{i.i.d.}{\sim} K_r$, $i = 1, \ldots, n$.

Return $(\tilde{X}_i^{(1:t)})_{1\le i\le n}$.

5.2 Choice of the kernel distribution

Technically any kernel distribution with differentiable density is valid for RRJ. We recommend the Gaussian kernel as a default choice for practical use, given its simplicity in sampling and computation. Aside from the kernel itself, the choice of the bandwidth r is more critical for the behavior of the proposed algorithm, as it balances the SMC accuracy and differentiability. For the sake of differentiability, we would like a large r; for high sampling accuracy, however, we would like a small r so that the bias introduced by RRJ is negligible.

Consider a generic step in Algorithm 2, where we analyze the resampling error similar to that in Li et al. (2022). There, the resampling step is unbiased in terms of

$$\mathbb{E}\left[\sum_{i=1}^{n}\frac{1}{n}\phi(\tilde{X}_i) \mid X, W\right] = \sum_{i=1}^{n} W_i\phi(X_i)$$

for any ϕ, so they only analyzed the resampling variance. Here, we are interested in the mean squared error (MSE) instead because resampling is now generally biased. When bias exists, under some loose conditions, specified in Theorem 1, a properly chosen r does not affect the order of the estimation error rate.

Theorem 1. *Consider the state space model and the SMC process in Algorithm 2 in which parameter θ is given, the length T of the state space model is fixed, and kernel K_r satisfies $k_r(x) = k(x/r)$ for a fixed density $k(\cdot)$ with bandwidth r given. Let $\pi_t(x^{(1:t)}) = p(x^{(1:t)} \mid y^{(1:t)})$ and let d be the dimension of the hidden state $x^{(t)}$. Assume*

(i) $f_t(y^{(t)} \mid \cdot)$ and $g_t(x^{(t)} \mid \cdot)$ are upper bounded by M and L-Lipschitz,

(ii) $f_t(y^{(t)} \mid \cdot)$ is lower bounded by $\underline{e} > 0$,

(iii) $r = O(1/\sqrt{dn})$.

Then for any bounded Lipschitz ϕ, we have

$$\left| \mathrm{E}\left\{ \frac{\sum_{j=1}^{n} W_j^{(t)} \phi(X_j^{(t)})}{\sum_{j=1}^{n} W_j^{(t)}} \right\} - \int \pi_t(x^{(1:t)}) \phi(x^{(t)}) dx^{(1:t)} \right| = O(1/\sqrt{n}), \quad (5.12)$$

$$\mathrm{Var}\left\{ \frac{\sum_{j=1}^{n} W_j^{(t)} \phi(X_j^{(t)})}{\sum_{j=1}^{n} W_j^{(t)}} \right\} = O(1/n), \quad (5.13)$$

and

$$\mathrm{E}\left[|\hat{\ell}(\theta) - \ell(\theta)|^2 \right] = O(1/n), \quad (5.14)$$

where $\ell(\theta)$ is the true log-likelihood.

We note that, while the rates in Theorem 1 do not explicitly depend on the dimension d, the dimension plays a role through the Lipschitz constant L. We therefore cannot claim that this result is dimension-independent. The complete proof and some empirical results on the likelihood bias of RRJ are provided in the Supplementary Materials Sections S2 and S3.

6. Experiments

6.1 Experiments overview

To verify the effectiveness of RRJ in different settings, we implement simulation studies under three models and analyze a real data case. As in Corenflos et al. (2021), we use the log-likelihood estimate as the training target. For simulations with well-specified models, we use the MSE of $\hat{\theta}$

as a performance measure; for real data analysis, in the absence of the ground truth, we run a vanilla SMC algorithm with a large n to evaluate the likelihood at $\hat{\theta}$ obtained by different resampling methods.

