Author:
Zhang, Chang

Title:
On the Improvements and Innovations of Monte Carlo Methods

On the Improvements and Innovations of Monte Carlo Methods

University of BRISTOL

Chang Zhang

Department of Mathematics

University of Bristol

June 2022

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of Doctor of Philosophy in the Faculty of Science

Abstract

Monte Carlo methods have played a central role in computational statistics for many years. While it has become a powerful tool for solving scientific problems over years of development, Monte Carlo algorithms also suffer from several limitations, making them inefficient when solving more challenging problems. This thesis is mainly concerned with improving the existing Monte Carlo algorithms as well as developing novel ideas that could potentially trigger extensive future works in the field of Monte Carlo methods.

In Chapter 1, we gave a review of the existing Monte Carlo methods and algorithms that are related to the work presented in this thesis. In Chapter 2, we focused on developing an improved algorithm for making inferences on the Piecewise-Deterministic Markov Processes (PDMP). We combined the idea of block sampling Doucet et al. (2006) with the existing particle filter for PDMP (Godsill and Vermaak, 2005) to obtain an improved algorithm. Simulations showed that the new algorithm is more capable of locating the jumps that are likely to be missed by the existing particle filter. A particle Gibbs sampler based on the new algorithm is also developed in the chapter. In Chapter 3 and 4, we developed an ABC-SMC algorithm based on Del Moral et al. (2012). Inspired by the fact that many problems solved by ABC-type algorithms involve a generator in which the generation process relies on the simulations of some latent random variables, we developed a modified ABC-SMC algorithm that, instead of generating these latent random variables from scratch every time, targets a specific joint distribution of the latent random variables instead. Under the same computational budget, we have numerically shown that the new algorithm achieves large improvement and can obtain more accurate approximations of the true posteriors compared to the standard ABC-SMC algorithms. Simulations also show that the new algorithm scales well in high dimensions. In Chapter 5, we turned to look at novel Monte Carlo approaches in light of Dau and Chopin (2020). We developed a novel SMC algorithm that samples Markov process snippets whose states, with proper weights, can be used to approximate expectations with respect to the target. Numerical examples indicate that this novel algorithm has significant performance improvement compared to its competitor. Lastly, the possible directions of future works based on the thesis are discussed in Chapter 5.6.

Acknowledgements

I would like to thank my supervisor Christophe Andrieu and Mark Beaumont for their patience and kindness over the past five years. It's they who did not only give me guidance on my projects but also showed me how to do research properly. It has been a rewarding journey and I have learned a lot from them.

I thank all my friends in Bristol who makes my life in Bristol full of joy and fun.

To my wife, Fan Chen - thank you for your love and support along the journey, and congratulations on officially being a Doctor. To our son, Linghe - thank you for bringing so much joy and happiness over the last seven months. You are the best gift I have ever had.

Finally, I would like to express my sincere thanks to my parents, who have selfless support for my education all the way. Without you, I would never achieve what I had achieved.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific references in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Contents

List of Figures

List of Tables

Chapter 1

Review of Monte Carlo Methods

1.1 The Monte Carlo Methods

The Monte Carlo (MC) is a well-known numerical method offering an approximation to any quantity that can be written in the form of the following integral

$$I = \mathbb{E}_\pi[\varphi(X)] = \int_{\mathcal{X}} \varphi(x)\pi(dx), \tag{1.1}$$

where $\pi(dx)$ is a probability density with respect to a probability measure. The approximation is obtained by simulating N independent random variables $X_1, X_2, .., X_N \sim \pi$ and set

$$\hat{I}^N := \frac{1}{N}\sum_{n=1}^{N} \varphi(X_n) \tag{1.2}$$

More specifically, the Monte Carlo method also provides an *particle* approximation of π through the empirical measure

$$\hat{\pi}^N(dx) := \frac{1}{N}\sum_{n=1}^{N} \delta_{X_i}(dx), \tag{1.3}$$

where δ_x represents the Dirac delta mass located at x. This means that any other quantities related to π can also be estimated by replacing π with $\hat{\pi}$. One would easily find that $\hat{I}^N = \mathbb{E}_{\hat{\pi}^N}(\varphi(X))$. One can also easily check that $\hat{I}^N$ is an unbiased estimator of I and $\mathbb{V}(\hat{I}^N) = \frac{1}{N}\mathbb{V}_\pi(\varphi)$ where $\mathbb{V}_\pi(\varphi) := \int_{\mathcal{X}} \varphi^2(x)\pi(dx) - \left[\int_{\mathcal{X}} \varphi(x)\pi(dx)\right]^2$. Clearly, the Monte Carlo methods rely on the ability to sample from π. Hence, sampling methods are indeed at the

centre of the works related to Monte Carlo methods. In the next section, two basic sampling methods will be introduced.

1.2 Rejection Sampling

Rejection sampling method dates back to Von Neumann (1963) and it is a fairly simple sampling mechanism that has been widely used in the literature. In principle, one can obtain samples from any probability distribution defined on any dimension with a density function given up to a normalising constant by *Rejection Sampling* method. Suppose that we are interested in sampling from a probability distribution π defined on $\mathbb{R}^d$ with density function

$$\pi(x) = \frac{1}{\mathcal{Z}_\pi}\tilde{\pi}(x)$$

We also assume that we can only evaluate the unnormalised density function, $\tilde{\pi}(x)$, pointwise. This means that the normalising constant $\mathcal{Z}_\pi$ is unknown. Moreover, suppose that we are able to directly sample from another distribution g and pointwise evaluate $\tilde{g}(x) = \mathcal{Z}_g g(x)$, the unnormalised density function of g. In addition, assume that the distribution g satisfies the following two conditions:

Condition 1. The support of g encompasses that of π, i.e. $f(x) > 0 \implies g(x) > 0$.

Condition 2. There exist $M > 0$, such that

$$\sup_{x \in \mathbb{R}^d} \frac{\tilde{\pi}(x)}{\tilde{g}(x)} = M < \infty$$

We will denote M the envelope constant of the *Rejection Sampling* algorithm thereafter. In practice, M does not need to be the exact supremum of the density ratio. One only needs to make sure that M is not smaller than this supremum. Based on the sampling distribution g, *Rejection Sampling* algorithm proceeds as follows:

Step 1. Generate a sample x from the distribution g

Step 2. Compute the ratio

$$\alpha(x) = \frac{\tilde{\pi}(x)}{M\tilde{g}(x)} \in [0, 1]$$

Step 3. Accept the sample x with probability $\alpha(x)$. Otherwise, GOTO *Step 1*.

The validity of the algorithm follows from the following theorem.

Theorem 1.2.1. *Let X be the random variable following distribution g. Then the accepted X in the foregoing procedure follows that target distribution π.*

Proof. Assuming that X is a continuous random variable. From the description of the rejection sampling procedure, for any $\mathcal{A} \subset \mathbb{R}^d$, we have

$$\begin{aligned}
\Pr(X \in \mathcal{A} | X \text{ is accepted}) &= \frac{\Pr(X \in \mathcal{A} \bigcap X \text{ is accepted})}{\Pr(X \text{ is accepted})} \\
&= \frac{\int_{\mathbb{R}^d} \mathbb{I}(x \in \mathcal{A}) \alpha(x) g(x)\, dx}{\int_{\mathbb{R}^d} \alpha(x) g(x)\, dx} \\
&= \frac{\int_{\mathcal{A}} \frac{\mathcal{Z}_\pi \pi(x)}{M \mathcal{Z}_g g(x)} g(x)\, dx}{\int_{\mathbb{R}^d} \frac{\mathcal{Z}_\pi \pi(x)}{M \mathcal{Z}_g g(x)} g(x)\, dx} \\
&= \frac{\mathcal{Z}_\pi / \mathcal{Z}_g \int_{\mathcal{A}} \pi(x)\, dx}{\mathcal{Z}_\pi / \mathcal{Z}_g} = \int_{\mathcal{A}} \pi(x)\, dx
\end{aligned}$$

Therefore, one can see that the accepted X is distributed according to π. If X is a discrete random variable, the argument follows similarly by replacing the integration with a summation.

□

Hence, it has been proved that *Rejection Sampling* algorithm does produce samples that are distributed according to the target distribution. More specifically, one may note that the probability of accepting a sample from g is given by

$$p = \int_{\mathbb{R}^d} \alpha(x) g(x)\, dx = \int_{\mathbb{R}^d} \frac{\mathcal{Z}_\pi \pi(x)}{M \mathcal{Z}_g g(x)} g(x)\, dx = \frac{\mathcal{Z}_\pi}{M \mathcal{Z}_g} \tag{1.4}$$

Let T be the random variable representing the number of samples from g required to produce one accepted sample, it is then easy to see that $T \sim Geo(p)$. Hence,

$$\mathbb{E}[T] = \frac{1}{p} = \frac{M \mathcal{Z}_g}{\mathcal{Z}_\pi}$$

which is the expected number of samples required to produce one accepted sample. Therefore,

the key to an efficient implementation of the *Rejection Sampling* algorithm is to find a 'good' sampling distribution g such that $\mathbb{E}[T]$ is small. In the case when $\mathcal{Z}_\pi$ and $\mathcal{Z}|g$ are known, one would aim to find a sampling distribution g such that M is small. Roughly speaking, this means that we want to find a sampling distribution g that has a similar shape to the target distribution π.

Rejection Sampling method is a generic sampling method that is exact and conceptually easy. Moreover, the samples obtained from it are independent. However, one may find it difficult to search for good sampling distributions that have similar shapes to the target distributions in practice. As a result, this may result in a large value of M, which means an inefficient implementation of the algorithm. Another main drawback for *Rejection Sampling* method is its reduced efficiency as the dimension of the sampling problem gets larger. Hence, for high-dimensional problems, one may prefer using other techniques such as Markov chain Monte Carlo and sequential Monte Carlo methods. The details of these methods will be reviewed in later sections of this chapter.

1.3 Importance Sampling

As we have seen before, standard Monte Carlo methods require one to be able to sample from some target density π. However, this is in most scenarios difficult or even impossible to achieve. In this case, one may consider using the technique called Importance Sampling (IS), which approximates the expectations with respect to π by simulations from a different distribution q, which is easy to sample from. Given that the support of π is contained in that of q, i.e. $q(x) > 0$ whenever $\pi(x) > 0$, the IS technique relies on the following identity

$$I = \mathbb{E}_\pi[\varphi(X)] = \int_{\mathcal{X}} \varphi(x)\pi(x)dx = \int_{\mathcal{X}} \varphi(x)\frac{\pi(x)}{q(x)}q(x)dx = \int_{\mathcal{X}} \varphi(x)w(x)q(x)dx, \quad (1.5)$$

where $w(x) = \pi(x)/q(x)$ is often referred as the importance weight, $\pi(x)$ the target density and $q(x)$ the proposal density. Since it is assumed that we can sample from q easily, given a sample $X_1, .., X_N$ from q, an estimator for I can then obtained by

$$\hat{I} = \frac{1}{N}\sum_{n=1}^{N} \varphi(X_n)\frac{\pi(X_n)}{q(X_n)} = \frac{1}{N}\sum_{n=1}^{N} w(X_n)\varphi(X_n) \quad (1.6)$$

The estimator in (1.6) assumes that one could evaluate both π and q pointwise. In the situation when π and/or q can only be evaluated up to proportionality, i.e. we only know $\pi(x) = \tilde{\pi}(x)/\mathcal{L}_\pi$ and $q(x) = \tilde{q}/\mathcal{L}_q$, where $\mathcal{L}_\pi$ and $\mathcal{L}_q$ are the normalising constants for π and q and are both intractable. In this case, we should have that

$$\int_{\mathcal{X}} \frac{\tilde{\pi}(x)}{\tilde{q}(x)} \varphi(x) q(x) dx = \frac{\mathcal{L}_\pi}{\mathcal{L}_q} I. \tag{1.7}$$

Hence, one can see that the estimator in (1.6) no longer works. In this case, one could use the identity

$$\int_{\mathcal{X}} \frac{\tilde{\pi}(x)}{\tilde{q}(x)} q(x) dx = \frac{\mathcal{L}_\pi}{\mathcal{L}_q}.$$

As a result, an estimator for I can then be obtained by

$$\hat{I} = \frac{\sum_{n=1}^N w(X_n)\varphi(X_n)}{\sum_{n=1}^N w(X_n)}, \tag{1.8}$$

given that $X_1, ..., X_N \sim q$. Importance Sampling provides an alternative way of estimating the expectation of interest even when the target distribution is intractable. However, the performance of IS relies heavily on the choice of the proposal distribution q. A good choice of q should satisfy two conditions: (1) It's a tractable distribution that is easy to sample from (2) It should result in an estimator $\hat{I}$ with as small MSE as possible. Practically, one should look for tractable distributions that are as close to π as possible in order to obtain good IS estimators.

To measure the efficiency of an IS estimator, a common choice is to use the Effective Sample Size (ESS), which is defined as

$$\text{ESS} = \frac{(\sum_{n=1}^N w(X_n))^2}{\sum_{n=1}^N w(X_n)^2}. \tag{1.9}$$

An interpretation for ESS is the equivalent sample size required to achieve the same level of precision if that sample came from π. It can be seen that in the optimal scenario, we should have $w(X_n) := 1$ for all $n = 1, .., N$. In this case, ESS $= N$, which is the maximum.

Although Importance Sampling solves the problem when we cannot sample directly from the

target, it is well-known that IS suffers from the curse of dimensionality. As the dimension of X increases, the variances of the importance weight $w(X)$ will also increase. This makes it extremely difficult to implement efficient Importance Sampling at high dimensions.

1.4 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods are a class of methods that are designed to solve statistical inference problems recursively. It can be seen as a special class of Importance Sampling (IS) methods which are Monte Carlo methods that construct an approximation using samples from a proposal distribution and the corresponding importance weights. The SMC methods provide a way to solve the dimensionality problems of the standard IS methods.

Suppose that we have a sequence of distributions, $(\pi_n^\theta)_{n\in\mathbb{N}}$, which are defined on spaces of increasing dimension, $\left(E^{(n)}\right)_{n\in\mathbb{N}}$. Furthermore, we define each distribution π_n^θ as a joint distribution of variables $x_{1:n} := (x_1, x_2, .., x_n)$, where $n = 1, 2, .., P$.

We can also assume, from now on, that only unnormalised versions of $(\pi_n^\theta)_{n\in\mathbb{N}}$ can be evaluated, i.e.

$$\pi_n^\theta := \gamma_n^\theta / Z_n^\theta,$$

where $Z_n^\theta > 0$ are normalising constants, and only γ_n^θ can be evaluated.

For each distribution π_n, suppose that samples of $x_{1:n}$ can be obtained from a proposal distribution $q_t(dx_{1:n})$, the Importance Sampling method creates an approximation of $\pi_n(dx_{1:n})$ by a collection of samples $\left\{X_{1:n}^j, j = 1, 2, ..., N\right\}$ and their corresponding normalised weights $\left\{W_n^j, j = 1, 2, ..., N\right\}$. The samples from $q_n(dx_{1:n})$ are also called particles in many situations and we will refer to these samples as particles in later parts. For each particle $X_{1:n}^j$, the corresponding unnormalised weight w_n^j can be calculated by

$$w_n^j = \frac{\gamma_n\left(x_{1:n}^j\right)}{q_n\left(x_{1:n}^j\right)}, \tag{1.10}$$

which can be normalised by $W_n^j := w_n^j / \sum_{i=1}^N w_n^i$. Based on the particles and the correspond-

ing weights, $\pi_n(dx_{1:n})$ can then be approximated by

$$\hat{\pi}_n(dx_{1:n}) := \sum_{j=1}^{N} W_n^j \delta_{x_{1:n}^j}(dx_{1:n}). \tag{1.11}$$

However, such a direct implementation of the Importance Sampling method is in fact far beyond practical. To obtain a good approximation of π_n from Importance Sampling, we should carefully design a proposal q_n that has a heavier tail than the target and concentrates on regions of high density of the target. A poorly designed target will make the importance weights have infinite variance and make the method computationally inefficient. However, it is often very hard to design such a good proposal, especially when the target becomes high dimensional as n increases. To get around such difficulties, we can instead employ the Importance Sampling sequentially and use a kind of divide-and-conquer idea to tackle the problem. This results in the Sequential Importance Sampling (SIS) method, which can be treated as a special case of Importance Sampling. In SIS, we design a proposal density with a Markovian structure. Instead of designing $q_n(dx_{1:n})$ for each n, the proposal density q_n can be seen as a propagation from q_{n-1} such that

$$q_n(dx_{1:n}) := q_{n-1}(dx_{1:n-1}) q_n(dx_n | x_{1:n-1}).$$

By choosing such a type of proposal, we divide the proposal problem into several conditional distributions. Therefore, one does not need to sample $x_{1:n-1}$ again when obtaining particles for approximating π_n. Instead, the particles $x_{1:n-1}^j$ obtained from the previous step can be reused to form $x_{1:n}^j$ by sampling $x_n \sim q_n\left(\cdot | x_{1:n-1}^j\right)$ and appending it to $x_{1:n-1}^j$. In this

situation, the unnormalized importance weight can be calculated by

$$w_n^j := \frac{\gamma_n\left(x_{1:n}^j\right)}{q_n\left(x_{1:n}^j\right)} = \frac{\gamma_n\left(x_{1:n}^j\right)}{q_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)}$$

$$= \frac{\gamma_{n-1}\left(x_{1:n-1}^j\right)}{q_{n-1}\left(x_{1:n-1}^j\right)} \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)} \tag{1.12}$$

$$= w_{n-1}^j \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)}.$$

We summarise the Sequential Importance Sampling method in Algorithm 1.1. Note that at each step, the importance weights are obtained by multiplying by an increment to the importance weights obtained from the previous step. To make the representations simpler, we define

$$G_n(x_{1:n}) := \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1}) q_n^\theta(x_n | x_{1:n-1})}$$

to be the incremental weight at step n. However, Sequential Importance Sampling also suffers from the problem that the estimation variance scales exponentially with the dimension of the problem. As n increases, the variances generally increase exponentially. If we look at the normalised weight at each step, we can see that the maximum unnormalized weight, $\max_{j=1,..,N} W_n^j$, will quickly become almost 1, making other weights approach zero as n increases. As a consequence, the target distributions at each stage will be approximated by only one effective particle, resulting in a large variance in the estimation. This is known as weight degeneracy

Such a drawback can be alleviated by cleverly choosing a proposal distribution that incorporates the information in $\hat{\pi}_{n-1}(dx_{1:n-1})$ obtained from the previous Monte Carlo estimation step. This results in the Sequential Monte Carlo methods. In the first step, Sequential Monte Carlo obtains samples $X_1^{1:N} \sim q_1(dx_1)$ just like the Sequential Importance Sampling does.

Algorithm 1.1: Sequential Importance Sampling (SIS)

for *n = 1,2,3,...,P* **do**
  for *j = 1,2,...,N* **do**
    Sample $X_{1:n}^j \sim q_n\left(\cdot|x_{1:n-1}^j\right)$;
    Set $X_{1:n}^j := \left(X_{1:n-1}^j, X_n^j\right)$;
    Calculate the unnormalized weight by ;

$$w_n^j := w_{n-1}^j \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j|x_{1:n-1}^j\right)}$$

  Set the normalized weight by

$$W_n^j := w_n^j / \sum_{i=1}^N w_n^i$$

  Approximate $\pi_n\left(dx_{1:n}\right)$ by

$$\hat{\pi}_n(dx_{1:n}) := \sum_{j=1}^N W_n^j \delta_{x_{1:n}^j}(dx_{1:n})$$

The importance weight at this iteration will then be given by

$$w_1^j := \frac{\gamma_1\left(x_1^j\right)}{q_1\left(x_1^j\right)}, \tag{1.13}$$

and normalised to $W_1^j := w_1^j / \sum_{i=1}^N w_1^i$. Approximation of $\pi_1(dx_1)$ can then be obtained by

$$\hat{\pi}_1\left(dx_1\right) := \sum_{j=1}^N W_1^j \delta_{X_1^j}\left(dx_1\right). \tag{1.14}$$

At later iterations of the SMC, we sample $X_{1:n}^{1:N}$ from different proposals as with the SIS method. Instead of using $q_{n-1}(dx_{1:n-1})q_n(dx_n|x_{1:n-1})$ as the proposal, particles are obtained from $\hat{\pi}_{n-1}(dx_{1:n-1})q_n(dx_n|x_{1:n-1})$. Simulation from $\hat{\pi}_{n-1}(dx_{1:n-1})q_n(dx_n|x_{1:n-1})$ can be broken into two steps: sampling $\tilde{X}_{1:n-1}^j \sim \hat{\pi}_{1:n-1}$ and $X_n^j \sim q_n\left(\cdot|\tilde{X}_{1:n-1}^j\right)$ and concatenating them to form $X_{1:n}^j$. Sampling from $\hat{\pi}_{n-1}$ is often referred to as a resampling since we are sampling from a distribution that itself is obtained from sampling. It can be seen as a random sampling from $X_{1:n-1}^{1:N}$ with replacement according to the weights $W_{n-1}^{1:N}$. This means that

the probability of choosing the j-th particle is just W_{n-1}^j. As a result, we will have that the expected number of times the j-th particle is resampled is $E\left\{\mathcal{N}_n^j\right\}$, will be given by

$$E\left(\mathcal{N}_n^j\right) = NW_n^j$$

Since the particles are sampled from $\hat{\pi}_{n-1}(dx_{1:n-1})q_n(dx_n|x_{1:n-1})$, they are approximately distributed according to $\pi_{n-1}(dx_{1:n-1})q_n(dx_n|x_{1:n-1})$. As a result, the corresponding importance weight will be obtained by

$$w_n^j := \frac{\gamma_n\left(\tilde{x}_{1:n-1}^j, x_n^j\right)}{\gamma_{n-1}\left(\tilde{x}_{1:n-1}^j\right) q_n\left(x_n^j|\tilde{x}_{1:n-1}^j\right)} \tag{1.15}$$

which is normalised to $W_n^j := w_n^j / \sum_{i=1}^N w_n^i$. For the resampling step, we can treat X_n^j as an offspring of particle A_{n-1}^j at iteration $n-1$. This interpretation was proposed in Andrieu et al. (2010). As a result, resampling particles can be viewed as sampling indices $\mathbf{A}_{n-1} := \left(A_{n-1}^1, ..., A_{n-1}^N\right) \sim r\left(\cdot|\mathbf{W}_{n-1}\right)$ with $r\left(\cdot|\mathbf{W}_{n-1}\right)$ being any kernel such that $r\left(j|\mathbf{W}_{n-1}\right) = W_{n-1}^j$ and $X_n^j \sim q_n\left(\cdot|X_{1:n-1}^{A_{n-1}^j}\right)$. One can also keep track of the ancestor lineage of the particles at time n by defining $B_{n|n}^i := i$ and

$$B_{t|n}^i = A_t^{B_{t+1|n}^i}$$

for $t = n-1, ..., 1$. Then each particle lineage at step n can be expressed in an alternative way,

$$X_{1:n}^i = \left(X_{1:n-1}^{A_{n-1}^i}, X_n^i\right) = \left(X_1^{B_{1|n}^i}, ..., X_n^{B_{n|n}^i}\right)$$

SMC method is sometimes also referred to as Sequential Importance Resampling method. Since resampling will introduce extra variance to the estimation of $\pi_n(dx_{1:n})$ as shown by Chopin (2004), it is generally preferable to use (1.11) to approximate $\pi_n(dx_{1:n})$ instead of using the equally weighted resampled particles. However, by inserting a resampling step, we can get rid of the particles of pretty low weights and focus our computation on regions with high probability density.

Algorithm 1.2: Sequential Importance Resampling (SIR)

for *n=1* **do**

Sample $X_1^j \sim q_1(dx_1)$ for $j = 1, ..., N$;

Compute the unnormalized weight by

$$w_1^j := \frac{\gamma_1\left(x_1^j\right)}{q_1\left(x_1^j\right)}$$

Set $W_1^j = w_1^j / \sum_{i=1}^N w_1^i$;

Approximate $\pi_1(dx_1)$ by

$$\hat{\pi}_1(dx_1) := \sum_{j=1}^N W_1^j \delta_{X_1^j}(dx_1)$$

for *n = 1,2,3,...,P* **do**

for *j = 1,2,...,N* **do**

Sample $A_{n-1}^j \sim \mathbf{r}(\cdot|\mathbf{W}_{n-1})$ with $\Pr(A_{n-1}^j = k|\mathbf{W}_{n-1}) = W_{n-1}^k$;

Sample $X_{1:n}^j \sim q_n\left(\cdot|x_{1:n-1}^{A_{n-1}^j}\right)$;

Set $X_{1:n}^j := \left(X_{1:n-1}^{A_{n-1}^j}, X_n^j\right)$;

Calculate the unnormalized weight by

$$w_n^j := \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^{A_{n-1}^j}\right) q_n\left(x_n^j|x_{1:n-1}^{A_{n-1}^j}\right)}$$

Set the normalised weight by

$$W_n^j := w_n^j / \sum_{i=1}^N w_n^i$$

Approximate $\pi_n\left(dx_{1:n}\right)$ by

$$\hat{\pi}_n(dx_{1:n}) := \sum_{j=1}^N W_n^j \delta_{x_{1:n}^j}(dx_{1:n})$$

1.5 Markov Chain Monte Carlo Methods

Markov Chain Monte Carlo (MCMC) methods are a broad class of methods that can be used to sample from a density π that is defined on a measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$, where one only

needs to evaluate π pointwise up to proportionality. The MCMC then tries to sample from a specially designed Markov chain whose stationary distribution is π. Within the MCMC field, the most famous algorithm is the Metropolis-Hastings (MH) algorithm. In the MH algorithm and suppose that the Markov chain is currently at state x, a new state y is then proposed according to a proposal distribution $q(x, y)$. Then, with probability

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\}, \tag{1.16}$$

the next state is set to be y. Otherwise, the Markov chain stays at x. The above procedure implicitly defines a Markov kernel of the form

$$K(x, y) = q(x, y)\alpha(x, y) + (1 - \alpha(x))\delta_x(y), \tag{1.17}$$

where $\alpha(x) = \int q(x, y)(1 - \alpha(x, y))dy$. The Markov kernel defined here is reversible since the detailed balance property is satisfied here:

$$\begin{aligned}\pi(x)q(x, y)\alpha(x, y) &= \pi(x)q(x, y)\min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\} \\ &= \min\{\pi(x)q(x, y), \pi(y)q(y, x)\} \\ &= \pi(y)q(y, x)\min\left\{1, \frac{\pi(x)q(x, y)}{\pi(y)q(y, x)}\right\} = \pi(y)q(y, x)\alpha(y, x).\end{aligned} \tag{1.18}$$

Therefore, it is to check that K will leave π invariant. The MH is a very powerful algorithm such that most MCMC algorithms can be viewed as the MH algorithm with specially designed proposal distribution or an extension to it. One of the common choice for q would be a Normal distribution centred at x with covariances chosen by the user, i.e. $q(x, y) = \mathcal{N}(y; x, \Sigma)$. If such a proposal is used, we often refer to the specific algorithm as the random-walk Metropolis-Hastings algorithm. We will use the same name to refer to this type of MH algorithm in later parts of the thesis.

1.6 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (Duane et al., 1987, Neal, 2011) is a class of Monte Carlo methods that promises better scalability compared to general-purpose MCMC algorithms such as the

random-walk Metropolis-Hastings algorithm. Suppose that we are interested in sampling from the distribution $\pi(x)$ where $x \in \mathbb{R}^d$. HMC introduces an auxiliary momentum variable $\mathbf{p}$ and defines the joint distribution $\bar{\pi}(x, \mathbf{p}) \propto \pi(x)\varpi(\mathbf{p})$ on the extended space $\mathbb{R}^d \times \mathbb{R}^d$. Moreover, ϖ is often chosen to be a multivariate Gaussian distribution with mean $\mathbf{0}$ and variance $\mathcal{I}_d$, i.e. $\varpi(\mathbf{p}) := \mathcal{N}(\mathbf{p}; \mathbf{0}, \mathcal{I}_d)$. Also, we can write $\pi(x) = \exp(-U(x))$ and define the corresponding Hamiltonian to be

$$H(x, \mathbf{p}) := U(x) + 1/2\mathbf{p}^T\mathbf{p},$$

where $U(x)$ is often viewed as the potential energy of a system at position x and $1/2\mathbf{p}^T\mathbf{p}$ can be viewed as the corresponding kinetic energy at that position. Assuming that $U(x)$ is differentiable, the evolution of the system can be described by the following Hamiltonian dynamics

$$\begin{aligned} \frac{dx}{dt} &= \mathbf{p} \\ \frac{d\mathbf{p}}{dt} &= -\nabla U(x). \end{aligned} \tag{1.19}$$

Contrary to the random-walk Metropolis-Hastings algorithm, which only generates proposals in a small neighbour of the current state, using the Hamiltonian dynamics can produce proposals that are far away from the current state. In addition, the Hamiltonian dynamics is also reversible, conservative and volume preserving (Neal, 2011). This means that proposals obtained from exact Hamiltonian dynamics will always be accepted. Hence, the mixing of the resulting Markov chain will be significantly improved.

In practice, however, the analytical solution to (1.19) is rarely available and an approximation to the solution for $t \in [0, \tau]$ is obtained by discretising the time with small stepsize ϵ for $L = \tau/\epsilon$ steps. A commonly chosen discretising scheme is the leap-frog method, which works as follows:

$$\begin{aligned} \mathbf{p}(t + \epsilon/2) &= \mathbf{p}(t) - \frac{\epsilon}{2}\nabla U(x(t)) \\ x(t + \epsilon) &= x(t) + \epsilon\mathbf{p}(t + \epsilon/2) \\ \mathbf{p}(t + \epsilon) &= p(t + \epsilon/2) - \frac{\epsilon}{2}\nabla U(x(t + \epsilon)) \end{aligned} \tag{1.20}$$

to produce one-step update starting from $(x(t), \mathbf{p}(t))$. We use $\psi_\epsilon : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ to

denote the one-step leap-frog update with stepsize ϵ, i.e. $(x(t+\epsilon), \mathbf{p}(t+\epsilon)) = \psi_\epsilon(x(t), \mathbf{p}(t))$. Given the current state $(x, \mathbf{p})$, the HMC algorithm completes one iteration by performing the following:

Step 1. A new momentum $\mathbf{p}_{new}$ is sampled from $\varpi(\cdot)$.

Step 2. Obtain the proposal by setting $(x', \mathbf{p}') := \sigma \circ \psi_\epsilon^L(x, \mathbf{p}_{new})$ with $\sigma(x, \mathbf{p}) := (x, -\mathbf{p})$.

Step 3. Accept $(x', \mathbf{p}')$ to be the next state with probability

$$\min\left\{1, \exp\left(-H(x', \mathbf{p}') + H(x, \mathbf{p}_{new})\right)\right\}.$$

Otherwise, stay at the current state.

Step 1 of the above scheme samples a new momentum from ρ and due to its independence to $\pi(x)$, such a sampling step will leave $\bar{\pi}$ invariant. Moreover, $\phi = \sigma \circ \psi_\epsilon^L$ is an involution. With the acceptance probability given by $\min(1, \bar{\pi} \circ \phi(x, \mathbf{p})/\bar{\pi}(\mathbf{p}))$, the Markov kernel defined in *Step 2* and *3* will also leave $\bar{\pi}$ invariant (Andrieu et al., 2020).

1.6.1 HMC with boundary reflections

In this section, we introduce the HMC samplers with boundary reflections, which can be viewed as a modification on the standard HMC methods. Suppose that we are interested in sampling from $\pi(x)$ with the constraint $C(x) \geq 0$ where $C : \mathbb{R}^d \to \mathbb{R}$ is a piecewise smooth function. This can also be viewed as sampling from

$$\tilde{\pi}(x) \propto \begin{cases} \pi(x), & C(x) \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1.21}$$

Such sampling problem has been discussed in several places in the literature (e.g. Betancourt (2011), Pakman and Paninski (2013, 2014), Mohasel Afshar and Domke (2015), Chevallier et al. (2018)). Although different in details of implementations, the methods proposed in the literature all make reflections against the boundary defined by $C(x) = 0$ when x reaches or steps outside the boundary. From the Hamiltonian point of view, such a constraint imposes an infinite potential energy barrier in $\mathbb{R}^d$ (Betancourt, 2011). Hence, it's reasonable to make

bounces when the Hamiltonian equation integrates outside the boundary from the physical intuition.

Consider the one-step leap-frog update starting from $(x(t), \mathbf{p}(t))$, if $x(t+\epsilon) := x(t) + \epsilon\mathbf{p}(t+\epsilon/2)$ is outside the boundary, i.e. $C(x(t+\epsilon)) < 0$, we follow Algorithm 1.3 to perform the leap-frog update with reflections. Note that when the set $\{t > 0 | C(x + t\mathbf{p}_{1/2}) = 0\} = \emptyset$, its infimum is set to be ∞. In fact, the leap-frog discretisation with reflections splits the straight line motion from x to $x+\epsilon\mathbf{p}_{1/2}$ into several line segments that all lie within the region bounded by $C(x) = 0$. We use $\psi_\epsilon^r : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$ to represent the map implicitly defined by Algorithm 1.3 and one can easily check that ψ_ϵ^r is also time-reversal. Hence, the proposal defined by $\phi_{\epsilon,L}^r := \sigma \circ \psi_\epsilon^{r,L}$ is still an involution, which implies that with the same form of acceptance probability, the HMC with reflections still leaves $\tilde{\pi}(x)$ invariant.

Algorithm 1.3: Reflective Leap-frog, $\psi_\epsilon^r(x, \mathbf{p})$

Input :
- The starting point $(x, \mathbf{p})$.
- The discretisation length, ϵ.
- The potential energy function $U(x)$ and its gradient function $\nabla U(x)$.
- The piecewise smooth function C defining the boundary.

Set $t_r := \epsilon$;
Set $\mathbf{p}_{1/2} = \mathbf{p} - \frac{\epsilon}{2}\nabla U(x)$;
Set $t_b := \textbf{Find\_First\_Bounce\_Time}(x, \mathbf{p}_{1/2}, C) := \inf\{t > 0 | C(x + t\mathbf{p}_{1/2}) = 0\}$;
while $t_b < t_r$ **do**
    Set $x := x + t_b\mathbf{p}_{1/2}$;
    Calculate $\mathbf{n} := \nabla C(x)/||\nabla C(x)||$;
    Set $\mathbf{p}_{1/2} := \mathbf{p}_{1/2} - \langle \mathbf{p}_{1/2}, \mathbf{n}\rangle \mathbf{n}$;
    Set $t_r = t_r - t_b$;
    Set $t_b := \textbf{Find\_First\_Bounce\_Time}(x, \mathbf{p}_{1/2}, C)$;
Set $x' := x + t_r\mathbf{p}_{1/2}$;
Set $\mathbf{p}' := x' - \frac{\epsilon}{2}\nabla U(x')$

The HMC with reflections discussed above can also be viewed as a generalisation of the billiard-walk sampler (Gryazina and Polyak (2014), Polyak and Gryazina (2014)) with fixed stepsize. Instead, one could also simulate a random stepsize from e.g. an Exponential distribution at each time. To avoid too many reflections at some point, one could also perform early rejections if the number of reflections exceeds a certain threshold (Chevallier et al. (2018)).

Chapter 2

Static Parameter Estimations of Piecewise Deterministic Markov Models using Particle Gibbs samplers

2.1 Introduction

Piecewise deterministic processes (PDPs) are stochastic processes that jump randomly at a countable number of time points but otherwise evolve deterministically in continuous time. Practically, we may only be able to observe such processes in discrete time. Most of the time, such discrete time observations also come with noise or one cannot even observe the process directly. This often makes it much more difficult to perform inferences on PDPs.

To make inferences on the PDPs, several particle filtering algorithms have been proposed in the literature. Godsill and Vermaak (2005) proposed algorithms termed as *variable rate particle filters* (VRPFs), that are based around the bootstrap approach and split the time into disjoint time blocks and sample the jumps in each time block sequential by a particle filter. A corresponding smoothing algorithm was later proposed in Bunch and Godsill (2013). However, if the time is discretised in a way such that there are jumps near the endpoints of the time blocks, the VRPF algorithm may likely miss these jumps and the missing jumps will not be recovered through smoothing either. This will make the estimation of the PDPs by

VRPF algorithms unreliable, especially for the times that are near the endpoints of the time blocks.

A more sophisticated particle filtering algorithm that is based on the sequential Monte Carlo samplers of Del Moral et al. (2006) was introduced by Whiteley et al. (2011), in which the sampled jumps are modified in light of new observations and the auxiliary backward kernels are included to make sure that the particle filter still has the actual target distribution as marginals. However, this algorithm is not suitable for smoothing techniques as most jumps contained in X_{n-1} would coincide with the jumps contained in X_n, making the backward transition kernel almost degenerate. Moreover, the way the algorithm is designed in Whiteley et al. (2011) introduces an approximation to the actual distribution of the PDPs and such an approximation becomes more obvious when it is likely to have more than one jump within a time block. Finke et al. (2014) reformulated the algorithm in Whiteley et al. (2011), making it more suitable for variance reduction techniques such as backward simulation. However, the approximating problem was not solved, and it was shown numerically in Finke et al. (2014) that this can bring bias to the static parameter estimations in certain cases.

For the estimation of the static parameters in PDPs, several attempts have also been made previously in the literature. Centanni and Minozzo (2006*a*) and Centanni and Minozzo (2006*b*) introduced a sampler that is based around the reversible-jump MCMC of Green (1995). An SMC approach was proposed by Del Moral et al. (2007) and was improved by Martin et al. (2013). Rao and Teg (2013) designed a Gibbs sampler that is suitable for the classes of PDPs in which the state space is discrete. Finke et al. (2014) proposed to apply the particle Gibbs sampler with the reformulated particle filter to make inferences on the static parameters of the PDPs.

In this chapter, we propose a modified particle filter that combines the VRPF sampler and the block sampling algorithm introduced in Doucet et al. (2006) to make inferences on the PDPs. This new algorithm is termed as *BlockVRPF*. We also apply the particle Gibbs sampler with the BlockVRPF algorithm to estimate the posterior distributions of the static parameters in the PDPs. Our main contributions are as follows.

1. We implement a new particle filter to estimate the PDPs of interest given discretely and

particle observed noisy observations, termed as the BlockVRPF sampler, that addresses the limitations faced by the original VRPF algorithm while at the same time does not introduce any approximations like the algorithms in Whiteley et al. (2011) and Finke et al. (2014) does. Moreover, the BlockVRPF sampler is still suitable for backward samplings.

2. We provide explanations of the importance of the rejuvenation step proposed in Finke et al. (2014) from a different perspective and discuss the importance of appropriately choosing the auxiliary backward kernels to ensure that the rejuvenation step does bring improvement on the mixing of the particle Gibbs sampler.

We show the improvements on the VRPF sampler brought by the BlockVRPF algorithm in a toy filtering problem. Moreover, we demonstrate that using BlockVRPF will not bring biases to the parameter estimations through two numerical examples.

2.2 Piecewise Deterministic Processes

In this section, we follow the descriptions in Whiteley et al. (2011) and Finke et al. (2014) to give an introduction of discretely observed piecewise deterministic processes (PDPs). These are stochastic processes that jump randomly at an almost surely countable number of random times but otherwise evolve deterministically in continuous time. We also provide examples considered throughout this work.

Let $(\tau_j, \phi_j)_{j\in\mathbb{N}}$ be a stochastic process that represents the random jump times and the corresponding jump values. Moreover, all the τ's will take values such that $\tau_0 = 0$ and $\tau_0 < \tau_1 < \tau_2 < ....$ We also define Φ to be the support for all the jump values $(\phi_j)_{j\in\mathbb{N}}$. A Piecewise Deterministic Process (PDP) is a continuous time stochastic process $(\zeta_t)_{t\geq 0}$ such that $\zeta_0 := \phi_0$ and

$$\zeta_t := F^{\theta}(t|\tau_{v_t}, \phi_{v_t})$$

where $v_t := \sup\{j \in \mathbb{N}|\tau_j \leq t\}$ represents the latest jump time before time $t \in \mathbb{R}_+$. Hence, a piecewise deterministic process will evolve deterministically according to F^{θ} after time τ_j until it reaches the next jump time τ_{j+1}. Here, we use θ to represent all the static parameters used in the model. Suppose that we are interested in such a process up to time T and define

$k := v_T$ to be the number of jumps before time T and $\zeta_{(a,b]} := \{\zeta_t | t \in (a,b]\}$ be the PDP in the time interval $(a,b]$. It is clear that the process $\zeta_{(0,T]}$ can be completely determined by $(k, \tau_{1:k}, \phi_{1:k}, \phi_0)$ and the deterministic function F^θ. For simplicity, we follow Whiteley et al. (2011) to propose a Markovian prior on the jump times and values in the interval $(0,T]$, i.e.

$$p^\theta(k, \tau_{1:k}, \phi_{1:k}, \phi_0) = S^\theta(\tau_k, T) q_0^\theta(\phi_0) \mathbb{I}(0 < \tau_1 < \tau_2 < .. < \tau_k < T) \times \prod_{j=1}^{k} f^\theta(\tau_j | \tau_{j-1}) g^\theta(\phi_j | \tau_j, \tau_{j-1}, \phi_{j-1}), \quad (2.1)$$

where $S^\theta(\tau_k, T) := 1 - \int_{\tau_k}^{T} f^\theta(s|\tau_k)\, ds$ denotes the probability that no jump occurring in the interval $(\tau_k, T]$ and f^θ, g^θ represents the conditional probability density of the jump times and the associated jump values. Note that we also included the density of k, the number of jumps up to T in the density p. The reason why we included it is that p can therefore be defined on the space

$$\tilde{E} = \bigcup_{k=0}^{\infty} \left\{ \{k\} \times \mathbb{T}_{(0,T],k} \times \Phi^{k+1} \right\},$$

where $\mathbb{T}_{(a,b],k} := \left\{ (\tau_1, ..., \tau_k) \in (0,\infty)^k : a < \tau_1 < ... < \tau_k \leq b \right\}$. The Markovian structure of the process implies that inter-jump times $\tau_n - \tau_{n-1}$ are independent with each other and the jump value ϕ_n at τ_n will only depend on the previous jump value ϕ_{n-1} and the latest inter-jump time $\tau_n - \tau_{n-1}$.

In real situations, such a continuous time stochastic process can only be observed at discrete times with some measurement errors. Let $y_{(s,t]}$ be the observations obtained in the interval $(s,t]$ and $p^\theta(y_{(s,t]} | \zeta_{(s,t]})$ be the density of the observations given the PDP. We also assume that the observations obtained in disjoint intervals are conditionally independent given the PDP. Hence, we will have that

$$p^\theta(y_{(0,T]} | \zeta_{(0,T]}) = p^\theta(y_{(\tau_k,T]} | \tau_k, \phi_k) \prod_{j=1}^{k} p^\theta(y_{(\tau_{j-1},\tau_j]} | \tau_{j-1}, \phi_{j-1}). \quad (2.2)$$

The posterior density of the jump times and values up to t will then be given by

$$\begin{aligned}\pi^\theta(\zeta_{(0,T]}|y_{(0,T]}) &= \pi^\theta(k,\tau_{1:k},\phi_{0:k}|y_{(0,T]}) = \gamma^\theta(k,\tau_{1:k},\phi_{0:k}|y_{(0,T]})/Z^\theta \\ &= p^\theta(k,\tau_{1:k},\phi_{1:k},\phi_0)p^\theta(y_{(0,T]}|\zeta_{(0,T]})/Z^\theta,\end{aligned} \tag{2.3}$$

where Z^θ is the normalising constant, which is typically unknown. We will refer to this posterior distribution as π in future discussions when θ is known. In practice, we are often interested in the Bayesian inference, based on the posterior distributions of the jumps defined in (2.3) with known static model parameters θ. It is more common to make Bayesian inference on the static parameter θ. In this case, we assign a prior $\pi(\theta)$ to the parameter and try to find the posterior distribution of θ

$$\pi(\theta|y_{(0,T]}) = \int \pi(\theta)\pi^\theta(dk, d\tau_{1:k}, d\phi_{0:k}|y_{(0,T]}).$$

Since the integral involved in the above expression is in general intractable, we will often turn to employ Monte Carlo methods to perform such inference.

2.2.1 Elementary Change-point Model

The first example we consider is the elementary change-point model. We assume that the interjump times are distributed according to a *Gamma* distribution with shape and scale parameters equal to α and β. Moreover, the interjump times are assumed to be independent of each other. Given the jump times, the corresponding jump values are assumed to follow a Gaussian AR(1)-process, i.e. $g(\phi_n|\phi_{n-1},\tau_{n-1},\tau_n) = \mathcal{N}(\phi_n;\rho\phi_{n-1},\sigma_\phi^2)$, where $\rho \in \mathbb{R}$ and $\sigma_\phi^2 \in (0,\infty)$ are the static parameters of the model. Having the jump times and values, the deterministic function of the change-point model is piecewise constant, i.e. $F(t|\tau,\phi) := \phi$. Observations are then obtained at regular times $k\Delta, k = 1,2,...$ and are obtained with a Gaussian noise of mean 0 and variance σ_y^2. Hence, the static parameters of the model are given by $\theta := (\rho,\sigma_\phi^2,\sigma_y^2,\alpha,\beta) \in \mathbb{R}\times(0,\infty)^4$.