Simulation studies consist of (1) the linear state space model, (2) the stochastic volatility model, and (3) the non-linear transition model. For each model, 50 independent replications are carried out. The proposed RRJ is applied to each of the simulated data to learn the corresponding parameters. We compare the results with the same model trained by the ensemble transform proposed by Corenflos et al. (2021), along with the method in Naesseth et al. (2018), where the resampling terms are simply ignored. The last two methods are referred to as "ET" and "multinomial", respectively.

All of the aforementioned methods, including the RRJ, are carried out in the procedure of gradient learning using ADAM (Kingma and Ba, 2014). For simulation studies, we use a fixed learning rate of 0.01; for the real data analysis, we follow the same setting as in Corenflos et al. (2021) and use learning rate $0.01 \times 0.9^{\lfloor \text{step}/200 \rfloor}$. All the other ADAM parameters are set to default values.

6.2 Linear state space models

The multidimensional linear state space model is a widely used model for time series data. Its general form can be expressed as:

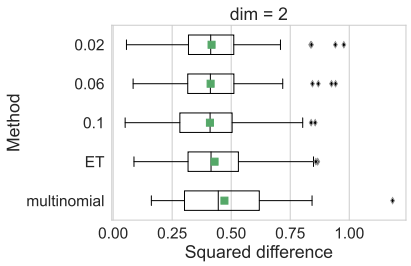
$$X^{(t)} \mid X^{(1:t-1)} \sim \mathcal{N}\left(\Phi X^{(t-1)}, \Sigma_x\right),$$
$$Y^{(t)} \mid X^{(1:t)}, Y^{(1:t-1)} \sim \mathcal{N}\left(X^{(t)}, \Sigma_y\right),$$

for $t = 2, \cdots, T$, with $X^{(1)} \sim \mathcal{N}(0, 1/3\mathbf{I})$. In the simulations, we set $\Sigma_x = \Sigma_y = 0.25I$, and let

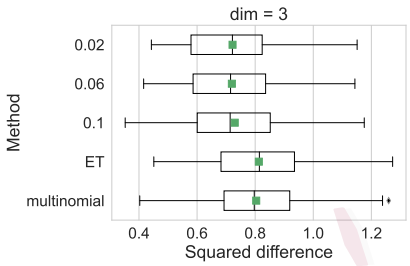
$$\Phi = [\rho^{|i-j|}]_{ij} \times \frac{0.8}{\lambda_{\max}([\rho^{|i-j|}]_{ij})}, \tag{6.15}$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of a matrix and the true value of ρ is set as 0.5.

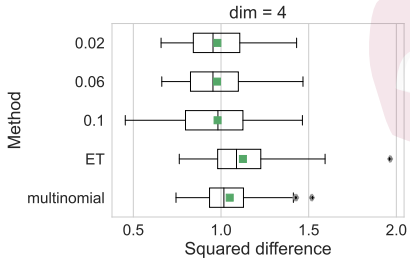
From Figure 2, we can see that RRJ with different bandwidth parameters show better performances than ensemble transform and multinomial. We also show pairwise comparisons with ensemble transform in Figure 3, where both methods are applied to the same data. We can see that RRJ often outperforms ensemble transform, especially in higher dimensions.



(a) $d = 2$.

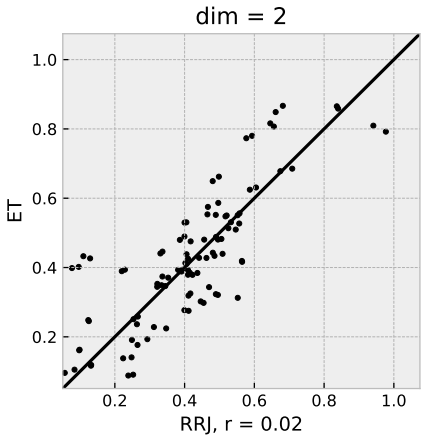


(b) $d = 3$.