Figure 2.1 shows the artificial data simulated from the model with $\alpha = 4, \beta = 10$ and $(\rho,\sigma_\phi^2,\sigma_y^2) := (0.9,1.0,0.5)$. Moreover, we take $\Delta := 1$ and $T := 1,0000$ for the simulated data.

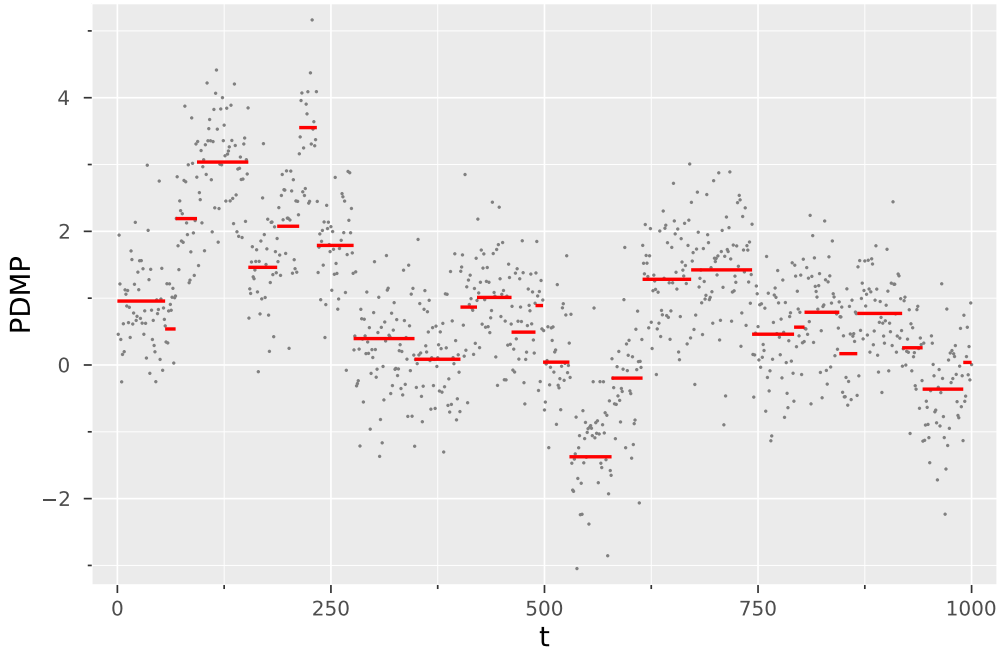

Figure 2.1: Simulated data of the change-point model described in this section. The static parameter take values $(\rho, \sigma_\phi^2, \sigma_y^2, \alpha, \beta) := (0.9, 1.0, 0.5, 4.0, 10.0)$. The grey points are the noisy observations recorded at every $\Delta := 1$. The red lines represent the actual PDMP

2.2.2 Shot-noise Cox Model

Another model we consider is the *Shot-noise Cox* model which was introduced by Cox and Isham(1980). It plays a central role in modelling claims arrivals in the insurance market, for example. The *Shot-noise Cox* model can be treated as a generalised Poisson point process with more flexibility on the intensity, allowing it to be a stochastic process as well. Let ζ_t be a shot-noise intensity process which is unobservable. In the context of insurance claims, ζ_t can be interpreted as follows. Catastrophic events occur at random times $\{\tau_n\}_{n=1,2,3,\ldots}$ resulting in a sudden jump on the intensity of claim arrivals. The corresponding jump values ϕ_n would then depend on the severity of the catastrophic event causing such a jump. Between τ_n and τ_{n+1}, the intensity will gradually decay as more and more claims have been settled until the advent of the next catastrophic event.

In this numerical example, the interjump times are assumed to be exponentially distributed with rate λ_τ, i.e.

$$f(\tau_n|\tau_{n-1}) = \lambda_\tau \exp(-\lambda_\tau \exp(\tau_n - \tau_{n-1})) \times \mathbb{I}(\tau_n > \tau_{n-1}).$$

Furthermore, we assume that the initial jump value, ϕ_0, is distributed according to an exponential distribution with rate λ_ϕ, i.e.

$$g(\phi_0) = \lambda_\phi(-\lambda_\phi\phi_0)\mathbb{I}(\phi_0 > 0).$$

Moreover, given τ_{n-1}, τ_n and the previous jump value ϕ_{n-1}, the conditional density of the jump value at ϕ_n, will be given by

$$g(\phi_n|\phi_{n-1}, \tau_{n-1}, \tau_n) = \lambda_\phi \exp(-\lambda_\phi(\phi_n - \phi_n^-))\mathbb{I}(\phi_n > \phi_n^-).$$

Here, $\phi_n^- := \phi_{n-1}\exp(-\kappa(\tau_n - \tau_{n-1}))$ represents the value of the intensity just before the jump at τ_n. Moreover, the intensity ζ_t at any time will only depend on the jump times and jump values and is given by

$$\zeta_t := \phi_{\nu_t}\exp(-\kappa(t - \tau_{\nu_t})),$$

where $\nu_t := \sup\{j \in \mathbb{N}|\tau_j \leq t\}$.

Given the intensity process ζ_t, claims will arrive according to a Poisson process with intensity ζ_t. In the example, we set the observations to be the claim arrival times. Hence, for any time interval $(s, t]$, the likelihood of the observations in the interval will be given by

$$p(y_{(s,t]}|\zeta_{(s,t]}) := \exp\left(-\int_s^t \zeta_s ds\right) \prod_{y\in(s,t]} \zeta_y.$$

Therefore, the static parameters in the model are $\theta := (\lambda_\tau, \lambda_\phi, \kappa)$. Figure 2.2 shows an example of the intensity process and the corresponding observations simulated from the shot-noise Cox model with $\lambda_\tau = 1/40, \lambda_\phi = 2/3$ and $\kappa = 1/100$.

2.3 Variable rate particle filter (VRPF)

In this section, we introduce the first particle filter for PDPs named *variable rate particle filter* (VRPF) proposed by Godsill and Vermaak (2005). This is actually a standard SMC algorithm on PDPs with a reparameterised presentation of the process. More specifically, the algorithm

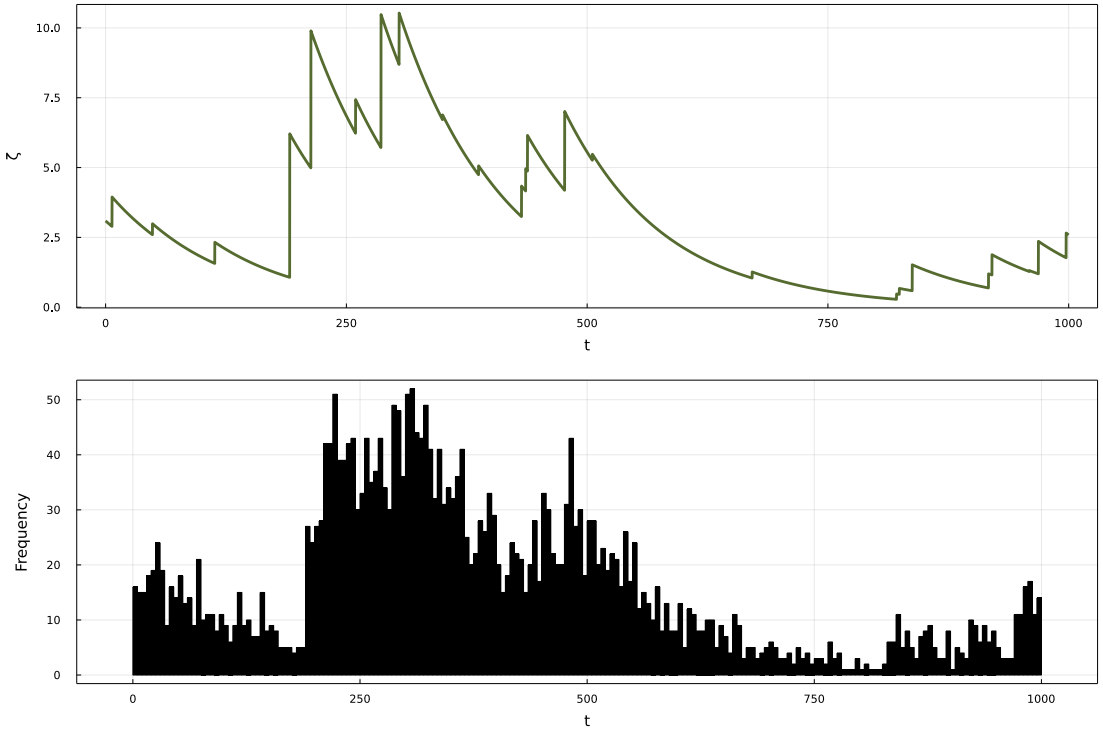

Figure 2.2: Data simulated from the shot-noise Cox model. Static parameters used are $\theta := (\lambda_\tau, \lambda_\phi, \kappa) = (1/40, 2/3, 1/100)$. Top: The intensity process ζ_t of the Cox model. Bottom: histogram of the event times with bin width equal to 2.5.

splits the PDP of interest into several disjoint intervals. Let $0 = t_0 < t_1 < t_2 < ... < t_P = T$, where $t_n, n > 0$ are pre-specified times. Let $(\tau_{n,k}, \phi_{n,k})$ be the k-th jump time and its associated jump value in the time interval $(t_{n-1}, t_n]$. Moreover, let $k_n \geq 0$ be the number of jumps in the interval $(t_{n-1}, t_n]$. Then, we can define the 'states' to be

$$X_1 := (k_1, \tau_{1,1:k_1}, \phi_{1,1:k_1}, \phi_0) \tag{2.4a}$$

$$X_n := (k_n, \tau_{n,1:k_n}, \phi_{n,1:k_n}) \tag{2.4b}$$

These 'states' takes values in

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k_1} \times \Phi^{k_1+1} \right), \tag{2.5a}$$

$$E_n := \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(t_{n-1},t_n],k_n} \times \Phi^{k_n} \right). \tag{2.5b}$$

Define $\mathcal{J}_n := \left(\hat{k}_n, \hat{\tau}_{n,1:\hat{k}_n}, \hat{\phi}_{n,0:\hat{k}_n}\right)$ to be the collection of all the jump times and their associated jump values we sample to define the PDP in the interval $(0, t_n]$. Then we will have $\hat{k}_n = \sum_{j=1}^{n} k_j$. Moreover, $\left(\hat{\tau}_{n,1:\hat{k}_n}, \hat{\phi}_{n,1:\hat{k}_n}\right)$ will be given by

$$\hat{\tau}_{n,1:\hat{k}_n} := \bigcup_{j=1}^{n} \left\{\tau_{j,1:k_j}\right\}, \tag{2.6a}$$

$$\hat{\phi}_{n,0:\hat{k}_n} := \{\phi_0\} \bigcup \left[\bigcup_{j=1}^{n} \left\{\phi_{j,1:k_j}\right\}\right]. \tag{2.6b}$$

with the conventions that $\left\{\tau_{j,1:k_j}\right\} := \emptyset$ and $\left\{\phi_{j,1:k_j}\right\} := \emptyset$ if $k_j = 0$. Moreover, we know that the piecewise deterministic process, $\zeta_{(0,t_n]}$, is completely determined by $\mathcal{J}_n$. Therefore, we can define a sequence of distributions $\{\pi_n\}_{n=1,2,..,P}$ with $\pi_n(x_{1:n}) := \gamma_n(x_{1:n})/Z_n$, on the space $E^n := \prod_{j=1}^{n} E_j$ that is given by

$$\begin{aligned}\gamma_n(x_{1:n}) := S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right) \times & q\left(\hat{\phi}_{n,0}\right) \times g(y_{(0,t_n]}|\mathcal{J}_n) \\ & \times \prod_{j=1}^{\hat{k}_n} \left\{f\left(\hat{\tau}_{n,j}|\hat{\tau}_{n,j-1}\right) q\left(\hat{\phi}_{n,j}|\hat{\phi}_{n,j-1}, \hat{\tau}_{n,j}, \tau_{n,\hat{j}-1}\right)\right\},\end{aligned} \tag{2.7}$$

where we assume that $\hat{\tau}_{n,0} := 0$. One can easily see that $\pi_P(x_{1:P})$ is actually the target distribution of interested defined in (2.3) and a standard SMC algorithm can be applied to $\{\pi_n\}_{n=1,2,...,P}$. At the n-th SMC step, X_n is sampled conditional on $X_{1:n-1}$ according to a proposal kernel $K_n\left(dx_n|X_{1:n-1}\right)$, where

$$K_n\left(x_n|x_{1:n-1}\right) = K_{n,1}\left(k_n|x_{1:n-1}\right) K_{n,2}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, x_{1:n-1}\right) \tag{2.8}$$

and the corresponding incremental weight is given by

$$G_n\left(x_{1:n}\right) := \frac{\gamma_n\left(x_{1:n}\right)}{\gamma_{n-1}\left(x_{1:n-1}\right) K_n(x_n|x_{1:n-1})}.$$

If $k_n = 0$, no jump times and values are sampled in the interval $(t_{n-1}, t_n]$. Therefore, $\hat{k}_n = \hat{k}_{n-1}$ and $\mathcal{J}_n := \mathcal{J}_{n-1}$ and

$$G_n(x_{1:n}) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n,\hat{k}_n}, t_{n-1}\right)} \times \frac{g\left(y_{(t_{n-1},t_n]}|\mathcal{J}_n\right)}{K_{n,1}(0|x_{1:n-1})}. \tag{2.9}$$

If $k_n \geq 1$, jump times $\tau_{n,1:k_n}$ and their corresponding jump values $\phi_{n,1:k_n}$ will be sampled in the interval $(t_{n-1}, t_n]$. In this scenario, $\hat{k}_n = \hat{k}_{n-1} + k_n$ and

$$\mathcal{J}_n := \left(\hat{k}_n, \mathcal{J}_{n-1} \backslash \{\hat{k}_{n-1}\}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right).$$

The corresponding incremental weight is given by

$$G_n\left(x_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathcal{J}_{n-1}\right) g\left(y_{(t_{n-1},t_n]}|\zeta_{(0,t_n]}\right)}{K_{n,1}(k_n|x_{1:n-1}) K_{n,2}(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, x_{1:n-1})} \tag{2.10}$$

where for $n \geq 1$,

$$\begin{aligned} h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathcal{J}_{n-1}\right) :=& f\left(\tau_{n,1}|\hat{\tau}_{n-1,\hat{k}_{n-1}}\right) q\left(\phi_{n,1}|\hat{\phi}_{n-1,\hat{k}_n-1}, \hat{\tau}_{n-1,\hat{k}_n-1}, \tau_{n,1}\right) \\ &\times \prod_{j=2}^{k_n} \left\{f(\tau_{n,j}|\tau_{n,j-1}) q(\phi_{n,j}|\phi_{n,j-1}, \tau_{n,j}, \tau_{n,j-1})\right\}. \end{aligned}$$

We also use the convention that $h(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathcal{J}_{n-1}) := 1$ if $k_n = 0$ and $h(\tau_{1,1:k_1}, \phi_{1,1:k_1}|\mathcal{J}_0) := p(k_1, \tau_{1,1:k_1}, \phi_{1,1:k_1})$. The VRPF method is detailed in Algorithm 2.1. The VRPF method has a potential problem due to the discretisation of the continuous time PDMP. If a jump occurs close to the end of a time block, say $(t_{n-1}, t_n]$, the VRPF method is likely to miss it since the observations in the block $(t_{n-1}, t_n]$ provide little information about such a jump. Even when such a jump is sampled, the corresponding sampled jump value is likely to be inaccurate as well since the information available up to that time is minimal. Moreover, the inaccuracy caused by this problem cannot be corrected by any of the smoothing methods either. Realising this problem we propose modifications to the VRPF method, which is based on the block sampling techniques introduced by Doucet et al. (2006). In the next section, we will give an introduction to the block sampling technique as well as the modifications we made on top of the VRPF method.

Algorithm 2.1: Variable Rate Particle Filter (VRPF)

for *n=1* **do**
  for $i = 1, 2, ..., N$ **do**
    Sample $X_1^i := \left(k_1^i, \tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i, \phi_0^i\right) \sim K_1(\cdot)$;
    Set $\mathcal{J}_1^i := \left(\tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i\right)$;
    Calculate the un-normalised weight

$$G_1\left(X_1^i\right) := \frac{\gamma_1\left(X_1^i\right)}{K_1\left(X_1^i\right)}$$

  for $i = 1, 2, ..., N$ **do**
    Calculate the normalised weight W_1^i such that

$$W_1^i \propto G_1(X_1^i), \quad \sum_{i=1}^{N} W_1^i = 1$$

for *n = 2,3,...,P* **do**
  for *i = 1,2,...,N* **do**
    Sample $A_{n-1}^i \sim \mathbf{r}\left(\cdot|\mathbf{W_{n-1}}\right)$;
    Sample $X_n^i := \left(k_n^i, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right) \sim K_n\left(\cdot|X_{1:n-1}^{A_{n-1}^i}\right)$ and set
    $X_{1:n}^i := \left(X_{1:n-1}^{A_{n-1}^i}, X_n^i\right)$;
    if $k_n^i = 0$ **then**
      Set $\mathcal{J}_n^i := \mathcal{J}_{n-1}^{A_{n-1}^i}$;
      Calculate the un-normalised weight, $G_n(X_{1:n}^i)$, using Equation (2.9);
    if $k_n^i \geq 1$ **then**
      Set $\mathcal{J}_n^i := \left(\hat{k}_{n-1}^{A_{n-1}^i} + k_n^i, \mathcal{J}_{n-1}^{A_{n-1}^i} \setminus \left\{\hat{k}_{n-1}^{A_{n-1}^i}\right\}, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right)$;
      Calculate the un-normalised weight, $G_n(X_{1:n}^i)$, using Equation (2.10);
  for $i = 1, 2, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

$$W_n^i \propto G_n(X_{1:n}^i), \quad \sum_{i=1}^{N} W_n^i = 1$$

2.4 Block Sampling Strategies and BlockVRPF Sampler

As discussed at the end of the previous section, the VRPF sampler may suffer from several problems, making estimations of jumps near the time interval breakpoints inaccurate. However, these problems can be alleviated by looking back at the previously sampled jumps as we collect more information and make modifications in light of the new observations. This inspires us to incorporate the block sampling strategy to perform such modifications. In this section, we first introduce the block sampling strategy of Doucet et al. (2006). We then describe the modification we make to the VRPF to tackle the aforementioned limitations.

2.4.1 Block SMC samplers

Suppose that we have already obtained $\{W^i_{n-1}, X^i_{1:n-1}\}_{i=1,..,N}$ at step $n-1$. Under the standard SMC scheme, only $\{X^i_n\}_{i=1,..,N}$ are sampled at step n and the paths $X^i_{1:n}$ are constructed by concatenating X^i_n with $X^i_{1:n-1}$ to approximate $\pi_n(x_{1:n})$. However, $\pi_n(x_{1:n})$ and $\pi_{n-1}(x_{1:n-1})$ may have a large discrepancy. As a consequence, the sampled paths $X^i_{1:n-1}$ may not be in the region of high density under $\pi_n(x_{1:n})$. The block sampling strategy, on the other hand, potentially provides a way of alleviating this limitation of the standard SMC scheme. Instead of only sampling the X^i_n's at step n, part of the previous paths are also resampled in light of $\pi_n(x_{1:n})$. Suppose that $X^{',i}_{n-L:n}$ for some $L > 1$ are sampled at step n according to a proposal density $K_n(dx^{'}_{n-L:n}|x_{1:n-1})$, the paths at step n will then be obtained by discarding $X_{n-L:n-1}$ from $X_{1:n-1}$ and adding $X^{'}_{n-L:n}$ to it. As a result, this new path at step n will be $(X_{1:n-L-1}, X^{'}_{n-L:n})$. If we define $\mathcal{Q}_n(x_{1:n-L-1}, x^{'}_{n-L:n})$ to be the proposed density of the path at step n, then $\mathcal{Q}_n(x_{1:n-L-1}, x^{'}_{n-L:n})$ is given by

$$\begin{aligned}\mathcal{Q}_n(x_{1:n-L-1}, x^{'}_{n-L:n}) &= \int \mathcal{Q}_n(x_{1:n-1}, x'_{n-L:n})dx_{n-L:n-1} \\ &= \int \mathcal{Q}_{n-1}(x_{1:n-1})K_n(x'_{n-L:n}|x_{1:n-1})dx_{n-L:n-1}\end{aligned}$$

Therefore, when the path is propagated from $X_{1:n-1}$ to $(X_{1:n-L-1}, X^{'}_{n-L:n})$, importance weight will become

$$W^i_n \propto \frac{\gamma_n(x_{1:n-L-1}, x^{'}_{n-L:n})}{\mathcal{Q}_n(x_{1:n-L-1}, x^{'}_{n-L:n})} = \frac{\gamma_n(x_{1:n-L-1}, x^{'}_{n-L:n})}{\int \mathcal{Q}_{n-1}(x_{1:n-1})K_n(x'_{n-L:n}|x_{1:n-1})dx_{n-L:n-1}}$$

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	...
Z_1	$X_1(1)$								
Z_2	$X_1(2)$	$X_2(1)$							
Z_3	$X_1(3)$	$X_2(2)$	$X_3(1)$						
Z_4	$X_1(4)$	$X_2(3)$	$X_3(2)$	$X_4(1)$					
Z_5		$X_2(4)$	$X_3(3)$	$X_4(2)$	$X_5(1)$				
Z_6			$X_3(4)$	$X_4(3)$	$X_5(2)$	$X_6(1)$			
Z_7				$X_4(4)$	$X_5(3)$	$X_6(2)$	$X_7(1)$		
Z_8					$X_5(4)$	$X_6(3)$	$X_7(2)$	$X_8(1)$	
...						...	...	...	...

Table 2.1: Illustrative example showing the components in z_n for $n = 1, 2, 3, .., 6$ and $L = 3$.

We can see that the calculation of the importance weights at step n involves an integral, which is, in most cases, impossible to be evaluated pointwise. As Doucet et al. (2006) pointed out, even if it is possible to calculate the integral exactly, the form of the importance weights will be no longer as simple as what we have in Algorithm 1 and Algorithm 2.

To deal with this problem, we can instead target a density defined on an extended space by using the auxiliary trick. Define $X_n(j)$ to be the jth time X_n is sampled. To simplify notation at a later stage, we also define Z_n to be the variable(s) sampled at step n. Hence,

$$Z_n := \begin{cases} (X_1(n), X_2(n-1), ..., X_{n-1}(2), X_n(1)), & \text{for } 1 \leq n \leq L \\ (X_{n-L}(L+1), X_{n-L+1}(L), ..., X_{n-1}(2), X_n(1)), & \text{for } n \geq L+1 \end{cases}$$

This allows us to transform between the X's and Z's by Equation (2.11)

$$Z_{n,k} := X_{n-k+1}(k) \qquad X_n(j) := Z_{n+j-1,j}, \tag{2.11}$$

where $Z_{n,k}$ represents the kth element in Z_n. For the ease of understanding, Table 2.1 shows the details of the components contained in the variable Z_n for $n = 1, 2, .., 8$ and $L = 3$. Hence, one can see that $Z_{1:n}$ will include all the random variables sampled up to step n. Based on the previous discussion, the proposal distribution of all these variables will be

$$\begin{aligned} \mathcal{Q}_n(z_{1:n}) &= K_1(z_1) \prod_{t=2}^{n} K_t(z_t | z_{1:t-1}) \\ &= K_1(x_1(1)) \prod_{t=2}^{L} K_t(x_1(t), ..., x_t(1) | z_{1:t-1}) \prod_{t=L+1}^{n} K_t(x_{t-L}(L+1), ..., x_t(1) | z_{1:t-1}). \end{aligned}$$

To be compatible with the importance density, the extended target should also include all the variables up to step n. Hence, when $n > L$, we have that

$$\tilde{\pi}_n(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) := \gamma_n(x_1(L+1), ..., x_{n-L}(L+1), x_{n-L+1}(L), ..., x_n(1)) \times$$
$$\prod_{t=2}^{L} \lambda_t(x_1(t-1), x_2(t-2), ..., x_{t-1}(1)|z_{1:t}) \prod_{t=L+1}^{n} \lambda_t(x_{t-L}(L), x_{t-L+1}(L-1), ..., x_{t-1}(1)|z_{1:t}), \tag{2.12}$$

where the λ_l's are auxiliary conditional densities of the rejuvenated variables at step l. When $n \leq L$, the corresponding extended target has a similar definition and is given by

$$\tilde{\pi}_n(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) := \gamma_n(x_1(n), x_2(n-1), ..., x_{n-1}(2), x_n(1)) \times \prod_{t=2}^{n} \lambda_t(x_1(t-1), x_2(t-2), ..., x_{t-1}(1)|x_{1:t}). \tag{2.13}$$

Importantly, the extended target density incorporates the actual target of interest as a marginal. One can see that by targeting the extended density defined in (2.12) and (2.13), we can avoid the need for the integral appearing in the previous set-up and the importance weight will become

$$W_n \propto \frac{\tilde{\gamma}_n(z_{1:n})}{\mathcal{Q}_n(z_{1:n})} = \frac{\tilde{\gamma}_n(z_{1:n})\mathcal{Q}_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}(z_{1:n-1})\mathcal{Q}_n(z_{1:n})} \times \frac{\tilde{\gamma}_{n-1}(z_{1:n-1})}{\mathcal{Q}_{n-1}(z_{1:n-1})} = w_{n-1}\frac{\tilde{\gamma}_n(z_{1:n})\mathcal{Q}_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}(z_{1:n-1})\mathcal{Q}_n(z_{1:n})}.$$

Note that $\mathcal{Q}_n(z_{1:n}) = \mathcal{Q}_{n-1}(z_{1:n-1})K_n(z_n|z_{1:n-1})$. Hence, the incremental weight for the SMC with the block sampling strategy is then given by

$$\mathcal{G}_n(z_{1:n}) := \frac{\tilde{\gamma}_n(z_{1:n})\mathcal{Q}_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}(z_{1:n-1})\mathcal{Q}_n(z_{1:n})} = \frac{\tilde{\gamma}_n(z_{1:n})}{\tilde{\gamma}_{n-1}(z_{1:n-1})K_n(z_n|z_{1:n-1})}. \tag{2.14}$$

When $n > L$, the incremental weight will be given by

$$\mathcal{G}_n(z_{1:n}) = \frac{\gamma_n(x_{1:n-L}(L+1), x_{n-L+1}(L), .., x_{n-1}(2), x_n(1))}{\gamma_{n-1}(x_{1:n-L-1}(L+1), x_{n-L}(L), .., x_{n-2}(2), x_{n-1}(1))}$$

$$\times \frac{\lambda_n(x_{n-1}(1), ..., x_{n-L}(L)|x_{1:n-L}(L+1), x_{n-L+1}(L), .., x_{n-1}(2), x_n(1))}{K_n(x_n(1), .., x_{n-L}(L+1)|x_{1:n-L-1}(L+1), x_{n-L}(L), .., x_{n-2}(2), x_{n-1}(1))}. \tag{2.15}$$

When $n \leq L$, the corresponding incremental weight will be given by

$$\mathcal{G}_n(z_{1:n}) := \frac{\gamma_n(x_1(n), x_2(n-1), .., x_{n-1}(2), x_n(1))}{\gamma_{n-1}(x_1(n-1), x_2(n-2), ..x_{n-1}(1))} \times$$

$$\frac{\lambda_n(x_1(n-1), .., x_{n-1}(1)|x_1(n), x_2(n-1), .., x_{n-1}(2), x_n(1))}{K_n(x_n(1), ..., x_1(n)|x_1(n-1), x_2(n-2), ..x_{n-1}(1))}. \quad (2.16)$$

If the resampling steps are also included in this scheme, the weights at step n will then be equal

Algorithm 2.2: SMC with Block Sampling

for $n = 1$ **do**
  for $i = 1, 2, .., N$ **do**
    Sample $X_1^i(1) \sim K_1(dx_1)$;
    Set $Z_1^i := X_1^i(1)$;
    Calculate and normalise the weights

$$W_1^i \propto \frac{\gamma_1(x_1^i(1))}{K_1(x_1^i(1))}$$

  for $n = 2, .., L$ **do**
    Sample $\mathbf{A}_{n-1} \sim r_{n-1}(\mathbf{W}_{n-1})$;
    for $i = 1, .., N$ **do**
      Sample $Z_n^i := (X_n^i(1), .., X_1^i(n)) \sim K_n(dz_n|Z_{1:n-1}^{A_{n-1}^i})$;
      Set $Z_{1:n}^i := (Z_{1:n-1}^{A_{n-1}^i}, Z_n^i)$;
      Calculate and normalise the weights W_n^i according to (2.16);
  for $n = L+1, .., P$ **do**
    Sample $\mathbf{A}_{n-1} \sim r_{n-1}(\mathbf{W}_{n-1})$;
    for $i = 1, ..N$ **do**
      Sample $Z_n^i := (X_n^i(1), ..X_{n-L}^i(L+1)) \sim K_n(dz_n|Z_{1:n-1}^{A_{n-1}^i})$;
      Set $Z_{1:n}^i := (Z_{1:n-1}^{A_{n-1}^i}, Z_n^i)$;
      Calculate and normalise the weights W_n^i according to (2.15);

to the incremental weights only. Details of the Block SMC method are given in Algorithm 2.2. It is also clear that the performance of the Block SMC method will depend on the choices of the artificial conditional densities $\{\lambda_n\}$ and the proposal densities $\{K_n\}$. To have optimal performance, these should be chosen to minimise the variance of the incremental weights defined in (2.15) and (2.16). Doucet et al. (2006) derived the following two propositions which provide us with guidance on how to choose these densities.

Proposition 1. *When $n > L$, the optimal conditional distribution*

$$\lambda_n(x_{n-1}(1), ..., x_{n-L}(L)|x_{1:n-L}(L+1), x_{n-L+1}(L), .., x_{n-1}(2), x_n(1))$$

that minimizes the variance of the incremental weight defined in (2.15) *is given by*

$$\lambda_n^{opt}(x_{n-L:n-1}|x_{1:n-L-1}, x'_{n-L:n}) = \frac{\gamma_{n-1}(x_{1:n-1})K_n(x'_{n-L:n}|x_{1:n-1})}{\int \gamma_{n-1}(x_{1:n-1})K_n(x'_{n-L:n}|x_{1:n-1})\, dx_{n-L:n-1}} \tag{2.17}$$

where we use define $x_{1:n-1} := (x_{1:n-L-1}(L+1), x_{n-L}(L), .., x_{n-2}(2), x_{n-1}(1)))$ *and* $x'_{n-L:n} := (x_n(1), x_{n-1}(2), ..., x_{n-L}(L+1))$ *for the ease of presentation. The optimal conditional distribution that minimizes the incremental weight of* (2.16) *will be given in a similar form.*

Proposition 2. *Given that the optimal artificial conditional distribution $\{\lambda_n^{opt}\}$ is chosen. The corresponding optimal proposal distribution is given by*

$$K_n^{opt}(x'_{n-L:n}|x_{1:n-1}) = \pi_n(x'_{n-L:n}|x_{1:n-L-1}) \tag{2.18}$$

where $x_{1:n-1}$ and $x'_{n-L:n}$ have the same definition as in proposition 1

One can see that an integral is involved in the optimal choice $\{\lambda_n^{opt}\}$ and it is generally not tractable. Hence, one needs to choose λ_n that approximates (2.17). Moreover, the optimal proposal distribution defined in (2.18) also suggests that $x_{n-L:n-1}$ should be resampled in light of π_n. In fact, as Doucet et al. (2006) pointed out, the choice of $\{\lambda_n\}$ and $\{K_n\}$ is quite crucial to the performance of the block sampling strategy. If they are not chosen appropriately, block sampling strategy may not always outperform the standard SMC sampler.

2.4.2 Block Variable Rate Particle Filter (Block-VRPF)

As discussed before, we want to make modifications on the jumps sampled in the previous time blocks before the jumps in the current time block are sampled. The block sampling strategy is therefore a suitable choice of doing this. Instead of resampling all the jumps in the previous block, however, we propose to only modify some of the earlier jumps. Hence, at the n-th SMC step, in addition to sampling jump times and values in the interval $(t_{n-1}, t_n]$, represented by $x_n(1)$, jumps already sampled in the interval $(t_{n-2}, t_{n-1}]$, represented by $x_{n-1}(1)$, will also be

modified. We follow Whiteley et al. (2011) to propose two types of modifications on $x_{n-1}(1)$ - a *birth* move and an *adjustment* move:

Modification 1. If a *birth* is proposed, a new jump time will be sampled in the interval $(\tau_{n-1,k_{n-1}} \vee t_{n-2}, t_{n-1}]$ together with its jump value. Here, $a \vee b$ represents the maximum of a and b.

Modification 2. If an *adjustment* is proposed, the last jump time in $x_{n-1}(1)$ and its associated jump value, $\left(\tau_{n-1,k_{n-1}}, \phi_{n-1,k_{n-1}}\right)$, will be discarded and a new jump time and its jump value will be proposed in the interval $(\tau_{n-1,k_{n-1}-1} \vee t_{n-2}, t_{n-1}]$. In addition, if $x_{n-1}(1)$ contains zero jump, the *adjustment* will keep $x_{n-1}(1)$ unchanged.

Note that the modifications are not constrained to those proposed above. Other modification proposals and even modifications on more jumps are also possible. One can even modify more than one previous block, as discussed in the previous section. For ease of presentation, we will focus on the *birth* and *adjustment* moves only for the moment. Moreover, when the time block is large enough to be likely to contain at least one jump, it is enough to only modify the jumps in the last block since modifying the last jump is what we are interested in.

Let $u_{n-1} := \left(\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$ be the modified jump time and associated value of the last jump in the interval $(t_{n-2}, t_{n-1}]$ with the convention that $U_{n-1} := \emptyset$ if $k_{n-1} = 0$. Also, let $M_{n-1} \in \{0, 1\}$ be the indicator of the type of modification made on $X_{n-1}(1)$ with 0 representing an *adjustment* and 1 representing a *birth*. Following the notation of block sampling strategies, we use $X_{n-1}(2)$ to denote $X_{n-1}(1)$ after modification and from the description above, we should have:

$$X_{n-1}(2) := \left(k_{n-1} + 1, \tau_{n-1,1:k_{n-1}}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}}, \mathring{\phi}_{n-1}\right) \tag{2.19a}$$

when $M_{n-1} = 1$. When $M_n = 0$, i.e. an *adjustment* is proposed, the modified $X_n(1)$ will be given by

$$X_{n-1}(2) := \left(k_{n-1}, \tau_{n-1,1:k_{n-1}-1}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}-1}, \mathring{\phi}_{n-1}\right). \tag{2.19b}$$

If the number of jump times in $X_{n-1}(1)$ is 0, $X_{n-1}(2) = X_{n-1}(1)$. We treat this as a special case of Equation (2.19b). After the state $X_{n-1}(1)$ is modified, the jump times and jump values in the interval $(t_{n-1}, t_n]$ will be sampled conditional on the modified PDP. Hence, the *Block-VRPF* actually samples $(M_{n-1}, U_{n-1}, X_n(1))$ at the n-th SMC step for $n > 1$, and they should be the actual 'states' we are considering. To ease notation, we re-define

$$Z_1 := X_1(1) \tag{2.20a}$$

$$Z_n := (M_{n-1}, U_{n-1}, X_n(1)) \tag{2.20b}$$

to be the 'states' at the n-th SMC step. These 'states' will take values in the subsets of

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k} \times \Phi^{k_1+1} \right) \tag{2.21a}$$

and

$$\begin{aligned} E_n := & \left[\left(\{0\} \times T^M_{(\tau_{n-1,k_{n-1}-1} \vee t_{n-2}, t_{n-1}]} \times \Phi \right) \bigcup \left(\{1\} \times T^M_{(\tau_{n-1,k_{n-1}} \vee t_{n-2}, t_{n-1}]} \times \Phi \right) \right] \\ & \times \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(0,t_n],k_n} \times \Phi^{k_n} \right), \end{aligned} \tag{2.21b}$$

where $T^M_{(s,t]} := \{\tau | \tau \in (s,t]\}$. Moreover, denote $\bar{U}_{n-1}$ the jump time and value in the interval $(t_{n-2}, t_{n-1}]$ that are modified in the n-th SMC step. According to the modifications we defined above, $\bar{U}_{n-1} := \left(\tau_{n-1,k_{n-1}}, \phi_{n-1,k_{n-1}}\right)$ when $M_{n-1} = 0$ with the convention that $(\tau_{n-1,0}, \phi_{n-1,0}) := \emptyset$ when $k_{n-1} = 0$. In addition, $\bar{U}_{n-1} := \emptyset$ when $M_{n-1} = 1$. Then, $\left(X_{n-1}(2), M_{n-1}, \bar{U}_{n-1}\right)$ and $(X_{n-1}(1), M_{n-1}, U_{n-1})$ actually represent the same set of random variables. Therefore, if we look at all the 'states' we have sampled up to the n-th SMC step, we can see that

$$\begin{aligned} Z_{1:n} &:= (X_1(1), M_1, U_1, ..., X_{n-1}(1), M_{n-1}, U_{n-1}, X_n(1)) \\ &:= \left(X_1(2), M_1, \bar{U}_1, ..., X_{n-1}(2), M_{n-1}, \bar{U}_{n-1}, X_n(1)\right) \end{aligned} \tag{2.22}$$

i.e. there is a one-to-one transformation between the two parameterizations.

Target Density

In the following, we are going to present the extended target distribution of the *Block-VRPF* algorithm. Since we have the one-to-one correspondence in (2.22), it is the same to express the joint distribution of $Z_{1:n}$ in terms of $(X_1(2), M_1, U_1, ..., X_{n-1}(2), M_{n-1}, U_{n-1}, X_n(1))$ as to express it in terms of $\left(X_1(2), M_1, \bar{U}_1, ..., X_{n-1}(2), M_{n-1}, \bar{U}_{n-1}, X_n(1)\right)$. The reason why we care about this is that the distribution of the PDP given the observations at the n-th SMC step, $\zeta_{(0,t_n]}$, is completely determined by $(X_1(2), X_2(2), ..., X_{n-1}(2), X_n(1))$. Hence, the target density should incorporate $\gamma_n(x_1(2), x_2(2), .., x_{n-1}(2), x_n(1))$ as marginal. Therefore, we define the extended target distribution of the algorithm, $\bar{\pi}_n(Z_{1:n}) := \bar{\gamma}_n(Z_{1:n})/\bar{\mathcal{Z}}_n$, to be

$$\begin{aligned}\bar{\gamma}_n\left(Z_{1:n}\right) := \gamma_n(x_{1:n-1}(2), x_n(1))\mu_n\left(m_{1:n-1}|x_{1:n-1}(2), x_n(1)\right) \\ \times \prod_{j=1}^{n-1} \lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:n-1}(2), x_n(1)\right), \quad (2.23)\end{aligned}$$

where $\{\mu_n\}$ and $\{\lambda_n\}$ are artificial conditional densities. These densities are included to define a target distribution on an extended space to avoid the need to evaluate an integral, as discussed in the previous section. Similarly, denote $\mathcal{J}_n := \left(\hat{k}_n, \hat{\tau}_{n,1:\hat{k}_n}, \hat{\phi}_{n,1:\hat{k}_n}\right)$ to be the jump times and values that define the PDP, $\zeta_{(0,t_n]}$ at the n-th SMC step. Then, $\mathcal{J}_n$ can be completely defined by $(x_1(2), .., x_{n-1}(2), x_n(1))$ and we will have that

$$\begin{aligned}\gamma_n\left(x_1(2), ..., x_{n-1}(2), x_n(1)\right) &= \gamma_n\left(\mathcal{J}_n\right) \\ &= S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right) q\left(\hat{\phi}_0\right) g\left(y_{(0,t_n]}|\mathcal{J}_n\right) \\ &\times \prod_{j=1}^{\hat{k}_n} \left\{ f\left(\hat{\tau}_j|\hat{\tau}_{j-1}\right) q\left(\hat{\phi}_j|\hat{\phi}_{j-1}, \hat{\tau}_j, \hat{\tau}_{j-1}\right)\right\}.\end{aligned}$$

Based on the modifications we proposed, $\mathcal{J}_n$ can also be determined recursively, i.e. we can set $\mathcal{J}_1 := (k_1, \tau_{1,1:k_1}, \phi_{1,k_1}, \phi_0)$ and $\mathcal{J}_n$ can be derived from $\mathcal{J}_{n-1}$ by

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \quad (2.24a)$$

when $M_{n-1} = 1$. If $M_{n-1} = 0$ and $k_{n-1} \geq 1$, then

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}, \hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right). \tag{2.24b}$$

If $M_{n-1} = 0$ and $k_{n-1} = 0$, then

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right). \tag{2.24c}$$

Here, we use the convention that if $k_n = 0$, then Equation (2.24a), (2.24b) and (2.24c) will be simplified to $\left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$, $\left(\hat{k}_{n-1}, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}, \hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right\}, \mathring{\tau}^m_{n-1}, \mathring{\phi}_{n-1}\right)$ and $\mathcal{J}_{n-1}$ respectively.

Proposal Kernel

As for the proposed kernel, we use $K_n\left(z_n | z_{1:n-1}\right) := K_{n,1}\left(m_{n-1} | z_{1:n-1}\right) K_{n,2}\left(u_{n-1} | m_{n-1}, z_{1:n-1}\right) \times K_{n,3}\left(x_n | z_{1:n-1}, m_{n-1}, u_{n-1}\right)$, in which $K_{n,1}\left(m_{n-1} | z_{1:n-1}\right) := K_{n,1}\left(m_{n-1} | \mathcal{J}_{n-1}\right)$ propose a modification type based on the PDP in the interval $(0, t_{n-1}]$. We follow the kernel proposed in Whiteley et al. (2011), i.e. $K_{n,1}\left(0 | \mathcal{J}_{n-1}\right) := S\left(\hat{\tau}_{n-1,,\hat{k}_{n-1}}, t_{n-1}\right)$. This is to say that the probability that an 'adjust' is proposed equals the prior density that no jump occurs after the last jump in $\mathcal{J}_{n-1}$ and before the end of last epoch, t_{n-1}. If a 'birth' is proposed, then U_{n-1} will be proposed according to

$$\begin{aligned} K_{n,2}\left(u_{n-1} | m_{n-1} = 1, z_{1:n-1}\right) :=& \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,1}\left(\mathring{\tau}_{n-1} | \mathcal{J}_{n-1}\right) \beta_{n-1,1}\left(\mathring{\phi}_{n-1} | \mathcal{J}_{n-1}\right) \end{aligned} \tag{2.25a}$$

If an 'adjust' is proposed, then U_{n-1} will be sampled according to

$$\begin{aligned} K_{n,2}\left(u_{n-1} | a, z_{1:n-1}\right) :=& \mathbb{1}\left(k_{n-1} \geq 1\right) \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}-1} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,0}\left(\mathring{\tau}_{n-1} | \mathcal{J}_{n-1}\right) \beta_{n-1,0}\left(\mathring{\phi}_{n-1} | \mathcal{J}_{n-1}\right) \\ & + \mathbb{1}(k_{n-1} = 0) \times \delta_{\emptyset}(u_{n-1}) \end{aligned} \tag{2.25b}$$

i.e. if $X_{n-1}(1)$ contains no jumps at all and an 'adjust' is proposed, then U_{n-1} will be an empty set. Here, $\alpha_{n-1,1}$ and $\alpha_{n-1,0}$ are the proposal densities for the modified jump time

on $X_{n-1}(1)$ for a *birth* and an *adjustment* move respectively and $\beta_{n-1,1}$ and $\beta_{n-1,0}$ are the proposal densities for the corresponding jump values. To sample the jump times and values in the interval $(t_{n-1}, t_n]$ according to $K_{n,3}$, we use similar proposals in VRPF method, i.e.

$$\begin{aligned} K_{n,3}\left(x_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) :=& K_{n,3}^1\left(k_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) \\ & \times K_{n,3}^2\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1}, u_{n-1}, z_{1:n-1}\right) \end{aligned} \tag{2.26}$$

Auxiliary Densities

As discussed before, the auxiliary densities are included to ensure that the importance weight will have the correct form and incorporate γ_n as marginals. However, when choosing these auxiliary densities, we need to ensure that their support is included in the support of the proposal kernels. Hence, for the density $\mu_n\left(m_{1:n-1}|x_{1:n-1}(2), x_n(1)\right)$ in (2.23), we propose a factorisable density for simplicity, i.e.

$$\mu_n\left(m_{1:n-1}|x_{1:n-1(2),x_n(1)}\right) := \prod_{j=1}^{n-1} \mu_n^j\left(m_j|x_{1:j}(2)\right).$$

Furthermore, we propose a uniform distribution over 'adjust' and 'birth' on m_j if $x_j(2)$ contains at least one jump, i.e.

$$\mu_n^j(m_j|x_{1:j}(2)) := \frac{1}{2} \times \mathbb{1}\left(\bar{k}_j > 0\right) + \mathbb{1}(\bar{k}_j = 0)\delta_0(m_j).$$

Other forms of densities are also possible as long as the densities have the correct support. The density for the modified jump time and jump value, $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right)$, must be chosen by the users. Note that if $M_j = 1$ or $M_j = 0$ with $k_j = 0$, there is actually nothing to be modified. In this case, $\bar{u}_j := \emptyset$ and $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right)$ can be replaced with 1. When $M_j = 0$ and $k_j \geq 1$, $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right) := \lambda_j\left(\hat{\tau}_{j,\hat{k}_j}, \hat{\phi}_{j,\hat{k}_j}|m_{1:j}, x_{1:j}(2)\right)$ should have support

$$\left(\hat{\tau}_{j,\hat{k}_j-1} \vee t_{n-2}, t_{n-1}\right] \times \Phi.$$

One easy choice is to assign $\hat{\tau}_{j,\hat{k}_j}$ a uniform distribution over its support. For $\hat{\phi}_{j,\hat{k}_j}$, we can set it to follow its prior.

Incremental Weight

The incremental weight, $G_n\left(z_{1:n}\right) := \hat{\gamma}_n\left(z_{1:n}\right) / \left[\hat{\gamma}_{n-1}\left(z_{1:n-1}\right) K_n\left(z_n | z_{1:n-1}\right)\right]$ is calculated as follows. In this section, we set

$$\begin{aligned} h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n} | \tau, \phi\right) :=& f\left(\tau_{n,1} | \tau\right) q\left(\phi_{n,1} | \phi, \tau, \tau_{n,1}\right) \\ & \times \prod_{j=2}^{k_n} \left\{ f(\tau_{n,j} | \tau_{n,j-1}) q(\phi_{n,j} | \phi_{n,j-1}, \tau_{n,j}, \tau_{n,j-1}) \right\} \end{aligned}$$

for any value of n. We will use this notation throughout this section. When $m_{n-1} = 0$, i.e. an *adjustment* is proposed,

1. If $k_{n-1} = 0$

$$G_n\left(z_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{\mu_{n-1}(m_{n-1} | \bar{x}_{n-1}) \lambda_{n-1}(\bar{u}_{n-1} | m_{1:n-1}, \bar{x}_{1:n-1})}{K_{n,1}\left(0 | \mathcal{J}_{n-1}\right)}$$

$$\times\, g\left(y_{(t_{n-1}, t_n]} | \mathcal{J}_n\right) \times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n} | \hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right)}{K^1_{n,3}\left(k_n | 0, \emptyset, z_{1:n-1}\right) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n} | k_n, 0, \emptyset, z_{1:n-1}\right)} \tag{2.27a}$$

2. If $k_{n-1} \geq 1$,

$$G_n\left(z_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{\mu_{n-1}(m_{n-1}|\bar{x}_{n-1})}{K_{n,1}\left(0|\mathcal{J}_{n-1}\right)} \times \frac{g\left(y_{\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \wedge \mathring{\tau}_{n-1}, t_n\right]}|\mathcal{J}_n\right)}{g\left(y_{\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \wedge \mathring{\tau}_{n-1}, t_{n-1}\right]}|\mathcal{J}_{n-1}\right)}$$

$$\times \frac{\lambda_{n-1}\left(\bar{u}_{n-1}|m_{1:n-1}, \bar{x}_{1:n-1}\right)}{\mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}-1} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \alpha_{n-1,0}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,0}\left(\mathring{\phi}_{n-1}|\mathring{\tau}_{n-1}, \mathcal{J}_{n-1}\right)}$$

$$\times \frac{f\left(\mathring{\tau}_{n-1}|\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right) q\left(\mathring{\phi}_{n-1}|\hat{\phi}_{n-1,\hat{k}_{n-1}-1}, \mathring{\tau}_{n-1}, \hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)}{f\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}|\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right) q\left(\hat{\phi}_{n-1,\hat{k}_{n-1}}|\hat{\phi}_{n-1,\hat{k}_{n-1}-1}, \tau_{n-1,\hat{k}_{n-1}}, \hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)}$$

$$\times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|0, u_{n-1}, z_{1:n-1}) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 0, u_{n-1}, z_{1:n-1}\right)} \quad (2.27b)$$

When $m_{n-1} = 1$, i.e. a *birth* is proposed,

$$G_n\left(z_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{\mu_{n-1}(m_{n-1}|\bar{x}_{n-1})}{K_{n,1}\left(m_{n-1}|\mathcal{J}_{n-1}\right)} \times \frac{g\left(y_{(\mathring{\tau}_{n-1}, t_n]}|\mathcal{J}_n\right)}{g\left(y_{(\mathring{\tau}_{n-1}, t_{n-1}]}|\mathcal{J}_{n-1}\right)}$$

$$\times \frac{f\left(\mathring{\tau}_{n-1}|\hat{\tau}_{n-1,\hat{k}_{n-1}}\right) q\left(\mathring{\phi}_{n-1}|\hat{\phi}_{n-1,\hat{k}_{n-1}}, \mathring{\tau}_n, \hat{\tau}_{n-1,\hat{k}_{n-1}}\right) \lambda_{n-1}(\bar{u}_{n-1}|m_{1:n-1}, \bar{x}_{1:n-1})}{\mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \alpha_{n-1,1}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,1}\left(\mathring{\phi}_{n-1}|\mathcal{J}_{n-1}\right)}$$

$$\times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|m_{n-1} = 1, u_{n-1}, z_{1:n-1}) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1} = 1, u_{n-1}, z_{1:n-1}\right)} \quad (2.28)$$

The BlockVRPF method is outlined in Algorithm 2.3. Note that U^i_{n-1} in line 12 of Algorithm 2.3 will become $\emptyset$ when $M^i_{n-1} = 0$ and $k^{A^i_{n-1}}_{n-1} = 0$ while $K^2_{n,3}$ and h should be replaced with 1 when $k^i_n = 0$. Moreover, λ_{n-1} will also be replaced by 1 when a *birth* is proposed or when an *adjustment* is proposed but $k_{n-1} = 0$. Also the BlockVRPF sampler we propose

Algorithm 2.3: Block Variable Rate Particle Filter (BlockVRPF)

for *n=1* **do**
  for $i = 1, 2, ..., N$ **do**
    Sample $X_1^i := \left(k_1^i, \tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i, \phi_0^i\right) \sim K_1(\cdot)$;
    Set $Z_1^i := X_1^i$ and $\mathcal{J}_1^i := \left(\tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i\right)$;
    Calculate the un-normalised weight

$$G_1\left(Z_1^i\right) := \frac{\gamma_1\left(Z_1^i\right)}{K_1\left(Z_1^i\right)}$$

  for $i = 1, 2, ..., N$ **do**
    Calculate the normalised weight W_1^i such that

$$W_1^i \propto G_1(Z_1^i), \quad \sum_{i=1}^{N} W_1^i = 1$$

for *n = 2,3,...,P* **do**
  for *i = 1,2,...,N* **do**
    Sample $A_{n-1}^i \sim \mathbf{r}\left(\cdot|\mathbf{W_{n-1}}\right)$;
    Sample $M_{n-1}^i \sim K_{n,1}\left(\cdot|\mathcal{J}_{n-1}^{A_{n-1}^i}\right)$;
    Sample $U_{n-1}^i := \left(\mathring{\tau}_{n-1}^i, \mathring{\phi}_{n-1}^i\right) \sim K_{n,2}\left(\cdot|m_{n-1}^i, z_{1:n-1}^{A_{n-1}^i}\right)$;
    Sample $X_n^i := \left(k_n^i, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right) \sim$
    $K_{n,3}^1\left(\cdot|m_{n-1}^i, u_{n-1}^i, z_{1:n-1}^{A_{N-1}^i}\right) K_{n,3}^2\left(\cdot|k_n^i, m_{n-1}^i, u_{n-1}^i, z_{1:n-1}^{A_{N-1}^i}\right)$;
    Set $Z_n^i := \left(M_{n-1}^i, U_{n-1}^i, X_{n-1}^i\right)$ and $Z_{1:n}^i := \left(Z_{1:n-1}^{A_{n-1}^i}, Z_n^i\right)$;
    Define $\mathcal{J}_n^i$ according to (2.24a), (2.24b) or (2.24c) based on $\mathcal{J}_{n-1}^{A_{n-1}^i}$;
    Calculate the incremental weight $G_n\left(z_{1:n}^i\right)$ according to (2.27a), (2.27b) or (2.28);
  for $n = 1, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

$$W_n^i \propto G_n(Z_{1:n}^i), \quad \sum_{i=1}^{N} W_n^i = 1$$

in this section provides us with an opportunity to correct the jumps in the last time block. Such a correction is more obvious when a jump appears near the end of a block in the actual process. This is illustrated in Figure 2.3. One can see that with the same number of samples, the VRPF sampler missed the jump occurring at the end of the first time block in the actual PDMP completely. On the contrary, due to the modification step in the BlockVRPF sampler, there is a chance to recover this missed jump (as shown in the graph).

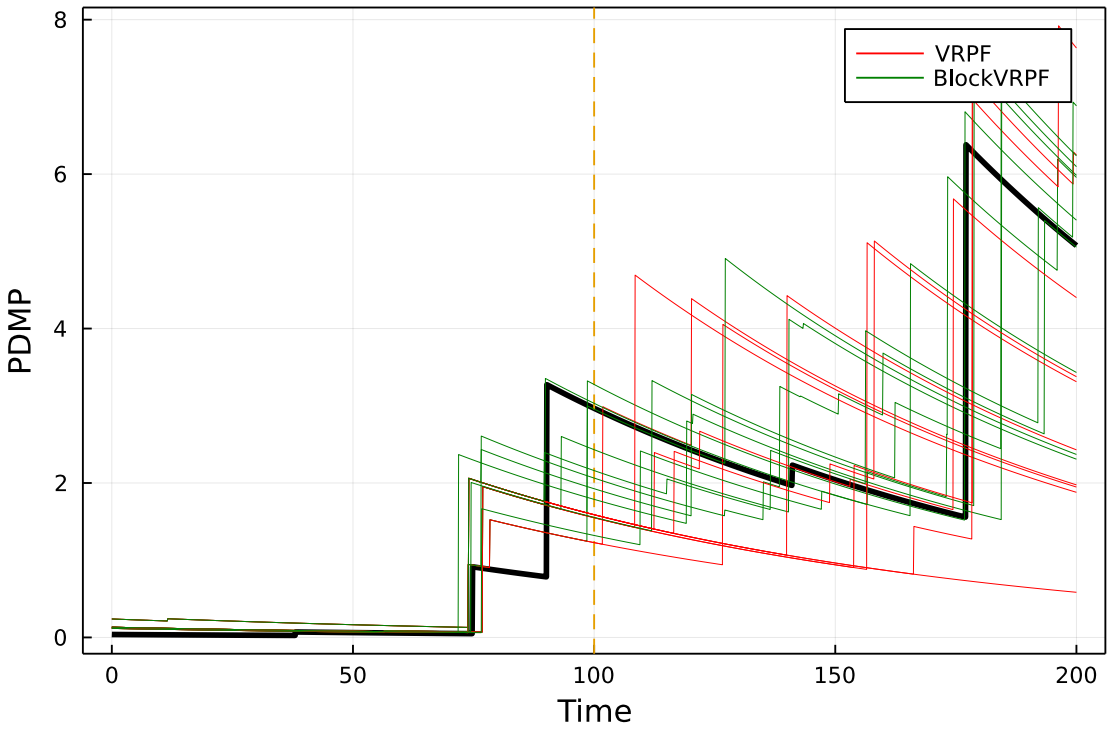

Figure 2.3: Illustration of the correcting power of BlockVRPF sampler. The black line represents the actual PDMP we are interested in. Given the same sampled jumps in the first block (time blocks are split by the yellow dashed line), both VRPF and BlockVPRF samplers were used to sample the jumps in the second block. These sampled processes are represented by red lines (VRPF) and green lines (BlockVRPF)

2.4.3 Comparison between VRPF and BlockVRPF Samplers

To compare the performance of the VRPF and BlockVRPF samplers, we apply both samplers to simulated data of the elementary change-point model shown in Figure 2.4, which are the first 500 observations shown in Figure 2.1. In addition, we deliberately discretise the time so that there are jumps near the end of each time block. The time discretisations are also shown in Figure 2.4. For both algorithms, the number of jumps in each time block is

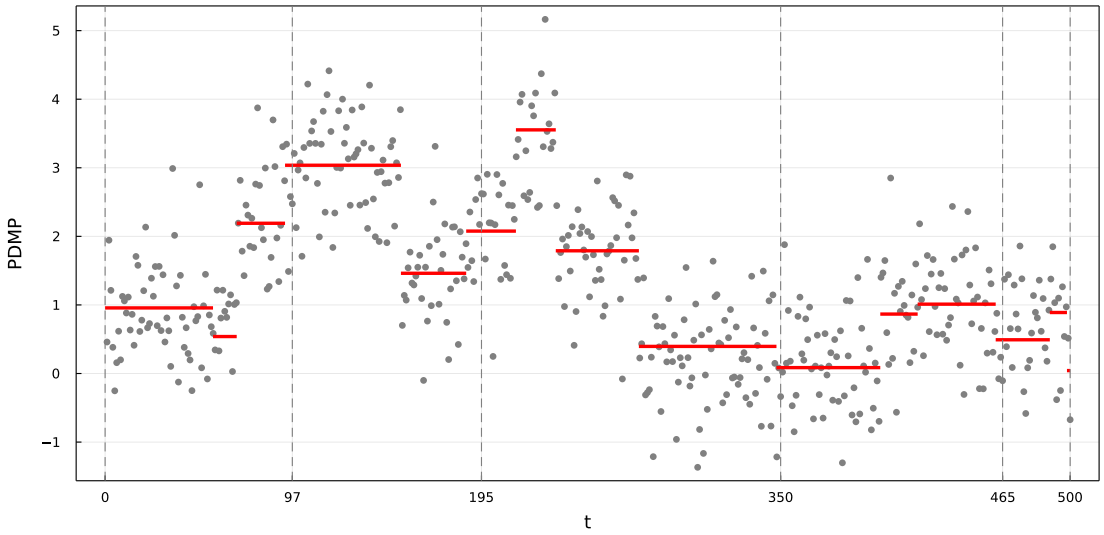

Figure 2.4: Data used in the simulations discussed in section 2.5.1. Grey dotted lines represent the time discretisations used in the simulations.

sampled from a Poisson distribution with mean equal to the length of the block divided by the mean interjump time. Given the number of jumps, we sample the corresponding jump times from the uniform distribution and order them. The jump values are then sampled from their full conditional distributions. For the BlockVRPF algorithm, a *birth* move is proposed with probability $1 - S(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1})$. Otherwise, an *adjustment* is proposed instead. For a *birth* move, the new jump times is proposed uniformly between $\hat{\tau}_{n-1,\hat{k}_{n-1}}$ and t_{n-1}. For an *adjustment*, a modified jump time is proposed from a Gaussian distribution centred at $\hat{\tau}_{n-1,\hat{k}_{n-1}}$ with variance 0.1^2. Conditional on the modified jump times, the corresponding jump values are also sampled from their full conditional distributions. When an *adjustment* is proposed and $k_{n-1} \neq 0$, we use a Gaussian distribution centred at $\mathring{\phi}_{n-1}$ with standard deviation 0.1 and a truncated Gaussian distribution in the interval $(\hat{\tau}_{n-1,\hat{k}_{n-1}-1} \vee t_{n-2}, t_{n-1}]$ with mean $\mathring{\tau}_{n-1}$ and standard deviation 0.2. Resampling is also performed whenever the effective sample size (ESS) drops below half the total number of particles.

For each sampler, we run the algorithm with 500 particles and the true parameter values for $1,000$ times. We also perform backward simulations at the end of each simulation to obtain one sample of the jumps conditional on the observations. From these samples, we find the last jump as well as the first jump in each time block. For each of the actual jumps, we also find the nearest sampled jump times for both VRPF and BlockVRPF methods. Figure 2.5 shows

the sampled jump times obtained from both methods just before and after $t = 350$, which is one of the time discretisation points. We can see that the VRPF method tends to ignore the last jump point before $t = 350$ and place high weights around the previous jump time. It also fails to detect this jump at a later stage, as shown in (b) of Figure 2.5. On the contrary, BlockVRPF successfully detects the jump near $t = 350$ and places a higher probability in the region around that jump as well. Figure 2.6 shows the histograms of the nearest sampled jump times to the actual one obtained from both methods. It is clear from the plots that the histogram for the BlockVRPF methods is more concentrated around the actual jump time near $t = 350$. However, a non-negligible proportion of the sample from the VRPF method is in a region centred at $t = 400$. This suggests that BlockVRPF are more robust in finding and recovering the jumps that are near the time discretisation points. On the contrary, using the VRPF method is more likely to miss these jumps or only find them at times later than the actual ones. Figure 2.7 shows the histograms of the sampled nearest jump times to the actual one near $t = 97$. It is clear that the BlockVRPF method is still robust in finding the jump at the correct position. The VRPF method also manages to find the jump this time. However, it only finds it at a slightly later time than the actual one. As a result, it may bring biases to the estimation when the VRPF method is used.

For the jump times that are further away from the time discretisation points, both VRPF and BlockVRPF sampler will produce similar estimations. For presentation simplicity, we will not include all the plots here and the details of all the simulation results will be included in Appendix. The simulation results discussed in this section indicate that the BlockVRPF method is more robust compared to the VRPF method. However, one may also notice that the calculation of incremental weights for the BlockVRPF sampler involves a likelihood ratio of the part of the observations in the previous block conditional on the new and old jump time and values. This suggests that if a modification significantly improves the estimation of the PDMP, the corresponding incremental weight will be significantly large, making it a dominating particle. This will introduce a larger variance to the incremental weights, resulting in the BlockVRPF potentially having smaller ESSs. The incremental weight also depends on the sampled jumps in the new block at each SMC step. Hence, having proposal kernels that are close approximations to the optimal ones described in Proposition 2 is also crucial to the

performance of the BlockVRPF sampler. Search for optimal proposal kernels or approximations of these kernels will therefore be possible directions of future work.

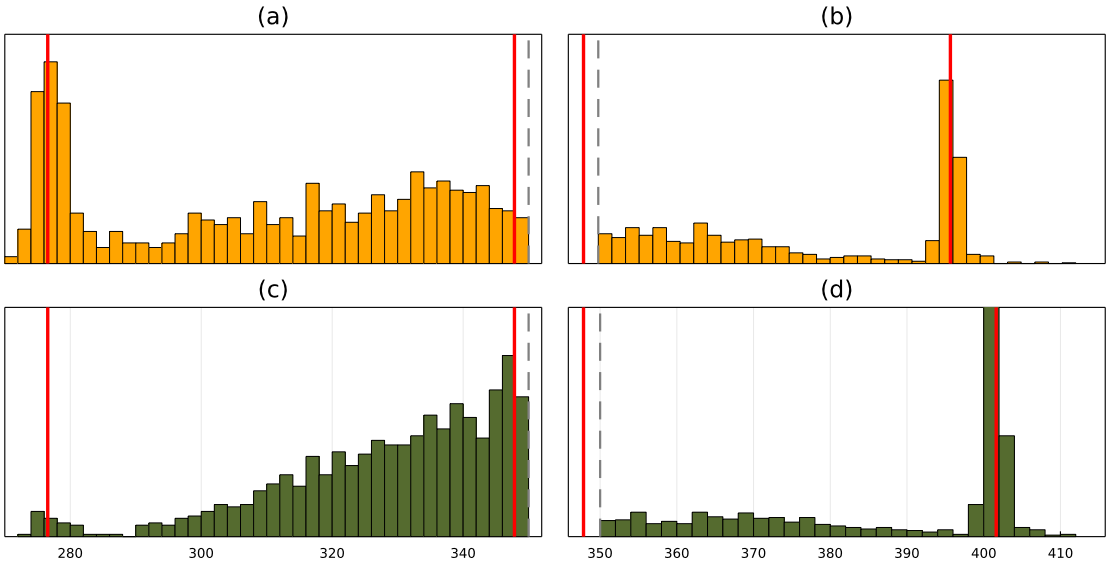

Figure 2.5: Histograms showing the sampled jump times just before and after a time discretisation point (represented by the grey dashed line). Left: last sampled jump times before the discretisation. Right: first sampled jump times after the discretisation. Results obtained using the VRPF and BlockVRPF samplers are in orange and green colour respectively. Red vertical lines are the two most recent actual jump times before and after the time discretisation point.

The BlockVRPF method is also suitable to be used within the particle Gibbs sampler to make inferences on the static parameter in the PDMP models. In this next section, we are going to give an introduction to the particle Gibbs sampler and apply it with both VRPF and BlockVRPF methods to obtain estimations of the posterior distributions of the static parameters for the two models considered in this work.

2.5 Static Parameter Estimation using Particle Gibbs

In the previous section, we proposed a novel method that combines block sampling and VRPF to perform filtering on the piecewise deterministic Markov models, given the values of the static parameters in the model are known. In this section, we are trying to infer these static parameters.

There have been many Bayesian methods based on the SMC sampler proposed by various authors to estimate the static parameter of Markov models, such as Neal (2001) and Chopin

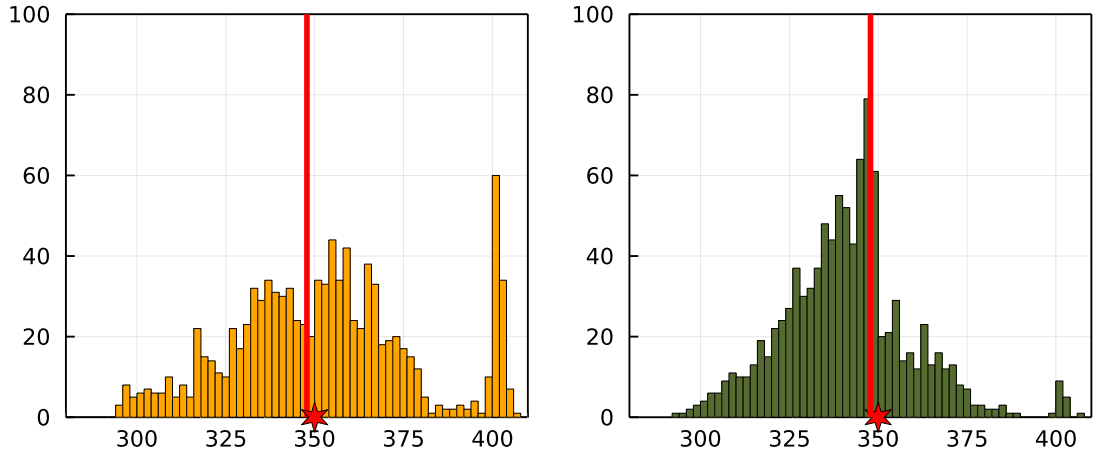

Figure 2.6: Histograms of the sampled nearest jump times to the jump just before $t = 350$ obtained with both methods. Left: results obtained using the VRPF sampler. Right: results obtained using the BlockVRPF sampler. The red line represents the actual jump times and the red star represents the time discretisation point.

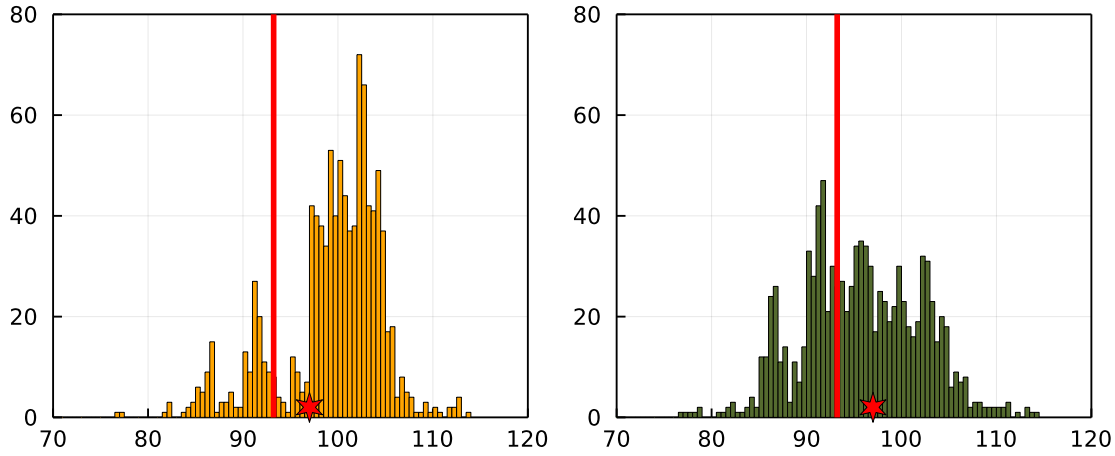

Figure 2.7: Histograms of the sampled nearest jump times to the jump just before $t = 97$ obtained from both methods. Left: results obtained using the VRPF sampler. Right: results obtained using the BlockVRPF sampler. The red line represents the actual jump times and the red star represents the time discretisation point.

(2002). More recently, Andrieu et al. (2010) and Chopin et al. (2013) showed that SMC methods could be combined with Markov Chain Monte Carlo (MCMC) methods to estimate the static parameters in state-space models.

2.5.1 Particle Markov Chain Monte Carlo (PMCMC) Methods

The Particle Markov Chain Monte Carlo (PMCMC) is a class of Bayesian inference methods that combines SMC with MCMC to approximately generate samples from the posterior distribution of the parameters for models whose likelihood is intractable due to the presence of latent variables. Suppose the parameter $\theta \in \Theta$ of the model of interest has a prior density $\pi(\theta)$. Given observations $y_{1:T}$, we are interested in the posterior density $\pi_P(\theta|y_{1:P}) \propto \pi(\theta)p(y_{1:P}|\theta)$. When there are latent variables present in the model, the likelihood $p(y_{1:P}|\theta)$ will be given by

$$p(y_{1:P}|\theta) = \int p(x_{1:P}, y_{1:P}|\theta)dx_{1:P} = \int p(x_{1:P}|\theta)p(y_{1:P}|x_{1:P}, \theta)dx_{1:P}, \tag{2.29}$$

which is intractable in most scenarios. This prevents us from designing standard MCMC samplers targeting $p(\theta|y_{1:P})$. To alleviate this problem, one could include $x_{1:P}$ as auxiliary variables and target the extended density $\pi_P(\theta, x_{1:P}|y_{1:P})$ using the MCMC sampler. By targeting the extended density, we have an opportunity to implement a Gibbs sampler to sample from $\pi_P(\theta, x_{1:P}|y_{1:P})$ by the following scheme:

Step 1. Sample $\theta' \sim \pi_P(\theta|x_{1:P}, y_{1:P})$

Step 2. Sample $x_{1:P} \sim \pi_P(x_{1:P}|\theta', y_{1:P})$

Performing the sampling task in *Step 1* is in general easy. If conjugate priors are used for θ, one can sample exactly from the posterior density. As the unnomarlised density $\pi_P(\theta, x_{1:P}, y_{1:P})$ can be evaluated point-wise, one can also replace *Step 1* with a Metropolis-Hastings step when direct sampling is not possible. On the other hand, sampling from the density in *Step 2* is in general intractable except for some specific scenarios such as linear Gaussian models and finite state hidden Markov models (Andrieu et al., 2010). Hence, practically one should replace *Step 2* with a Metropolis-Hastings update. In order to ensure good performance of the MH update in *Step 2*, one should normally search for a 'good' proposal distribution $q(x_{1:P})$ that is similar to $\pi_P(x_{1:P}|\theta', y_{1:P})$. As a result, a natural idea would be to use the empirical

distribution obtained by running an SMC algorithm targeting $\pi_P(x_{1:P}|\theta', y_{1:P})$ with N particle as the proposal distribution for the MH update. From the previous chapter, we know that the SMC algorithm produces an approximation of its target density by

$$\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P}) := \sum_{n=1}^{N} W_P^n \delta_{x_{1:P}^n}(dx_{1:P}), \tag{2.30}$$

This gives us a way of designing a proposal distribution that is in fact an approximation of the actual density $\pi_P(x_{1:P}|\theta', y_{1:P})$. In fact, one can use the approximation defined in (2.30) as the proposal distribution and this is the key idea behind the PMCMC methods. Sampling from $\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P})$ is simple as one only needs the outputs from a single run of the corresponding SMC algorithm. However, the calculation of the acceptance probability of the MH update requires one to compute the proposal density $q(dx_{1:P})$ that is in this case given by

$$q(x_{1:P}) := \mathbb{E}_{W_{1:P}^{1:N}, x_{1:P}^{1:N}} \left[\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P})\right] \tag{2.31}$$

where the expectation is taken with respect to all the random variables generated by the SMC algorithm (Andrieu et al., 2010). This is in general intractable, making the MH update in *Step 2* impractical. One natural way to solve this problem would be using the 'auxiliary trick' again - to include all the random variables produced during the SMC algorithm as auxiliary variables and interpret the SMC algorithm as a proposal kernel that generates a 'single sample' at each time. More specifically, let $\mathbf{X}_n := \left(X_n^1, X_n^2, ..., X_n^N\right)$ and $\mathbf{A}_{n-1} := \left(A_{n-1}^1, A_{n-1}^2, ..., A_{n-1}^N\right)$ be the particles and ancestor indices generated at step n of the SMC algorithm. Then, all the random variables generated during an SMC algorithm will be $(\mathbf{X}_1, ..., \mathbf{X}_P, \mathbf{A}_1, ..., \mathbf{A}_{P-1})$ and the joint density of these random variables will be given by

$$\psi_P^{N,\theta}(\mathbf{x}_1, ..., \mathbf{x}_P, \mathbf{a}_1, ..., \mathbf{a}_{P-1}) := \left\{\prod_{j=1}^{N} K_1^\theta\left(x_1^j\right)\right\} \prod_{n=2}^{P} \left\{r^\theta\left(\mathbf{a}_{n-1}|\mathbf{W}_{n-1}\right) \prod_{j=1}^{N} K_n^\theta\left(x_n^j|x_{1:n-1}^{a_{n-1}^j}\right)\right\}. \tag{2.32}$$

Such an SMC sampler will produce N distinct paths, i.e. $X_{1:P}^1, X_{1:P}^2, ..., X_{1:P}^N$. For a specific

path, $X_{1:P}^k$, we can denote it as

$$X_{1:P}^k := X_{1:P}^{B_{1:P}^k} = \left(X_1^{B_1^k}, X_2^{B_2^k}, ..., X_P^{B_P^k}\right).$$

with $B_P^k = k$ and $B_n^k = A_n^{B_{n+1}^k}$. To sample a single path from the proposal, one just need to sample the lineage index k with probability W_P^k and set $x_{1:P}^{'} := x_{1:P}^k := x_{1:P}^{b_{1:P}}$ with $b_P = k$. Since all the random variables from an SMC algorithm are now included in the proposal distribution, we should also define a corresponding extended target distribution that includes θ and $(\mathbf{X}_1, ..., \mathbf{X}_P, \mathbf{A}_1, ..., \mathbf{A}_{P-1})$. The design of the target distribution should fulfil the following two conditions:

Condition 1. The target distribution should still admit $\pi_P(\theta, x_{1:P}|y_{1:P})$ as a marginal.

Condition 2. The target distribution should be as close as possible to the proposal distribution in a certain sense.

The first condition needs to be fulfilled to ensure that we are still able to target the correct distribution as a marginal. We try to achieve the second condition in order to make sure that the MH update targeting this extended distribution is as efficient as possible. Inspired by this, we should consider factorising the extended target density in the form

$$\tilde{\pi}_P(\theta, b_{1:P}, \mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) = \frac{\pi_P(\theta, x_{1:P}^{b_{1:P}}, b_{1:P})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}} | \theta, x_{1:P}^{b_{1:P}}, b_{1:P}), \quad (2.33)$$

where we define $\mathbf{x}_{1:P}^{-b_{1:P}} := \mathbf{x}_{1:P} \backslash x_{1:P}^{b_{1:P}}$ and similarly $\mathbf{a}_{1:P-1}^{-b_{2:P}} := \mathbf{a}_{1:P-1} \backslash a_{1:P-1}^{b_{2:P}}$. The first part of (2.33) is included to meet the first condition. Practically, one could simply sample a lineage index k with probability W_P^k and set $b_P := k$ and this would implicitly define the values of $b_{P-1}, ..., b_1$ through the relationship $b_n = a_n^{b_{n+1}}$. If the support of the proposal distributions used in the SMC algorithm, $K_1(dx_1) \prod_{j=2}^P K_j(dx_j|x_{1:j-1})$, encompasses that of π_P, the marginal density $\pi_P(\theta, x_{1:P}^{b_{1:P}}, b_{1:P})/N^P$ would be the same as π_P, which is the actual density of our interest. The second part of (2.33) is designed to meet the second requirement. As we need to define the distribution of the rest of the particles and ancestor indices and hope to design a distribution that is close to $\psi_P^{N,\theta}$. One way to do this is to define a conditional distribution of the rest of the particles and ancestor indices. Hence, we can define the joint

distribution of $\mathbf{X}_{1:P}^{-k}$ and $\mathbf{A}_{1:P-1}^{-k}$ to be

$$\begin{aligned}\psi_P^{N,\theta}\left(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}} | x_{1:P}^{b_{1:P}}, b_{1:P}\right) &:= \frac{\psi_P^{N,\theta}(\mathbf{x}_{1:P}, \mathbf{a}_{1:P})}{K_1\left(x_1^{b_1}\right)\prod_{n=2}^{P}\left\{r\left(b_{n-1}|\mathbf{W}_{n-1}\right)K_n\left(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}\right)\right\}} \\ &= \left\{\prod_{1\leq j\leq N, j\neq b_1} K_1\left(x_1^j\right)\right\}\prod_{n=2}^{P} r\left(\mathbf{a}_{n-1}^{-b_n}|\mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \\ &\quad \times \prod_{1\leq j\leq N, j\neq b_n} K\left(x_n^j | x_{1:n-1}^{a_{n-1}^j}\right).\end{aligned} \tag{2.34}$$

Since Equation (2.34) is the conditional distribution of $\left(\mathbf{X}_{1:P}^{-b_{1:P}}, \mathbf{A}_{1:P-1}^{-b_{2:P}}\right)$ given the path $\left(x_{1:P}^{b_{1:P}}, b_{1:P}\right)$ in $\tilde{\pi}_P^N$. This actually inspires the idea of the particle Gibbs sampler described in Andrieu et al. (2010). At each iteration, given we have obtained $\left(\theta, b_{1:P}, x_{1:P}^{b_{1:P}}, \mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}}\right)$, we can do the following to obtain a new set of random variables

Step 1. Sample $\theta^* \sim \tilde{\pi}_P^N\left(\theta | b_{1:P}, x_{1:P}^{b_{1:P}}, \mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}}\right)$

Step 2. Sample $\mathbf{X}_{1:P}^{*,-b_{1:P}}, \mathbf{A}_{1:P-1}^{*,-b_{2:P}} \sim \psi_P^{N,\theta^*}\left(\mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}} | b_{1:P}, x_{1:P}^{b_{1:P}}\right)$

Step 3. Sample $b_{1:P}^*, x_{1:P}^{b_{1:P}^*} \sim \tilde{\pi}_P^N\left(b_{1:P}^*, x_{1:P}^{b_{1:P}^*} | \theta^*, \mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}}, b_{1:P}, x_{1:P}^{b_{1:P}}\right)$

As described in Andrieu et al. (2010), *Step 1* of the above sampling scheme can be simplified to sampling $\theta^* \sim \pi_P(\theta^* | x_{1:P}^{b_{1:P}})$ and this still leaves the target density $\tilde{\pi}_P^N$ invariant. This is a special type of Gibbs sampler known as the 'collapsed' Gibbs sampler which was discussed in Liu (2001) and Van Dyk and Park (2008). For complex models, directly sampling from $\pi_P(\theta^* | x_{1:P}^{b_{1:P}})$ may not be possible either. In this case, one can insert a Metropolis-Hastings update to *Step 1*. Given a transition kernel $q(\, d\theta' | \theta)$, one can sample a candidate θ' and set $\theta^* := \theta'$ with probability

$$\alpha(\theta, \theta') := 1 \wedge \frac{\pi_P(\theta' | x_{1:P}^{b_{1:P}}) q(\theta | \theta')}{\pi_P(\theta | x_{1:P}^{b_{1:P}}) q(\theta' | \theta)}$$

Such a technique is termed as *Metropolis-Hastings within Partially Collapsed Gibbs* (MHwPCG) and its correctness was described in Van Dyk and Park (2008). *Step 2* of the scheme involves sampling from the conditional distribution defined by $\psi_P^{N,\theta^*}\left(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}} | x_{1:P}^{b_{1:P}}, b_{1:P}\right)$. One can see that this conditional distribution only depends on the transition kernels $K_{1:P}$ and the resampling scheme r used in the SMC algorithm. Hence, to sample from the conditional,

one can employ an algorithm similar to the standard SMC algorithm, except for keeping a particular particle trajectory $x_{1:P}^{b_{1:P}}$ fixed. Such an SMC sampler is termed as the conditional SMC (cSMC) sampler and was first introduced in Andrieu et al. (2010). The details of the sampler are listed in Algorithm 2.4.

Algorithm 2.4: Conditional SMC Sampler (cSMC)

Given a path $(X_{1:P}^*, B_{1:P})$;
for *n=1* **do**
  for $j = 1, 2, 3, ..., N$ **do**
    if $j \neq B_1$ **then**
      Sample $X_1^j \sim K_1(\cdot)$;
    if $j = B_1$ **then**
      Set $X_1^j = X_1^*$;
    Calculate the weight w_1^j and normalise the weights $W_1^j \propto w_1^j$;
for $n = 2, 3, .., P$ **do**
  for $j = 1, 2, .., N$ **do**
    if $j \neq B_n$ **then**
      Sample $A_{n-1}^j \sim r\left(\cdot|\mathbf{W}_{n-1}\right)$;
      Sample $X_n^j \sim K_n\left(\cdot|X_{1:n-1}^{A_{n-1}^j}\right)$;
    if $j = B_n$ **then**
      Set $A_{n-1}^j = B_{n-1}$;
      Set $X_n^j = X_n^*$;
    Calculate the weights w_n^j and normalise them $W_n^j \propto w_n^j$;

Note that one could permute the indices of the particles in an SMC sampler while keeping the SMC sampler invariant. Hence, one could practically fix $b_{1:P}$ to some convenient values (e.g $b_i = 1, i = 1, 2, .., P$) to simplify the implementation of the cSMC sampler.

Step 3 samples the ancestor indices $b_{1:P}$ and the corresponding trajectory defined by the indices $x_{1:P}^{b_{1:P}}$. Following (Lindsten and Schön, 2013, Chapter 5), the marginal is given by $\pi_P(\theta, x_{1:p}) \propto \pi(\theta)\pi_P^\theta(x_{1:P})$ and we can rewrite $\pi_P^\theta(x_{1:P})$ as

$$\pi_P^\theta(x_{1:P}) := \pi_1^\theta(x_1) \prod_{n=2}^{P} \frac{\pi_n^\theta(x_{1:n})}{\pi_{n-1}^\theta(x_{1:n-1})}.$$

Moreover, in an SMC sampler, the unnormalised weight w_n is given by

$$w_n := \frac{\pi_n^\theta(x_{1:n})}{\pi_{n-1}^\theta(x_{1:n-1})K_n^\theta(x_n|x_{1:n-1})}.$$

Hence, we have that

$$\pi_P^\theta(x_{1:P}) := w_1 K_1(x_1) \prod_{n=2}^{P} w_n K_n^\theta(x_n|x_{1:n-1}).$$

Hence, if we plugin $x_{1:P}^{b_{1:P}}$, we would obtain

$$\begin{aligned}
\pi_P^\theta(x_{1:P}^{b_{1:P}}) &:= w_1^{b_1} K_1(x_1^{b_1}) \prod_{n=2}^{P} w_n^{b_n} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \\
&= \left\{ \frac{w_1^{b_1}}{\sum_{i=1}^N w_1^i} K_1(x_1^{b_1}) \prod_{n=2}^{P} \frac{w_n^{b_n}}{\sum_{i=1}^N w_n^i} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\} \\
&= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^{P} W_{n-1}^{b_{n-1}} K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\} \\
&= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^{P} r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\}.
\end{aligned} \tag{2.35}$$

If we substitute this into (2.33), we will obtain, after simplification

$$\tilde{\pi}_P(\theta, b_{1:P}, \mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) = \frac{\pi(\theta) W_P^{b_P}}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\} \propto W_P^{b_P}.$$

Hence, we can see that to perform *Step 3*, one simply choose $b_P = k$ with probability W_P^k and set $b_n := a_n^{b_{n+1}}$ for $n = P-1, P-2, ..., 2, 1$. The corresponding trajectory will then be obtained deterministically. These three steps complete one sweep of the particle Gibbs sampler, and its full implementation is listed in Algorithm 2.5.

2.5.2 Particle Gibbs with Backward Sampling

We discussed the particle Gibbs sampler in the previous section. One of the major advantages of such type of sampler is that it does not rely on the asymptotics of N to be a valid MCMC sampler. However, the particle Gibbs sampler we discussed in the previous section is likely to

Algorithm 2.5: particle Gibbs sampler

Initialise at any $X_{1:P}^{(0)}$,$k^{(0)}$ and $B_{1:P}^{(0)}$;
for $n = 1, 2, ..$ **do**
  Sample $\theta^{(n)} \sim \pi_P\left(\cdot|x_{1:P}^{(n-1)}\right)$;
  Sample $\mathbf{X}_{1:P}^{(n),-k^{(n-1)}}, \mathbf{A}_{1:P-1}^{(n),-k^{(n-1)}}$ by running a cSMC sampler conditional on $X_{1:P}^{(n-1)}$ and $B_{1:P}^{(n-1)}$, outlined in Algorithm 2.4;
  Sample $k^{(n)} = j$ with probability W_P^j. Set $B_P^{(n)} = j$ and $B_m^{(n)} = A_m^{B_{m+1}^{(n)},k^{(n)}}$ for $m = P-1, .., 2, 1$;
  Set $X_{1:P}^{(n)} = \left(X_1^{(n),B_1^{(n)}}, ..., X_P^{(n),B_P^{(n)}}\right)$;

have mixing issues. In the particle Gibbs sampler we discussed before, we only sample the ancestor index at P according to the importance weight $\mathbf{W}_P$, and trace the ancestral lineage back of the particles to get a full path $X_{1:P}$ at each step. This may result in the particle Gibbs sampler having poor mixing when there is significant degeneracy in the cSMC sampler. Such a problem is especially obvious when P is large and N is small since a longer time and a small number of particles often come with degeneracy. As the particle trajectory sampled in the previous iteration must be kept fixed in the cSMC algorithm, the next sampled particle trajectory would be therefore very similar to the previous one, resulting in poor mixing of the Gibbs sampler. One way to solve this problem was proposed by Whiteley (2010) and further explored later in Lindsten and Schön (2012). The idea is to insert a backward simulation step to the particle Gibbs sampler to mitigate path degeneracy issues. In the context of particle Gibbs sampler, one can interpret this changing *Step 3* of the particle Gibbs sweep to simulations of $b_P, b_{P-1}, .., b_1$. Hence, instead of performing the original *Step 3*, we do the following

- Sample $b_P^* \sim \tilde{\pi}_P^N(b_P^*|\theta^*, \mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}}, b_{1:P}, x_{1:P}^{b_{1:P}})$
- Sample $b_{P-1}^* \sim \tilde{\pi}_P^N(b_{P-1}^*|\theta^*, \mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}}, b_{1:P-1}, x_{1:P-1}^{b_{1:P-1}}, x_P^{b_P^*}, b_P^*)$
- .....
- Sample $b_t^* \sim \tilde{\pi}_P^N(b_t^*|\theta^*, \mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}}, b_{1:t}, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}, b_{t+1:P}^*)$
- .....
- Sample $b_1^* \sim \tilde{\pi}_P^N(b_1^*|\theta^*, \mathbf{x}_{1:P}^{*,-b_{1:P}}, \mathbf{a}_{1:P-1}^{*,-b_{2:P}}, b_1, x_1^{b_1}, x_{2:P}^{b_{2:P}^*}, b_{2:P}^*)$

Now, we are going to derive the conditional distribution of b_t^* listed above. One can see that the conditional distributions of b_t^* is independent of $\mathbf{x}_{t+1:P}^{-b_{t+1:P}^*}$ and $\mathbf{a}_{t:P-1}^{-b_{t+1:P}^*}$. Therefore, by marginalising $\psi_P^{N,\theta}$ over $\mathbf{x}_{t+1:P}^{-b_{t+1:P}}$ and $\mathbf{a}_{t:P-1}^{-b_{t+1:P}}$, we obtain

$$\psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}} | x_{1:P}^{b_{1:P}}, b_{1:P}) := \prod_{\substack{j=1 \\ j\neq b_1}}^{N} K_1\left(x_1^j\right) \prod_{n=2}^{t} \left\{ r\left(\mathbf{a}_{n-1}^{-b_n} | \mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \prod_{\substack{j=1 \\ j\neq b_n}}^{N} K_n(x_n^j | x_{1:n-1}^{a_{n-1}^j}) \right\}. \tag{2.36}$$

Hence, the conditional distribution of b_t^* will then be given by

$$\tilde{\pi}_P^N(b_t^* | \theta^*, \mathbf{x}_{1:t}^{*,-b_{1:t}}, \mathbf{a}_{1:t-1}^{*,-b_{2:t}}, b_{1:t}, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}, b_{t+1:P}^*)$$

$$= \frac{\pi_P(\theta^*, b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}} | b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})$$

$$= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})\pi(\theta^*)}{N^P} \prod_{\substack{j=1 \\ j\neq b_1}}^{N} K_1\left(x_1^j\right) \prod_{n=2}^{t} \left\{ r\left(\mathbf{a}_{n-1}^{-b_n} | \mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \prod_{\substack{j=1 \\ j\neq b_n}}^{N} K_n(x_n^j | x_{1:n-1}^{a_{n-1}^j}) \right\}$$

$$= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi(\theta^*) W_t^{b_t}}{N^P} \psi_t^{N,\theta^*}(\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \left\{ \prod_{n=1}^{t} \sum_{i=1}^{N} w_n^i \right\} \propto W_t^{b_t} \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})}, \tag{2.37}$$

where the last line is derived in a similar way to what we did in (2.35). In fact, the backward sampling step inserted in the particle Gibbs corresponds to the backward smoothing schedule of a standard SMC algorithm. We summarise the corresponding particle Gibbs with backward sampling (PGBS) sampler in Algorithm 2.6

The PGBS sampler generally has much better convergence properties than the standard particle Gibbs sampler. Due to the addition of the backward sampling step, we are able to explore all the possible trajectories created by the combinations of particles obtained from the cSMC algorithm. As a result, the new sampled trajectory is highly likely to be significantly different from the previous one. This brings a big improvement in the mixing in the state space, hence

Algorithm 2.6: particle Gibbs with Backward Simulation

Start at $\theta^{(0)}$, $X_{1:P}^{(0)}$ and $B_{1:P}^{(0)}$ arbitrarily;
for *n=1,2,...* **do**
  Sample $\theta^{(n)} \sim \pi_P\left(\cdot|X_{1:P}^{(n-1)}\right)$ using e.g. a MH kernel;
  Run a cSMC algorithm conditional on $x_{1:P}^{(n-1)}, b_{1:P}^{(n-1)}$ and $\theta^{(n)}$, outlined in Algorithm 2.4, to get $\mathbf{X}_{1:P}^{(n)}$ and $\mathbf{A}_{1:P-1}^{(n)}$ and $\mathbf{W}_{1:P}^{(n)}$;
  for $t = P, P-1, ..., 2, 1$ **do**
    Set $b_t^{(n)} = k$ with probability

$$W_t^{(n),k} \frac{\pi_P^{\theta^{(n)}}(x_{1:t}^{(n),k}, x_{t+1:P}^{(n),b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{(n),k})}$$

    where $b_t = k$ and $b_j = a_j^{(n),b_{j+1}}$
  Set $x_{1:P}^{(n)} := x_{1:P}^{(n),b_{1:P}^{(n)}}$

the particle Gibbs sampler overall.