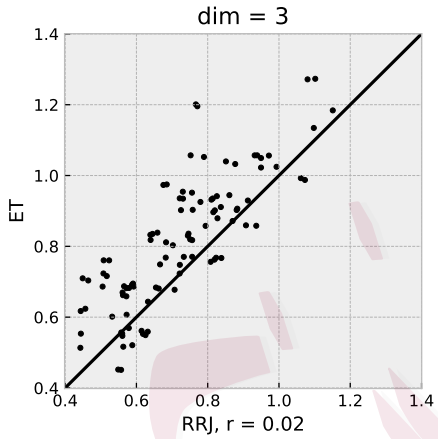


(c) $d = 4$.

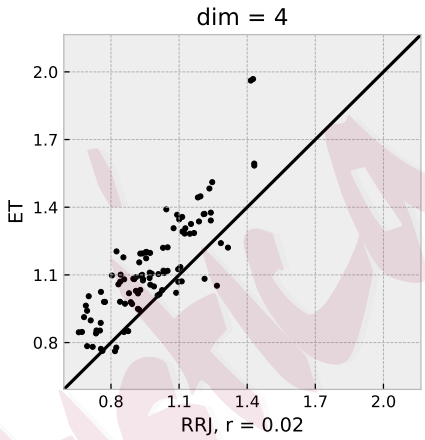
Figure 2: Simulation results for the linear transition state space model in different dimensions. Boxplots of squared differences over 50 independent experiments, with mean values indicated by green squares. Here, $T = 40$, particle numbers are 50.



(a) $d = 2$.



(b) $d = 3$.



(c) $d = 4$.

Figure 3: Pairwise comparison between the RRJ and the ensemble transform in different dimensions. Each dot represents an independent experiment, where the x-coordinate is the squared loss of RRJ with kernel bandwidth $r = 0.02$, and the y-coordinate is the squared loss of ensemble transform. Here, $T = 40$, particle numbers are 50.

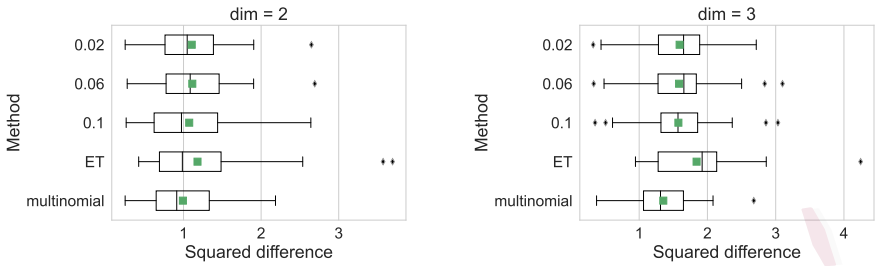
6.3 Stochastic volatility model

The multidimensional stochastic volatility model is commonly applied model in financial economics. The model is

$$X^{(t)} \mid X^{(1:t-1)} \sim \mathcal{N}\left(\Phi X^{(t-1)}, \Sigma\right),$$
$$Y^{(t)} \mid X^{(1:t)}, Y^{(1:t-1)} \sim \mathcal{N}\left(0, \beta^2 \operatorname{diag}(\exp(X^{(t)}))\right),$$

for $t = 2, \cdots, T$ with $X^{(1)} \sim \mathcal{N}(0, 1/3\mathbf{I})$, where Φ and Σ are $d \times d$ parameter matrices.

According to Chapter 14.2 in Chopin and Papaspiliopoulos (2020), the likelihood function of the stochastic volatility model is highly pathological, and it is often challenging to estimate all its parameters simultaneously. To validate the effectiveness of the proposed method more conveniently, in this study we fix $\Sigma = 0.25I$ and only estimate part of the parameters, i.e, Φ and β, with the true value of β set to 0.5 and the true of Φ set the same as in (6.15). From Figure 4, we can see that RRJ with different parameters show better performances than ensemble transform, but not as good as multinomial.