For the VRPF method, let $\check{\mathcal{J}}_n := (\check{k}_n, \check{\tau}_{n,1:\check{k}_n}, \check{\phi}_{n,1:\check{k}_n})$ be the collection of jump times and values that define the PDP in the interval (t_{n-1}, t_P), we then have that

$$\check{k}_n := \sum_{j=n}^{P} k_j \qquad \check{\tau}_{n,1:\check{k}_n} := \bigcup_{j=n}^{P} \{\tau_{j,1:k_j}\} \qquad \check{\phi}_{n,1:\check{k}_n} := \bigcup_{j=n}^{P} \{\phi_{j,1:k_j}\}.$$

Suppose the ancestor indices $b_{n+1}^*, .., b_P^*$ have already been sampled and the corresponding particles $x_n^{b_n^*}, ..., x_P^{b_P^*}$ form $\check{\mathcal{J}}_{n+1}^*$. If $\check{k}_{n+1}^* > 0$, the unnormalised backward sampling incremental weight will be given by

$$\mathcal{G}_{n|P}(x_{1:n}^{b_{1:n}}) := \frac{f(\check{\tau}_{n+1,1}^*|\hat{\tau}_{n,\hat{k}_n}) g(\check{\phi}_{n+1,1}^*|\hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}_{n+1,1}^*) p(y_{(t_n,\check{\tau}_{n+1,1}^*]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}. \tag{2.38}$$

If $\check{k}_{n+1}^* = 0$, the corresponding backward sampling incremental weight is given by

$$\mathcal{G}_{n|P}(x_{1:n}^{b_{1:n}}) = \frac{S(\hat{\tau}_{n,\hat{k}_n}, t_P) p(y_{(t_n,t_P]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}. \tag{2.39}$$

For the BlockVRPF sampler, we use the same notation $\check{\mathcal{J}}_n$ to denote the collection of jump times and values defined by $Z_n, .., Z_p$. Given $Z_{n+1}^{b_{n+1}^*}, .., Z_P^{b_P^*}$ and the corresponding $\check{\mathcal{J}}_n^*$, if

$M_n^* = 1$, the corresponding backward sampling incremental weight would be given by

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{\mathbb{I}(\mathring{\tau}_n^* > \hat{\tau}_{n,\hat{k}_n}) f(\mathring{\tau}_n^* | \hat{\tau}_{n,\hat{k}_n}) g(\mathring{\phi}_n^P * | \hat{\phi}_{n,\hat{k}_n}, \mathring{\tau}_n^*, \hat{\tau}_{n,\hat{k}_n}) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n) p(y_{(\mathring{\tau}_n^*, t_n]} | \hat{\phi}_{n,\hat{k}_n})}. \quad (2.40)$$

When $M_n^* = 0$ and $U_n^* = \emptyset$, if $\check{k}_n = 0$, we have

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{\mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1}) S(\hat{\tau}_{n,\hat{k}_n}, t_P) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}) p(y_{(t_n, t_P]} | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}. \quad (2.41)$$

If $\check{k}_n^* > 0$, we then have

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{f(\check{\tau}_{n+1,1}^* | \hat{\tau}_{n,\hat{k}_n}) g(\check{\phi}_{n+1,1}^* | \hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}_{n+1,1}^*) p(y_{(t_n, \check{\tau}_{n+1,1}^*]} | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}$$
$$\times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1}) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}). \quad (2.42)$$

In the case $U_n^* \neq \emptyset$, we should have

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{f(\mathring{\tau}_n^* | \hat{\tau}_{n,\hat{k}_n - 1}) g(\mathring{\phi}_n^* | \hat{\phi}_{n,\hat{k}_n - 1}, \hat{\tau}_{n,\hat{k}_n - 1}, \mathring{\tau}_n^*)}{f(\hat{\tau}_{n,\hat{k}_n} | \hat{\tau}_{n,\hat{k}_n - 1}) g(\hat{\phi}_{n,\hat{k}_n} | \hat{\phi}_{n,\hat{k}_n - 1}, \hat{\tau}_{n,\hat{k}_n - 1}, \mathring{\tau}_n^*)} \times \frac{p(y_{(\hat{\tau}_{n,\hat{k}_n} \wedge \mathring{\tau}_n^*, \mathring{\tau}_n^*)} | \hat{\phi}_{n,\hat{k}_n - 1})}{p(y_{(\hat{\tau}_{n,\hat{k}_n} \wedge \mathring{\tau}_n^*, t_n)} | \hat{\phi}_{n,\hat{k}_n - 1}, \hat{\phi}_{n,\hat{k}_n})}$$

$$\times \frac{\mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}) \lambda(\hat{\tau}_{n,\hat{k}_n}, \hat{\phi}_{n,\hat{k}_n} | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} > t_{n-1}, \mathring{\tau}_n^* > \hat{\tau}_{n,\hat{k}_n - 1}). \quad (2.43)$$

2.5.3 Auxiliary Variable Rejuvenation

In this section, we describe the auxiliary rejuvenation step proposed by Finke et al. (2014) that is crucial to ensure the correctness of particles Gibbs with BlockVRPF sampler. Recall that when using the BlockVRPF sampler, the target distributions of the particle filter are given by

$$\bar{\gamma}_t^\theta(Z_{1:t}) := \gamma_t^\theta(x_{1:t-1}(2), x_t(1)) \mu_t^\theta(m_{1:t-1} | x_{1:t-1}(2), x_t(1))$$
$$\times \prod_{j=1}^{t-1} \lambda_j^\theta(\bar{u}_j | m_{1:j}, x_{1:j}(2), x_{j+1}(1)) \quad (2.44)$$

for $t = 1, 2, .., P$. Note that in the context of the PGS, the static parameter θ of the model is explicitly stated in the target density to remind us that they need to be sampled from the sampler as well. Hence, at the n-th sweep of the particle Gibbs sampler, a new parameter set $\theta^{(n)}$ should be sampled conditional on $Z_{1:P}^{(n-1)}$ according to (2.44). However, due to the presence of the auxiliary variables $m_{1:n-1}^{(n-1)}$ and $\bar{u}_{1:P-1}^{(n-1)}$, sampling from the full distribution would be computationally expensive. Moreover, since what we are actually interested in is the marginal distribution $\gamma_P^{\theta}(x_{1:P-1}(2), x_P(1))$, sampling from the full distribution would also potentially bring extra complexity. Hence, one could alternatively sample $\theta^{(n)}$ from the target distribution $p(\theta|x_{1:P-1}^{(n-1)}(2), x_P^{(n-1)}(1)) \propto \pi(\theta)\gamma_P^{\theta}(x_{1:P-1}^{(n-1)}(2), x_P^{(n-1)}(1))$ only, where $\pi(\theta)$ is the prior distribution of θ. To ensure that the particle Gibbs sampler still targets the correct distribution, the auxiliary variables $m_{1:P-1}, \bar{u}_{1:p-1}$ should be sampled given $\theta^{(n)}$ according to $\mu_P^{\theta^{(n)}}$ and $\{\lambda_j^{\theta^{(n)}}\}_{j=1,..,P-1}$. This is the *auxiliary variable rejuvenation* step and it can be viewed as being a partially collapsed Gibbs sampler (Liu et al., 1994, Van Dyk and Park, 2008). For notational simplicity, we use $\mathcal{J}_P$ to represent the collection $(x_{1:P-1}(2), x_P(1))$. Also, we denote $\mathcal{M}_P$ to represent the collection $(m_{1:P-1}, \bar{u}_{1:P-1})$ and $\Gamma_P^{\theta}(\mathcal{M}_P|\mathcal{J}_P) := \mu_t^{\theta}(m_{1:t-1}|\mathcal{J}_P)\prod_{j=1}^{t-1}\lambda_j^{\theta}(\bar{u}_j|m_{1:j}, \mathcal{J}_P)$. Hence, once we have obtained $(\theta^{(n-1)}, b_{1:P}^{(n-1)}, \mathcal{J}_P^{(n-1)}, \mathcal{M}_P^{(n-1)}, \mathbf{Z}_{1:P}^{(n-1),-b_{1:P}^{(n-1)}}, \mathbf{A}_{1:P-1}^{(n-1),-b_{2:P}^{(n-1)}})$ at step $n-1$, the particle Gibbs sampler with auxiliary variable rejuvenation could perform the following steps at sweep n:

S1. Sample $\theta^{(n)}, b_{1:P}^*, \mathcal{M}_P^*, \mathbf{Z}_{1:P}^{*,-b_{1:P}^*}, \mathbf{A}_{1:P-1}^{*,-b_{2:P}^*} \sim p(\cdot|\mathcal{J}_P^{(n-1)})$

S2. Sample $\mathcal{M}_P^{**}, b_{1:P}^{**}, \mathbf{Z}_{1:P}^{**,-b_{1:P}^{**}}, \mathbf{A}_{1:P-1}^{**,-b_{2:P}^{**}} \sim p(\cdot|\theta^{(n)}, \mathcal{J}_P^{(n-1)})$

S3. Obtain the conditional path $Z_{1:P}^*$ from $\mathcal{M}_P^{**}$ and $\mathcal{J}_P^{(n-1)}$ and sample

$$b_{1:P}^{***}, \mathbf{Z}_{1:P}^{(n),-*}, \mathbf{A}_{1:P-1}^{(n),-*} \sim p(\cdot|\theta^{(n)}, Z_{1:P}^*)$$

where $\mathbf{Z}_{1:P}^{(n),-*}, \mathbf{A}_{1:P-1}^{(n),-*}$ represent the new particles and ancestor indices obtained by running cSMC algorithm conditional on $Z_{1:P}^*$ and $\theta^{(n)}$.

S4. Sample $b_{1:P}^{(n)} \sim p(\cdot|\theta^{(n)}, Z_{1:P}^*, \mathbf{Z}_{1:P}^{(n),-*}, \mathbf{A}_{1:P-1}^{(n),-*})$ and obtain $\mathcal{J}_P^{(n)}$ accordingly.

One can see that *S1* can be collapsed by removing $b_{1:P}^*, \mathcal{M}_P^*, \mathbf{Z}_{1:P}^{*,-b_{1:P}^*}, \mathbf{A}_{1:P-1}^{*,-b_{2:P}^*}$ without af-

fecting the validity of the Gibbs sampler. As a result, $p(\cdot|\mathcal{J}_P^{(n-1)})$ will be a density proportional to $\pi(\theta)\gamma_P^\theta(x_{1:P-1}^{(n-1)}(2), x_P^{(n-1)}(1))$. *S1* is important to make sure that the resulting Gibbs sampler converges to the correct stationary distribution hence it must be kept. However, one can see that $b_{1:P}^{**}, \mathbf{Z}_{1:P}^{**,-b_{1:P}^{**}}, \mathbf{A}_{1:P-1}^{**,-b_{2:P}^{**}}$ will not affect later simulations hence can be omitted as well. By trimming these variables, *S2* becomes

S2'. Sample $\mathcal{M}_P^{**} \sim p(\cdot|\theta^{(n)}, \mathcal{J}_P^{(n-1)}) := \Gamma_P^{\theta^{(n)}}(\mathcal{M}_P|\mathcal{J}_P^{(n-1)})$

This is exactly the rejuvenation step we discussed before. In *S3*, $b_{1:P}^{***}$ can also be trimmed for the same reason and then this can be done by running a cSMC algorithm conditional on the lineage $Z_{1:P}^*$ obtained from $\mathcal{J}_P^{(n-1)}$ and $\mathcal{M}_P^{**}$ from the rejuvenation step. *S4* is finally performed to obtain the new sampled PDMP $\mathcal{J}_P^{(n)}$ and this step can also be replaced by backward simulation outlined in Algorithm 2.6.

We can now see that the rejuvenation step is crucial to ensure the validity of the particle Gibbs sampler when the BlockVRPF method is used. The resulting algorithm is given in Algorithm 2.7. In addition to ensuring the correctness of the sampler, the rejuvenation step also has the potential to improve the mixing of particle Gibbs sampler when the BlockVRPF method is used. Looking at the incremental weight at step n given in (2.27a), (2.27b) and (2.28), one notices that when a *birth* or an *adjustment* is proposed, the likelihood ratio of part of the observations in the interval $(t_{n-2}, t_{n-1}]$ given the modified and original PDMP is included in the incremental weights. Hence, when a modification moves the PDMP closer to the true process, this likelihood ratio is going to be large, resulting in the particle filter having a few dominant particle(s) at certain SMC step(s). Consequently, the cSMC sampler with backward simulation will be likely to produce the same sampled path for many iterations, making the sampling of the hidden PDMP stuck at a local mode. Rejuvenating the auxiliary variables potentially solves this problem. By resampling the modifications and 'old' jumps, the modifications contained in $x_{1:P-1}(2)$ may become less significant. As a result, the incremental weights will decrease, making the sampler more likely to escape from a local mode.

Although the rejuvenation step has the potential to improve mixing, the actual effect will heavily depend on the choice of the backward kernels $\{\mu_n\}$ and $\{\lambda_n\}$ appearing in the target distribution. Ideally, we want to sample $\{\bar{u}_j\}$ that are not far from the modifications $\{u_j\}$ to

Algorithm 2.7: particle Gibbs with rejuvenation

Initialise the chain at $\theta^{(0)}, b_{1:P}^{(0)}, \mathcal{J}_P^{(0)}, \mathcal{M}_P^{(0)}, \mathbf{Z}_{1:P}^{(0),-b_{1:P}^{(0)}}, \mathbf{A}_{1:P-1}^{(0),-b_{2:P}^{(0)}}$;

for *n=1,2,..,N* **do**

- Sample $\theta^{(n)} \sim \pi(\theta)\gamma_P^{\theta}(x_{1:P}^{(n-1)}(2), x_P^{(n-1)}(1))$. If directly sampling is not possible, $\theta^{(n)}$ can be obtained by e.g. applying Metropolis-Hastings kernel;
- Perform the rejuvenation step, i.e. Sample $\mathcal{M}_P^* \sim \Gamma_P^{\theta^{(n)}}(\mathcal{M}_P|\mathcal{J}_P^{(n-1)})$;
- Obtain the conditional path $Z_{1:P}^*$ from $\mathcal{J}_P^{(n-1)}$ and $\mathcal{M}_P^*$ through a deterministic transformation;
- Run cSMC algorithm given $Z_{1:P}^*$ and $\theta^{(n)}$ to obtain $\mathbf{Z}_{1:P}^{(n),-*}, \mathbf{A}_{1:P-1}^{(n),-*}$;
- Through backward simulation, obtain $b_{1:P}^{(n)}$ hence obtain $\mathcal{J}_P^{(n)}$

make sure that the incremental weights after rejuvenation are not too large. In the numerical studies of this chapter, we propose to use a Gaussian kernel centred at the modified jump times and values with a small variance. However, using Gaussian kernels creates a new problem - if the variance is set to be too small, a large change in the last jump proposed by u_j (which often indicates a correction to a wrong jump value/time) would have tiny or even 0 weight. Hence, one should carefully design the backward kernels to ensure that the rejuvenation steps indeed bring improvements to the mixings.

2.5.4 Numerical Examples

In this section, we apply the particle Gibbs sampler with both the VRPF and BlockVRPF method to perform Bayesian inference on two challenging examples discussed in section 2.2 - the *Elementary Change-point Model* and the *Short-noise Cox Model*. We follow what we have done in section 2.4.3 to perform the SMC and cSMC algorithms. In addition, we split the time horizon into 10 equal intervals when performing the (c)SMC algorithm. Moreover, we apply the random-walk Metropolis-Hastings algorithm with 500 iterations to obtain new parameter sets at every Gibbs sweep. The proposal covariance matrices used in the random-walk MH algorithm are obtained through pilot runs of the particle Gibbs sampler. For both models, we run the particle Gibbs for $60,000$ iterations with the first $10,000$ iterations discarded as the burning period.

Shot-noise Cox Model

In this section, we represent the simulation results for the shot-noise Cox model. To avoid resampling *births* during rejuvenation, the backward kernel μ_n will be set to be a *Bernoulli*(0.1). When $m_n = 0$, the corresponding backward kernel for the jump times and values will be Gaussian kernels centred at the modified jump time and value with standard deviation equal to 2 and 1 respectively. For the parameters $\theta := (\lambda_\tau, \lambda_\phi, \kappa)$, we assign Gaussian priors with zero mean and covariance $Diag(10, 10^2, 10)$, constrained to $(0, \infty)^3$. We ran both VRPF-PG and BlockVRPF-PG samplers with 10, 50 and 100 particles. Figure 2.8 shows the estimated marginal posterior distributions obtained by both algorithms. One can see that the BlockVRPF-PG can produce comparable estimations of these posterior distributions even when only 10 particles are used. One should note that although different ways of discretising the time horizon may result in different mixing speeds of the PGS, the BlockVRPF-PG sampler will always yield unbiased estimation. This can be viewed as an advantage of the BlockVRPF-PG sampler compared to the algorithm proposed by Finke et al. (2014), for which discretisation does affect the results. This is due to the fact that the algorithm proposed in Finke et al. (2014) in fact yields an approximation of the actual posterior distributions and the accuracy of such approximation depends on the time discretisations. This was illustrated through results in Finke et al. (2014) on the same model. For the BlockVRPF-PG sampler, on the contrary, one can choose any way to discretise the time horizon without worrying about the estimation accuracies. Hence, choosing hyperparameters for the BlockVRPF-PG sampler is easier.

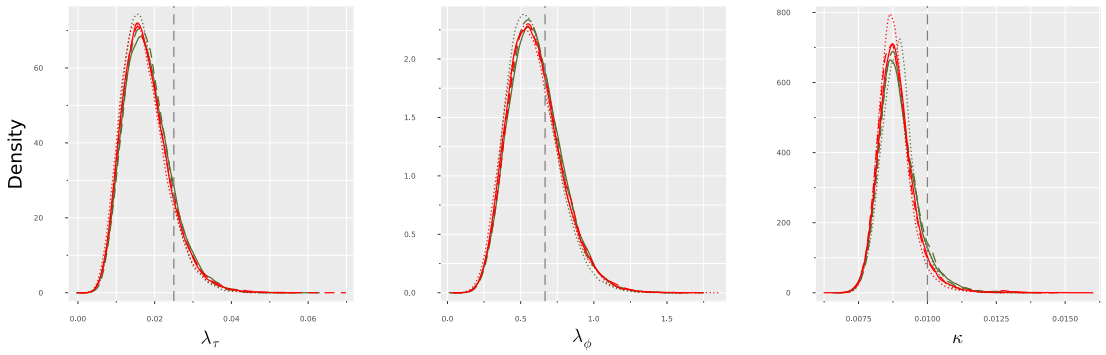

Figure 2.8: Kernel density estimation of the posteriors of the static parameters in short-noise Cox model. Results obtained from VRPF-PG are in red and those obtained from BlockVRPF-PG are in green. Results obtained using 10, 50 and 100 particles are represented by dot, dashed and solid lines, respectively. Grey vertical dashed lines represent the true parameter values.

Figure 2.9 shows the autocorrelations of the two algorithms. One can see that using the same number of particles, VRPF-PG has smaller auto correlations compared to the BlockVRPF-PG sampler. This is possibly due to the sub-optimal choice of backward kernels in the BlockVRPF sampler, as discussed in section 2.5.3. The choice of backward kernel variances may make the incremental weights even larger after rejuvenation. Consequently, the sampled PDMPs may become more correlated, resulting in higher correlations as shown in the figure. This suggests that one may need to choose the backward kernels carefully to yield optimal performance.

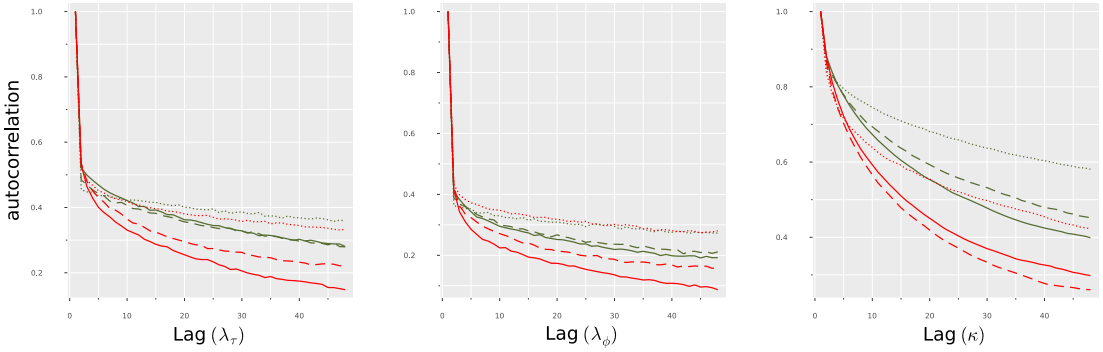

Figure 2.9: Autocorrelations obtained with both algorithms. Results obtained with the VRPF-PG and BlockVRPF-PG samplers are in red and green colour respectively. Dot, dashed and solid lines represent the results obtained when using 10, 50 and 100 particles.

Elementary Change-point Model

In this section, we represent the simulation results for the elementary change-point model. We use the same form of backward kernels as we did in the shot-noise Cox model. However, the standard deviations for the modified jump time and value are set to be 0.1 and 0.05 respectively. Moreover, the parameters $\theta := (\rho, \sigma_\phi, \sigma_y, \alpha, \beta)$ are assigned to have Gaussian priors with zero mean and covariance Diag$(10^2, 10^2, 10, 10^3, 10^2)$, truncated in the region $\mathbb{R} \times (0, \infty)^4$. Figure 2.10 shows the posterior estimations of the parameters obtained using both VRPF-PG and BlockVRPF-PG algorithms. One can see that both algorithms yield comparable estimations, regardless of the number of particles used in the SMC algorithm. We notice that BlockVRPF-PG produces results that are more concentrated at the mode for parameter β and this becomes more obvious when less number of particles are used. This is potentially due to the extra variance introduced to the BlockVRPF method because of the modification move at each SMC step, which results in the sampled PDMP becoming more

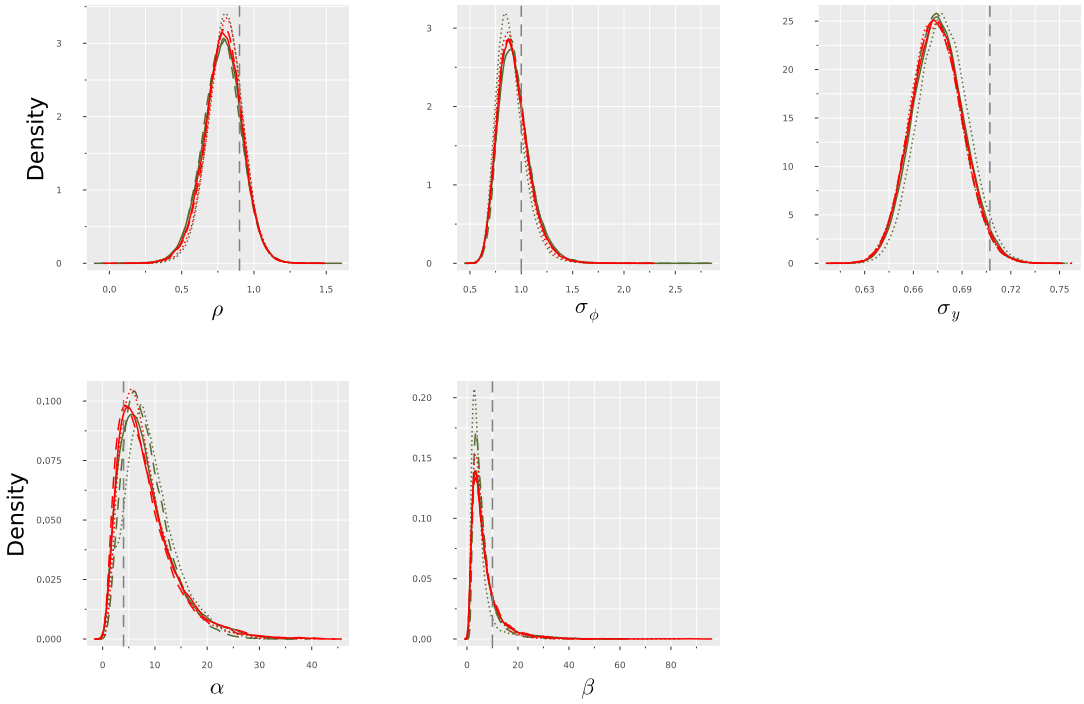

Figure 2.10: Kernel density estimators of the posterior distributions of parameters in the elementary change-point model. The results obtained by using VRPF and BlockVRPF methods are represented in red and green colour respectively. Results obtained by using 25, 50 and 100 particles are represented in dot, dashed and solid lines respectively.

likely to get stuck at local modes if the backward kernels are badly chosen. In fact, the choices of the backward kernels are crucial to the performance of the BlockVRPF-PG sampler. We also run the BlockVRPF-PG with 25 particles and different values for the standard deviations of the backward Gaussian kernels for the modified jump time and value (0.1 and 0.05). Figure 2.11 shows the autocorrelations obtained from the two runs. One can see that only changing the standard deviation used in the backward kernel will result in significantly different performances of the BlockVRPF-PG sampler. Hence, one should try to find or approximate the optimal choices of the backward kernels to achieve the best performance for the BlockVRPF sampler and searching for such optimal kernels is potentially a direction of future work as well.

2.6 Conclusion

In this chapter, we have implemented a particle filter that addresses the limitations faced by the VRPF sampler and still yields unbiased estimations of the PDPs of interest. Numerical

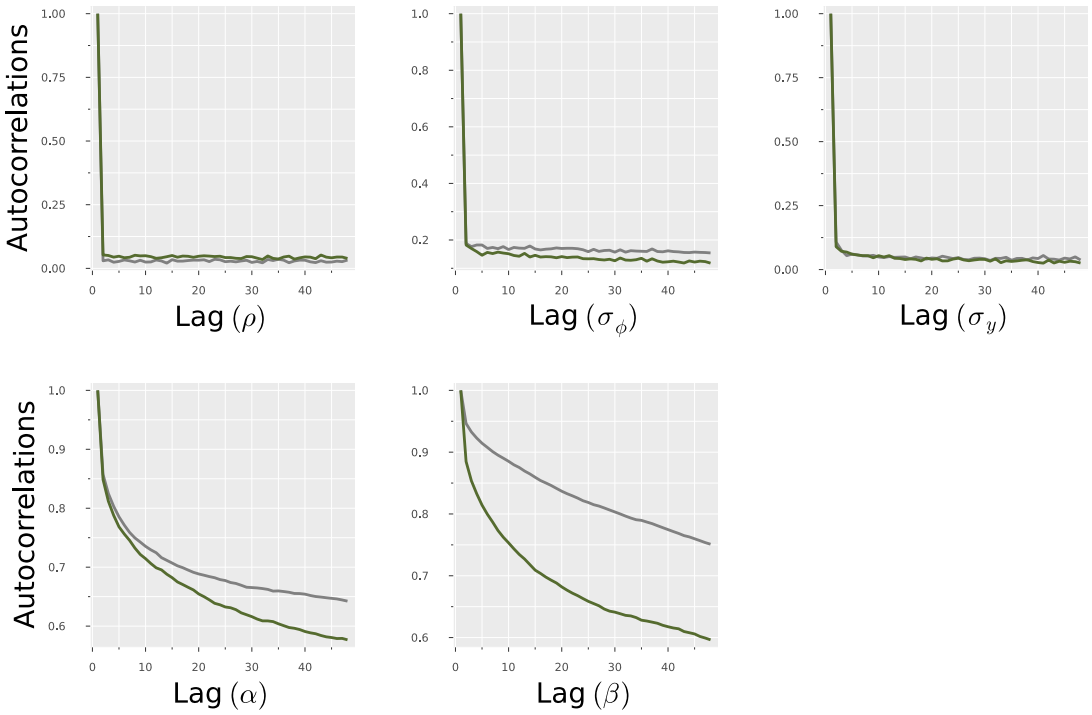

Figure 2.11: Autocorrelations obtained by running BlockVRPF-PG with 25 particles and different choices of backward kernels. Grey lines represent the results obtained using standard deviations 0.1 and 0.05 while green lines represent results obtained using standard deviations 0.1 and 0.2.

studies provide a comprehensive illustration of the improvement brought by this new algorithm as well as the unbiasedness of the estimations obtained by applying particle Gibbs with this particle filter. We also realise the potential implementation issues that affect its performance. Potential directions of future studies could be the following.

1. We follow Whiteley et al. (2011) and Finke et al. (2014) to propose two type of modifications on the jumps. More sophisticated modifications could also be included. For example, it would be reasonable to also include a *deletion* move, i.e. delete an inappropriate jump proposed in the previous block. Moreover, the proposal distributions for the type of modification as well as the position the modification should be placed can be designed in a more sophisticated way.

2. The performance of the particle filter heavily depends on the quality of the proposal distributions used. So far, we used a bootstrap-like way to design the proposals, which is far from ideal for this type of problem. One may consider using flow-based models to define the proposals to improve the performance of the particle filter.

3. In the simulation studies, we showed that the choice of the auxiliary backward kernels does have a significant impact on the performance of the algorithms. Hence, the search for the optimal backward kernel or approximations of these kernels would also be an interesting future direction.

4. In this chapter, we only consider going back one block. It is easy to extend the scheme to multiple blocks. Hence, it's going to be interesting to investigate the effect of the block numbers on the performance. Moreover, it would also be interesting to try to design an algorithm that chooses the block number adaptively within the particle filter.

Chapter 3

Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) refers to a class of methods and algorithms that are designed to perform Bayesian analysis using an approximation to the true posterior distribution of a model, in which the likelihood function implied by the data generating process is computationally intractable. Since its early introduction in the literature of population genetics (Tavaré et al., 1997, Pritchard et al., 1999), it has been extensively and rapidly developed in the past two decades. The underlying mechanism for ABC is simple - generating data from the model of interest and comparing them to the observed data to search for regions of potentially high posterior density in the parameter space. Due to the simplicity of this mechanism, ABC methods have become popular in many research areas and they are now viewed as a standard Bayesian tool. We can now see applications and developments of ABC methods in various scientific areas, e.g. population genetics (Beaumont et al., 2002), coalescence models (Tavaré et al., 1997), system biology (Liepe and Stumpf, 2018), climate (Holden et al., 2018) and ecology (Fasiolo and Wood, 2018), to name a few.

In this chapter, we give an overview of the idea behind ABC methods and introduced some of the main ABC methods in the literature.

3.1 The ABC Posteriors

ABC methods are trying to solve the same Bayesian inferential problem where we are given a Bayesian model whose parameter $\theta \in \Theta$ has a prior density of $\pi(\theta)$ with respect to a probability

measure $d\theta$. Moreover, for data generated from the model, $\mathbf{y} \in \mathbb{R}^{n_y} := \mathcal{Y}$, we define $l(\mathbf{y}|\theta)$ to be the corresponding likelihood function for $\theta \in \Theta$. Suppose that observations $\mathbf{y}^*$ are obtained, we are then interested in the posterior distribution of the parameter θ, $\pi(\theta|\mathbf{y}^*)$, which is given by

$$\pi(\theta|\mathbf{y}^*) = \frac{1}{\mathcal{Z}}\pi(\theta)l(\mathbf{y}^*|\theta) \tag{3.1}$$

where

$$\mathcal{Z} = \int_\Theta \pi(\theta)l(\mathbf{y}^*|\theta)\, d\theta$$

represents the normalising constant of the posterior density. However, if the likelihood function $l(\mathbf{y}|\theta)$ is expensive or impossible to evaluate, it is going to be difficult to use standard Monte Carlo methods to sample from $\pi(\theta|\mathbf{y}^*)$. ABC methods then provide us with an alternative way of tackling this problem that could bypass the evaluation of the likelihood function and it only requires the ability to easily generate pseudo-data from the underlying model. Let $\|\cdot\|$ be a suitable distance metric (e.g. Euclidean distance metric) and $\epsilon > 0$ be a positive real number. In the ABC context, we often call ϵ the tolerance of the ABC posterior. Then, given $\theta \in \Theta$, we define

$$\begin{aligned} L_\epsilon^{ABC}(\mathbf{y}^*|\theta) &= \int_{\mathcal{Y}} l(\mathbf{y}|\theta)\mathbb{I}\left(\|\mathbf{y}-\mathbf{y}^*\| < \epsilon\right)\, d\mathbf{y} \\ &= \int_{\mathcal{B}_\epsilon} l(\mathbf{y}|\theta)\, d\mathbf{y} \end{aligned} \tag{3.2}$$

where $\mathcal{B}_\epsilon := \{\mathbf{y} \in \mathcal{Y} : \|\mathbf{y}-\mathbf{y}^*\| < \epsilon\} \subset \mathcal{Y}$ is the ϵ-ball centred at $\mathbf{y}^*$ in the data space $\mathcal{Y}$. Loosely speaking, (3.2) describes the idea of estimating the true likelihood $l(\mathbf{y}^*|\theta)$ by the pseudo-data generated from the model given θ that are similar to the observations and this similarity is measure by the distance metric $\|\cdot\|$ (Fearnhead and Prangle, 2012). Hence, one can view (3.2) as an approximation of the actual likelihood function. Hence, if we replace the actual likelihood function with $L_\epsilon^{ABC}(\mathbf{y}^*|\theta)$ in (3.1), we obtain an approximation to the actual posterior distribution, which is given by

$$\pi_\epsilon^{ABC}(\theta|\mathbf{y}^*) = \frac{1}{\mathcal{Z}^{ABC}}\pi(\theta)L_\epsilon^{ABC}(\mathbf{y}^*|\theta) \tag{3.3}$$

where

$$\mathcal{Z}^{ABC} = \int_{\Theta} \pi(\theta) L_{\epsilon}^{ABC}(\mathbf{y}^*|\theta)\, d\theta = \int_{\Theta}\int_{\mathcal{B}_\epsilon} \pi(\theta) l(\mathbf{y}|\theta)\, d\mathbf{y}\, d\theta$$

We will refer to $\pi_{\epsilon}^{ABC}(\theta|\mathbf{y}^*)$ as the ABC posterior with tolerance ϵ in the latter part of the thesis. To see the limit of the ABC posterior as $\epsilon \to 0$, we need the Lebesgue differentiation theorem (Stein and Shakarchi, 2009).

Theorem 3.1.1 (Lebesgue Differentiation Theorem)**.** *If f is integrable on $\mathbb{R}^d$, then for almost every $x \in \mathbb{R}^d$,*

$$\lim_{\substack{m(\mathcal{B})\to 0 \\ x\in\mathcal{B}}} \frac{1}{m(\mathcal{B})} \int_{\mathcal{B}} f(y)\, dy = f(x) \tag{3.4}$$

where $m(\mathcal{B})$ denotes the Lebesgue measure of a ball $\mathcal{B}$ in $\mathbb{R}^d$.

We omit the proof here and refer the readers to Stein and Shakarchi (2009) for the detailed proof of the theorem. With the aim of the Lebesgue differentiation theorem, we can then obtain the following theorem.

Theorem 3.1.2. *Let $\theta \in \Theta \subset \mathbb{R}^{n_\theta}$ and $\mathbf{y} \in \mathbb{R}^{n_y}$ be random variables having density $\pi(\theta)l(\mathbf{y}|\theta)$ with respect to a probability measure. Moreover, assume that $p(\mathbf{y}^*) := \int_{\Theta} \pi(\theta) l(\mathbf{y}|\theta)\, d\theta > 0$. Then, we have*

$$\lim_{\epsilon\to 0} \pi_{\epsilon}^{ABC}(\theta|\mathbf{y}^*) = \pi(\theta|\mathbf{y}^*)$$

for almost every $\mathbf{y}^* \in \mathcal{Y}$.

Proof. The main idea of the proof is adopted from Prangle (2017). Clearly, since

$$\mathcal{B}_\epsilon := \{\mathbf{y} \in \mathcal{Y} : \|\mathbf{y} - \mathbf{y}^*\| < \epsilon\}$$

Hence, $\mathbf{y}^* \in \mathcal{B}_\epsilon$ for all $\epsilon > 0$ and $m(\mathcal{B}_\epsilon) \to 0$ as $\epsilon \to 0$. Therefore, for almost every $\mathbf{y}^* \in \mathcal{Y}$,

we have

$$\begin{aligned}\lim_{\epsilon\to 0}\pi_\epsilon^{ABC}(\theta|\mathbf{y}^*) &= \lim_{\epsilon\to 0}\frac{\pi(\theta)L_\epsilon^{ABC}(\mathbf{y}^*|\theta)}{\mathcal{Z}^{ABC}} \\ &= \lim_{\epsilon\to 0}\frac{\pi(\theta)\frac{1}{m(\mathcal{B}_\epsilon)}\int_{\mathcal{B}_\epsilon} l(\mathbf{y}|\theta)\,d\mathbf{y}}{\frac{1}{m(\mathcal{B}_\epsilon)}\int_\Theta\int_{\mathcal{B}_\epsilon}\pi(\theta)l(\mathbf{y}|\theta)\,d\mathbf{y}\,d\theta} \\ &= \lim_{\substack{m(\mathcal{B}_\epsilon)\to 0\\ x\in\mathcal{B}_\epsilon}}\frac{\pi(\theta)\frac{1}{m(\mathcal{B}_\epsilon)}\int_{\mathcal{B}_\epsilon} l(\mathbf{y}|\theta)\,d\mathbf{y}}{\frac{1}{m(\mathcal{B}_\epsilon)}\int_\Theta\int_{\mathcal{B}_\epsilon}\pi(\theta)l(\mathbf{y}|\theta)\,d\mathbf{y}\,d\theta} \\ &= \frac{\pi(\theta)l(\mathbf{y}^*|\theta)}{\int_\Theta\pi(\theta)l(\mathbf{y}^*|\theta)\,d\theta} = \pi(\theta|\mathbf{y}^*)\end{aligned}$$

where the third equality in the above equation directly follows the Lebesgue differentiation theorem. □

Theorem 3.1.2 implies that the ABC posterior distribution will converge to the actual posterior as the tolerance goes to 0. Hence, for small values of ϵ, the corresponding ABC-posterior distribution provides a good approximation of $\pi(\theta|\mathbf{y}^*)$. In reality, the pseudo-data $\mathbf{y}$ and the observations $\mathbf{y}^*$ may be in high dimension and this will make it very inefficient to use (3.3) for inference. Hence, we typically replace $\mathbf{y}$ and $\mathbf{y}^*$ with appropriate lower dimensional summary statistics of them. Define $\mathbf{s} := S(\mathbf{y})$ and $\mathbf{s}^* := S(\mathbf{y}^*)$ where $S : \mathcal{Y} \to \mathcal{S} := \mathbb{R}^{n_s}$ denotes some summary statistics function. We can define an approximation to the actual posterior

$$\pi(\theta|\mathbf{s}^*) := \frac{1}{\mathcal{Z}_s}\pi(\theta)l(\mathbf{s}^*|\theta) \tag{3.5}$$

where $\mathcal{Z}_s$ denotes the normalising constant and $l(\mathbf{s}|\theta)$ denotes the likelihood function of the summary statistics implied by $l(\mathbf{y}|\theta)$. The corresponding ABC posterior density would then be given by

$$\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*) = \frac{1}{\mathcal{Z}_s^{ABC}}\pi(\theta)L_\epsilon^{ABC}(\mathbf{s}^*|\theta) \tag{3.6}$$

with

$$L_\epsilon^{ABC}(\mathbf{s}^*|\theta) = \int_{\mathcal{S}} l(\mathbf{s}|\theta)\mathbb{I}\left(\|\mathbf{s}-\mathbf{s}^*\| < \epsilon\right)\,d\mathbf{s} = \int_{\mathcal{B}_\epsilon^s} l(\mathbf{s}|\theta)\,d\mathbf{s} \tag{3.7}$$

where $\mathcal{B}_\epsilon^s := \{\mathbf{s}\in\mathcal{S} : \|\mathbf{s}-\mathbf{s}^*\| < \epsilon\}$. The limiting distribution of $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$ was discussed in Prangle (2017) and we will obtain a similar limiting result if the following conditions are satisfied.

1. The set $\mathcal{B}_\epsilon^s := \{\mathbf{s} \in \mathcal{S} : \|\mathbf{s} - \mathbf{s}^*\| < \epsilon\}$ is Lebesgue measurable
2. $\lim_{\epsilon \to 0} m(\mathcal{B}_\epsilon^s) = 0$
3. For $\epsilon > 0$, $\mathcal{B}_\epsilon^s$ shrink regularly or have bounded eccentricity at $\mathbf{s}^*$, i.e. there is a constant $c > 0$ such that for every $\epsilon > 0$, there is a ball $\mathcal{B}_\epsilon$ such that

$$\mathbf{s}^* \in \mathcal{B}_\epsilon, \qquad \mathcal{B}_\epsilon^s \subset \mathcal{B}_\epsilon, \qquad \text{and} \quad m(\mathcal{B}_\epsilon^s) \geq m(\mathcal{B}_\epsilon)$$

If the above three conditions are satisfied, we would have that

$$\lim_{\epsilon \to 0} \pi_\epsilon^{ABC}(\theta|\mathbf{s}^*) = \pi(\theta|\mathbf{s}^*)$$

Compared to (3.3), (3.6) has been adopted more frequently in the ABC literature. In fact, one can view (3.3) and a special case of (3.6) when we define $S(\mathbf{y}) = \mathbf{y}$ as the identity function. Moreover, the level of approximation depends on the choice of the summary statistics function. If $\mathbf{s}$ is a sufficient summary statistics for θ, we will have $\pi(\theta|\mathbf{s}^*) = \pi(\theta|\mathbf{y}^*)$ so there is no loss of information when the summary statistics is used. If $\mathbf{s}$ is not a sufficient summary statistics, then there will be some loss of information and $\pi(\theta|\mathbf{s}^*)$ would only be an approximation of the actual posterior. In this case, the use of summary statistics would add another layer of approximation on the ABC posterior.

The ABC posterior distribution can be further generalised by using an appropriate kernel $K_\epsilon(\mathbf{u})$ that satisfies the following condition

1. $\int_{\mathbb{R}^d} K_\epsilon(\mathbf{u})\, d\mathbf{u} = 1$
2. $|K_\epsilon(\mathbf{u})| \leq A\epsilon^{-d}$ for all $\epsilon > 0$
3. $|K_\epsilon(\mathbf{u})| \leq A\epsilon/|\mathbf{u}|^{d+1}$ for all $\epsilon > 0$ and $\mathbf{u} \in \mathbb{R}^d$.

We refer to the class of kernels satisfying the above three conditions as approximations to the identity (Stein and Shakarchi, 2009). Informally, we have that

$$K_\epsilon \to \mathcal{D} \qquad \text{as} \qquad \epsilon \to 0$$

where $\mathcal{D}(\mathbf{u})$ denotes the Dirac delta function centred at $\mathbf{0}$. Then, the following theorem would be useful in the generalisation of the ABC posteriors.

Theorem 3.1.3. *Let $\{K_\epsilon\}_{\epsilon>0}$ be an approximation to the identity and f is integrable on $\mathbb{R}^d$, the for almost every $\mathbf{u} \in \mathbb{R}^d$,*

$$(f * K_\epsilon)(\mathbf{u}) := \int_{\mathbb{R}^d} f(\mathbf{y}) K_\epsilon(\mathbf{u}-\mathbf{y})\, d\mathbf{y} \to f(\mathbf{u}) \quad as \quad \epsilon \to 0 \tag{3.8}$$

If we replace the function f in Theorem 3.1.3 with the likelihood function $l(\mathbf{s}^*|\theta)$, we can define a generalised ABC likelihood as

$$L_\epsilon^{ABC}(\mathbf{s}^*|\theta) = (l * K_\epsilon)(\mathbf{s}^*) := \int_{\mathcal{S}} l(\mathbf{s}|\theta) K_\epsilon(\mathbf{s}^* - \mathbf{s})\, d\mathbf{s} \tag{3.9}$$

According to Theorem 3.1.3, $L_\epsilon^{ABC}(\mathbf{s}^*|\theta) \to l(\mathbf{s}^*|\theta)$ as $\epsilon \to 0$. Note that the ABC likelihood function defined in (3.7) is in fact special cases of (3.9) when the kernel is chosen to be

$$K_\epsilon(\mathbf{s}-\mathbf{s}^*) = \frac{1}{m(\mathcal{B}_\epsilon^s)} \mathbb{I}(\|\mathbf{s}-\mathbf{s}^*\| < \epsilon) \tag{3.10}$$

This is known as the uniform kernel in the literature. Therefore, the generalised ABC posterior will then be given by

$$\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*) = \frac{1}{\mathcal{Z}_s^{ABC}} \pi(\theta) \int_{\mathcal{S}} l(\mathbf{s}|\theta) K_\epsilon(\mathbf{s}^* - \mathbf{s})\, d\mathbf{s} \tag{3.11}$$

This is the general ABC posterior and it is central in all ABC methods and algorithms. There have been different interpretations of the ABC posterior in the literature. Following the derivations in (3.9), one can view the ABC likelihood as a kernel density estimation of the actual likelihood function and then the ABC can be treated as a standard Bayesian inference with an approximated likelihood function. While ABC is usually considered as an approximate Bayesian method, Wilkinson (2013) considered ABC as an exact Bayesian method under the assumption that there is an error in the observations, i.e. $e = \mathbf{s}^* - \mathbf{s}$. In this case, the kernel K_ϵ can then be treated as the density function of this error term.

Despite the different interpretations of ABC, our main purpose is then to sample from (3.11)

and utilise the samples from the ABC posterior to perform Bayesian analysis. However, we may notice that there is still an integral in the ABC posterior defined in (3.11) and this integral is intractable in almost all scenarios. One may also notice that the ABC posterior of (3.11) can be viewed as a marginal density of the joint distribution

$$\pi_\epsilon(\theta, \mathbf{s}|\mathbf{s}^*) \propto \pi(\theta) l(\mathbf{s}|\theta) K_\epsilon(\mathbf{s}^* - \mathbf{s}) \tag{3.12}$$

This lends us to a broad range of Monte Carlo methods that can be used to instead sample from the joint density $\pi_\epsilon(\theta, \mathbf{s}|\mathbf{s}^*)$ which is free of integrals and the ABC posterior can then be obtained as a marginal. This is also where the possibility to sample pseudo-data from the underlying statistical model comes into play.

Alternatively, it is also useful to notice that the ABC likelihood defined in (3.9) can be viewed as the expectation of $K_\epsilon(\mathbf{s}^* - \mathbf{s})$, which is a function of the summary statistics $\mathbf{s}$. Let $\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^n$ be n independent samples of the summary statistics $\mathbf{s}$ from the model $l(\mathbf{s}|\theta)$ given θ, a common unbiased estimator of the expectation would then be given by

$$\widehat{\mathbb{E}}\left[K_\epsilon(\mathbf{s}^* - \mathbf{s})\right] := \frac{1}{n}\sum_{i=1}^{n} K_\epsilon(\mathbf{s}^* - \mathbf{s}^i) \tag{3.13}$$

This is also an unbiased estimator of the ABC likelihood of (3.9). Replacing the ABC likelihood in (3.11) with (3.13), we then obtain an estimator for the ABC posterior that is given by

$$\widehat{\pi}_\epsilon^{ABC}(\theta|\mathbf{s}^*) \propto \pi(\theta) \left\{\frac{1}{n}\sum_{i=1}^{n} K_\epsilon(\mathbf{s}^* - \mathbf{s})\right\} \tag{3.14}$$

and it is in fact an unbiased estimator for the ABC posterior up to proportionality. One may therefore consider directly targeting $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$ by making use of the unbiased estimator of it given in (3.14). Another interpretation of the estimator is through the joint posterior distribution given by

$$\pi_\epsilon^{ABC}(\theta, \mathbf{s}^1, ...\mathbf{s}^n|\mathbf{s}^*) \propto \pi(\theta) \left\{\prod_{i=1}^{n} l(\mathbf{s}^i|\theta)\right\} \left\{\frac{1}{n}\sum_{i=1}^{n} K_\epsilon(\mathbf{s}^* - \mathbf{s}^i)\right\} \tag{3.15}$$

which is the joint distribution of the parameter θ and all n summary statistics $\mathbf{s}^1, ..., \mathbf{s}^n$. One

of the important results is that

$$\begin{aligned}
&\int\int\cdots\int \pi_\epsilon^{ABC}(\theta, \mathbf{s}^1\mathbf{s}^2, ..., \mathbf{s}^n|\mathbf{s}^*)\, d\mathbf{s}^1\, d\mathbf{s}^2 ... d\mathbf{s}^n \\
\propto &\int\int\cdots\int \pi(\theta)\left\{\prod_{i=1}^n l(\mathbf{s}^i|\theta)\right\}\left\{\frac{1}{n}\sum_{i=1}^n K_\epsilon(\mathbf{s}^* - \mathbf{s}^i)\right\} d\mathbf{s}^1\, d\mathbf{s}^2 ... d\mathbf{s}^n \\
= &\frac{1}{n}\sum_{j=1}^n \int\int\cdots\int \pi(\theta)\left\{\prod_{i=1}^n l(\mathbf{s}^i|\theta)\right\} K_\epsilon(\mathbf{s}^* - \mathbf{s}^i)\, d\mathbf{s}^1\, d\mathbf{s}^2 ... d\mathbf{s}^n \\
= &\frac{1}{n}\sum_{j=1}^n \pi(\theta) L_\epsilon^{ABC}(\mathbf{s}^*|\theta) = \pi(\theta) L_\epsilon^{ABC}(\mathbf{s}^*|\theta)
\end{aligned}$$

Hence, this extended joint distribution still incorporates the original ABC posterior as marginal. In practice, most of the standard ABC methods are designed to target either (3.12) or (3.15). As a result, the ABC methods can be viewed as standard Monte Carlo methods targeting a specially designed distribution whose limiting distribution coincides with the actual Bayesian posterior distribution of interest. In the remaining of the chapter, we will review some of the classical ABC algorithms that frequently appear in the ABC literature and their limitations will also be discussed.

3.2 Rejection ABC Samplers

The earliest ABC samplers (Tavaré et al., 1997, Pritchard et al., 1999) are in fact rejection sampling algorithms targeting $\pi_\epsilon^{ABC}(\theta, \mathbf{s}|\mathbf{s}^*)$ defined in (3.12) where

$$\mathcal{Z}_{\theta,s}^{ABC} := \int_\Theta \int_\mathcal{S} \pi(\theta) l(\mathbf{s}|\theta) K_\epsilon(\mathbf{s}^* - \mathbf{s})\, d\mathbf{s}\, d\theta$$

denotes the corresponding unknown normalising constant. Accordingly, the proposal sampling distribution of $(\theta, \mathbf{s})$, $g(\theta, \mathbf{s})$ will be in the form

$$g(\theta, \mathbf{s}) = q(\theta) l(\mathbf{s}|\theta) \tag{3.16}$$

where q is a distribution of θ defined on $\Theta' \supset \Theta$. In practice, one can sample a candidate parameter vector θ from q and then generate a pseudo-dataset from the model followed by the calculation of the corresponding summary statistics to obtain a $(\theta, \mathbf{s})$ tuple that is distributed

according to g. As a result, we have

$$M := \sup_{\theta,\mathbf{s}} \frac{\pi(\theta)l(\mathbf{s}|\theta)K_\epsilon(\mathbf{s}^* - \mathbf{s})}{q(\theta)l(\mathbf{s}|\theta)} = \sup_{\mathbf{s}} K_\epsilon(\mathbf{s}^* - \mathbf{s}) \sup_{\theta} \frac{\pi(\theta)}{q(\theta)} = K_\epsilon^{max} \sup_{\theta} \frac{\pi(\theta)}{q(\theta)}$$

where K_ϵ^{max} is the of the chosen kernel function. Then, the probability of accepting a $(\theta, \mathbf{s})$ tuple from g is given by

$$\alpha(\theta, \mathbf{s}) = \frac{\pi(\theta)l(\mathbf{s}|\theta)K_\epsilon(\mathbf{s}^* - \mathbf{s})}{Mg(\theta, \mathbf{s})} = \frac{\pi(\theta)K_\epsilon(\mathbf{s}^* - \mathbf{s})}{Mq(\theta)}$$

It is clear to see that under the specific construction of the proposal sampling distribution given in (3.16), the calculation of the acceptance probability does not require us to be able to evaluate the intractable likelihood function $l(\mathbf{s}|\theta)$. Algorithm 3.1 outlines the detailed implementation of the rejection ABC sampler.

Algorithm 3.1: Rejection ABC Sampler

Target: $\pi_\epsilon^{ABC}(\theta, \mathbf{s}|\mathbf{s}^*) \propto \pi(\theta)l(\mathbf{s}|\theta)K_\epsilon(\mathbf{s}^* - \mathbf{s})$

Input :

- An integer $N > 0$.
- A proposal sampling distribution, $q(\theta)$.
- The envelope constant $M := K_\epsilon^{max} \sup_\theta \frac{\pi(\theta)}{q(\theta)}$.

for $n = 1, 2, ..., N$ **do**

Step 1. Sample $\theta \sim q(\theta)$;

Step 2. Generate pseudo-dataset $\mathbf{y}$ from the model $l(\mathbf{y}|\theta)$;

Step 3. Calculate the summary statistics $\mathbf{s} := S(\mathbf{y})$;

Step 4. With probability

$$\frac{\pi(\theta)K_\epsilon(\mathbf{s}^* - \mathbf{s})}{Mq(\theta)}$$

Set $\theta_n := \theta$. Otherwise, GOTO *Stpe 1.*

Output: A sample $\theta_1, ..., \theta_N \sim \pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$

In the original development of rejection ABC sampler (Pritchard et al., 1999), the proposal sampling distribution is set to be the prior, i.e. $q(\theta) := \pi(\theta)$ and the uniform kernel is employed, one should have

$$K_\epsilon^{max} := \sup_{\mathbf{s}} \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon) = 1$$

Consequently, the acceptance probability simplifies to

$$\alpha(\theta, \mathbf{s}) = K_\epsilon(\mathbf{s}^* - \mathbf{s}) = \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)$$

Therefore, a proposed tuple $(\theta, \mathbf{s})$ would be either directly accepted or rejected, depending on whether the summary statistics calculated from a generated pseudo-dataset are 'close' enough to the summary statistics of the observations. Correspondingly, *Step 4* of Algorithm 3.1 will change to

Step 4. If $\|\mathbf{s}^* - \mathbf{s}\| < \epsilon$, set $\theta_n := \theta$. Otherwise, GOTO *Step 1.*

One can also apply the same rejection scheme on the target defined in (3.15) using the proposal sampling distribution

$$g(\theta, \mathbf{s}^1, ..., \mathbf{s}^n) = q(\theta) \prod_{i=1}^{n} l(\mathbf{s}^i|\theta) \tag{3.17}$$

Similarly, the corresponding envelope constant is then given by

$$M := \sup_{\theta, \mathbf{s}^1, .., \mathbf{s}^n} \frac{\pi(\theta) \left\{\prod_{i=1}^n l(\mathbf{s}^i|\theta)\right\} \left\{\frac{1}{n}\sum_{i=1}^n K_\epsilon(\mathbf{s}^* - \mathbf{s}^i)\right\}}{q(\theta) \prod_{i=1}^n l(\mathbf{s}^i|\theta)} = K_\epsilon^{max} \sup_\theta \frac{\pi(\theta)}{q(\theta)}$$

which is the same as that for the target distribution of (3.12). Also, the acceptance probability is then given by

$$\alpha(\theta, \mathbf{s}^1, .., \mathbf{s}^n) = \frac{\pi(\theta) \sum_{i=1}^n K_\epsilon(\mathbf{s}^* - \mathbf{s})}{nMq(\theta)}$$

Again, if $q(\theta) := \pi(\theta)$ and $K_\epsilon(\mathbf{s}^* - \mathbf{s}) := \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)$, the acceptance probability can then simplify to

$$\alpha(\theta, \mathbf{s}^1, .., \mathbf{s}^n) = \frac{1}{n} \sum_{i=1}^{n} K_\epsilon(\mathbf{s}^* - \mathbf{s})$$

In this case, one can view $\alpha(\theta, \mathbf{s}^1, ..\mathbf{s}^n)$ as an unbiased estimator of the probability of generating pseudo-datasets with summary statistics lying within an ϵ-ball around $\mathbf{s}^*$ given parameter θ. This is called the marginal ABC sampler and is widely explored in the literature (e.g. Marjoram et al. (2003), Reeves and Pettitt (2005), Sisson et al. (2007), Del Moral et al. (2012)).

Chapter 4

ABC Methods with Latent Variables

In the previous chapter, we reviewed some of the classical ABC methods in the literature and discussed their limitations. While ABC methods benefit from the ability to easily generate data from the model to replace the evaluation of the likelihood function, it also leads to the major bottleneck of classical ABC methods. Relying on the re-generations of pseudo-data to accept or reject a proposal for the parameters, classical ABC methods will have extremely low acceptance probabilities, especially when the observations are of high dimension. This is due to the fact that generating pseudo-data that are close to the observations, even under the perfect parameter choice, will become harder when the dimension of observations increases. This is known as the curse of dimensionality, and it happens since there are increasing numbers of random components to be matched to generate close pseudo-data to the observations as the dimension increases.

However, we notice that for most problems considered in the ABC context, the underlying data generation process can be fully characterised by a sequence of latent random variables, $\mathbf{u} \in \mathcal{U}^d$, which come from a known distribution, e.g. standard Gaussian or Uniform distribution. More specifically, given parameter values θ, we assume that there exist a deterministic function $\phi_\theta : \mathcal{U}^d \to \mathcal{Y}$, such that a set of pseudo-data, $\mathbf{y}$, can be obtained by the deterministic transformation of $\mathbf{u}$ through ϕ_θ, i.e.

$$\mathbf{y} := \phi_\theta(\mathbf{u}) \tag{4.1}$$

In other words, we are considering models in which the randomness of the data generation process can be solely determined by the latent random variables $\mathbf{u}$ through the function ϕ_θ. Define $f(\mathbf{u})$ to be the joint probability density function of the latent random variables $\mathbf{u}$. One can now see that given θ, $\mathbf{s}_\theta(\mathbf{u}) := S(\phi_\theta(\mathbf{u}))$ would be the summary statistics of the pseudo-data generated by the latent random variables $\mathbf{u}$. Hence, the likelihood function of a specific summary statistics $\mathbf{s}$ given θ, $l(\mathbf{s}|\theta)$, is then given by

$$l(\mathbf{s}|\theta) = \int_{\mathcal{U}^d} f(\mathbf{u})\delta_{\mathbf{s}}(\mathbf{s}_\theta(\mathbf{u}))\, d\mathbf{u} \tag{4.2}$$

The ABC likelihood can therefore be defined by the latent random variables, which is of the form

$$\begin{aligned} L_\epsilon^{ABC}(\mathbf{s}^*|\theta) &= \int_{\mathcal{S}} l(\mathbf{s}|\theta) K_\epsilon(\mathbf{s}^* - \mathbf{s})\, d\mathbf{s} \\ &= \int_{\mathcal{S}} \left\{ \int_{\mathcal{U}^d} f(\mathbf{u})\delta_{\mathbf{s}}(\mathbf{s}_\theta(\mathbf{u}))\, d\mathbf{u} \right\} K_\epsilon(\mathbf{s}^* - \mathbf{s})\, d\mathbf{s} \\ &= \int_{\mathcal{U}^d} f(\mathbf{u}) \int_{\mathcal{S}} K_\epsilon(\mathbf{s}^* - \mathbf{s})\delta_{\mathbf{s}}(\mathbf{s}_\theta(\mathbf{u}))\, d\mathbf{s}\, d\mathbf{u} \\ &= \int_{\mathcal{U}^d} f(\mathbf{u}) K_\epsilon(\mathbf{s}^* - s_\theta(\mathbf{u}))\, d\mathbf{u} \end{aligned} \tag{4.3}$$

Then, the ABC posterior can be correspondingly written as

$$\begin{aligned} \pi_\epsilon^{ABC}(\theta|\mathbf{s}^*) &\propto \pi(\theta) L_\epsilon^{ABC}(\mathbf{s}^*|\theta) \\ &= \int_{\mathcal{U}^d} \pi(\theta) f(\mathbf{u}) K_\epsilon(\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}))\, d\mathbf{u} \end{aligned} \tag{4.4}$$

Hence, instead of targeting the joint density of $(\theta, \mathbf{s})$, it is also possible to target the joint density of the parameters and the latent random variables, $(\theta, \mathbf{u})$, and this still admits the ABC posterior as marginal. Enlightened by (4.4), we aim to design an algorithm that tries to solve the same Bayesian problem as those standard ABC methods but targets a different joint distribution. This is the main idea developed in this chapter.

There have been several methods proposed in the literature that considered using the latent random variables in the ABC context. Neal (2012) proposed two types of coupled-ABC algorithms to solve the household epidemic problem. The couple-ABC algorithms search one

or more parameter vectors given a random seeds vector sampled from the prior that leads to close matches to the observations. These form an approximation of the posterior by certain post-processing steps. A similar idea was also explored in Meeds and Welling (2015), where they used optimisation techniques to search for parameters that minimise the distance between pseudo-observations and the data. Similar optimisation ideas were also explored in Forneron and Ng (2016). Graham and Storkey (2017) proposed an alternative method that targets density in the joint space $(\theta, \mathbf{u})$ conditional on the exact match to the data using Hamiltonian Monte Carlo. This lies in the same class of sampling methods with a target defined on a manifold. Similar to what we proposed in this chapter, Prangle et al. (2018) proposed a rare event approach that uses the SMC sampler to explore that latent variable space and construct an estimate of the likelihood function defined in (4.3) with the uniform kernel. This is then used within a Pseudo-marginal Metropolis-Hastings algorithm to sample from the ABC posterior density.

The rest of the chapter is organised as follows. In Section 4.1, we give an introduction of the rare-event approach developed in Prangle et al. (2018). We then describe the algorithm we developed in Section 4.2. The algorithm we developed is based on the ABC-SMC sampler of Del Moral et al. (2012) but targets a different joint distribution from the standard ABC-SMC sampler does. Informally, we can think of the algorithm we developed as a relaxation of the method in Graham and Storkey (2017). When the uniform kernel is used, our algorithm can be treated as sampling within a "tube" around the manifold defined by $S(\phi_\theta(\mathbf{u})) = \mathbf{s}^*$ instead of sampling on the manifold, as what Graham and Storkey (2017) did. The performance of the algorithm developed in this chapter will be illustrated through three numerical examples that are widely used in the ABC literature in Section 4.4. We also compare its performance with the standard ABC-SMC sampler and the rare-event approach of Prangle et al. (2018).

4.1 The Rare Event Approach

As discussed in the previous chapter, most of the standard ABC methods rely on replacing the evaluation of the ABC likelihood, which is generally intractable, with the evaluation of a kernel function $K_\epsilon(\mathbf{s}^* - \mathbf{s})$. One can view the kernel as an unbiased estimator of the actual ABC likelihood. Hence, most of the standard ABC methods can be viewed as pseudo-marginal

methods targeting the actual ABC posterior. However, when the dimension of summary statistics becomes large, it is hard to simulate pseudo-data with close-match summary statistics. As a result, when the tolerance is chosen to be small and the uniform kernel is used, we may get many rejections since the ABC likelihood, in this case, is estimated to be 0 for the majority of the time, making these standard ABC methods highly inefficient when the high-dimensional scenario is considered.

The rare-event approach of Prangle et al. (2018) tried to solve this problem by forming more accurate estimations of the ABC likelihood. If the kernel is chosen to be uniform, i.e.

$$K_\epsilon(\mathbf{s}^* - \mathbf{s}) = \frac{1}{m(\mathcal{B}_\epsilon^{\mathbf{s}^*})}\mathbb{I}\left(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon\right) \propto \mathbb{I}\left(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon\right)$$

the ABC likelihood function the becomes

$$L_\epsilon^{ABC}(\mathbf{s}^*|\theta) \propto \int_{\mathcal{S}} l(\mathbf{s}|\theta)\mathbb{I}\left(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon\right)\, d\mathbf{s} = \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon|\theta)$$

Hence, the ABC likelihood can be viewed as, up to proportionality, the probability of generating pseudo-data with summary statistics that is within a distance of ϵ from $\mathbf{s}^*$, given the parameters θ. Standard ABC methods estimate this probability by generating the datasets from the prior and calculating a crude Monte Carlo estimator. When this probability is small, as is the case when ϵ is small or $dim(\mathbf{s})$ is high, the relative error of the crude Monte Carlo estimator has a high variance. Hence, Prangle et al. (2018) instead considered using the rare-event technique to form a new estimator of the probability whose relative error has a smaller variance compared to the crude Monte Carlo estimator. The main idea of the rare event technique is as follows. Consider a sequence of events $A_1 \supset A_2 \supset ... \supset A_P$ and we are interested in the probability of event A_P, which is very small. Instead of directly calculating the probability of A_P, we instead write

$$\Pr(A_P) = \Pr\left(\bigcap_{n=1}^{P} A_n\right) = \Pr(A_1)\Pr(A_2|A_1)...\Pr(A_P|A_{P-1})$$

One can then form Monte Carlo estimators of each of the probabilities in the product and multiply them together to get an estimate of $\Pr(A_P)$. If these probabilities are relatively

large, the relative error of the estimation formed in this way would have a smaller variance compared to that formed by directly estimating $\Pr(A_P)$. Going back to the ABC problem, suppose there is a sequence of decreasing tolerances $\epsilon_0 > \epsilon_1 > \epsilon_2 > ... > \epsilon_P := \epsilon$, where the last tolerance is the one used in the ABC posterior. We can write the ABC likelihood, which is proportional to $\Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)$, by

$$\begin{aligned} L_\epsilon^{ABC}(\mathbf{s}^*|\theta) &\propto \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon) \\ &= \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_1|\theta) \prod_{n=2}^{P} \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_n|\theta, \|\mathbf{s}^* - \mathbf{s}\| < \epsilon_{n-1}) \\ &= \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_1|\theta) \prod_{n=2}^{P} \frac{\Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_n|\theta)}{\Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_{n-1}|\theta)} \\ &= \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_1|\theta) \prod_{n=2}^{P} \frac{L_{\epsilon_n}^{ABC}(\mathbf{s}^*|\theta)}{L_{\epsilon_{n-1}}^{ABC}(\mathbf{s}^*|\theta)} \end{aligned}$$

Hence, if ϵ_1 is chosen to be large and the tolerance decreases at a slow rate, each part in the above product should have a relatively large value and we can then consider estimating them separately and multiplying them together to create an estimator of the ABC likelihood up to proportionality.

From (4.3), one could see that the ABC likelihood, $L_{\epsilon_n}^{ABC}(\mathbf{s}^*|\theta)$, can also be interpreted as the normalising constant of the distribution $\pi_{\epsilon_n}(\mathbf{u}|\theta) \propto f(\mathbf{u})\mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon_n)$. Denote $\mathcal{Z}_n$ to be the normalising constant of $\pi_{\epsilon_n}(\mathbf{u}|\theta)$, one can then easily find out that the ABC likelihood can therefore be written as

$$L_\epsilon^{ABC}(\mathbf{s}^*|\theta) = \mathcal{Z}_1 \prod_{n=2}^{P} \frac{\mathcal{Z}_n}{\mathcal{Z}_{n-1}}$$

Hence, one could see that this can be estimated within an SMC sampler targeting the sequence of distributions $\{\pi_{\epsilon_n}(\mathbf{u}|\theta)\}_{n=0,1,2,...,P}$. In Prangle et al. (2018), they used the two algorithms described in Cérou et al. (2012) for estimating this rare event probability with an SMC approach. One of the algorithms requires an input of the sequences of tolerances described above whereas these tolerances can be adaptively determined in the other algorithm. We followed Prangle et al. (2018) to name the FIXED-RE-SMC and ADAPT-RE-SMC algorithms in this section. Both algorithms use the same mechanism as the SMC sampler of Del Moral et al. (2006). Given the sequence of decreasing tolerances, $\{\epsilon_n\}_{n=0,1,2,..,P}$, an SMC sampler

is designed to target a sequence of joint distributions which are defined in increasing spaces and have the form

$$\tilde{\pi}_{\epsilon_n}(\mathbf{u}_{1:n}|\theta) = \pi_{\epsilon_n}(\mathbf{u}_n|\theta) \prod_{j=0}^{n-1} \mathcal{L}_j(\mathbf{u}_j|\mathbf{u}_{j+1})$$

Since the uniform kernel is used, the corresponding incremental weight is then given in the form

$$\mathcal{G}(\mathbf{u}_n) = \mathbb{I}\left(\|\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}_n)\| < \epsilon_n\right)$$

Hence, the RE-SMC algorithm generally proceeds as follows. N sets of latent random variables (called particles) are sampled from their prior distribution at the beginning. At the n-th SMC iteration, a new tolerance ϵ_n is set such that N_{acc} of the particles are still "alive". A set of N particles are then sampled uniformly from these alive particles and applied through a Markov kernel that targets the density $\pi_{\epsilon_n}(\mathbf{u}_n|\theta)$. An estimation of the ratio $\mathcal{Z}_n/\mathcal{Z}_{n-1}$ can then be obtained as a byproduct of the SMC sampler and is given by

$$\hat{R}_n := \widehat{\frac{\mathcal{Z}_n}{\mathcal{Z}_{n-1}}} = \frac{N_{acc}}{N}$$

Algorithm 4.1 outlines the details of the ADAPT-RE-SMC algorithm described in Prangle et al. (2018). We did not include the details of the FIXED-RE-SMC algorithm in this section as it's only a slight modification of the ADAPT-RE-SMC algorithm. In fact, one could just replace line 6 of Algorithm 4.1 with predefined tolerance values. With a fixed sequence of tolerance values, it is possible that in a certain SMC step we obtain $|I_n| = 0$, resulting in $\hat{R}_n = 0$. In this case, one could terminate the SMC sampler early and return $\hat{R} = 0$. It was proved in Cérou et al. (2012) that the FIXED-RE-SMC algorithm would produce an unbiased estimator of the rare-event probability whereas the ADAPT-RE-SMC algorithm would instead give an estimator with $\mathcal{O}(N^{-1})$ bias. The asymptotic variance for the relative error of the FIXED-RE-SMC algorithm is also smaller than that of the ADAPT-RE-SMC algorithm.

For the Markov kernel used within the RE-SMC algorithms, Prangle et al. (2018) chose to use the Pseudo-marginal slice sampling of Murray and Graham (2016) and focused on the case where the latent random variables have a joint uniform distribution over $[0, 1]^d$. One of the main advantages of using slice sampling is that particles obtained from it are unique, resulting

Algorithm 4.1: ADAPT-RE-SMC algorithm

Input :

- The parameter of the model, θ
- The number of particles, N
- The terminal tolerance, ϵ
- Number of active particles kept at each SMC step, N_{acc}

for $i = 1, .., N$ **do**
  Sample $\mathbf{u}_0^i \sim f(\mathbf{u}_0)$;
  Calculate $D_0^i := \|\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}_0^i)\|$;
  Set $\epsilon_0 := \infty$;
for $n = 1, 2, ...$ **do**
  Set $\epsilon_n := \max\left\{\epsilon, D_{n-1}^{(N_{acc})}\right\}$, where $D_{n-1}^{(N_{acc})}$ is the N_{acc}-th value of $D_{n-1}^{1:N}$ when ordered increasingly. ;
  Obtain the set $I_n = \left\{i : D_{n-1}^i \leq \epsilon_n\right\}$ and $\hat{R}_n := |I_n|/N$;
  for $i = 1, ..., N$ **do**
    Sample index A_{n-1}^i uniformly from I_n, and apply a Markov kernel to $\mathbf{u}_{n-1}^{A_{n-1}^i}$ with invariant density $\pi_{\epsilon_n}(\mathbf{u}_n|\theta)$ to obtain $\mathbf{u}_n^i$;
    Set $D_n^i := \|\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}_n^i)\|$;
  if $\epsilon_n = \epsilon$ **then**
    Break the loop and set $P := n$
return $\hat{R} = \prod_{n=1}^P \hat{R}_n$

in a more stable Monte Carlo estimate of the probability compared to the Metropolis-Hastings kernel, where particles obtained may contain duplicates. Algorithm 4.2 lists the details of the slice sample used in Prangle et al. (2018). Note that if the latent random variables do not have a joint uniform distribution over $[0, 1]^d$ the slice sampling will proceed in a bit different but similar way. We will not go into details of the algorithm and refer the readers to Murray and Graham (2016) for more details of the algorithm. By default, the initial search width is set to 1. However, when ϵ is small, the number of loops required for the slice sampling will increase accordingly. To ensure that the numbers of loops are fairly constant within the RE-SMC algorithm, one could adaptively adjust the initial search width at the beginning of each SMC step.

The estimation of the likelihood obtained from RE-SMC algorithm can then be used within the Metropolis-Hastings algorithm targeting the ABC-posterior $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*) \propto \pi(\theta)L_\epsilon^{ABC}(\mathbf{s}^*|\theta)$. Hence, we can replace the likelihood function with the estimation obtained in the RE-SMC algorithm. As the estimator is unbiased, the algorithm is still valid, and it is known as the

Algorithm 4.2: Slice sampling kernel within RE-SMC

Input:

- The current state of the latent random variables, $\mathbf{u} \in \mathcal{U}^d$
- Initial search width, w
- Summary statistics of the observations, $\mathbf{s}^*$ and the transformation function, $\mathbf{s}_\theta$
- The tolerance value, ϵ

Define $m \equiv y \mod 2$ and the function

$$r(x) := \begin{cases} m, & m < 1 \\ 2 - m, & m \geq 1 \end{cases}$$

Sample $\mathbf{v} \sim \mathcal{N}(0, I_d)$;
Sample $u \sim \text{Uniform}(0, w)$. Let $a = -u, b = w - u.$;
Sample $z \sim \text{Uniform}(a, b)$ and set $\mathbf{u}' := r(\mathbf{u} + z\mathbf{v})$;
while $\|\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}')\| \geq \epsilon$ **do**
    if $z < 0$ **then**
        Set $a = z$;
    else
        Set $b = z$;
    Sample $z \sim \text{Uniform}(a, b)$;
    Set $\mathbf{u}' := r(\mathbf{u} + z\mathbf{v})$;

pseudo-marginal Markov chain Monte Carlo method (Andrieu and Roberts, 2009). Given the current state θ_{n-1}, a proposed state θ' is generated from the proposal density $q(\theta'|\theta_{n-1})$. Then, with probability

$$\alpha(\theta, \theta') := \min\left\{1, \frac{\hat{L}_\epsilon^{ABC}(\mathbf{s}^*|\theta')\pi(\theta')q(\theta|\theta')}{\hat{L}_\epsilon^{ABC}(\mathbf{s}^*|\theta)\pi(\theta)q(\theta'|\theta)}\right\} \tag{4.5}$$

The resulting pseudo-marginal Metropolis-Hastings algorithm was described in Prangle et al. (2018) and listed in Algorithm 4.3. In this paper, we refer to this algorithm as the PMMH-RESMC algorithm.

The computational cost of RE-SMC can be further saved by having an early rejection step within the PMMH-RESMC algorithm. As described in Algorithm 4.3, a proposal is rejected when

$$u > \frac{\pi(\theta')\hat{L}'q(\theta_{n-1}|\theta')}{\pi(\theta_{n-1})\hat{L}_{n-1}q(\theta'|\theta_{n-1})} \tag{4.6}$$

Algorithm 4.3: PMMH-RESMC algorithm

Input :

- The initial state, θ_1
- The number of iterations, M
- The terminal tolerance, ϵ

for $n = 1$ **do**

  Calculate and set $\hat{L}_1 := \hat{L}_\epsilon^{ABC}(\mathbf{s}^*|\theta_1)$ using ADAPT-RE-SMC algorithm described in Algorithm 4.2. ;

for $n = 2, ..., M$ **do**

  Propose $\theta' \sim q(\theta'|\theta_{n-1})$;

  Calculate $\hat{L}' := \hat{L}_\epsilon^{ABC}(\mathbf{s}^*|\theta')$ using ADAPT-RE-SMC algorithm. ;

  Generate $u \sim \text{Uniform}(0, 1)$;

  if $u > \frac{\pi(\theta')\hat{L}'q(\theta_{n-1}|\theta')}{\pi(\theta_{n-1})\hat{L}_{n-1}q(\theta'|\theta_{n-1})}$ **then**

    Reject the proposal and set $\theta_n = \theta_{n-1}$ and $\hat{L}_n := \hat{L}_{n-1}$;

  else

    Set $\theta_n = \theta'$ and $\hat{L}_n := \hat{L}'$

Output: $\theta_1, ..., \theta_M$

Rearranging the above formula, we obtained that a proposal will be rejected when

$$\hat{L}' < \frac{u\pi(\theta_{n-1})\hat{L}_{n-1}q(\theta'|\theta_{n-1})}{\pi(\theta')q(\theta_{n-1}|\theta')} \tag{4.7}$$

Therefore, we don't need to wait for the RE-SMC algorithm to finish and obtain an estimation $\hat{L}'$. In fact, one can terminate the RE-ABC algorithm at the t-th step as long as $\prod_{j=1}^{t} \hat{R}_j$ becomes smaller than the fraction in (4.7). This will ensure that the final estimation $\hat{L}'$ would definitely lead to a rejection since all the estimation $\hat{R}_n$'s are less than 1. Hence, the final estimation $\hat{L}'$ will always be less than a partial product of it. Prangle et al. (2018) argued that adding the early rejection step will also solve the problem that the ADAPT-RE-SMC algorithm might be stuck and never terminate.

4.2 The Latent ABC-SMC (L-ABC-SMC) Sampler

In this section, we described the algorithm we developed in this chapter. We note that (4.4) indicates that one can view the ABC posterior as the marginal distribution of the joint distribution of the parameters θ and the latent random variables $\mathbf{u}$. One can instead design an algorithm targeting $\pi_\epsilon(\theta, \mathbf{u}|\mathbf{s}^*) \propto \pi(\theta)f(\mathbf{u})K_\epsilon(\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u}))$. For the ease of notation, we

define $\zeta := (\theta, \mathbf{u}) \in \Theta \times \mathcal{U}^d$ and write $\mathbf{s}_\theta(\mathbf{u}) := \Psi(\zeta)$. Then the joint density we are targeting can be written as

$$\pi_\epsilon^{ABC}(\zeta|\mathbf{s}^*) \propto \rho(\zeta) K_\epsilon(\mathbf{s}^* - \Psi(\zeta)) \tag{4.8}$$

where $\rho(\zeta) := \pi(\theta) f(\mathbf{u})$ denotes the joint prior distribution of θ and $\mathbf{u}$. The proposed method follows the ABC-SMC algorithm of Del Moral et al. (2012) but targets the distribution $\pi_\epsilon(\zeta|\mathbf{s}^*)$ instead. In fact, the algorithm only modified the Metropolis-Hastings kernel used within the ABC-SMC sampler. Instead of proposing $\theta' \sim q(\theta'|\theta)$ and generating pseudo-data sets $\mathbf{y}$ based on θ', our method treats the parameters and the latent random variables as a whole random vector whose posterior distribution given $\mathbf{s}^*$ is of our interest. Given the current state ζ, a proposal ζ' is proposed from the proposal distribution $q(\zeta'|\zeta)$. Hence, the data generation process has been done implicitly through the generation of ζ' from the proposal distribution followed by transforming it through the function Ψ. The proposal ζ' can then be accepted with probability

$$\min\left\{1, \frac{\pi_\epsilon^{ABC}(\zeta'|\mathbf{s}^*) q(\zeta|\zeta')}{\pi_\epsilon^{ABC}(\zeta|\mathbf{s}^*) q(\zeta'|\zeta)}\right\}$$

When the uniform kernel is applied, the acceptance probability can be simplified to

$$\min\left\{1, \frac{\rho(\zeta')\mathbb{I}\left(\|\mathbf{s}^* - \Psi(\zeta')\| < \epsilon\right) q(\zeta|\zeta')}{\rho(\zeta)\mathbb{I}\left(\|\mathbf{s}^* - \Psi(\zeta)\| < \epsilon\right) q(\zeta'|\zeta)}\right\} = \min\left\{1, \frac{\rho(\zeta') q(\zeta|\zeta')}{\rho(\zeta) q(\zeta'|\zeta)} \mathbb{I}\left(\|\mathbf{s}^* - \Psi(\zeta')\| < \epsilon\right)\right\}$$

The above acceptance probability can be interpreted as follow. Given the current state ζ, we sample a new realisation ζ' that is close to the current state through a proposal distribution $q(\zeta'|\zeta)$. If the summary statistics of the pseudo-data transformed from ζ' is "close enough" to the observed summary statistics $\mathbf{s}^*$ (measured by the distance metric $\|\cdot\|$), the proposal ζ' is then kept in the sample to approximate the posterior distribution. This is exactly the potential gain of our algorithm compared to the standard ABC methods. When we obtain ζ with $\Psi(\zeta)$ close enough to $\mathbf{s}^*$, we spend some time exploring a small neighbourhood around ζ. If the transformation Ψ is smooth enough, we may expect that $\Psi(\zeta')$ would not change too much if ζ' is near to ζ. This would increase the probability of finding a new combination of θ and $\mathbf{u}$ with $\|\mathbf{s}^* - \mathbf{s}_\theta(\mathbf{u})\| < \epsilon$ significantly, thus improving the efficiency of the algorithm

and allowing the new algorithm to reach even smaller tolerances with the same computational budget.

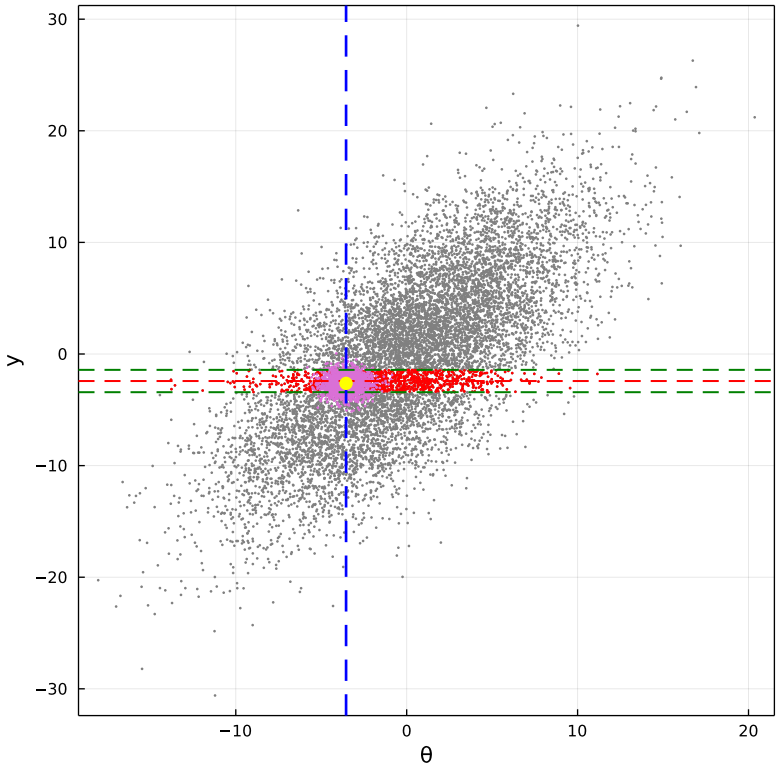

Figure 4.1: A 2-D example illustrating the advantages of the proposed algorithm. In the example the parameter θ has a prior distribution of $\mathcal{N}(0, 5^2)$ and given θ, the observations is assumed to generated from $\mathcal{N}(\theta, 10^2)$. Red horizontal lines represent the observations we want to match. Grey points in the graph are generated from the joint prior of θ and y.

Figure 4.1 illustrates the idea we discussed in the previous paragraph using a simple 2-D Gaussian example. Given an observation, $\mathbf{y}^*$, which is represented by the red horizontal line on the graph, standard ABC algorithms either try to look for close matches to $\mathbf{y}^*$ from the joint prior distribution (i.e. try to obtain one of the red points on the graph) or try to find close matches to $\mathbf{y}^*$ given specific parameter value θ (i.e. try to find a red point on the blue line). On the contrary, our proposed method tried to explore the region around an accepted point (the yellow point). As illustrated by the graph, one of the pink points would be obtained by using the proposed methods. We can see that there is a much higher chance to obtain a new point that's still within a distance of ϵ from $\mathbf{y}^*$.

Algorithm 4.4 outlines the modified ABC-MCMC sampler applied within our proposed algorithm. One can see that the data generation step in the original ABC-MCMC sampler has

Algorithm 4.4: Modified ABC-MCMC Samplers:$\mathbf{MABCMCMC}(\zeta_0, \mathbf{M}, \mathbf{s}^*, \epsilon, \lambda, \mathbf{\Sigma})$

Input :

- A distance metric, d.
- The tolerance for the ABC posterior, ϵ
- Number of MCMC loops, N
- The observed data $\mathbf{y}^*$
- An initial state, ζ_0 satisfying $\|\mathbf{s}^* - \Psi(\zeta_0)\| < \epsilon$
- The scale factor for the proposal, λ
- The variance for the proposal distribution, Σ

Initialise the chain at ξ_0 ;
Initialise an indicator, $I = 0$;
for $n = 1, 2, ..., N$ **do**
    Sample $\zeta' \sim \mathcal{N}(\zeta_{n-1}, \lambda^2\Sigma)$;
    Sample $u \sim \mathcal{U}(0, 1)$;
    if $u > \min\{1, [\rho(\zeta')q(\zeta_{n-1}|\zeta')] \,/\, [\rho(\zeta_{n-1})q(\zeta'|\zeta_{n-1})]\}$ **then**
        Set $\zeta_n := \zeta_{n-1}$;
    else
        Set $\mathbf{y} := \Psi(\zeta')$;
        if $\|\mathbf{s}^* - \Psi(\zeta')\| < \epsilon$ **then**
            Set $\zeta_n := \zeta'$;
            Set $I = I + 1$;
        else
            Set $\zeta_n := \zeta_{n-1}$;

Output: The last state ζ_N; the indicator I

been replaced by the evaluation of the transformation function Ψ. Following Del Moral et al. (2012), we can similarly define a sequence of extended densities $\{\tilde{\pi}_n(\zeta_{1:n}|\mathbf{s}^*)\}_{n=0,1,2,...}$ that are defined by

$$\tilde{\pi}_n(\zeta_{0:n}|\mathbf{s}^*) = \frac{1}{\tilde{\mathcal{Z}}_n}\rho(\zeta_n)K_{\epsilon_n}(\mathbf{s}^* - \Psi(\zeta_n)) \prod_{j=0}^{n-1} \mathcal{L}_j(\zeta_j|\zeta_{j+1}) \tag{4.9}$$

where $\tilde{\mathcal{Z}}_n := \int \rho(\zeta_n)K_{\epsilon_n}(\mathbf{s}^* - \Psi(\zeta_n))\, d\zeta_n$ denotes the normalising constant of $\tilde{\pi}_n(\zeta_{0:n}|\mathbf{s}^*)$. If the forward kernel is defined through the Modified ABC-MCMC sampler described in Algorithm 4.4 and the backward kernels are of the same form as Del Moral et al. (2012), the incremental weight for the proposed algorithm is still given by

$$\mathcal{G}_n(\zeta_{0:n}) = \frac{\mathbb{I}(\|\mathbf{s}^* - \Psi(\zeta_{n-1})\| < \epsilon_n)}{\mathbb{I}(\|\mathbf{s}^* - \Psi(\zeta_{n-1})\| < \epsilon_{n-1})} = \mathbb{I}(\|\mathbf{s}^* - \Psi(\zeta_{n-1})\| < \epsilon_n) \tag{4.10}$$

We will denote the algorithm we developed in this chapter by the L-ABC-SMC sampler in the

later parts of this chapter.