(a) $d = 2$. (b) $d = 3$.

Figure 4: Simulation results for the stochastic volatility model in different dimensions. Boxplots of squared differences over 50 independent experiments, with mean values indicated by green squares. Here, $T = 40$, particle numbers are 50.

6.4 Nonlinear transition state space model

We consider the state space model with nonlinear transition density (Kitagawa, 1996) below

$$\begin{aligned} X^{(t)} \mid X^{(1:t-1)} &\sim \mathcal{N}\left(0.5\theta_3 X^{(t-1)} + \frac{10\theta_4 X^{(t-1)}}{1+(X^{(t-1)})^2}, \theta_1 V^{(t)}\right), \\ Y^{(t)} \mid X^{(t)} &\sim \mathcal{N}\left(0.05\theta_5 X^{(t)}, \theta_2 W^{(t)}\right), \end{aligned} \tag{6.16}$$

where $V^{(t)}$ and $W^{(t)}$ are independent random variables sampled from the standard normal distribution $\mathcal{N}(0,1)$, and $X^{(1)} \sim \mathcal{N}(0,2)$. In this model, θ_1, θ_2, θ_3, θ_4 and θ_5 are the parameters to be estimated, whose real values are all set to 1. Compared to the linear transition model, the parameters are more difficult to estimate for this non-linear transition model. Figure 5

and Table 1 shows that RRJ outperforms the other methods for θ_3 and θ_4, while not as good for θ_1 and θ_2. For θ_5, errors of RRJ methods have much smaller median, while the means are comparable.

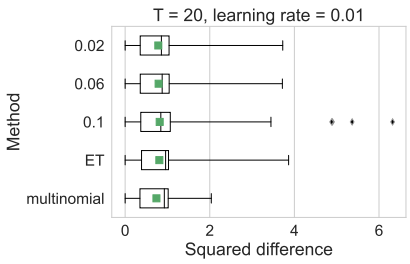
Mean loss	θ_1	θ_2	θ_3	θ_4	θ_5
$\text{RRJ}_{0.02}$	0.782(0.021)	0.067(0.004)	0.318(0.016)	0.248(0.012)	0.047(0.004)
$\text{RRJ}_{0.06}$	0.791(0.021)	0.067(0.004)	0.320(0.017)	0.269(0.021)	0.266(0.027)
$\text{RRJ}_{0.1}$	0.816(0.027)	0.069(0.004)	0.331(0.018)	0.381(0.042)	0.623(0.063)
ET	0.808(0.021)	0.063(0.005)	0.398(0.018)	0.639(0.092)	0.766(0.063)
multinomial	0.740(0.017)	0.054(0.004)	0.346(0.016)	0.621(0.088)	0.361(0.028)

(a) The mean squared error (MSE) and the standard deviation across multiple independent experiments.

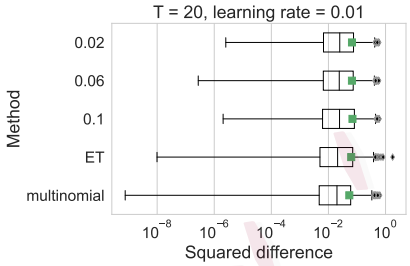
Median loss	θ_1	θ_2	θ_3	θ_4	θ_5
$\text{RRJ}_{0.02}$	0.857	0.025	0.198	0.149	0.016
$\text{RRJ}_{0.06}$	0.875	0.024	0.195	0.137	0.030
$\text{RRJ}_{0.1}$	0.843	0.024	0.182	0.156	0.051
ET	0.956	0.021	0.222	0.190	0.424
multinomial	0.926	0.020	0.221	0.185	0.163

(b) The median of the squared errors across multiple independent trials.