Implementation Details

In this part, we discussed the tuning strategies to ensure a successful and stable performance of the L-ABC-SMC algorithm. Since the L-ABC-SMC algorithm is also based on the SMC sampler, there are several tuning choices we can adopt and these typically can be automated. The first thing we considered is the choice of proposal distributions for the variable ζ. In this chapter, we focused on using a random-walk Metropolis-Hastings algorithm to move the particles. Hence, at time n of the L-ABC-SMC algorithm, a move from ζ_{n-1} to ζ_n is proposed by sampling ζ_n from a multivariate gaussian distribution centred at ζ_{n-1}. To make the algorithm more efficient, it would be ideal if the variance of the proposal density is the same or similar to that of the target density, $\pi_{\epsilon_n}^{ABC}(\zeta_n|\mathbf{s}^*)$. In the setting of L-ABC-SMC algorithm, we would expect that ϵ_n and ϵ_{n-1} are close, implying that the associated densities $\pi_{\epsilon_{n-1}}^{ABC}(\zeta_{n-1}|\mathbf{s}^*)$ and $\pi_{\epsilon_n}^{ABC}(\zeta_n|\mathbf{s}^*)$ would be similar to each other as well. Hence, it would be reasonable to use the approximation of the variance of $\pi_{\epsilon_{n-1}}^{ABC}(\zeta_{n-1}|\mathbf{s}^*)$ as the variance of the proposal density for ζ_n. An estimation of this variance would be easily obtained through the particles and their weights obtained at the $n-1$-th SMC step. As discussed previously, if a uniform kernel is used, the weights of the particles would be either 1 or 0 up to proportionality. Hence, one would only need to calculate the empirical variance of all the particles with non-zero weights to at $n-1$-th step to obtain an estimation of the variance of $\pi_{\epsilon_{n-1}}^{ABC}(\zeta_{n-1}|\mathbf{s}^*)$

In the L-ABC-SMC algorithm, particles are moved according to an MCMC kernel. Hence, it is possible that some particles in the previous iteration would not be moved due to rejections of proposals. Since a resampling step is always performed at the beginning of each SMC iteration, $\pi_{\epsilon_n}^{ABC}(\zeta_n|\mathbf{s}^*)$ would end up being represented by many duplicates of a few distinct particles, resulting in high Monte Carlo variances. Such a problem would become more severe when the tolerance becomes small and moving a particle becomes more difficult or when the SMC step n becomes larger. One way to alleviate this problem is to apply the MCMC kernel with multiple iterations. Instead of applying the MCMC algorithm with only 1 iteration at time n, we could apply the algorithm for M_n iterations. The value M_n can be chosen to ensure that a proportion of at least μ of the particles is expected to be moved under the

M_n-step MCMC kernel. Given the average acceptance, probability is α_n, the value of M_n would then be chosen such that

$$(1-\alpha_n)^{M_n} < 1-\mu \Rightarrow M_n > \frac{\log(1-\mu)}{\log(1-\alpha_n)} \tag{4.11}$$

The value of α_n is typically not known analytically, but we can find an approximation of it from the output in the previous iteration. More specifically, one can record the total number of acceptance in the previous iteration, $\mathcal{A}_{n-1}$, and an estimation of α_{n-1} can then be obtained by $\hat{\alpha}_{n-1} := \mathcal{A}_{n-1}/(NM_{n-1})$. Again, as the $\pi_n^{ABC}(\zeta_n|\mathbf{s}^*)$ and $\pi_{n-1}^{ABC}(\zeta_{n-1}|\mathbf{s}^*)$ are close, $\hat{\alpha}_{n-1}$ can also be used as an estimate of α_n. Hence, we will choose M_n such that $M_n > \log(1-\mu)/\log(1-\alpha_{n-1})$ in practice.

One last implementation strategy we considered is the scale of proposal distributions. Having obtained the estimation of the variance of $\hat{\Sigma}_n$, a proposal of ζ_n will then be obtained from the distribution $\mathcal{N}(\zeta_{n-1}, \lambda_n^2\hat{\Sigma}_n)$ where λ_n can be viewed as the step size of the proposals. If the step size is chosen to be too large, the acceptance probability will become small, resulting in a larger value of M_n which implies a longer simulation run-time. However, if λ_n is chosen to be small, the proposals will then not depart far from the current state. As a result, the particles are close to each other, resulting in an underestimation of the variance of the target. In our implementation, we adaptively change the value of λ_n such that a certain average acceptance probability is maintained at each SMC step while keeping the value of λ above a certain minimum value λ_{min} at the same time. Hence, if α^* is the average acceptance probability we want to achieve, and the average acceptance probability at step $n-1$ is estimated to be $\hat{\alpha}_{n-1}$ when the scale factor is λ_{n-1}, the scale factor λ_n at step n can then be chosen adaptively to be

$$\lambda_n := \max\left\{\exp(\log(\lambda_{n-1}) + s(\hat{\alpha}_{n-1} - \alpha^*)), \lambda_{min}\right\} \tag{4.12}$$

Such a tuning strategy is reminiscent of the adaptive MCMC algorithm of Andrieu and Thoms (2008). The difference here is that nothing is needed to guarantee the correctness of the algorithm since the choice of λ_n is only part of the importance distribution used in the SMC algorithm. Algorithm 4.5 lists the details of the implementation of the L-ABC-SMC algorithm.

Algorithm 4.5: Chap RESMC - Alg: L-ABC-SMC Algorithm

Input :

- The number of particles at each SMC step, N
- A terminal tolerance level, ϵ
- The observed data, $\mathbf{y}^*$
- A distance metric, $\|\cdot\|$
- Parameter controlling the choice of the tolerance sequence, η
- The initial scale factor λ_1
- The mimum scale factor λ_{min}
- Initial number of MCMC steps, M_1
- The minimum proportion of particles to be moved, μ

for $n = 0$ **do**
  Set $\epsilon_0 := \infty$;
  for *i = 1* **to** *N* **do**
    Sample $\zeta_0^i \sim \rho(d\zeta)$;
    Set $W_0^i := 1/N$;
    Calculate $D_0^i = \|\mathbf{s}^* - \Psi(\zeta_0^i)\|$;
for $n = 1, 2, ...$ **do**
  Estimate the covariance matrix of $\pi_n^{ABC}(\zeta_n|\mathbf{s}^*)$, $\hat{\Sigma}_n$, based on the particles with non-zero weights obtained in time $n-1$, $\left\{\xi_{n-1}^{\mathbf{A}_n - 1}\right\}$;
  Resample the particles according to their weights $W_{n-1}^{1:N}$ and set $\mathbf{A}_{n-1} := \left(A_{n-1}^1, .., A_{n-1}^N\right)$ to be the indices of the resampled particles;
  Determine the next tolerance level by $\epsilon_n = \max\left\{\epsilon, \mathbf{D}_{n-1}^{\mathbf{A}_{n-1}, (\lfloor \eta N \rfloor)}\right\}$, where $\lfloor \eta N \rfloor$ is the largest integer that is not bigger than ηN and $\mathbf{D}_{n-1}^{\mathbf{A}_{n-1}, (\lfloor \eta N \rfloor)}$ represents the $\lfloor \eta N \rfloor$-th smallest value of $\mathbf{D}_{n-1}^{\mathbf{A}_{n-1}}$;
  for *i=1* **to** *N* **do**
    Set $\zeta_n^i, I_n^i := \textbf{MABCMCMC}\left(\zeta_{n-1}^{A_{n-1}^i}, M_n, \mathbf{s}^*, \epsilon_n, \lambda_n, \hat{\Sigma}_n\right)$;
    Obtain the unnormalised weights by

$$w_n^i := \mathbb{I}\left(\left\|\mathbf{s}^* - \Phi\left(\zeta_{n-1}^{A_{n-1}^i}\right)\right\| < \epsilon_n\right)$$

    and normalise to get $W_n^i = w_n^i / \sum_{j=1}^N w_n^j$;
    Set $\mathcal{A}_n := \sum_{i=1}^N I_n^i$ and $\hat{\alpha}_n := \mathcal{A}_n/(NM_n)$;
    Set M_{n+1} such that $M_{n+1} > \frac{\log(1-\mu)}{\log(1-\hat{\alpha}_n)}$;
    Set $\lambda_{n+1} := \max\left\{\exp(\log(\lambda_n) + s(\hat{\alpha}_n - \alpha^*)), \lambda_{min}\right\}$;
  if $\epsilon_n = \epsilon$ **then**
    break;

4.3 Computational Cost Comparison

In this section, we are going to compare the asymptotic computational cost needed for the L-SMC-ABC algorithm and the rare-events approach of Prangle et al. (2018) to generate 1 sample to approximate the target $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$. For the justification to be correct, we need to make the following assumptions about the algorithms:

A1. The distance metric used in the algorithms is the Euclidean distance.

A2. Running the ABC-MCMC kernel once within the L-ABC-SMC algorithm requires $\mathcal{O}(1)$ functional evaluations and running the slice sampling algorithm once within the RE-SMC algorithm also requires $\mathcal{O}(1)$ functional evaluations.

A3. The numbers of loops of ABC-MCMC kernel at different SMC iterations are roughly constant, i.e. M_n's stays roughly the same across different SMC steps. Also, the number of slice sampling loops within the RE-SMC algorithm also stays roughly constant at different SMC steps.

A4. The time required for the simulator to sample one set of pseudo-observations is bounded above and below by non-zero constants that do not depend on the inputs of the simulator.

A5. The uniform ABC kernels are used in both algorithms, i.e. $K_\epsilon(\mathbf{s}^* - \mathbf{s}) \propto \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)$

A6. The joint prior density, $\rho(\zeta)$, and the likelihood function $l(\mathbf{s}^*|\theta)$, are bounded above by M_ζ and $M_{\theta,\mathbf{s}^*}$ for all values of ζ, θ and $\mathbf{s}^*$.

In the L-ABC-SMC algorithm, an estimate of the normalising constant $\tilde{\mathcal{Z}} := \int \rho(\zeta_P) K_{\epsilon_P}(\mathbf{s}^* - \Psi(\zeta_P))\, d\zeta_P$, with $\epsilon_P := \epsilon$ denotes the final tolerance achieved by the algorithm, can by obtained by

$$\hat{\tilde{\mathcal{Z}}}_P = \prod_{n=1}^{P} \sum_{i=1}^{N} W_{n-1} \mathcal{G}_n(\zeta_{0:n}^i)$$

When the uniform kernel is used in the algorithm, we should have

$$\tilde{\mathcal{Z}}_P = \int \rho(\zeta_P) \mathbb{I}(\|\mathbf{s}^* - \Psi(\zeta_P)\| < \epsilon)\, d\zeta_P = \int_{\mathcal{B}_\epsilon^{\mathbf{s}^*}} \rho(\zeta_P)\, d\zeta_P \leq M_\zeta m(\mathcal{B}_\epsilon^{\mathbf{s}^*})$$

where $\mathcal{B}_\epsilon^{\mathbf{s}^*} := \{\zeta : \|\mathbf{s}^* - \Psi(\zeta)\| < \epsilon\}$. Denote $\mathcal{V}(\epsilon) := m(\mathcal{B}_\epsilon^{\mathbf{s}^*})$ for simplicity. We can see that

when $\epsilon \ll 1$, $\tilde{\mathcal{Z}}_P \approx \mathcal{V}(\epsilon)$. In L-ABC-SMC algorithm, particles are resampled at every SMC step, hence $W_n^i := 1/N$ for all $n := 0, 1, 2, ..., P$ and $i = 1, 2, .., N$. Moreover, since the uniform kernels were used in the L-ABC-SMC algorithm, we would have that $\mathcal{G}_n(\zeta_{0:n}^i)$ should be either 0 or 1, depending on the choice of ϵ_n and ζ_{n-1}, as shown in (4.10). According to line 10 Algorithm 4.5, we can see that ϵ_n are adaptively chosen such that approximately ηN of the N particles obtained at $n-1$-th SMC step will produce summary statistics with a distance of smaller than ϵ_n from $\mathbf{s}^*$. As a result, one could see that $\sum_{i=1}^N \mathcal{G}_n(\zeta_{0:n}^i) \approx \eta N$ for all $n = 1, 2, .., P$. Hence, we have

$$\begin{aligned}
\hat{\tilde{\mathcal{Z}}}_P &= \prod_{n=1}^{P} \sum_{i=1}^{N} W_{n-1} \mathcal{G}_n(\zeta_{0:n}^i) \\
&= \prod_{n=1}^{P} \sum_{i=1}^{N} \frac{1}{N} \mathcal{G}_n(\zeta_{0:n}^i) \\
&= \prod_{n=1}^{P} \frac{1}{N} \sum_{i=1}^{N} \mathcal{G}_n(\zeta_{0:n}^i) \\
&\approx \prod_{n=1}^{P} \frac{1}{N} \eta N = \eta^P
\end{aligned}$$

Hence, when a small terminal tolerance ϵ is used, we can obtain an estimation of the P, the number of SMC steps required for the L-ABC-SMC algorithm to reach ϵ, by

$$P \approx \log(\tilde{\mathcal{Z}}_P)/\log(\eta) \approx \log(\mathcal{V}(\epsilon))/\log(\eta)$$

If *A3* is satisfied (it is possible to achieve this when the scale factor is chosen adaptively), the computational cost for one SMC step will then be $\mathcal{O}(\eta N)$ as only approximately ηN particles needed to be moved at each SMC step. Therefore, the total computational cost would be $\mathcal{O}(\eta N \log(\mathcal{V}(\epsilon))/\log(\eta))$. At the end of the L-ABC-SMC algorithm, we would expect to obtain ηN alive particles that can be used to approximate $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$. Therefore, the computational cost for obtaining 1 sample approximating $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$ would then be $\mathcal{O}(\log(\mathcal{V}(\epsilon))/\log(\eta))$.

Similar to the L-ABC-SMC sampler, the rare-event approach of Prangle et al. (2018) relies on the implementation of the RE-SMC algorithm to obtain estimations of the likelihood of different parameter values. As outlined in Algorithm 4.1, we can see that the estimation of

the rare-event probability is obtained by taking the product of $\hat{R}_n$ obtained at each SMC step within RE-SMC algorithm. Also, we can see that the value of ϵ_n is chosen such that $\hat{R}_n \approx \eta = N_{acc}/N$ for all $n = 1, 2, .., P$. Hence, we will have that an estimation of the normalising constant, which is also $\Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon|\theta)$, is given by

$$\Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon|\theta) \approx \eta^P$$

On the other hand, by the Lebesgue differentiation theorem, we should have that

$$\lim_{\epsilon \to 0} \frac{1}{m(\mathcal{B}_\epsilon^{\mathbf{s}^*})} \int l(\mathbf{s}|\theta) \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)\, d\mathbf{s} = l(\mathbf{s}^*|\theta) \tag{4.13}$$

Hence, for $\epsilon \ll 1$, we should have that

$$\int l(\mathbf{s}|\theta) \mathbb{I}(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon)\, d\mathbf{s} = \Pr(\|\mathbf{s}^* - \mathbf{s}\| < \epsilon|\theta) \approx l(\mathbf{s}^*|\theta) m(\mathcal{B}_\epsilon^{\mathbf{s}^*}) \sim m(\mathcal{B}_\epsilon^{\mathbf{s}^*})$$

Hence, for small values of ϵ, we could obtain a similar estimate for the rare-event probability, which is $P \approx \log(\mathcal{V}(\epsilon))/\log(\eta)$. Hence, the computational cost for running the RE-SMC algorithm once will be $\mathcal{O}(N \log(\mathcal{V}(\epsilon))/\log(\eta))$ and this is also the computational cost for obtaining a sample approximating $\pi_\epsilon^{ABC}(\theta|\mathbf{s}^*)$ from the PMMH-RESMC algorihtm.

4.4 Numerical Examples

4.4.1 The g-and-k distributions

The first example we considered is the g-and-k distribution, which is defined as

$$x := a + b \left(1 + 0.8 \frac{1 - \exp(-gz)}{1 + \exp(-gz)}\right) (1 - z)^k z \tag{4.14}$$

where z represents the realisations from the standard Normal distribution and $\theta = (a, b, g, k)$ the parameters. We will denote $g_\theta(z)$ as the transformation defined in (4.14). The g-and-k distribution is one of the commonly used numerical examples in the ABC literature. Given a specific parameter vector, it is easy to generate samples from the g-and-k distribution - one could do this by generating samples from the standard Normal distribution and then trans-

forming them according to $g_\theta(z)$ to obtain samples from the g-and-k distribution. However, the probability density function of the g-and-k distribution is intractable to calculate. To see this, let X be the random variable following the g-and-k distribution, and then one can see that

$$X := g_\theta(Z)$$

where Z is a random variable following the standard Normal distribution. Therefore, it is easy to see that

$$f_X(x) = \phi(g_\theta^{-1}(x)) \left| \frac{dg_\theta^{-1}(x)}{dx} \right|$$

which is intractable since we are not able to find an analytical form of the inverse function g_θ^{-1}. However, there is a way of calculating the density function with high precision, making it still possible to devise an MCMC sampler to target the posterior distribution. The details of the implementation of the MCMC sampler are described in Appendix A. We can then use the outputs from the MCMC sampler as ground truth to investigate the correctness and accuracy of the various ABC samplers.

To compare the performance and efficiency among various ABC algorithms, we generated 20 data from the g-and-k distribution with parameter $\theta = (3.0, 1.0, 2.0, 0.5)$ and applied the standard ABC-SMC algorithm, the rare-event approach (RE-SMC) designed by Prangle et al. (2018) and the L-ABC-SMC algorithm we developed in this chapter to produce estimations of the posterior distribution of the parameters $\theta := (a, b, g, k)$. We assigned the parameters a uniform prior on $[0, 10]^4$. For both ABC-SMC and L-ABC-SMC samplers, we chose to use the uniform kernel and use the full dataset, i.e. $S(y) := y$. The distance metric was chosen to be the Euclidean distance for both SMC samplers. We also followed the same tunning strategies - the empirical variance of 'alive' particles at each SMC step was calculated and used as the proposal variance in the next SMC step. Tolerances were also chosen adaptively so that the effective sample size at each SMC step would remain above $0.8N$, where N represents the number of particles. Both ABC-SMC and L-ABC-SMC algorithms used random-walk ABC-MCMC sampler for explorations and proposals of new particles and the stepsize and the number of iterations of the ABC-MCMC sampler were also tuned adaptively within the SMC algorithm. More specifically, acceptance probabilities were estimated at each SMC step and

used to tune the scale factor and MCMC steps for the next SMC step. Scale factors were tuned to ensure that the acceptance probability remains above 0.2 and the MCMC steps were chosen so that 99% of the particles are expected to be moved. Moreover, the scale factor was kept above 0.1 to prevent the particles from collapsing into a narrow region. Early stopping rules were also adopted hence both ABC-SMC and L-ABC-SMC samplers will be terminated early if the acceptance probability drops below 0.1%.

For the RE-SMC algorithm, we followed Prangle et al. (2018) to tune the algorithm. More specifically, we used the adaptive RE-SMC algorithm in which the tolerance sequence is chosen adaptively in the same way as the standard ABC-SMC algorithm. Also, we chose N_{acc} to be half of the total number of particles, i.e. $N_{acc} := 0.5N$. Slice sampling with adaptive search width was used within RE-SMC to move the particles around. For the corresponding PMMH algorithm, we started the chain at the true parameter values to reduce the burn-in period and used the random-walk kernel as the proposal distribution. To ensure efficient proposals, we ran L-ABC-SMC sampler with the same terminal tolerance to estimate the variance of the parameters and used it as the proposal variance. We ran the PMMH algorithm for $2,000$ iterations and all the SMC algorithms were run with $5,000$ particles. With the same simulated observations, we ran the three algorithms with terminal tolerances $\epsilon = 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 15.0, 20.0$

We first compare the computational cost required by each algorithm to obtain one sample from the approximate posterior distribution with different terminal tolerance values. Table 4.1 shows the results. As expected, when the terminal tolerance is set to be large, both SMC algorithms have comparable computational costs but the rare event approach of Prangle et al. (2018) has a significantly higher computational cost. However, the ABC-SMC sampler was not able to reach smaller terminal tolerances and the algorithm was terminated early due to the low acceptance probability of the ABC-MCMC sampler within the ABC-SMC algorithm. On the contrary, both the L-ABC-SMC algorithm and the RE-SMC method were able to produce samples with terminal tolerances as low as 0.2. This suggests that one can target ABC posteriors that are closer to the actual posterior distribution by using L-ABC-SMC and RE-ABC methods, hence reducing the level of approximations. One should also notice that the computational cost for the L-ABC-SMC algorithm is consistently lower than that of the RE-SMC method, which justifies the asymptotic arguments in the previous section.

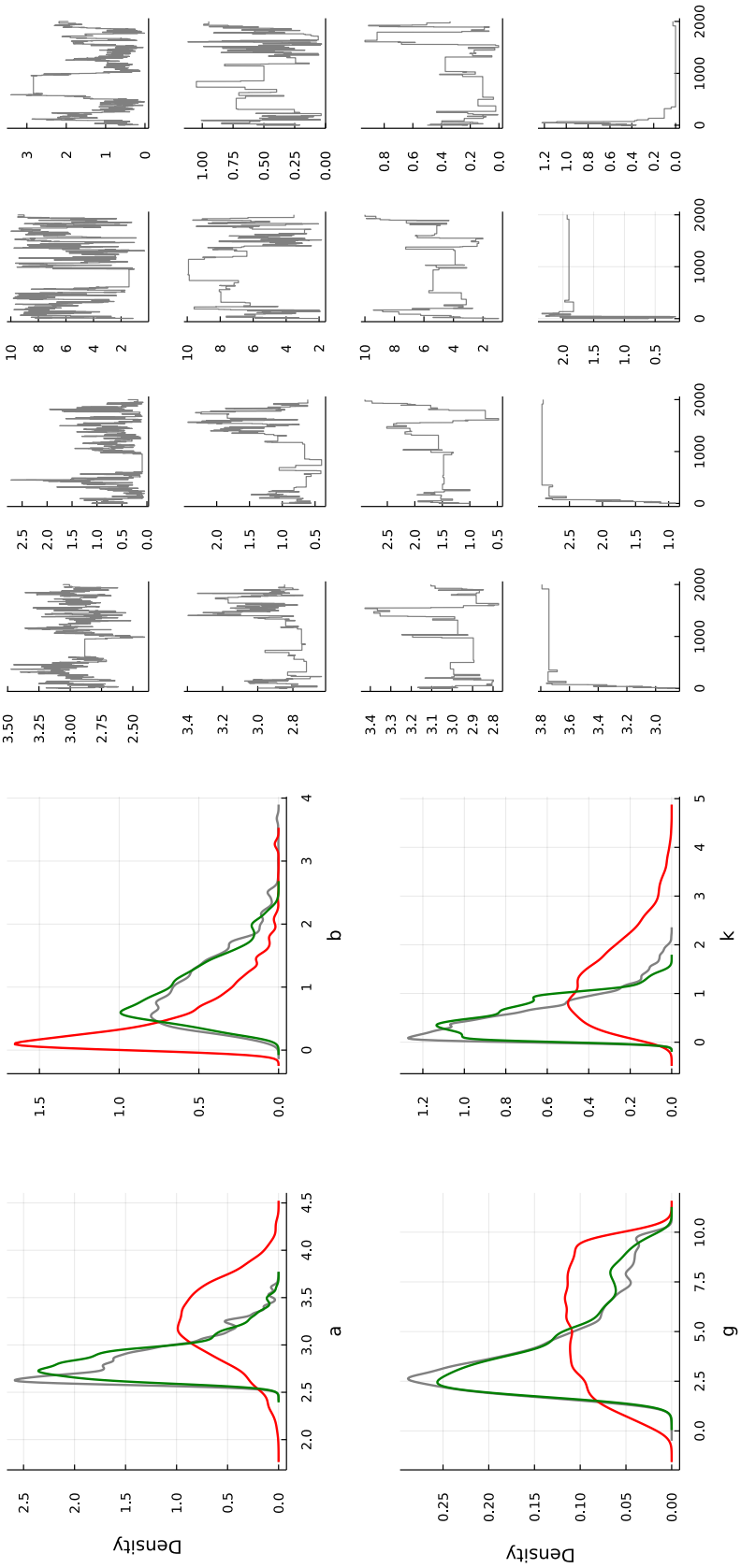

Figure 4.2: Simulation results of the g-and-k distribution with 20 observations. Left panel: kernel density plots of the posterior distributions of θ. The plots show kernel density estimations obtained from L-ABC-SMC (green lines), ABC-SMC (red lines) and MCMC sampler (grey lines). The final tolerance for the L-ABC-SMC sampler is 0.2 whereas the final tolerance for the ABC-SMC sampler is 5.44. Left Panel: Trace plots of the PMMH algorithm of the rare-even approach in Prangle et al. (2018). Tolerances used were 2.0 (first row), 1.0 (second row), 0.5 (third row) and 0.2 (last row)

Terminal Tolerance	L-ABC-SMC	ABC-SMC	RE-ABC
25.0	2.37	**2.34**	4.02
20.0	2.45	**2.44**	4.03
15.0	2.54	**2.51**	3.98
10.0	**2.69**	2.70	4.24
5.0	**3.10**	4.80	5.20
2.0	**3.35**	-	5.70
1.0	**3.49**	-	5.93
0.5	**3.60**	-	6.02
0.2	**3.78**	-	5.96

Table 4.1: The computational cost for obtaining one sample from the approximate posterior distribution with different terminal tolerances. The computational cost was measured by the number of pseudo-data generations required to produce one sample. Numbers in the table were given in the log10-scale.

Figure 4.2 further illustrates the performance of the algorithms we considered. The left panel shows the kernel density estimation of the parameters' posterior distributions obtained by L-ABC-SMC, ABC-SMC and the MCMC sampler. With 20 observations, the L-ABC-SMC sampler could reach a terminal tolerance of 0.2 while the ABC-SMC sampler was terminated early with the final tolerance equal to 5.44. Hence, one can obtain nearly the true posteriors through the L-ABC-SMC sampler, as shown in the figure. Compared to the L-ABC-SMC sampler, the ABC-SMC sampler was not able to produce accurate estimations, due to the fact that it was terminated earlier and hence could not reach small tolerances. The RE-SMC method was also able to produce samples from an ABC-posterior with small tolerances. However, with small tolerances (e.g. $\epsilon = 0.2, 0.5$) the chain obtained from the PMMH algorithm was stuck in a certain state for a long time, resulting in very high variances in the posterior estimations. This is illustrated by the trace plots in the left panel of Figure 4.2. One can see that the RE-SMC method could only produce reliable estimations when the tolerance was 2.0 (top row of the left panel). When tolerance 0.2 was used (bottom row of the left panel), the chain was stuck at one state for a long time, making the results not suitable for posterior estimations.

The L-ABC-SMC sampler also scales fairly well and can produce accurate estimations when the dimension of the latent variables gets higher. Figure 4.3 shows the kernel density estimations when 50 and 100 data were observed. One could still obtain samples from ABC posterior of tolerance 0.5 using the L-ABC-SMC sampler, creating estimations that are almost identical to

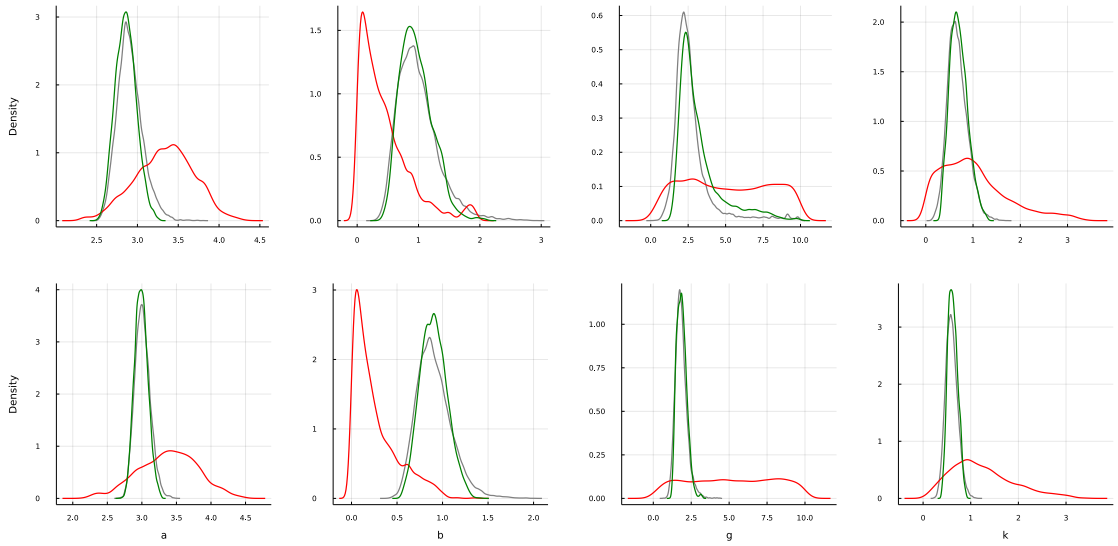

Figure 4.3: Kernel density estimation of the g-and-k distribution with 50 (top) and 100 (bottom) observations. Estimations were obtained by L-ABC-SMC(green), ABC-SMC(red) and MCMC(grey) samplers. The final tolerances for the L-ABC-SMC sampler are both 0.5 and the final tolerances for the ABC-SMC sampler are 12.93 for 50 observations and 22.62 for 100 observations.

the true posteriors. Compared to L-ABC-SMC, the final tolerances reached by the ABC-SMC sampler are much higher, resulting in inaccurate estimations.

4.4.2 The Lotka-Volterra Model

In this section, we are inferring the static parameters in the SDE variant of the Lotka-Volterra predator-prey model (Wilkinson, 2018) that is commonly considered in the ABC literature (e.g. Meeds and Welling (2015); Papamakarios and Murray (2016)). In particular, the Lotka-Volterra model is a stochastic process describing the time evolution of the population of predators interacting with a population of prey. Given the synthetic observed predator-prey population data, we are trying to infer the parameters of the following SDEs

$$dX = (\theta_1 X - \theta_2 XY)\, dt + \sigma_X\, dn_X \tag{4.15a}$$

$$dY = (\theta_4 XY - \theta_3 Y)\, dt + \sigma_Y\, dn_Y \tag{4.15b}$$

where X represents the prey population and Y represents the predator population, $\theta := (\theta_1, \theta_2, \theta_3, \theta_4)$ are the parameters to be inferred and n_X and n_Y are two zero mean white noise processes with variance σ_X^2 and σ_Y^2. In this example, we used Euler–Maruyuma scheme

Nsengiyumva et al. (2013) to generate realisations of the process at discrete times. Suppose that the initial value of the process is (x_0, y_0) and the time step δt and process is at (x_{n-1}, y_{n-1}) at step $n-1$, the values at step n, (x_n, y_n) can then be simulated by

$$\begin{aligned} x_n &= (\theta_1 x_{n-1} - \theta_2 x_{n-1} y_{n-1})\delta t + \sigma_X \sqrt{\delta t} u_{n,1} \\ y_n &= (\theta_4 x_{n-1} y_{n-1} - \theta_3 y_{n-1})\delta t + \sigma_Y \sqrt{\delta t} u_{n,2} \end{aligned} \tag{4.16}$$

where $u_{n,1}, u_{n,2}$ are random variables simulated from a standard Normal distribution. Hence, we can see that pseudo-data at d discrete time steps can be then generated by the parameters $\theta := (\theta_1, \theta_2, \theta_3, \theta_4)$ and $2d$ random variables from the standard Normal distributions. Therefore, we can treat all these standard Normal random variables as the random seeds $\mathbf{u} \in \mathbb{R}^{2d}$ and the observations $(x_{1:d}, y_{1:d})$ can then be interpreted as outputs from a deterministic function $\phi_\theta(\mathbf{u})$. Algorithm 4.6 outlines the definition of the function $\phi_\theta(\mathbf{u})$.

Algorithm 4.6: Simulator $\phi_\theta(\mathbf{u})$

Input :
- The static parameter, θ
- The random seeds $\mathbf{u}$
- The initial values (x_0, y_0)
- The standard deviations, σ_X and σ_Y

Set $N := dim(\mathbf{u})/2$;
for $n = 1, 2, .., N$ **do**
  Set $x_n = (\theta_1 x_{n-1} - \theta_2 x_{n-1} y_{n-1})\delta t + \sigma_X \sqrt{\delta t} u_{2n-1}$;
  Set $y_n = (\theta_4 x_{n-1} y_{n-1} - \theta_3 y_{n-1})\delta t + \sigma_Y \sqrt{\delta t} u_{2n}$;

Output: The pseudo-data $(x_1, y_1, .., x_N, y_N)$

For the experiment, we simulated 100 artificial observed data, $(x_1, y_1, .., x_{50}, y_{50})$, with $\sigma_X = \sigma_Y = 1$ and $x_0 = y_0 = 100$. For the discrete time step, we chose $\delta t = 1$ and the parameters were set to be $\theta_1 = 0.4, \theta_2 = 0.005, \theta_3 = 0.05, \theta_4 = 0.001$. Figure 4.4 shows the data we will be using for the experiment.

Given the observations shown in figure 4.4, we tried to make inferences on the model parameter $\theta := (\theta_1, \theta_2, \theta_3, \theta_4)$, assuming that $x_0, y_0, \sigma_X, \sigma_Y$ and δt is known. For the static parameters θ, we assign a Log-Normal prior with location $\mu = -2$ and scale $\sigma = 1$. Hence, $\log \theta_i \sim \mathcal{N}(-2, 1)$ for $i = 1, 2, 3, 4$. In the Lotka-Volterra model, we chose the priors informatively to minimise the prior probability of generating infeasible parameter settings. The Lotka-Volterra model

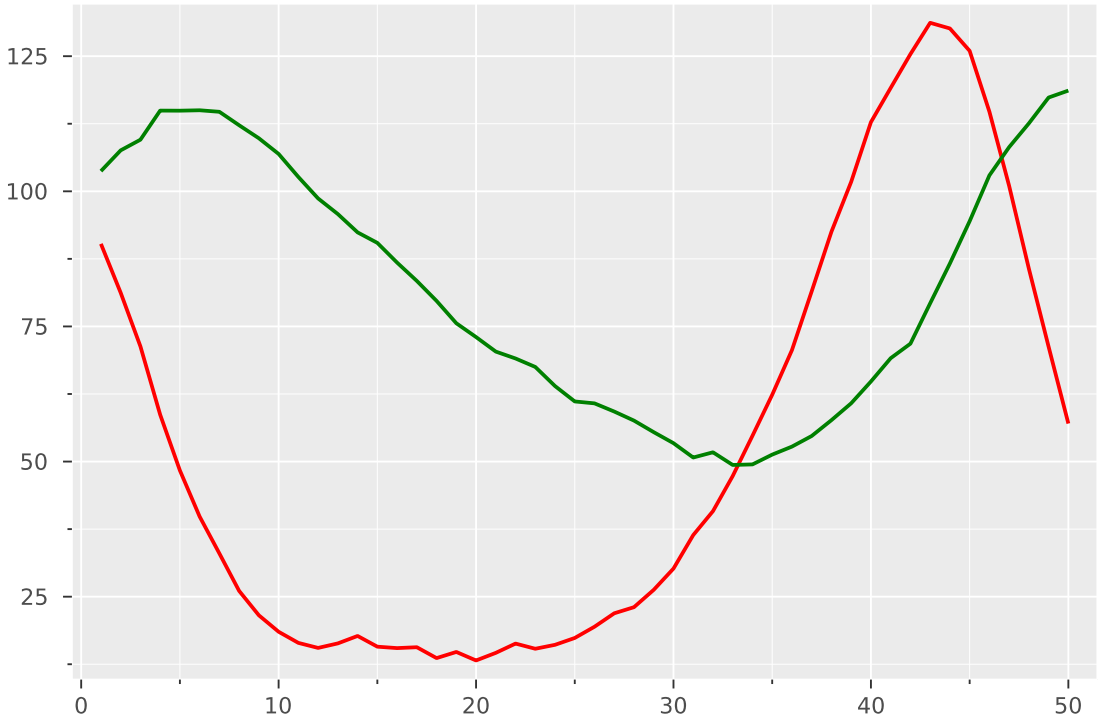

Figure 4.4: Observations of the Lotka-Volterra model used in the numerical example. The data were simulated with $x_0 = y_0 = 100, \sigma_X = \sigma_Y = 1.0$ and $\delta t = 1$. The green line represents the predator population and the red line represents the prey population.

appears to be unstable for many parameter settings, with the prey population blowing up exponentially if the predator population becomes zero. Such a blow-up is both biologically implausible and likely to cause numerical issues. Hence, we chose to bias the prior towards smaller values to make it more likely to produce parameter settings with stable dynamics.

We used the same implementation details as that in the g-and-k distribution example. The terminal tolerance was set to be 5.0 for the L-ABC-SMC algorithm. As observations of the Lotka-Volterra model are quite sensitive to the parameter settings, the rare-event approach of Prangle et al. (2018) was likely to get stuck when an initial value was chosen from the prior density of θ - the RE-SMC sampler within the PMMH-RESMC algorithm failed to decrease to 5.0 effectively, making the algorithm fail to estimate the ABC-likelihood with $\epsilon = 5.0$. Such a problem was illustrated in Figure 4.5. When the initial parameters are sampled from the prior, it is almost impossible for the RE-SMC algorithm to reach the terminal tolerance with the parameter settings. Hence, we will be unable to obtain estimates of the ABC-likelihoods from the RE-SMC algorithm, making the PMMH-RESMC algorithm impossible to proceed

with. Such a problem will persist even when parameters that are very close to the true values are used (as shown by the grey lines in Figure 4.5). An estimate of the ABC-likelihood with $\epsilon = 5.0$ can only be obtained when the actual parameters are used in the RE-SMC algorithm. However, we will not be able to know these actual values in reality, making the rare-event approach of Prangle et al. (2018) impractical to use.

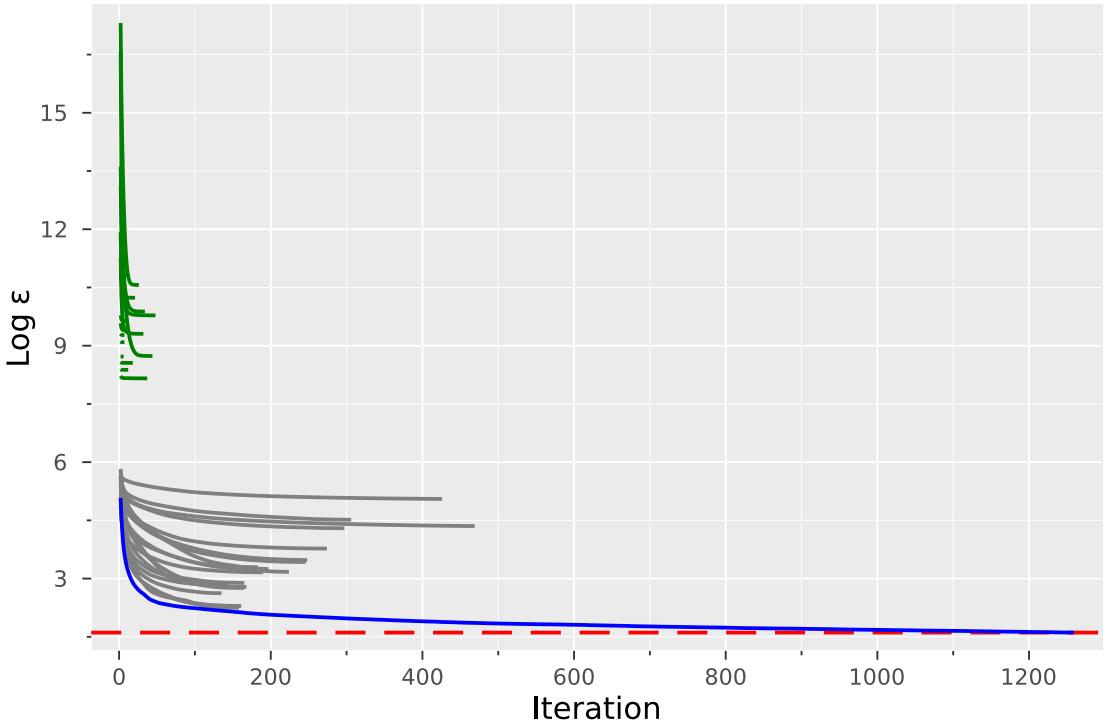

Figure 4.5: Evolution of the ϵ values of ADAPT-RE-SMC algorithm for Lotka-Volterra model. Red dashed lines: the log-terminal tolerance to be achieved, which is $\log 5.0$. Green lines: the evolution of the tolerances with parameters sampled from the prior. Grey lines: evolution of the tolerances with parameters sampled from $\mathcal{N}(\log \theta_i, 0.1^2)$ for $i = 1, 2, 3, 4$. Blue line: evolution of the tolerances with parameters chosen to be the true parameters. Each ADAPT-RE-SMC algorithm will be stopped if the percentage change in ϵ is smaller than 0.1%

Figure 4.6 showed the results when the informative prior is used. One could see that when applying the L-ABC-SMC algorithm to data sets with highly dependent structures, it is also able to produce accurate estimations of the posterior by reaching much smaller final tolerance values with given computational constraints. We also tried to apply the algorithms when the priors of the parameters are not chosen appropriately and investigated the effect of the prior choices on the algorithms. We considered two cases under the Lotka-Volterra model. The first is when diffuse priors are assigned to the parameters. In this case, we assign a prior of $LogNormal(0.0, 5.0^2)$ to each of the parameters. Hence, there is a higher chance for the

parameters sampled from the prior to have very large or very small values with this prior. In the second case, we considered using a prior of $LogNormal(0, 1^2)$ for each of the parameters. By using this prior, we are unlikely to get values close to the true parameters from the prior. In other words, we put the prior mass in a "wrong" region in the parameter space.

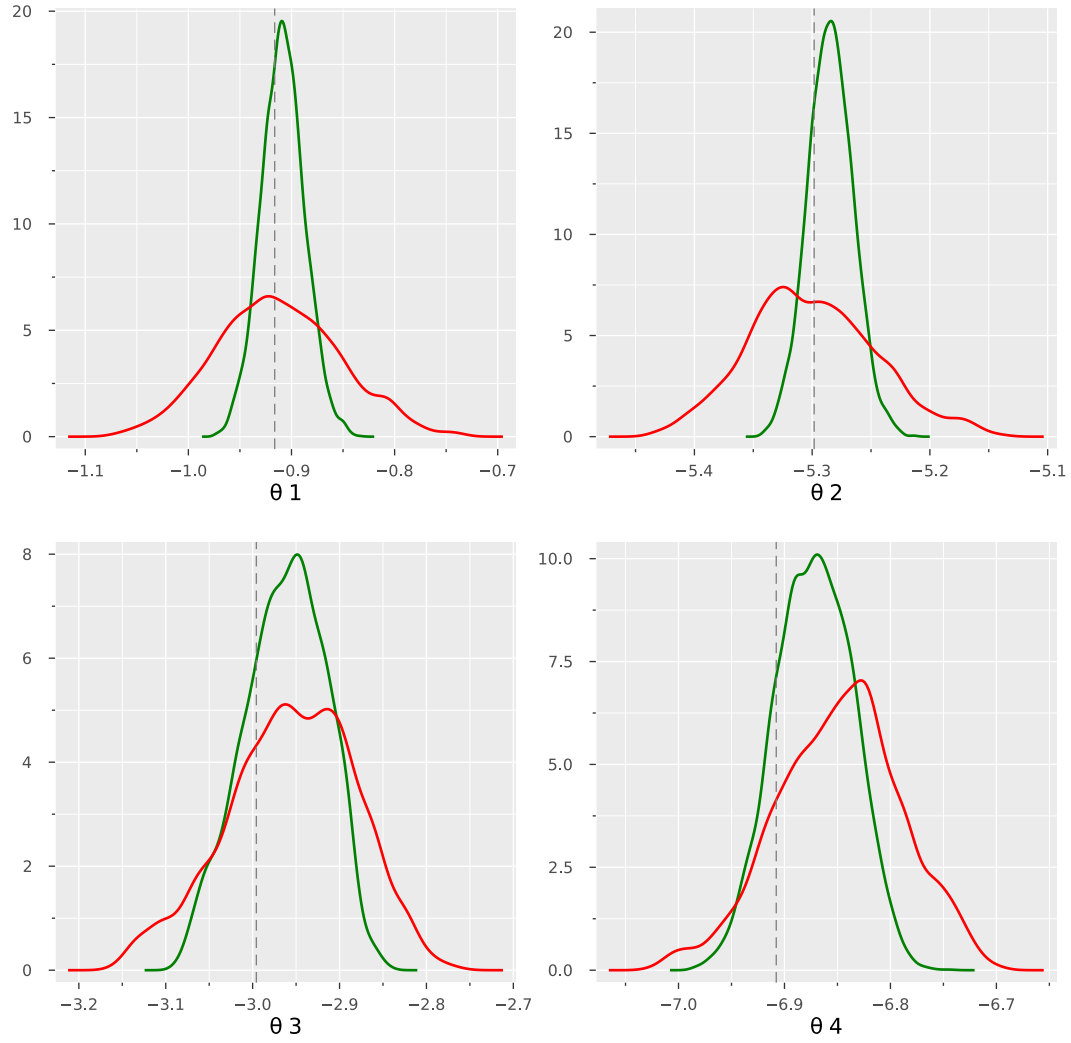

Figure 4.6: Posterior estimations with priors is chosen to be $LogNormal(-2, 1^2)$ for all the parameters. Green lines are the ABC posterior estimations obtained by the L-ABC-SMC algorithm with terminal tolerance equal to 5.0. Red lines are the ABC posterior estimations obtained by the ABC-SMC algorithm with final tolerance equal to 27.5. Grey dashed lines are the true parameter values.

Figure 4.7 shows the results obtained using the two inappropriate priors. We can see that the L-ABC-SMC algorithm is quite robust to inappropriate choice of priors and continues producing accurate estimations of the posterior densities. ABC-SMC algorithm is also robust to inappropriate priors, but it can only produce less accurate estimations due to the early

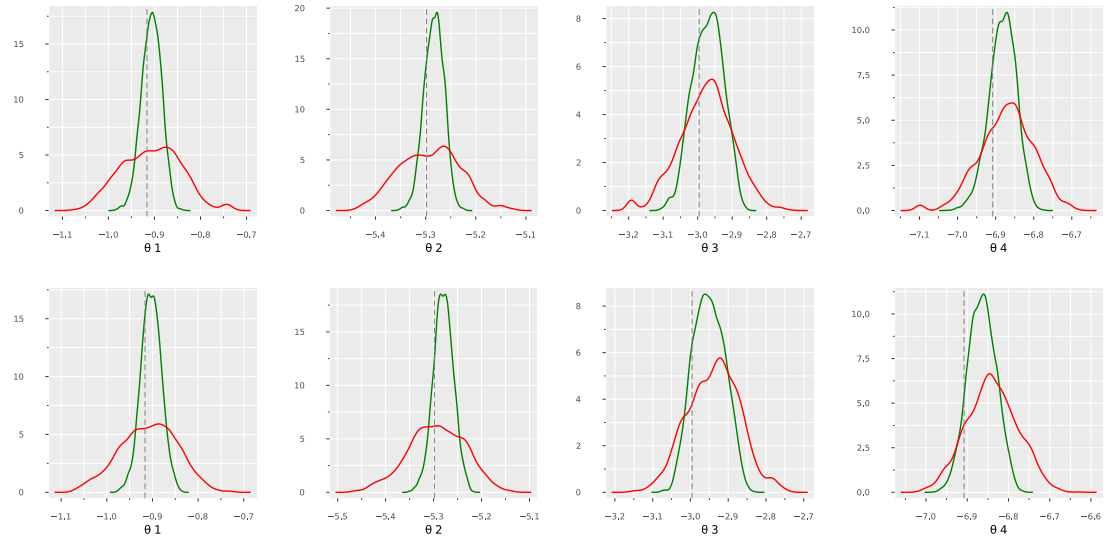

Figure 4.7: Poster estimations with diffuse and misspecified prior. Top row: Parameter estimations with diffuse prior. Bottom row: Parameter estimations with misspecified prior. Green lines are estimations obtained by the L-ABC-SMC algorithm and red lines are estimations obtained by the ABC-SMC algorithm. Grey dashed lines are the true parameter values. The final tolerances for L-ABC-SMC for both cases are 5.0.

termination of the algorithm caused by a low acceptance rate. However, we were not able to successfully obtain estimations of the ABC likelihood using the RE-SMC algorithm when the prior are chosen inappropriately. This is due to the same reason as we discussed before and when the priors are chosen inappropriately, such a problem would be even worse. Consequently, the PMMH-RESMC algorithm failed to produce any results in this case.

4.4.3 Conclusions

In this chapter, we proposed a new method, named the L-ABC-SMC algorithm, that uses the same posterior approximation as standard ABC algorithms (e.g. Rejection ABC, ABC-MCMC or ABC-SMC samplers). Our proposed method utilised the latent random variables that are purely responsible for the randomness in the simulated data and hence can be used to generate pseudo-data through a deterministic function. Compared to the standard ABC-SMC algorithm of Del Moral et al. (2012), the L-ABC-SMC algorithm could reach much smaller tolerance values given the same computational constraints, hence producing more accurate estimations of the posterior densities. Such superiority has been shown by two numerical examples - the g-and-k distribution and the Lotka-Volterra model. A rare-event approach

has also been developed by Prangle et al. (2018) recently which also uses SMC samplers to explore the latent variable space. Unlike the L-ABC-SMC algorithm, the SMC algorithm in Prangle et al. (2018), which is termed as RE-SMC algorithm is used to find an estimation of the ABC likelihood given a specific parameter setting. Such estimation is then used within a pseudo-marginal MCMC scheme to produce samples from the corresponding ABC posterior. In this chapter, we refer to this rare-event approach as the PMMH-RESMC algorithm. As shown in the numerical example, the PMMH-RESMC algorithm is likely to produce chains that are stuck at a certain state for a long time when the tolerance is chosen to be as small as that used by the L-ABC-SMC algorithm. As a result, it will produce estimations with very high variances, which is not acceptable in practice. Even worse, when the observations are sensitive to the parameter settings (like those in the Lotka-Volterra model), the RE-SMC algorithm may fail to produce an estimation of the ABC likelihood for inappropriate choice of parameter values, making the PMMH-RESMC algorithm implausible in practice. Moreover, the successful implementation of the PMMH-RESMC algorithm relies on several tuning choices. As discussed in Prangle et al. (2018), a pilot run of the algorithm is suggested to find an estimation of the target variance that can be used as the proposal variance. It is also recommended that for each parameter setting, an ADAPT-RE-SMC algorithm is run to find the appropriate tolerance schedule followed by one run of the FIXED-RE-SMC algorithm with the chosen tolerance schedule to estimate the corresponding likelihood. Compared to the RE-ABC algorithm, the tuning of the L-ABC-SMC algorithm is fairly automatic and all the tuning steps can be done within the algorithm without any pilot runs. This makes our algorithm easier to implement and more available to the general community.

There are several possible extensions that can be considered as future work:

1. We noticed that a good estimation of the variances is quite crucial to the good performance of our method. In this paper, we used the empirical variances of the alive particles at each SMC step as the estimation of the variances. Other methods may be considered to improve the accuracy of the variance estimation.

2. We considered using the random-walk Metropolis-Hastings algorithm to explore the ABC-posteriors at each SMC step. Although it was successful in both of the numerical experiments, it has been shown many times in the literature that gradient-based and

non-reversible algorithms would have better performance compared to the random-walk algorithm. We tried to adopt Metropolis Adjusted Langevin Algorithm (MALA) and Bouncy Particle Samplers (BPS) within the ABC-SMC framework to explore the ABC-posteriors. However, it was not very successful and did not produce an improvement in performance with increased cost. It would be interesting to explore further in this direction for the possibilities of improving the proposed method with advanced MCMC algorithms.

3. We considered the model where the number of random seeds is fixed in this paper. There are models that rely on a variable number of random seeds (e.g. genealogy tree in population genetics) and the method we proposed here will no longer be suitable for those models. Hence, an interesting extension to the current method would be allowing the number of random seeds to be variable. This would be similar to the Reversible-jump MCMC algorithm.

Chapter 5

Markov Snippet SMC (Monte Carlo in general)

5.1 Introduction

The work presented in this chapter is mainly inspired by the waste-free SMC algorithm proposed by Dau and Chopin (2020). As discussed in previous chapters, SMC samplers are a class of stochastic algorithms targeting an arbitrary sequence of distributions $\pi_t(dx), t = 1, 2, .., P$, where π_P may be the actual density of our interest. Within an SMC sampler, importance sampling, resampling and Markov steps are often involved. The waste-free SMC algorithm makes improvements to the Markov steps, at which particles are moved by a k-fold MCMC kernel that leaves π_t invariant at each SMC iteration. In certain scenarios, a large value of k is needed in order to ensure good performances at the Markov steps. However, in the standard SMC samplers, only the last state obtained from the k-fold MCMC kernel will be used as the offspring of the current particle. Hence, the intermediate outputs of the k MCMC steps are seemingly wasted.

The waste-free SMC algorithm of Dau and Chopin (2020) focuses on deriving a way of using all the MCMC outputs within the SMC sampler, making the MCMC steps of the algorithm waste-free. By doing this, only N out of the NT particles from the previous SMC iteration are resampled. Then, each resampled particle will be moved $T - 1$ times through a chosen

MCMC kernel. These resampled particles together with all their intermediates outputs from the MCMC kernel are then gathered together to form a form sample of NT particles. Algorithm 5.1 shows the detail of the waste-free SMC algorithm proposed in Dau and Chopin (2020). The correctness of the algorithm was established in Dau and Chopin (2020) and numerical experiments have also shown its superior performance compared to the standard SMC samplers.

We saw the possibility of further improving the waste-free SMC algorithm based on two observations. First, the Markov kernel M_t involves an accept-reject step. This means that some of the proposed states may not be visited under the current scheme. Moreover, when an HMC kernel is applied, the intermediate states of the leapfrog integrators are still not used directly, incurring another form of 'waste'. In this chapter, we proposed a novel approach that, instead of only using the last state of the HMC trajectory, makes use of all the intermediate states of the HMC trajectory without any rejections. By properly weighting these 'particles', we are still able to approximate expectations with respect to the distribution of interest. This novel algorithm can be viewed as a standard SMC algorithm targeting the distributions of the HMC trajectories, hence treating the HMC trajectories as the particle. We will show later that our new algorithm represents a generalisation of the waste-free SMC sampler of Dau and Chopin (2020).

5.1.1 Overview and motivation

Assume interest is in sampling from a probability distribution μ on $(\mathsf{Z}, \mathscr{Z})$. In this chapter, we explore a framework where this problem is solved by embedding sampling from μ into that of sampling of Markov process snippets such that, properly weighted, the states of these snippets can be used to approximate expectations with respect to μ. We focus here primarily on Sequential Monte Carlo (SMC) algorithms, but the approach can be used in the context of Markov chain Monte Carlo algorithms as briefly explained in Corollary 5.6.1.2. As we shall see this presents several advantages and through numerical simulation, we establish the very good performance of the algorithms. This work has links with related recent attempts in the literature Rotskoff and Vanden-Eijnden (2019), Dau and Chopin (2020), Thin et al. (2021); these links and motivations are further discussed in Subsection 5.2.4. The chapter is organised

Algorithm 5.1: Waste-free SMC Algorithm

for *t=1,2,...,P* **do**
  if *t=1* **then**
    for $n = 1, ..NT$ **do**
      Sample $X_t^n \sim \pi_t(dx)$;
      Set $W_t^n = \frac{1}{NT}$;
  else
    Sample the ancestor indices $A_t^{1:N} \sim Multinomial(N, \mathbf{W}_{t-1})$;
    for $n = 1, .., N$ **do**
      Set $\tilde{X}_t^{m,1} \sim X_{t-1}^{A_t^m}$;
      for $p = 2, 3, .., T$ **do**
        Get $\tilde{X}_t^{m,p} \sim M_t(\tilde{X}_t^{m,p-1}, dx_t)$ where M_t is a Markov kernel of user's choice, e.g a random-walk Metropolis-Hastings kernel, that leaves π_{t-1} invariant ;
        Set $w_t^{m,p} := \frac{\pi_t(x_t^{m,p})}{\pi_{t-1}(x_t^{m,p})}$
    Gather $\tilde{X}_t^{m,p}$ for $m = 1, .., N$ and $p = 1, .., T$ to get $X_t^{1:NT}$;
    Set $W_t^n := \frac{w_t^n}{\sum_{j=1}^{NT} w_t^j}$

as follows. In Section 5.2 we provide a high-level description of the class of algorithms we propose. In Sections 5.4-5.5 we provide a theoretical background to show how Markov snippets can be used as intended above. Section 5.6 specialises our results to the scenario considered in Section 5.2.

5.1.2 Notation

We will write $\mathbb{N} = \{1, 2, \dots\}$ for the set of natural numbers, $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$, and $\mathbb{R}_+ = (0, \infty)$ for positive real numbers. Throughout we will be working on a general measurable space, generically denoted $(\mathsf{E}, \mathscr{E})$ in this subsection.

- For a set $A \in \mathscr{E}$, its complement in E is denoted by $A^{\complement}$. We denote the corresponding indicator function by $\mathbf{1}_A : \mathsf{E} \to \{0, 1\}$.
- For a μ a probability measure on $(\mathsf{E}, \mathscr{E})$ and a measurable function $f\colon \mathsf{E} \to \mathbb{R}$ and , we let $\mu(f) := \int f(x)\mu(\mathrm{d}x)$.
- For two probability measures μ and ν on $(\mathsf{E}, \mathscr{E})$ we let $\mu \otimes \nu\,(A \times B) = \mu\,(A)\,\nu\,(B)$ for $A, B \in \mathscr{E}$. For a Markov kernel $P\,(x, \mathrm{d}y)$ on $\mathsf{E} \times \mathscr{E}$, we write for $\bar{A} \in \mathscr{E} \otimes \mathscr{E}$, the minimal product σ-algebra, $\mu \otimes P\big(\bar{A}\big) = \int_{\bar{A}} \mu\,(\mathrm{d}x)\,P\,(x, \mathrm{d}y)$.

- A point mass distribution at x will be denoted by $\delta_x(\mathrm{d}y)$.

5.2 A simple example

Assume interest is in sampling from a probability distribution π on $(\mathsf{X}, \mathscr{X})$ and that to achieve this we use an SMC (Sequential Monte Carlo) sampler Del Moral et al. (2006) relying on HMC (Hybrid Monte Carlo), that is Metropolis-Hastings (MH) updates using a discretisation of Hamilton's equations Duane et al. (1987). More specifically define $\mu_n(\mathrm{d}z) := \pi_n(\mathrm{d}x)\varpi_n(\mathrm{d}v)$ for $n \in [\![0, P]\!]$ a sequence of distributions on $(\mathsf{Z}, \mathscr{Z}) = (\mathsf{X} \times \mathsf{V}, \mathscr{X} \otimes \mathscr{V})$ with $\pi_P = \pi$ and ϖ_n on $(\mathsf{V}, \mathscr{V})$, and consider mappings $\psi_n \colon \mathsf{Z} \to \mathsf{Z}$ with the property that $\psi_n^{-1} = \sigma_n \circ \psi_n \circ \sigma_n$ for some mapping $\sigma_n \colon \mathsf{Z} \to \mathsf{Z}$ such that $\sigma_n^2 = \mathrm{Id}$ and $\mu_n^{\sigma_n} = \mu_n$, where for a probability distribution ν on $(\mathsf{E}, \mathscr{E})$ and a suitable mapping $\phi \colon \mathsf{E} \to \mathsf{E}$, $\nu^\phi(A) = \nu(\phi^{-1}(A))$ for any $A \in \mathscr{E}$, that is ν^ϕ is the push-forward of ν under ϕ. We assume the existence of a density for μ_n with respect to some measure υ, denoted $\mu_n(z) =:= \mathrm{d}\mu_n/\mathrm{d}\upsilon(z)$ for $z \in \mathsf{Z}$.

Example 5.2.1. *The Stormer–Verlet integrator of Hamilton's equations, $\dot{x}_t = v_t, v_t = -\nabla U_n(x_t)$ for the potential $H_n(x, v) = U_n(x) + \frac{1}{2}|v|^2$ where $U_n \colon \mathsf{X} \to \mathbb{R}$, is given by $\psi_n(x, v) = \psi_n^B \circ \psi_n^A \circ \psi_n^B(x, v)$ where for $\varepsilon > 0$, $\psi_n^A(x, v) := (x + \varepsilon/2v, v)$ and $\psi_B(x, v) = (x, v - \varepsilon\nabla U_n(x))$ satisfies the conditions set above for a spherically symmetric distribution ϖ_n, with $\sigma_n \circ f(x, v) = f(x, -v)$ for any $f \colon \mathsf{Z} \to \mathsf{Z}$. When $U_n(x)$ is taken to be constant we have $\psi_n(x, v) = (x + \varepsilon v, v)$, corresponding to the guided Random walk of Gustafson (1998). One could as well use the leapfrog integrator for Hamilton's equations.*

5.2.1 Algorithm outline: unfolded perspective

For $n \in [\![0, P]\!]$, $k \in \mathbb{N}$ and $z \in \mathsf{Z}$ let

$$w_{n,k}(z) := \frac{\mu_n \circ \psi_n^k(z)}{\mu_n(z)},$$

with the understanding that $\psi_n^0 = \mathrm{Id}$ and hence $w_0(z) := w_{0,0}(z) = 1$, which we recognise to be the acceptance ratio of the HMC update for k integration steps. The primary interest in this paper is in SMC sampler algorithms as given in Algorithm 5.2. Let $N, T \in \mathbb{N} \setminus \{0\}$ and assume that sampling from μ_0 is possible.

Algorithm 5.2: Unfolded Hamiltonian Snippet SMC algorithm

Sample $z_0^{(i)} \overset{\text{iid}}{\sim} \mu_0$ for $i \in [\![N]\!]$;
for $n = 1, \ldots, P$ **do**
  for $(i,k) \in [\![N]\!] \times [\![0,T]\!]$ **do**
    Compute $\left(z_{n-1,k}^{(i)} = (x_{n-1,k}^{(i)}, v_{n-1,k}^{(i)}) := \psi_{n-1}^k(z_{n-1}^{(i)}), w_{n-1,k}(z_{n-1}^{(i)})\right)$;
  for $i \in [\![N]\!]$ **do**
    From the $N \times (T+1)$ weighted sample $\{z_{n-1,k}^{(i)}, w_{n-1,k}(z_{n-1}^{(i)}), (i,k) \in [\![N]\!] \times [\![0,T]\!]\}$, resample N particles, $\{\bar{z}_n^{(i)} := (\bar{x}_{n-1}^{(i)}, \bar{v}_{n-1}^{(i)}), i \in [\![N]\!]\}$, using the weights

$$\frac{\mu_n(z_{n-1,k}^{(i)})}{\mu_{n-1}(z_{n-1,k}^{(i)})} w_{n-1,k}(z_{n-1}^{(i)}), (i,k) \in [\![N]\!] \times [\![0,T]\!],$$

    Rejuvenate the velocities $z_n^{(i)} = (\bar{x}_{n-1}^{(i)}, v_n^{(i)})$ with $v_n^{(i)} \sim \varpi_n$.

The SMC sampler in Algorithm 5.2 therefore consists of N "seed" particles, the mutation mechanism used consists of generating N process snippets started at every seed particle $z \in \mathsf{Z}$, $\mathsf{z} := (z, \psi_{n-1}(z), \psi_{n-1}^2(z), \ldots, \psi_{n-1}^T(z))$, resulting in $N \times (T+1)$ particles which are then whittled down to a set of N seed particles using a standard resampling scheme; this yields the next generation of seed particles. This SMC sampler should be contrasted with standard implementations of SMC samplers where, after resampling, a seed particle normally gives rise to a unique particle in the mutation step. This procedure shares some similarities with Dau and Chopin (2020) but differs in that we here sample and use the full trajectory of length $T+1$ whereas direct use of their idea using a standard HMC (Metropolis-Hastings), assuming no velocity refreshment along the trajectory, would lead to the exploration of a random number of states of this trajectory due to the accept/reject mechanism involved. Simulations demonstrating the interest of this approach are given in subsection 5.2.5.

5.2.2 Outline of the justification of the algorithm

We now outline the main ideas underpinning the theoretical justification of this algorithm and present variations and extensions of these ideas derived from the general results of Sections 5.4, 5.5 and 5.6. Key to this is establishing that Algorithm 5.2 is a standard SMC sampler targeting a sequence of probability distributions $\{\bar{\mu}_n, n \in [\![0,P]\!]\}$ from which samples can be processed to approximate expectations with respect to $\{\mu_n, n \in [\![0,P]\!]\}$. This is touched on

in Dau and Chopin (2020), but we here provide full details and show how these ideas can be pushed much further, in interesting directions.

First define the probability distributions on $\left([\![0,T]\!] \times \mathsf{Z}, \mathscr{P}([\![0,T]\!]) \otimes \mathscr{Z}\right)$

$$\bar{\mu}_n(k, \mathrm{d}z) = \frac{1}{T+1} w_{n,k}(z) \mu_n(\mathrm{d}z)\,,$$

for $n \in [\![0,P]\!]$. We will show that Algorithm 5.2 can be interpreted as an SMC sampler targeting the sequence of marginal distributions on $\left(\mathsf{Z}, \mathscr{Z}\right)$

$$\bar{\mu}_n(\mathrm{d}z) = \mu_n(\mathrm{d}z) w_n(z) \text{ with } w_n(z) := \frac{1}{T+1} \sum_{k=0}^{T} w_{n,k}(z)\,,$$

which we may refer as a mixture, for $n \in [\![0,P]\!]$. It may appear at first sight that sampling from $\bar{\mu}_n$ is unsuitable to estimate expectations with respect μ_n. However the mixture structure of $\bar{\mu}_n$ can be exploited, using standard arguments. More precisely, from Lemma 5.4.1, one has for $f : \mathsf{Z} \to \mathsf{Z}$

$$\sum_{k=0}^{T} \frac{1}{T+1} \int f \circ \psi_n^k(z) w_{n,k}(z) \bar{\mu}_n(\mathrm{d}z) = \mu_n(f)\,, \tag{5.1}$$

which implies that samples from the mixture $\bar{\mu}_n$ can be used to unbiasedly estimate $\mu_n(f)$, thanks to a weighted average.

We now turn to the description of an SMC algorithm targeting $\left\{\bar{\mu}_n, n \in [\![0,P[\![\right\}$, Algorithm 5.3, and then establish that it is probabilistically equivalent to Algorithm5.2 in a sense made precise below. Let $\mathsf{z} = (z_0, z_1, \ldots, z_T) := \left(z_0, \psi_n(z_0), \ldots, \psi_n^T(z_0)\right) \in \mathsf{Z}^{(T+1)}$, $z_k = (x_x, v_k)$ for $k \in [\![0,T]\!]$, then for $n \in [\![P]\!]$ we introduce the following mutation kernel

$$\bar{M}_n(z, \mathrm{d}z') := \sum_{k=0}^{T} \bar{\mu}_{n-1}(k \mid z)(\delta_{x_k} \otimes \varpi_{n-1})(\mathrm{d}z'_0) \bar{M}_n^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right),$$

where we let $\mathsf{z}_{-0} := (z_1, \ldots, z_T)$ and

$$\bar{M}_n^{\otimes T}\left(z_0, \mathrm{d}\mathsf{z}_{-0}\right) := \prod_{k=1}^{T} M_n(z_{k-1}, \mathrm{d}z_k)\,.$$

We note that $\bar{\mu}_{n-1}(k \mid z) \propto w_{n-1,k}(z)$. It can be shown (Lemma 5.4.3) that for the near-optimal backward kernel

$$\bar{L}_n(z'\mathrm{d}z) = \frac{\bar{\mu}_n \otimes \bar{M}_n\big(\mathrm{d}(z,z')\big)}{\bar{\mu}_n \bar{M}_n(\mathrm{d}z')},$$

the step n importance weights are of the form

$$\bar{w}_n(z) = \frac{\bar{\mu}_n \otimes \bar{L}_n\big(\mathrm{d}(z',z)\big)}{\bar{\mu}_{n-1} \otimes \bar{M}_n\big(\mathrm{d}(z,\mathrm{d}z')\big)} = \frac{\mu_n(z)}{\mu_{n-1}(z)} \frac{1}{T+1} \sum_{k=0}^{T} w_{n-1,k}(z) = \frac{\mu_n(z)}{\mu_{n-1}(z)} w_{n-1}(z)\,. \tag{5.2}$$

Note the slight abuse of notation above, since $\bar{L}_n$ is not to L_n what $\bar{M}_n$ is to M_n. The standard SMC sampler corresponding to these choices is given in Algorithm 5.2; note that the weights $w_{n,k}$ appear as computed twice, for the only reason that it facilitates here the explanation of some of our arguments.

Algorithm 5.3: Folded Hamiltonian Snippet SMC algorithm

Sample $\check{z}_0^{(i)} \overset{\text{iid}}{\sim} \bar{\mu}_0 = \mu_0$ for $i \in [\![N]\!]$;
for $n = 1, \ldots, P$ **do**
  for $i \in [\![N]\!]$ **do**
    for $k \in [\![0, T]\!]$ **do**
      Compute $\left(\check{z}_{n-1,k}^{(i)} = (\check{x}_{n-1,k}^{(i)}, \check{v}_{n-1,k}^{(i)}) := \psi_{n-1}^k(\check{z}_{n-1}^{(i)}), w_{n-1,k}\big(\check{z}_{n-1}^{(i)}\big)\right)$;
    Draw $a_i \sim \mathrm{Cat}\left(1, w_{n-1,1}\big(\check{z}_{n-1}^{(i)}\big), w_{n-1,2}\big(\check{z}_{n-1}^{(i)}\big), \ldots, w_{n-1,T}\big(\check{z}_{n-1}^{(i)}\big)\right)$;
    Set $\tilde{z}_n^{(i)} = (\check{x}_{n-1,a_i}^{(i)}, \tilde{v}_n^{(i)})$ with $\tilde{v}_n^{(i)} \sim \varpi_n$;
  Resample N particles from $\big\{\tilde{z}_n^{(i)}, i \in [\![N]\!]\big\}$, using a multinomial distribution of parameter

$$\bar{w}_{n+1}\big(\tilde{z}_n^{(i)}\big) = \frac{\mu_{n+1}\big(\tilde{z}_n^{(i)}\big)}{\mu_n\big(\tilde{z}_n^{(i)}\big)} w_n\big(\tilde{z}_n^{(i)}\big), \quad i \in [\![N]\!]\,,$$

  and obtain $\check{z}_n^{(i)}$ for $i \in [\![N]\!]$;

We now provide the simple probabilistic argument justifying the alternative presentation in Algorithm 5.2. We show that for $n \in [\![P]\!]$ the distribution of $\big\{\tilde{z}_n^{(i)}, i \in [\![N]\!]\big\}$ given $\big\{\tilde{z}_{n-1}^{(i)}, i \in [\![N]\!]\big\}$ in Algorithm 5.2 is probabilistically equivalent to that of $\big\{z_n^{(i)}, i \in [\![N]\!]\big\}$ given $\big\{z_{n-1}^{(i)}, i \in [\![N]\!]\big\}$ in Algorithm 5.3. Using conditional independence we focus on the probability, in Algorithm 5.3, that for $j \in [\![N]\!]$, $\tilde{x}_n^{(j)}$ is obtained as the first component of $\psi_{n-1}^k\big(\tilde{z}_{n-1}^{(i)}\big)$ for some $(i,k) \in [\![N]\!] \times [\![0,T]\!]$, that is it has $\tilde{z}_{n-1}^{(i)}$ as the ancestor and is obtained

after k iterations of ψ_{n-1}, given $\{\tilde{z}_{n-1}^{(m)}, m \in [\![N]\!]\}$. Ignoring the distribution of $\tilde{v}_n^{(j)}$ we have

$$\begin{aligned}\mathbb{P}\big(\tilde{x}_n^{(j)} = \check{x}_{n-1,k}^{(j)}, \check{z}_{n-1}^{(j)} = \tilde{z}_{n-1}^{(i)} \mid \tilde{z}_{n-1}^{(m)}, m \in [\![N]\!]\big) &= \frac{\mu_n\big(\tilde{z}_{n-1}^{(i)}\big)}{\mu_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big)} w_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big) \times \frac{w_{n-1,k}\big(\tilde{z}_{n-1}^{(j)}\big)}{w_{n-1}\big(\tilde{z}_{n-1}^{(j)}\big)} \\ &= \frac{\mu_n\big(\tilde{z}_{n-1}^{(i)}\big)}{\mu_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big)} w_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big) \times \frac{w_{n-1,k}\big(\tilde{z}_{n-1}^{(i)}\big)}{w_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big)} \\ &= \frac{\mu_n\big(\tilde{z}_{n-1}^{(i)}\big)}{\mu_{n-1}\big(\tilde{z}_{n-1}^{(i)}\big)} w_{n-1,k}\big(\tilde{z}_{n-1}^{(i)}\big)\,.\end{aligned}$$

Now observe that this expression coincides with that of the conditional (including $v_n^{(j)}$) distribution of $z_n^{(j)} = \psi_{n-1}^k(z_{n-1}^{(i)})$ in Algorithm 5.2. In other words for $n \in [\![P]\!]$, $\{\tilde{z}_n^{(i)}, i \in [\![N]\!]\}$ and $\{z_n^{(i)}, i \in [\![N]\!]\}$ are distributionally equivalent for $n \in [\![P]\!]$ given particles at $n = 0$, particles which share the same initial distribution. Since the justification of the latter interpretation of the algorithm is straightforward, as a standard SMC sampler targeting instrumental distributions $\{\bar{\mu}_n, n \in [\![0, P[\![\}$, we will adopt this perspective in the remainder of the chapter for simplicity. We remind the reader that (5.1) provides us with a way of using all the values of $z_{k,n}$ to estimate expectations w.r.t. μ_n.

5.2.3 Straightforward generalisations

It should be clear that the algorithm we have described lends itself to numerous generalisations, which we briefly list here.

- each seed particle can give rise to multiple Markov chain snippets, which may be of interest to parallel machines,
- for $n \in [\![0, P]\!]$ the mappings $\{\psi_n^k, k \in [\![0, T]\!]\}$ for a given $\psi_n \colon \mathsf{Z} \to \mathbb{R}$ can be replaced with a family of invertible mappings $\{\psi_{n,k} \colon \mathsf{Z} \to \mathbb{R}, k \in [\![0, T]\!]\}$ where the $\psi_{n,k}s$ are now not required to be measure preserving, in which case the expression for $w_{n,k}$ may involve some form of "Jacobian", that is

$$w_{n,k}(z) := \frac{\mu_n \circ \psi_{n,k}(z)}{\mu_n(z)} \frac{\mathrm{d}\upsilon^{\psi_{n,k}^{-1}}}{\mathrm{d}\upsilon}(z)\,.$$

This is because, for $f : \mathsf{Z} \to \mathbb{R}$ bounded and measurable, on the one hand

$$\begin{aligned}\int f \circ \psi(z) \frac{\mathrm{d}\mu^{\psi^{-1}}}{\mathrm{d}\mu}(z)\mu(\mathrm{d}z) &= \int f \circ \psi(z)\mu^{\psi^{-1}}(\mathrm{d}z) \\ &= \int f(z)\mu(\mathrm{d}z)\,,\end{aligned}$$

and on the other hand

$$\begin{aligned}\int f(z)\frac{\mu \circ \psi(z)}{\mu(z)}(z)\frac{\mathrm{d}\upsilon^{\psi^{-1}}}{\mathrm{d}\upsilon}(z)\mu(z)\upsilon(\mathrm{d}z) &= \int f(z)\mu \circ \psi(z)\frac{\mathrm{d}\upsilon^{\psi^{-1}}}{\mathrm{d}\upsilon}(z)\upsilon(\mathrm{d}z) \\ &= \int f(z)\mu \circ \psi(z)\upsilon^{\psi^{-1}}(\mathrm{d}z) \\ &= \int f \circ \psi^{-1}(z)\frac{\mathrm{d}\mu}{\mathrm{d}\upsilon}(z)\upsilon(\mathrm{d}z) \\ &= \int f \circ \psi^{-1}(z)\mu(\mathrm{d}z) \\ &= \int f(z)\mu^{\psi^{-1}}(\mathrm{d}z) \\ &= \int f(z)\frac{\mathrm{d}\mu^{\psi^{-1}}}{\mathrm{d}\mu}(z)\mu(\mathrm{d}z)\,.\end{aligned}$$

- it is possible to generalise the definition of $\bar{\mu}_n$ and give different weights to the seed particles

$$\bar{\mu}_n(k, \mathrm{d}z) = \mu_n(\mathrm{d}z)\sum_{k=0}^{T}\omega_{n,k}\, w_{n,k}(z)\,,$$

with $\sum_{k=0}^{T}\omega_{n,k} = 1$ and $\omega_{n,k} \geq 0$ for $k \in [\![0, T]\!]$.

- one could use continuous time processes snippets. For example piecewise deterministic Markov processes such as the Zig-Zag process Bierkens and Roberts (2017) or the Bouncy Particle Sampler Bouchard-Côté et al. (2018) could be used since finite time horizon trajectories can be parametrized in terms of a finite number of parameters.

We leave the exploration of such generalisations for future work, although we think that the choice of the leapfrog integrator is particularly natural and attractive here.

5.2.4 Computational considerations, motivations and theoretical support

- It is worth pointing out that the computation of $w_{n,k}$ and $z_{n,k}$ will most often involve common quantities, leading to negligible computational overhead stemming from the

use of a leapfrog integrator when such a trajectory-based strategy is used within SMC. This was exploited and observed in our numerical examples.

- The motivation for using an integrator of Hamilton's equations is similar to that of HMC. Let $d \in \mathbb{N}$ and let $\mathsf{X} = \mathbb{R}^d$. We know that in certain scenarios Beskos et al. (2013), Calvo et al. (2021), the distributions $\{\mu_{n,d}, d \in \mathbb{N}, n \in [\![T(d)]\!]\}$ as such that for $n \in \mathbb{N}$, $\log \frac{\mu_{n,d} \circ \psi^k_{n,d}(z)}{\mu_{n,d}(z)} \rightarrow_{d\to\infty} \mathcal{N}(\mu_n, \sigma_n^2)$, that is the weights do not degenerate to zero or one: in the context of SMC this means that the part of the importance weight (5.2) arising from the mutation mechanism does not degenerate. Further, an appropriate choice of schedule, i.e. sequence $\{\mu_{n,d}, n \in [\![T(d)]\!]\}$ for $d \in \mathbb{N}$, ensures that the contribution $\frac{\mu_{n,d}(z)}{\mu_{n-1,d}(z)}$ to the importance weights (5.2) is also stable as $d \to \infty$. As shown in Beskos et al. (2013), Calvo et al. (2021), while direct importance sampling an exponential number of samples as d grows, the use of such a schedule reduces this complexity to a polynomial order.

- The scheme we propose differs from an approach which would use a standard HMC update with the waste-free algorithm of Dau and Chopin (2020). Indeed such an approach is not guaranteed to explore all the states $\{z_{n,k}, k \in [\![T]\!]\}$, but a random number of states, based on the accept-reject mechanism of the Metropolis-Hastings mechanism. In contrast, our scheme explores all the states, and as can be seen below can be easily extended to accommodate, for example, constraints.

- Averaging can be advantageous in two respects. We know that with $H_n(z) := -\log \mu_n(z)$, then for $z \in \mathsf{Z}$ the mapping $k \mapsto H_n \circ \psi^k_n(z)$ is typically oscillatory, motivating for example the x-tra chance algorithm of Campos and Sanz-Serna (2015). We also note that in the limit as $\varepsilon \to 0$ while $\varepsilon \times T$ is kept constant, the average in (5.2) corresponds to the numerical integration of the contour of $H_n(z)$, effectively leading to some form of Rao-Blackwellization of this contour.

- Our work is at first sight linked to Rotskoff and Vanden-Eijnden (2019), Thin et al. (2021), but differs in the details. Indeed, in the discrete-time setup considered in Rotskoff and Vanden-Eijnden (2019) the mixture introduced is, for a sequence $\{\omega_k \geq$

$0, k \in \mathbb{Z}\}$ such that $\#\{\omega_k \neq 0, k \in \mathbb{Z}\} < \infty$ and $\omega := \sum_{l\in\mathbb{Z}} \omega_k = 1$,

$$\bar{\mu}(\mathrm{d}z) = \mu(\mathrm{d}z) \sum_{k\in\mathbb{Z}} w_k(z)\,,$$

with (assuming volume preservation)

$$w_k(z) = \omega_k \frac{\mu \circ \psi^k(z)}{\sum_{l\in\mathbb{Z}} \omega_{k+l}\, \mu \circ \psi^l(z)}\,,$$

which is used as an importance sampling proposal in order to estimate expectations with respect to μ. However, the way this mixture is used is completely different since NEO relies on samples from μ, typically exact samples, which then undergo a transformation and are subsequently properly weighted, while we aim to sample from $\bar{\mu}$, typically using an iterative method. One could naturally apply the ideas developed throughout this paper to this particular choice of instrumental target distributions.

- We are currently considering adaptation schemes to tune the parameters ε and T of the Hamiltonian integrator, taking advantage of the population of samples, in the spirit of Hoffman et al. (2021), Hoffman and Sountsov (2022).

5.2.5 Numerical illustration: logistic regression

In this section, we consider sampling from the posterior distribution of a logistic regression model, focusing on the computation of the normalising constant. We follow Dau and Chopin (2020) and consider the sonar dataset, previously used in Chopin and Ridgway (2017). With intercept terms, the dataset has responses $y_i \in \{-1, 1\}$ and covariates $z_i \in \mathbb{R}^p$, where $p = 61$. The likelihood of the parameters $x \in \mathsf{X} := \mathbb{R}^p$ is then given by

$$L(x) = \prod_i^n F(z_i^\top x \cdot y_i), \tag{5.3}$$

where $F(x) := 1/(1+\exp(-x))$. We ascribe x a product of independent normal distributions of zero mean as a prior, with standard deviation equal to 20 for the intercept and 5 for the other parameters. Denote $p(dx)$ the prior distribution of x, we define a sequence of tempered distributions of the form $\pi_n(x) \propto p(dx)L(x)^{\lambda_n}$ for $\lambda_n \colon [\![0, P]\!] \to [0, 1]$ non-decreasing and such

that $\lambda_0 = 0$ and $\lambda_P = 1$. We apply both Hamiltonian Snippet-SMC and the implementation of waste-free SMC of Dau and Chopin (2020) and compare their performance.

For both algorithms, we set the total number of particles at each SMC step to be $N(T+1) = 10,000$. For the waste-free SMC, the Markov kernel is chosen to be a random-walk Metropolis-Hastings kernel with covariances adaptively computed as $2.38^2/d\ \hat{\Sigma}$, where $\hat{\Sigma}$ is the empirical covariance matrix obtained from the particles in the previous SMC step. For the Hamiltonian Snippet-SMC algorithm, we set ψ_n to be the one-step leap-frog integrator with stepsize ε, $U_n(x) = -\log(\pi_n(x))$ and ϖ a $\mathcal{N}(0, \mathrm{Id})$. To investigate the stability of our algorithm, we ran Hamiltonian Snippet SMC with $\varepsilon = 0.05, 0.1, 0.2$ and 0.3. For both algorithms, the temperatures λ_n are adaptively chosen so that the effective sample size (ESS) of the current SMC step will be αESS_{max}, where ESS_{max} is the maximum ESS achievable at the current step. In our experiments, we have chosen $\alpha = 0.3, 0.5$ and 0.7 for both algorithms.

Performance comparison

Figure 5.1 shows the boxplots of estimates of the logarithm of the normalising constant obtained from both algorithms, for different choices of N and ε for the Hamiltonian Snippet SMC algorithm. The boxplots are obtained by running both algorithms 100 times for different of algorithm parameters, with $\alpha = 0.5$ in all setups. Several points are worth observing. For a suitably choice of ε, the Hamiltonian Snippet SMC algorithm can produce stable and consistent estimates of the normalising constant with $10,000$ particles at each iteration. On the other hand, however, the waste-free SMC algorithm fails to produce accurate results for the same computational budget. It is also clear that with larger values of N (meaning the smaller value of T and hence shorter snippets), the waste-free SMC algorithm produces results with larger biases and variability. For the Hamiltonian Snippet SMC algorithm, the results are stable both for short and long snippets when ε is equal to 0.1 or 0.2. Another point is that when $\varepsilon = 0.05$ or 0.3, the Hamiltonian Snippet SMC algorithm becomes unstable with short (i.e. $\varepsilon = 0.05$) and long (i.e. $\varepsilon = 0.3$). Possible reasons are for too small a stepsize the algorithm is not able to explore the target distribution efficiently, resulting in unstable performances. On the other hand, when the stepsize is too large, the leapfrog integrator becomes unstable, therefore affecting the variability of the weights; this is a common problem

of HMC algorithms. Hence, a long trajectory will result in deteriorate estimations. Hence, to obtain the best performance, one should find a way of tuning the stepsize and trajectory length along with the SMC steps.

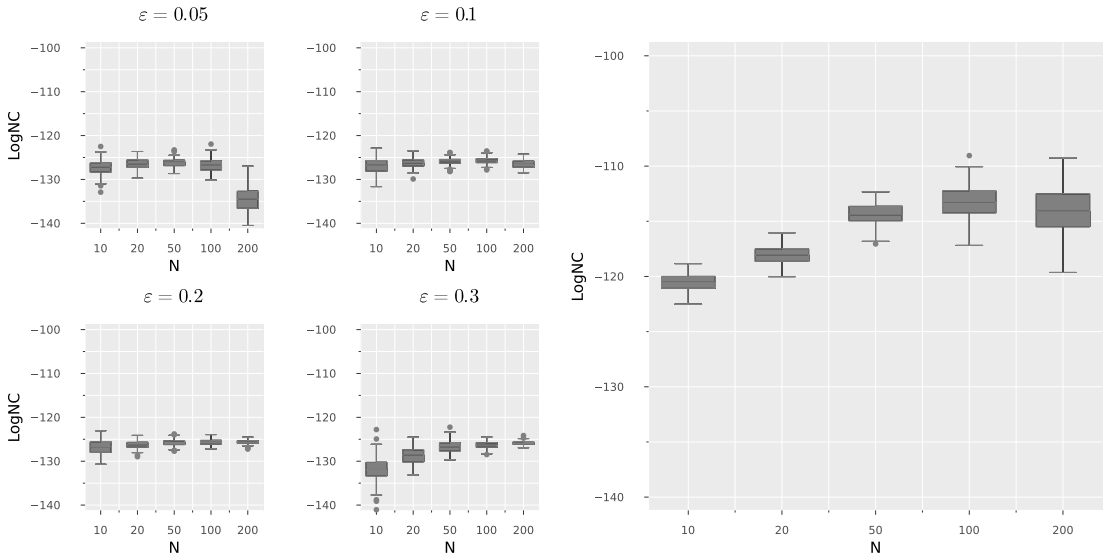

Figure 5.1: Estimates of the normalising constant (in log scale) obtained from both Hamiltonian Snippet SMC and waste-free SMC algorithm. Left: Estimate obtained from Hamiltonian Snippet SMC algorithm with different values of ε. Right: Estimates obtained from waste-free SMC algorithm.

In Figure 5.2 we display boxplots of the estimates of the posterior expectations of the mean of all coefficients, i.e. $\mathbb{E}_{\pi_P}(d^{-1}\sum_{i=1}^{d} x_i)$. This quantity is referred to as the *mean of marginals* in Dau and Chopin (2020) and we use this terminology. One can see that the same properties can be seen from the estimations of the mean of marginals, with the instability problems exacerbated with small and large stepsizes.

Computational Cost

In this section, we compare the running time of both algorithms. Since the calculations of the potential energy and its gradient often share common intermediate steps, we can recycle these to save computational costs. As the waste-free SMC also requires density evaluations, the Hamiltonian Snippet SMC algorithm will not require significant additional computations. Figure 5.3 shows boxplots of the simulation time of both algorithms from 100 runs. The simulations were run on an Apple M1-Pro CPU with 16G of memory. One can see that in comparison to the waste-free SMC the additional computational time is only marginal for the

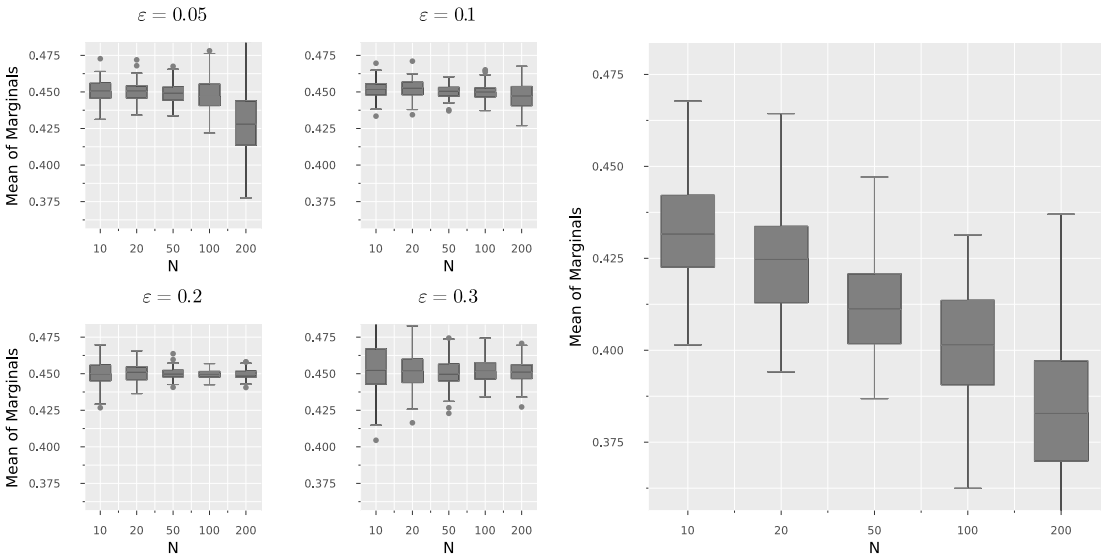

Figure 5.2: Estimates of the mean of marginals obtained from both Hamiltonian Snippet SMC and the waste-free SMC algorithms. Left: Estimate obtained from the Hamiltonian Snippet SMC algorithm with different values of ε. Right: Estimates obtained from the waste-free SMC algorithm.