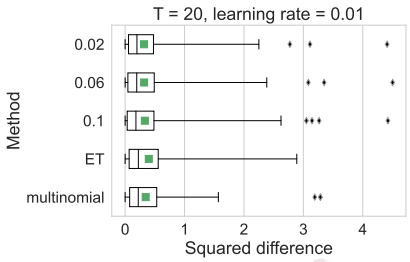
Table 1: The results of the non-linear experiments are presented in these two tables.



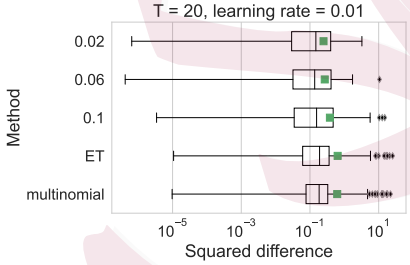
(a) θ_1.



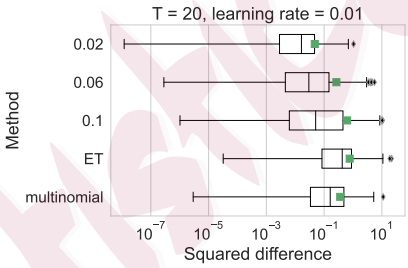
(b) θ_2 (log-scale).



(c) θ_3.



(d) θ_4 (log-scale).



(e) θ_5 (log-scale).

Figure 5: Simulation results for the nonlinear transition state space model. Boxplots of squared differences over 600 independent experiments, with mean values indicated by green squares. Here, $T = 20$.

6.5 VRNN on polyphonic music data

The variational recurrent neural network (VRNN) proposed by Chung et al. (2015) combines the concepts of RNN and variational autoencoder, which can ensure the flexibility of the dynamic model to a large extent. Following the notation in Corenflos et al. (2021), the model is represented in equation (6.17).

$$\begin{aligned}(R^{(t+1)}, O^{(t+1)}) &= \mathrm{RNN}_\theta(R^{(t)}, Y^{(1:t)}, \tau_\theta(Z^{(t)}),\\ Z^{(t+1)} &\sim \mathcal{N}(\mu_\theta(O^{(t+1)}), \sigma_\theta(O^{(t+1)})),\\ \hat{p}^{(t+1)} &= h_\theta(\tau_\theta(Z^{(t+1)}), O^{(t+1)}),\\ Y^{(t)} \mid R^{(t)}, Z^{(t)} &\sim \mathrm{Ber}(\hat{p}^{(t)}),\end{aligned} \tag{6.17}$$

where $R^{(t)}$ and $O^{(t)}$ represent the RNN state and output in a regular LSTM model, respectively. $Z^{(t)}$ is a Gaussian random variable and τ_θ, h_θ, μ_θ, σ_θ are fully connected neural networks. $Y^{(t)}$ denotes the binary observations. We initialize $(R^{(1)}, O^{(1)})$ to be zeros and $Z^{(1)}$ to be a sample from the standard multivariate Gaussian distribution. We use 36 particles to run the experiments and then run a separate SMC with multinomial resampling and 500 particles for performance validation. The learning rate is set to $0.01 \times 0.9^{\lfloor \mathrm{step}/1000 \rfloor}$ for a total of 10,000 steps. Further details of the experiments can be found in Supplementary Materials Section S4.

We present our experimental findings based on three polyphonic music datasets outlined in Corenflos et al. (2021), which adhere to the methodology of Boulanger-Lewandowski et al. (2012). Within each dataset, the binary vector $Y^{(t)}$ encompasses 88 dimensions, corresponding to the piano note range spanning from A0 to C8. Table 2 showcases the sample mean and standard deviation of the last-2000-step-average loss derived from independent experiments, while Figure 6 illustrates the loss history of a representative run. Notably, our results demonstrate that RRJ performs on par with or surpasses ensemble transform concerning the achieved log-likelihood. It is worth noting that ensemble transform tends to exhibit greater stability across distinct experimental runs.

	RRJ	ET
JSB	99.76 (16.09)	156.19 (16.38)
Nottingham	99.45 (20.88)	147.78 (13.52)
Muse	135.60 (33.59)	189.35 (15.47)

Table 2: Sample mean and standard deviation of the last-2000-step-average minus log-likelihood, calculated over 50 independent runs.

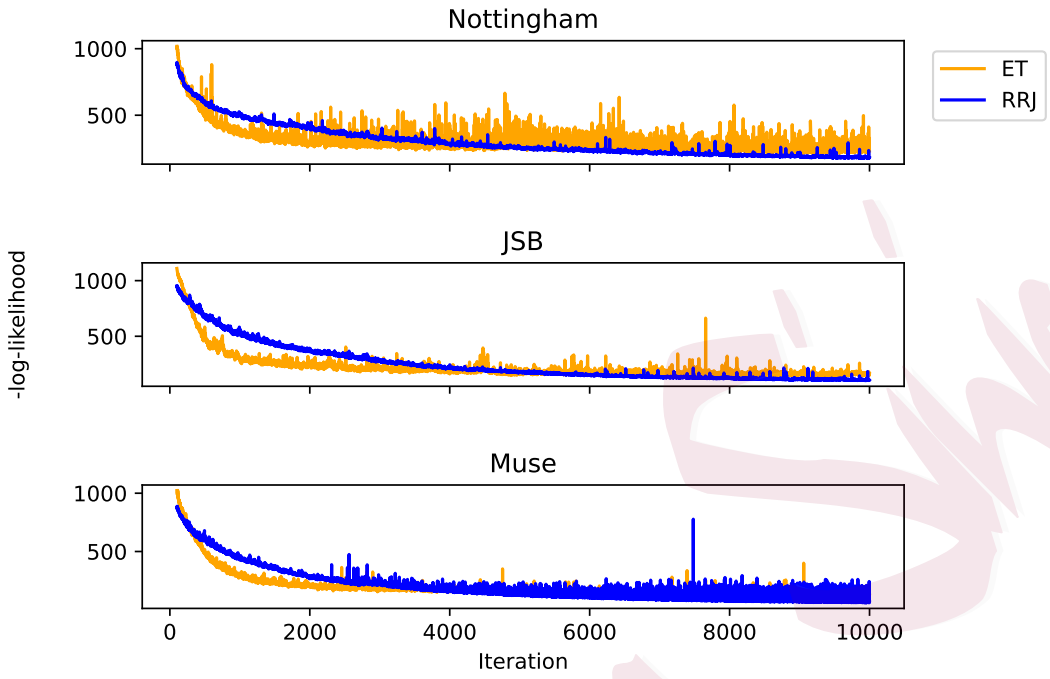


Figure 6: The likelihood evaluation for VRNN model in a single run. The first 100 burn-in steps are omitted.

7. Discussion

This artical introduces and investigates a method called the *reparametrized resampling via jittering* in particle filters for efficiently estimating the gradient of the likelihood function of state space models. Some theoretical insights and numerical experiments are provided to demonstrate the effectiveness of the proposed method. We conclude the article with some additional thoughts and open problems.

In a conventional application of particle filters, stratified resampling is often preferred as it introduces less randomness compared to multinomial resampling. However, it is not straightforward to backpropagate when stratified resampling is employed. Our method can be easily generalized to stratified resampling. Instead of using n i.i.d. samples $U_i \sim [0,1]^d$ (see Section 5), we can stratify the space and use a low discrepancy set such as in Gerber and Chopin (2015) and Li et al. (2022). It is expected to improve the gradient estimation.

The stratified multiple-descendant growth (SMG) algorithm proposed in Li et al. (2022) has been proved to converge faster than the traditional SMC in terms of the estimation MSE. The main idea of SMG is to shrink the number of particles at the resampling step and grow it back at the transition step in a stratified manner. This design is particularly conducive to convergence rate analysis. It may be possible to borrow ideas from the analysis of SMG so as to obtain a more refined convergence rate for RRJ.

In addition to the smooth jittering, there are also other ways to "smooth" discrete random variables, enabling the application of the reparametrization trick. For example, Maddison et al. (2016) apply the Gumbel-max trick, relaxing the multinomial sampling to a continuous distribution supported on a simplex. It would be useful to compare different relaxation approaches

in the context of designing differential particle filters.

Acknowledgements

The research is partly supported by the National Science Foundation of USA (2015411, JSL PI), National Natural Science Foundation of China (Grant 11931001, KD Co-PI), and the Guo Qiang Institute of Tsinghua University.

References

Aitchison, L. (2019). Tensor monte carlo: particle methods for the gpu era. *Advances in Neural Information Processing Systems*, 32.

Ali, A., Dobriban, E., and Tibshirani, R. (2020). The implicit regularization of stochastic gradient flow for least squares. In *International Conference on Machine Learning*, pages 233–244. PMLR.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Andrieu, C., Doucet, A., and Tadic, V. B. (2005). On-line parameter esti-

mation in general state-space models. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 332–337. IEEE.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

Chopin, N. and Papaspiliopoulos, O. (2020). *An introduction to sequential Monte Carlo*. Springer.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Yystems*, 28.

Corenflos, A., Thornton, J., Deligiannidis, G., and Doucet, A. (2021). Differentiable particle filtering via entropy-regularized optimal transport. In *International Conference on Machine Learning*, pages 2100–2111. PMLR.

Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo Methods in Practice*, pages 3–14. Springer.

Fearnhead, P. and Clifford, P. (2003). On-line inference for hidden Markov

models via particle filters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(4):887–899.

Gerber, M. and Chopin, N. (2015). Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579.

Gerber, M., Chopin, N., Whiteley, N., et al. (2019). Negative association, ordering and convergence of resampling methods. *The Annals of Statistics*, 47(4):2236–2260.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET.

Graves, A. (2016). Stochastic backpropagation through mixture density distributions. *arXiv preprint arXiv:1607.05690*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian

nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

Lai, J., Domke, J., and Sheldon, D. (2022). Variational marginal particle filters. In *International Conference on Artificial Intelligence and Statistics*, pages 875–895. PMLR.

Le, T. A., Igl, M., Rainforth, T., Jin, T., and Wood, F. (2017). Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*.

Lee, A. (2008). *Towards smooth particle filters for likelihood estimation with multivariate latent variables*. PhD thesis, University of British Columbia.

Li, Y., Wang, W., Deng, K., and Liu, J. S. (2022). Stratification and optimal resampling for sequential Monte Carlo. *Biometrika*, 109(1):181–194.

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.

Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete dis-

tribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. (2020). Monte Carlo gradient estimation in machine learning. *J. Mach. Learn. Res.*, 21(132):1–62.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. (2018). Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR.

Pitt, M. K. (2002). Smooth particle filters for likelihood evaluation and maximisation. Technical report.

Poyiadjis, G., Doucet, A., and Singh, S. S. (2011). *Biometrika*, 98(1):65–80.

Reich, S. (2013). A nonparametric ensemble transform method for Bayesian inference. *SIAM Journal on Scientific Computing*, 35(4):A2013–A2024.

Ścibior, A. and Wood, F. (2021). Differentiable particle filtering without modifying the forward pass. *arXiv preprint arXiv:2106.10314*.

Shephard, N. and Flury, T. (2009). Learning and filtering via simulation: smoothly jittered particle filters.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Zhu, M., Murphy, K., and Jonschkowski, R. (2020). Towards differentiable resampling. *arXiv preprint arXiv:2004.11938*.

Tsinghua University & Harvard University

E-mail: yichaoli@fas.harvard.edu

E-mail: wenshuowang1997@gmail.com

Tsinghua University

E-mail: kdeng@tsinghua.edu.cn

Harvard University

E-mail: jliu@stat.harvard.edu