Hamiltonian Snippet SMC algorithm and mostly due to the larger memory needed to store the intermediate values.

5.3 Auxiliary process proposal and more general examples

In this section, we establish a set of general results which make finding the expression for the importance weights or acceptance ratios given in this paper rather simple. The main idea is that in the situation where μ is dominated by a measure υ, which together with M and another kernel M^*form a (υ, M, M^*) triplet, the weight or acceptance ratio involved is straightforward to obtain. While this may appear an overkill in simple scenarios, we will see in Example 5.3.2 that it makes establishing the correctness of a more sophisticated algorithm straightforward.

5.3.1 A general framework

Lemma 5.3.1. *Let μ be a probability distribution on $(\mathsf{Z}, \mathscr{Z})$, υ be a measure on $(\mathsf{Z}, \mathscr{Z})$ such that $\upsilon \gg \mu$ and assume that we have a pair or Markov kernels $M, M^* \colon \mathsf{Z} \times \mathscr{Z} \to [0, 1]$ such*

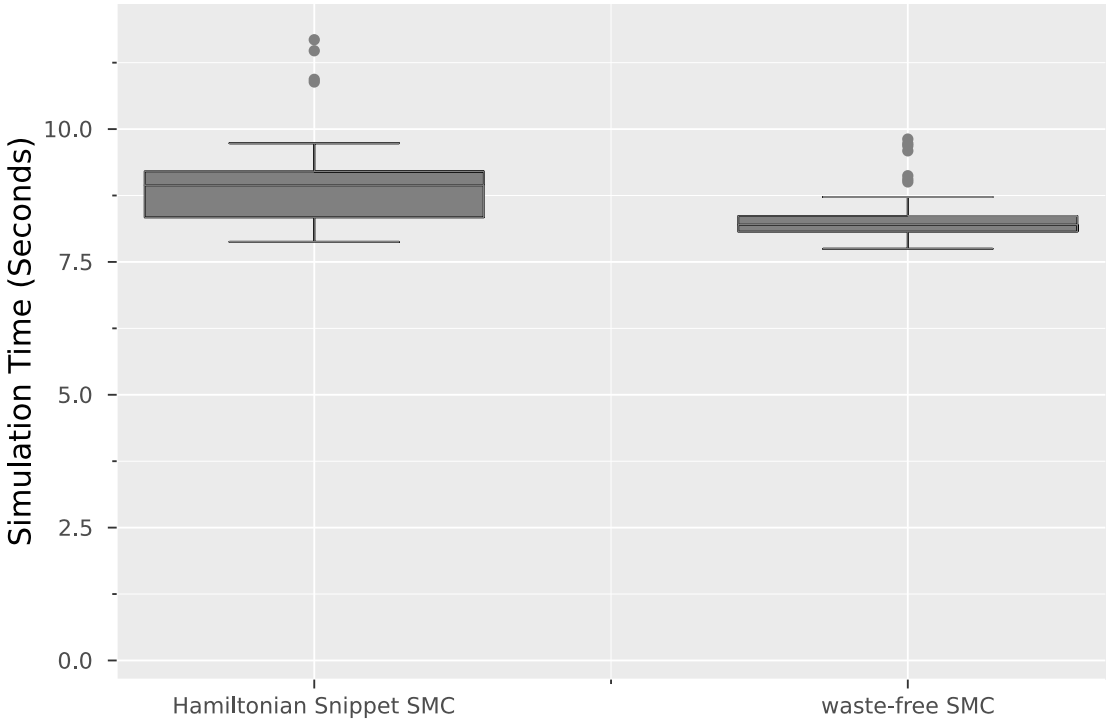

Figure 5.3: Simulation times for both algorithms. Left: Simulation time for Hamiltonian Snippet SMC algorithm, with $N = 100, \varepsilon = 0.2, \alpha = 0.5$. Right:Simulation times for the waste-free SMC with $N = 100$

that

$$\upsilon(\mathrm{d}z)M(z,\mathrm{d}z') = \upsilon(\mathrm{d}z')M^*(z',\mathrm{d}z)\,.$$

We call this property (υ, M, M^*) *reversibility. Then for* $z, z' \in$ Z *such that* $\mu(z) := \mathrm{d}\mu/\mathrm{d}\upsilon(z) > 0$ *we have*

$$\frac{\mu \otimes M^*\big(\mathrm{d}(z',z)\big)}{\mu \otimes M\big(\mathrm{d}(z,z')\big)} = \frac{\mathrm{d}\mu/\mathrm{d}\upsilon(z')}{\mathrm{d}\mu/\mathrm{d}\upsilon(z)} = \frac{\mu(z')}{\mu(z)}\,.$$

Proof. For $z, z' \in$ Z such that $\mathrm{d}\mu/\mathrm{d}\upsilon(z) > 0$ we have

$$\begin{aligned}
\mu(\mathrm{d}z')M^*(z',\mathrm{d}z) &= \frac{\mathrm{d}\mu}{\mathrm{d}\nu}(z')\upsilon(\mathrm{d}z')M^*(z',\mathrm{d}z) \\
&= \frac{\mathrm{d}\mu}{\mathrm{d}\nu}(z')\upsilon(\mathrm{d}z)M(z,\mathrm{d}z') \\
&= \frac{\mathrm{d}\mu/\mathrm{d}\upsilon(z')}{\mathrm{d}\mu/\mathrm{d}\upsilon(z)}\frac{\mathrm{d}\mu}{\mathrm{d}\nu}(z)\upsilon(\mathrm{d}z)M(z,\mathrm{d}z') \\
&= \frac{\mathrm{d}\mu/\mathrm{d}\upsilon(z')}{\mathrm{d}\mu/\mathrm{d}\upsilon(z)}\mu(\mathrm{d}z)M(z,\mathrm{d}z')\,.
\end{aligned}$$

□

Corollary 5.3.1.1. *With μ, υ, M and $L := M^*$ as above, for any $z, z' \in \mathsf{Z}$ such that $\mathrm{d}\mu/\mathrm{d}\upsilon(z) > 0$ and $k \in \mathbb{N}$*

$$\frac{\mu(\mathrm{d}z')L^k(z', \mathrm{d}z)}{\mu(\mathrm{d}z)M^k(z, \mathrm{d}z')} = \frac{\mathrm{d}\mu/\mathrm{d}\upsilon(z')}{\mathrm{d}\mu/\mathrm{d}\upsilon(z)}.$$

Example 5.3.1. *Let $\upsilon \gg \mu$, so that $\mu(z) := \mathrm{d}\mu/\mathrm{d}\upsilon(z)$ is well defined. Let, for $i \in \{1, 2\}$, $\psi_i \colon \mathsf{Z} \to \mathsf{Z}$ be invertible and such that $\upsilon^{\psi_i} = \upsilon$, $\psi_i^{-1} = \sigma \circ \psi_i \circ \sigma$ for $\sigma \colon \mathsf{Z} \to \mathsf{Z}$ such that $\sigma^2 = \mathrm{Id}$ and $\upsilon^\sigma = \upsilon$ then consider the delayed rejection MH transition probability*

$$M(z, \mathrm{d}z') = \alpha_1(z)\delta_{\psi_1(z)}(\mathrm{d}z') + \bar{\alpha}_1(z)\left[\alpha_2(z)\delta_{\psi_2(z)}(\mathrm{d}z') + \bar{\alpha}_2(z)\delta_z(\mathrm{d}z')\right].$$

In the standard HMC scenario υ is the Lebesgue measure on $\mathsf{X} \times \mathsf{V}$ and ψ_i is of the Stormer-Verlet type. Following Andrieu et al. (2020) notice that υ has density $v(z) = 1/2$ with respect to the measure $\upsilon + \upsilon^{\psi_i} = 2\upsilon$. Now define $S_1 := \{z \in \mathsf{Z} \colon v(z) \wedge v \circ \psi_1(z) > 0\}$, $\alpha_1(z) = 1 \wedge r_1(z)$ and $\bar{\alpha}_1(z) := 1 - \alpha_1(z)$ with

$$r_1(z) := \begin{cases} \frac{v \circ \psi_1(z)}{v(z)} = 1 & \text{for } z \in S_1 \\ 0 & \text{otherwise} \end{cases},$$

and with $S_2 := \{z \in \mathsf{Z} \colon [\bar{\alpha}_1(z)\, v(z)] \wedge [\bar{\alpha}_1 \circ \psi_2(z)\, v \circ \psi_2(z)] > 0\}$

$$r_2(z) := \begin{cases} \frac{\bar{\alpha}_1 \circ \psi_2(z)\, v \circ \psi_2(z)}{\bar{\alpha}_1(z)\, v(z)} = 1 & \text{for } z \in S_2 \\ 0 & \text{otherwise} \end{cases}.$$

ψ_1 can be a HMC update, while

$$\psi_2 = \psi_1 \circ \mathrm{b} \circ \psi_1 \text{ with } \mathrm{b}(x, v) = \left(x, v - 2\langle v, n(x)\rangle n(x)\right) \tag{5.4}$$

for some vector field $n \colon \mathsf{X} \to \mathsf{X}$. This can therefore be used as part of Algorithm 5.5; care must be taken when compute the weights $w_{n,k}$ and $\bar{w}_n$, see Section 5.4-5.6 provide the tools to achieve this. For example, in the situation where υ is the Lebesgue measure on $\mathsf{Z} = \mathbb{R}^d$ and $\psi_2 = \mathrm{Id}$ then we recover Algorithm 5.3 or equivalently Algorithm 5.2.

Example 5.3.2. *An interesting instance of Example 5.3.1 is concerned with the scenario where interest is in sampling μ constrained to some set $C \subset \mathsf{Z}$ such that $\upsilon(C) < \infty$. Define $\mu_C(\cdot) := \mu(C \cap \cdot)/\mu(C)$ and $\upsilon_C(\cdot) := \upsilon(C \cap \cdot)/\upsilon(C)$. We let M be defined as above but targeting υ_C. Naturally υ_C has a density w.r.t. υ, $\upsilon_C(z) = \mathbf{1}_C(z)/\upsilon(C)$ for $z \in \mathsf{Z}$ and for $i \in \{1, 2\}$ we have $\upsilon_C \circ \psi_i(z) = \mathbf{1}_{\psi_i^{-1}(C)}(z)$, $\upsilon \circ \psi_1 \circ \psi_2(z) = \mathbf{1}_{\psi_2^{-1} \circ \psi_1^{-1}(C)}(z)$. Consequently $S_1 := \left\{z \in \mathsf{Z} \colon \mathbf{1}_{C \cap \psi_1^{-1}(C)}(z) > 0\right\}$ and $S_2 = \left\{z \in \mathsf{Z} \colon \mathbf{1}_{C \cap \psi_1^{-1}(C^{\complement})}(z) \mathbf{1}_{\psi_2^{-1}(C) \cap \psi_2^{-1} \circ \psi_1^{-1}(C^{\complement})}(z) > 0\right\}$ and as a result*

$$\alpha_1(z) := \mathbf{1}_{A \cap \psi_1^{-1}(C)}(z)$$
$$\alpha_2(z) := \mathbf{1}_{A \cap \psi_1^{-1}(C^{\complement}) \cap \psi_2^{-1}(C) \cap \psi_2^{-1} \circ \psi_1^{-1}(C^{\complement})}(z)\,.$$

The corresponding kernel $M^{\otimes T}$ is described algorithmically in Algorithm 5.4. In the situation where $C := \{x \in \mathsf{X} \colon c(x) = 0\}$ for a continuously differentiable function $c \colon \mathsf{X} \to \mathbb{R}$, the bounces described in (5.4) can be defined in terms of the field $x \mapsto n(x)$ such that

$$n(x) := \begin{cases} \nabla c(x)/|\nabla c(x)| & \text{for } \nabla c(x) \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

This justifies the ideas of Betancourt (2011), where a process of the type given in Algorithm 5.4 is used as a proposal within a MH update, although the possibility of a rejection after the second stage seems to have been overlooked in that reference.

Algorithm 5.4: $M^{\otimes T}$ for M the delayed rejection algorithm targeting the uniform distribution on C

```
Given z_0 = z ∈ C ⊂ Z
for k = 1, ..., T do
    if ψ_1(z_{k-1}) ∈ C then
        z_k = ψ_1(z_{k-1})
    else if ψ_2(z_{k-1}) ∈ C and ψ_1 ∘ ψ_2(z_{k-1}) ∉ C then
        z_k = ψ_2(z_{k-1})
    else
        z_k = z_{k-1}.
```

Naturally a rejection of both transformations ψ_1 and ψ_2 of the current state means that the algorithm gets stuck. We note that it is also possible to replace the third update case with a

full refreshment of the velocity, which can be interpreted as a third delayed rejection update, of acceptance probability one.

5.3.2 Numerical illustration: orthant probabilities

In this section, we consider the problem of calculating the Gaussian orthant probabilities, which is given by

$$p(\mathbf{a}, \mathbf{b}, \Sigma) := \mathbb{P}(\mathbf{a} \leq \mathbf{X} \leq \mathbf{b}),$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$ are known vectors of dimension d and $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \Sigma)$ with Σ a covariance matrix of size $d \times d$. Consider the Cholesky decomposition of Σ which is given by $\Sigma := LL^\top$, where $L := (l_{ij})_{1 \leq i,j \leq d}$ is a lower triangular matrix with positive diagonal entries. It is clear that $\mathbf{X}$ can be viewed as $\mathbf{X} := L\eta$, where $\eta \sim \mathcal{N}_d(\mathbf{0}, \mathrm{Id}_d)$. Consequently, one can instead rewrite $p(\mathbf{a}, \mathbf{b}, \Sigma)$ as a product of d probabilities given by

$$p_1 = \mathbb{P}(a_1 \leq l_{11}\eta_1 \leq b_1) = \mathbb{P}\left(a_1/l_{11} \leq \eta_1 \leq b_1/l_{11}\right), \tag{5.5}$$

and

$$p_n = \mathbb{P}\left(a_t \leq \sum_{j=1}^{n} l_{nj}\eta_j \leq b_t\right) = \mathbb{P}\left(\frac{a_t - \sum_{j=1}^{n-1} l_{nj}\eta_j}{l_{nn}} \leq \eta_t \leq \frac{b_t - \sum_{j=1}^{n-1} l_{nj}\eta_j}{l_{nn}}\right), \tag{5.6}$$

for $n = 2, \ldots, d$. For notational simplicity, we let $\mathcal{B}_n(\eta_{1:n-1})$ denote the interval

$$\left[\frac{a_n - \sum_{j=1}^{n-1} l_{nj}\eta_j}{l_{nn}}, \frac{b_n - \sum_{j=1}^{n-1} l_{nj}\eta_j}{l_{nn}}\right],$$

with the convention $\mathcal{B}_1(\eta_{1:0}) := [a_1/l_{11}, b_1/l_{11}]$. Then, $p(\mathbf{a}, \mathbf{b}, \Sigma)$ can be written as the product of $p_n\, s$ for $n = 1, 2, \ldots, d$. Moreover, one can see that p_n is also the normalising constant of the conditional distribution of η_n given $\eta_{1:n-1}$. To calculate the orthant probability, Ridgway (2016) have proposed an SMC algorithm targeting the sequence of distributions

$\pi_n(\eta_{1:n}) := \pi_1(\eta_n) \prod_{k=2}^{n} \pi_n(\eta_k|\eta_{1:k-1})$ for $n \in [\![1, d]\!]$, given by

$$\pi_1(\eta_1) \propto \phi(\eta_1)\mathbf{1}\{\eta_1 \in \mathcal{B}_1\} = \gamma_1(\eta_1) \tag{5.7}$$

$$\pi_n(\eta_n|\eta_{1:n-1}) \propto \phi(\eta_n)\mathbf{1}\{\eta_n \in \mathcal{B}_n(\eta_{1:n-1})\} = \gamma_n(\eta_n|\eta_{1:n-1}). \tag{5.8}$$

where ϕ denotes the probability density of a $\mathcal{N}(0, 1)$. One could also note that $\pi_n(\eta_n|\eta_{1:n-1}) = 1/\Phi(\mathcal{B}_n(\eta_{1:n-1}))\phi(\eta_n)\mathbf{1}\{\eta_n \in \mathcal{B}_n(\eta_{1:n-1})\}$ and $\gamma_n(\eta_{1:n}) = \phi(\eta_{1:n}) \prod_{k=1}^{n} \mathbf{1}\{\eta_k \in \mathcal{B}_k(\eta_{1:k-1})\}$, where $\Phi(\mathcal{B}_n(\eta_{1:n-1}))$ represents the probability of a standard Normal random variable being in the region $\mathcal{B}_n(\eta_{1:n-1})$. Therefore, the SMC algorithm proposed by Ridgway (2016) then proceeds as follows. (1) At time t, particles $\eta_{1:t-1}^n$ are extended by sampling $\eta_n^n \sim \pi_n(\mathrm{d}\eta_t|\eta_{1:t-1}^{(i)})$. (2) Particles $\eta_{1:t}^{(i)}$ are then reweighted by multiplying the incremental weights $\Phi(\mathcal{B}_n(\eta_{1:n-1}^{(i)}))$ to $w_{n-1}^{(i)}$. (3) If the ESS is below a certain threshold, resample the particles and move them through an MCMC kernel that leaves π_t invariant for k iterations. For the MCMC kernel, Ridgway (2016) recommended using the Gibbs sampler that leaves π_t invariant to move the particles at step (3). The orthant probability we are interested in can then be viewed as the normalising constant of π_d and this can be estimated as a by-product of the SMC algorithm.

Since we are trying to sample from the constrained Gaussian distributions, the Hamiltonian equation can be solved exactly and $w_{n,k}$ is always 1. As a result, the incremental weights for the trajectories simplify to $\Phi(\mathcal{B}_n(u_{n-1}))$ and each particle on the trajectory starting from z_t will have an incremental weight proportional to $\Phi(\mathcal{B}_n(u_{n-1}))$. To obtain a trajectory, we follow Pakman and Paninski (2014) who perform HMC with reflections to sample. As the dimension increases, the number of reflections performed under ψ_n will also increase given a fixed integration time. We adaptively tuned the integrating time ε to ensure that the average number of reflections at each SMC step does not exceed a given threshold. In our experiment, we set this threshold to be 5. To show that the waste-recycling RSMC algorithm scales well in high dimension, we set $d = 150$, $a = (1.5, 1.5, ...)$ and $b = (\infty, \infty, ...)$. Also, we use the same covariance matrix in Dau and Chopin (2020) and perform variable re-ordering as suggested in Ridgway (2016) before the simulation.

Figures 5.4 and 5.5 show the results obtained with $N \times (T + 1) = 50,000$ and various values for N. With a quarter of the number of particles used in Dau and Chopin (2020), the waste-

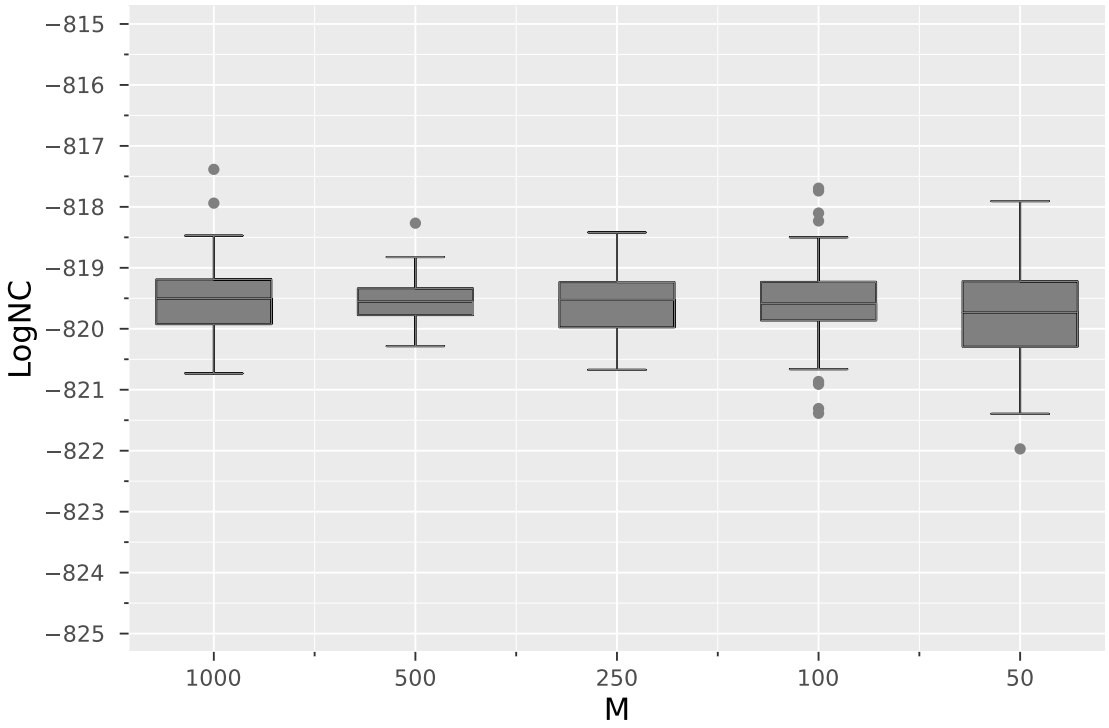

Figure 5.4: Orthant probability example: estimates of the normalising constant (i.e. the orthant probability) obtained from the waste-recycling HSMC algorithm with $N = 50,000$.

recycling HSMC algorithm achieves comparable performance when estimating the normalising constant (i.e. the orthant probability). Moreover, the estimates are stable for different choices of N values, although one observes that the algorithm achieves the best performance when $N = 500$ (i.e. each trajectory contains 100 particles). This also suggests that the integrating time should be adaptively tuned in a different way to achieve the best performance given a fixed computational budget. Estimates of the function $\varphi(x_{0:d}) = \mathbb{E}(1/d \sum_{i=1}^{d} x_i)$ with respect to the Gaussian distribution $\mathcal{N}_d(\mathbf{0}, \Sigma)$ truncated between $\mathbf{a}$ and $\mathbf{b}$ are also stable for different choices of N, although they are more variable than those obtained in Dau and Chopin (2020). This indicates that the waste-recycling HSMC algorithm does scale well in high dimensions. We note that this higher variance compared to the waste-free SMC of Dau and Chopin (2020), is obtained in a scenario where they are able to exploit the particular structure of the problem and implement an exact Gibbs sampler to move the particles. The waste-recycling HSMC algorithm is however more general and applicable to scenarios where such a structure is not present.

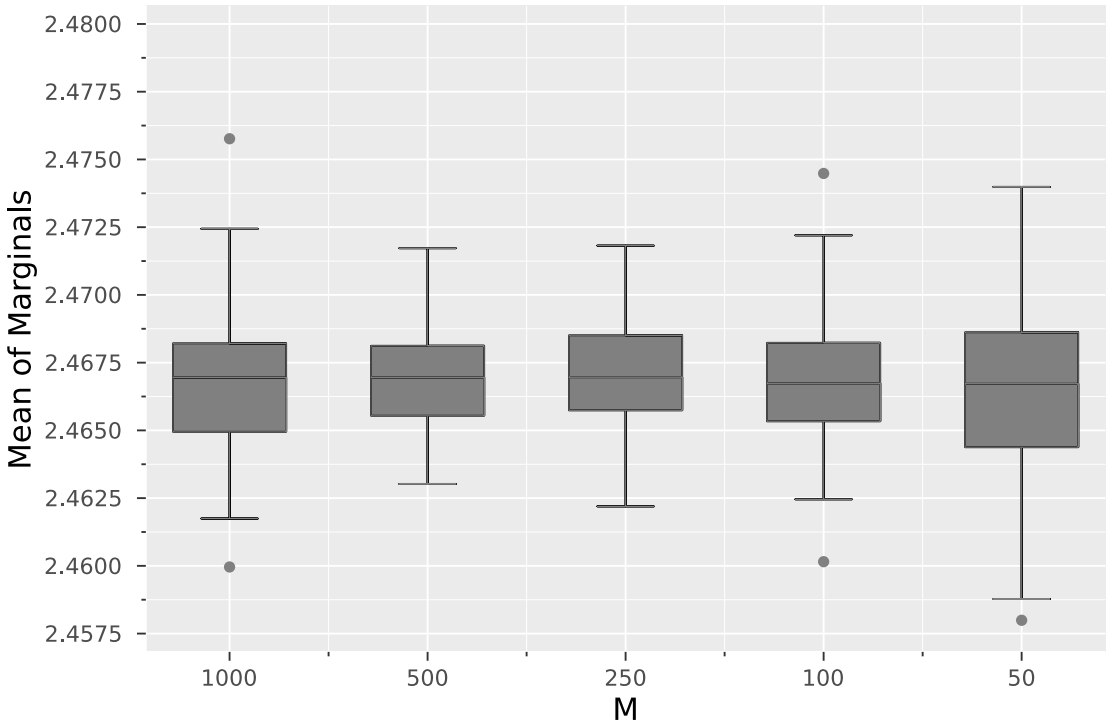

Figure 5.5: Orthant probability example: estimates of the mean of marginals obtained from the waste-recycling HSMC algorithm with $N = 50,000$.

5.4 Sampling a mixture: general justification

In this section, we first establish properties of a type of mixture of probability distributions arising from a snippet $\mathsf{z} := (z_0, z_1, \ldots, z_T) \in \mathsf{Z}^{T+1}$ of a Markov chain initialised with some probability distribution $z_0 \sim \mu$ and of Markov kernel, M, producing in particular $T+1$ weighted samples to compute expectations with respect to μ.

Lemma 5.4.1. *Let μ be a probability distribution on $\left(\mathsf{Z}, \mathscr{Z}\right)$. For $T \in \mathbb{N} \setminus \{0\}$ introduce the mixture defined on $\left(\mathsf{Z}^{T+1}, \mathscr{Z}^{\otimes(T+1)}\right)$*

$$\bar{\mu}(\mathrm{dz}) = \frac{1}{T+1} \sum_{k=0}^{T} \bar{\mu}(k, \mathrm{dz}) \,,$$

where for $(k, z) \in [\![0, T]\!] \times \mathsf{Z}$

$$\bar{\mu}(k, \mathrm{dz}) := \frac{1}{T+1} w_k(\mathsf{z}) \mu \otimes M^{\otimes T}(\mathrm{dz}) \,,$$

and for $M\colon \mathsf{Z} \times \mathscr{Z} \to [0,1]$ *and* $\mathsf{z} := (z_0, z_1, \ldots, z_T) \in \mathsf{Z}^{T+1}$,

$$M^{\otimes T}(z_0, \mathrm{d}\mathsf{z}_{-0}) := \prod_{i=1}^{T} M(z_{i-1}, \mathrm{d}z_i)\,.$$

Assume the existence of $L\colon \mathsf{Z} \times \mathscr{Z} \to [0,1]$ *such that for* $\mathsf{z} := (z_0, z_1, \ldots, z_T) \in \mathsf{Z}^{T+1}$ *the Radon-Nikodym derivative is well defined*

$$w_k(\mathsf{z}) := \frac{\mu \otimes L^k\big(\mathrm{d}(z_k, z_0)\big)}{\mu \otimes M^k\big(\mathrm{d}(z_0, z_k)\big)}\,,$$

with the convention $w_0(\mathsf{z}) = 1$. *Then, for any* $f\colon \mathsf{Z} \to \mathbb{R}$ *such that* $\mu(|f|) < \infty$

1. we have
$$\int f(z') \frac{\mu \otimes L^k\big(\mathrm{d}(z', z)\big)}{\mu \otimes M^k\big(\mathrm{d}(z, z')\big)} \mu(\mathrm{d}z) M^k(z, \mathrm{d}z') = \mu(f)\,,$$
2. with $\bar{f}(k, \mathsf{z}) := f(z_k)$ then
   (a)
   $$\bar{\mu}(\bar{f}) = \mu(f)\,,$$
   (b)
   $$\int \sum_{k=0}^{T} \bar{f}(k, \mathsf{z}) \frac{w_k(\mathsf{z})}{\sum_{l=0}^{T} w_l(\mathsf{z})} \bar{\mu}(\mathrm{d}\mathsf{z}) = \mu(f)\,.$$

Proof. The first statement follows directly from the definition of the Radon-Nikodym derivative

$$\begin{aligned}\int f(z') \frac{\mu \otimes L^k\big(\mathrm{d}(z', z)\big)}{\mu \otimes M^k\big(\mathrm{d}(z, z')\big)} \mu(\mathrm{d}z) M^k(z, \mathrm{d}z') &= \int f(z') \mu(\mathrm{d}z') L^k(z', \mathrm{d}z) \\ &= \int f(z') \mu(\mathrm{d}z')\,.\end{aligned}$$

The second statement follows from

$$\frac{1}{T+1} \sum_{k=0}^{T} \int \bar{f}(k, \mathsf{z}) w_k(\mathsf{z}) \mu \otimes M^{\otimes T}(\mathrm{d}\mathsf{z}) = \frac{1}{T+1} \sum_{k=0}^{T} \int f(z_k) w_k(\mathsf{z}) \mu \otimes M^k(\mathrm{d}z_k)\,,$$

and the first statement. The last statement follows from the second statement upon noticing that for $k \in [\![0, T]\!]$,

$$\bar{\mu}(k \mid \mathsf{z}) = \frac{w_k(\mathsf{z})}{\sum_{l=0}^{T} w_l(\mathsf{z})},$$

and using the tower property of expectations. □

Corollary 5.4.1.1. *Assume that* $\mathsf{z} \sim \bar{\mu}(\cdot)$*, then*

$$\sum_{k=0}^{T} f(z_k) \frac{w_k(\mathsf{z})}{\sum_{l=0}^{T} w_l(\mathsf{z})}$$

is an unbiased estimator of $\mu(f)$*. This justifies algorithms which sample from the mixture* $\bar{\mu}$ *directly in order to estimate expectations with respect to* μ*.*

Lemma 5.4.2. *With* μ *and* $\bar{\mu}$ *as in Lemma 5.4.1, let* $\bar{M} \colon \mathsf{Z}^{T+1} \times \mathscr{Z}^{\otimes(T+1)} \to [0,1]$ *be the Markov kernel*

$$\bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}') := \sum_{k=0}^{T} \bar{\mu}(k \mid \mathsf{z}) R(z_k, \mathrm{d}z'_0) M^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right),$$

with $R \colon \mathsf{Z} \times \mathscr{Z} \to [0,1]$*. Then for any* $\mathsf{z}, \mathsf{z}' \in \mathsf{Z}^{T+1}$

1. *we have*

$$\bar{\mu}(\mathrm{d}\mathsf{z})\bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}') = \mu(\mathrm{d}z_0) M^{\otimes T}(z_0, \mathrm{d}\mathsf{z}_{-0}) \left\{ \frac{1}{T+1} \sum_{l=0}^{T} w_l(\mathsf{z}) R(z_l, \mathrm{d}z'_0) \right\} M^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right),$$

2. *and*

$$\bar{\mu}\bar{M}(\mathrm{d}\mathsf{z}') = \mu R(\mathrm{d}z'_0)\, M^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right).$$

Proof. Note that

$$\begin{aligned}
\bar{\mu}(\mathrm{d}\mathsf{z})\bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}') &= \mu(\mathrm{d}z_0) M^{\otimes T}(z_0, \mathrm{d}\mathsf{z}_{-0}) \frac{1}{T+1} \sum_{k,l=0}^{T} w_k(\mathsf{z}) \bar{\mu}(l \mid \mathsf{z}) R(z_l, \mathrm{d}z'_0) M^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right) \\
&= \mu(\mathrm{d}z_0) M^{\otimes T}(z_0, \mathrm{d}\mathsf{z}_{-0}) \left\{ \frac{1}{T+1} \sum_{l=0}^{T} w_l(\mathsf{z}) R(z_l, \mathrm{d}z'_0) \right\} M^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right).
\end{aligned}$$

From this and using that $\bar{\mu}(l \mid \mathsf{z}) \propto w_l(\mathsf{z})$, which is a function of z_0 and z_l only, we obtain

$$\begin{aligned}
\bar{\mu}\bar{M}(\mathrm{d}\mathsf{z}') &= \int \mu(\mathrm{d}z_0) M^{\otimes T}(z_0, \mathrm{d}\mathsf{z}_{-0}) \frac{1}{T+1} \sum_{k,l=0}^{T} w_k(\mathsf{z}) \bar{\mu}(l \mid \mathsf{z}) R(z_l, \mathrm{d}z_0') M^{\otimes T}\left(z_0', \mathrm{d}\mathsf{z}_{-0}'\right) \\
&= \frac{1}{T+1} \left\{ \sum_{l=0}^{T} \int \mu(\mathrm{d}z_l) M^l(z_l, \mathrm{d}z_0) R(z_l, \mathrm{d}z_0') \right\} M^{\otimes T}\left(z_0', \mathrm{d}\mathsf{z}_{-0}'\right) \\
&= \frac{1}{T+1} \left\{ \sum_{l=0}^{T} \int \mu(\mathrm{d}z_l) R(z_l, \mathrm{d}z_0') \right\} M^{\otimes T}\left(z_0', \mathrm{d}\mathsf{z}_{-0}'\right) \\
&= \mu R(\mathrm{d}z_0')\, M^{\otimes T}\left(z_0', \mathrm{d}\mathsf{z}_{-0}'\right).
\end{aligned}$$

□

Lemma 5.4.3. *With μ and M as in Lemmas 5.4.1 and 5.4.2, then consider the backward kernel $\bar{L}$*

$$\bar{L}(\mathsf{z}', \mathrm{d}\mathsf{z}) := \frac{\bar{\mu}(\mathrm{d}\mathsf{z}) \bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}')}{\bar{\mu}\bar{M}(\mathrm{d}\mathsf{z}')}.$$

Then for ν another probability distribution $\left(\mathsf{Z}, \mathscr{Z}\right)$ and assuming $\int \mu(\mathrm{d}z) R(z, \mathrm{d}z_0') = \mu(\mathrm{d}z_0')$ then

$$\frac{\bar{\nu}(\mathrm{d}\mathsf{z}') \bar{L}(\mathsf{z}', \mathrm{d}\mathsf{z})}{\bar{\mu}(\mathrm{d}\mathsf{z}) \bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}')} = \frac{\nu(\mathrm{d}z_0')}{\mu(\mathrm{d}z_0')} \frac{1}{T+1} \sum_{k=0}^{T} w_k(\mathsf{z}').$$

Proof. From the definition of $\bar{L}$ and Lemma 5.4.2

$$\begin{aligned}
\frac{\bar{\nu}(\mathrm{d}\mathsf{z}') \bar{L}(\mathsf{z}', \mathrm{d}\mathsf{z})}{\bar{\mu}(\mathrm{d}\mathsf{z}) \bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}')} &= \frac{\bar{\nu}(\mathrm{d}\mathsf{z}')}{\bar{\mu}\bar{M}(\mathrm{d}\mathsf{z}')} \\
&= \frac{\nu \otimes M^{\otimes T}(\mathrm{d}\mathsf{z}')}{\mu R(\mathrm{d}z_0')\, M^{\otimes T}\left(z_0', \mathrm{d}\mathsf{z}_{-0}'\right)} \frac{1}{T+1} \sum_{k=0}^{T} w_k(\mathsf{z}') \\
&= \frac{\nu(\mathrm{d}z_0')}{\mu R(\mathrm{d}z_0')} \frac{1}{T+1} \sum_{k=0}^{T} w_k(\mathsf{z}'),
\end{aligned}$$

and we conclude with the assumption of R. □

5.5 Justification of the waste-free SMC sampler

Given a sequence $\left\{\mu_n, n \in [\![0, P]\!]\right\}$ of probability distributions defined on a measurable space $(\mathsf{Z}, \mathscr{Z})$ introduce the sequence of distributions defined on $\left([\![0, T]\!] \times \mathsf{Z}^{T+1}, \mathscr{P}([\![0, T]\!]) \otimes\right.$

$\mathscr{Z}^{\otimes(T+1)}\big)$ such that for any$(n,k,z) \in [\![0,P]\!] \times [\![0,T]\!] \times \mathsf{Z}$

$$\bar{\mu}_n(k, \mathrm{d}\mathsf{z}) := \frac{1}{T+1} w_{n,k}(\mathsf{z}) \mu_n \otimes M_n^{\otimes T}(\mathrm{d}\mathsf{z})$$

yielding the marginals

$$\bar{\mu}_n(\mathrm{d}\mathsf{z}) := \frac{1}{T} \sum_{k=0}^{T} \bar{\mu}_n(k, \mathrm{d}\mathsf{z}),$$

where for $k \in [\![0,P]\!]$

$$w_{n,k}(\mathsf{z}, \mathsf{z}') = \frac{\mu_n(\mathrm{d}\mathsf{z}') L_n^k(\mathsf{z}', \mathrm{d}\mathsf{z})}{\mu_n(\mathrm{d}\mathsf{z}) M_n^k(\mathsf{z}, \mathrm{d}\mathsf{z}')},$$

$w_{n,0}(\mathsf{z}) = 1$ for any $\mathsf{z} \in \mathsf{Z}^{T+1}$. Consider $\bar{M}_n \colon \mathsf{Z}^{T+1} \times \mathscr{Z}^{\otimes(T+1)} \to [0,1]$

$$\bar{M}_n(\mathsf{z}, \mathrm{d}\mathsf{z}') := \sum_{k=0}^{T} \bar{\mu}_{n-1}(k \mid \mathsf{z}) R_n(z_k, \mathrm{d}z'_0) M_n^{\otimes T}\left(z'_0, \mathrm{d}\mathsf{z}'_{-0}\right),$$

where $M_n, R_n \colon \mathsf{Z} \times \mathscr{Z} \to [0,1]$ and $\mu_{n-1} R_n = \mu_{n-1}$. Note that Dau and Chopin (2020) set $R_n = M_n$, and therefore impose that M_n is $\mu_{n-1}-$invariant, which is not necessary here. Considering the optimal backward kernel $\bar{L}_n \colon \mathsf{Z}^{T+1} \times \mathscr{Z}^{\otimes(T+1)} \to [0,1]$ of the form

$$\bar{L}_n(\mathsf{z}, \mathrm{d}\mathsf{z}') \propto \bar{\mu}_{n-1}(\mathrm{d}\mathsf{z}') \bar{M}_n(\mathsf{z}', \mathrm{d}\mathsf{z}),$$

from Lemma 5.4.1 and Lemma 5.4.2 one obtains

$$\bar{w}_n(\mathsf{z}, \mathsf{z}') = \frac{\mu_n(\mathrm{d}z'_0)}{\mu_{n-1}(\mathrm{d}z'_0)} \frac{1}{T+1} \sum_{k=0}^{T} w_{n-1,k}(\mathsf{z}, \mathsf{z}'). \tag{5.9}$$

The corresponding folded version of the algorithm is given in Algorithm 5.5.

5.6 Sampling HMC trajectories

In this scenario we have $\mu(\mathrm{d}z) = \pi(\mathrm{d}x)\varpi(\mathrm{d}v)$ assumed to have a density with respect to a measure υ, and ψ is an invertible mapping $\psi \colon \mathsf{Z} \to \mathsf{Z}$ such that $\upsilon^{\psi} = \upsilon$ and $\psi^{-1} = \sigma \circ \psi \circ \sigma$ with $\sigma \colon \mathsf{Z} \to \mathsf{Z}$ such that $\mu \circ \sigma(z) = \mu(z)$. In this chapter, we focus primarily on the scenario where ψ is a discretization of Hamilton's equations for a potential $U \colon \mathsf{X} \to \mathbb{R}$ e.g. a leapfrog or Stormer-Verlet integrator. We now consider the scenario where, in the framework developed

Algorithm 5.5: Hamiltonian Snippet SMC algorithm

sample $z_0^{(i)} \overset{\text{iid}}{\sim} \mu_0$ and set $w_0(z_0^{(i)}) = 1$ for $i \in [\![N]\!]$.
for $n = 1, \ldots, P$ **do**
  for $i \in [\![N]\!]$ **do**
    sample $a_i \sim \mathrm{Cat}\left(1, w_{n-1,1}(\mathsf{z}_{n-1}^{(i)}), w_{n-1,2}(\mathsf{z}_{n-1}^{(i)}), \ldots, w_{n-1,T}(\mathsf{z}_{n-1}^{(i)})\right)$
    sample $\tilde{z}_{n,0}^{(i)} \sim R_n(z_{a_i,n-1}, \cdot)$
    for $k \in [\![1, T]\!]$ **do**
      sample $\tilde{z}_{n,k}^{(i)} \sim M_n(\tilde{z}_{n,k-1}^{(i)}, \cdot)$
      compute

$$w_{n,k}\left(\mathsf{z}_{n-1}^{(i)}, \tilde{\mathsf{z}}_n^{(i)}\right) = \frac{\mu_n(\mathrm{d}\tilde{z}_{n,k}^{(i)}) L_n^k(\tilde{z}_{n,k}, \mathrm{d}\tilde{z}_{n,0}^{(i)})}{\mu_n(\mathrm{d}\tilde{z}_{n,0}^{(i)}) M_n^k(\tilde{z}_{n,0}^{(i)}, \mathrm{d}\tilde{z}_{n,k}^{(i)})}$$

  resample N particles from $\left\{\tilde{\mathsf{z}}_n^{(i)}, i \in [\![N]\!]\right\}$, using a multinomial distribution of parameter

$$\bar{w}_{n+1}\left(\mathsf{z}_{n-1}^{(i)}, \tilde{\mathsf{z}}_n^{(i)}\right) = \frac{\mu_{n+1}\left(\tilde{\mathsf{z}}_n^{(i)}\right)}{\mu_n\left(\tilde{\mathsf{z}}_n^{(i)}\right)} w_n\left(\mathsf{z}_{n-1}^{(i)}, \tilde{\mathsf{z}}_n^{(i)}\right), \quad i \in [\![N]\!],$$

  and obtain $\mathsf{z}_n^{(i)}$ for $i \in [\![N]\!]$.

in Section 5.5,we let $M(z, \mathrm{d}z') := \delta_{\psi(z)}(\mathrm{d}z')$ be the deterministic kernel which maps the current state $z \in \mathsf{Z}$ to $\psi(z)$. Define $\Psi(z, \mathrm{d}z') = \delta_{\psi(z)}(\mathrm{d}z')$ and $\Psi^*(z, \mathrm{d}z') = \delta_{\psi^{-1}(z)}(\mathrm{d}z')$; we exploit the ideas of (Andrieu et al., 2020, Proposition 4) to establish that Ψ^* is the $\nu-$adjoint of Ψ if υ is invariant under ψ.

Lemma 5.6.1. *Let μ be a probability measure and υ a measure, on $(\mathsf{Z}, \mathscr{Z})$ such that $\upsilon \gg \mu$. Denote $\mu(z) := \mathrm{d}\mu/\mathrm{d}\upsilon(z)$ for any $z \in \mathsf{Z}$. Let $\psi \colon \mathsf{Z} \to \mathsf{Z}$ be an invertible and volume preserving mapping, i.e. such that $\upsilon^\psi(A) = \upsilon\left(\psi^{-1}(A)\right) = \upsilon(A)$ for all $A \in \mathscr{Z}$, then*

1. *(υ, Ψ, Ψ^*) form a reversible triplet, that is for all $z, z' \in \mathsf{Z}$,*

$$\upsilon(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z') = \upsilon(\mathrm{d}z')\delta_{\psi^{-1}(z')}(\mathrm{d}z),$$

2. *for all $z, z' \in \mathsf{Z}$ such that $\mu(z) > 0$*

$$\mu(\mathrm{d}z')\delta_{\psi^{-1}(z')}(\mathrm{d}z) = \frac{\mu \circ \psi(z)}{\mu(z)} \mu(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z')\,.$$

Proof. For the first statement

$$\begin{aligned}\int f(z)g\circ\psi(z)\upsilon(\mathrm{d}z) &= \int f\circ\psi^{-1}\circ\psi(z)g\circ\psi(z)\upsilon(\mathrm{d}z)\\ &= \int f\circ\psi^{-1}(z)g(z)\upsilon^{\psi}(\mathrm{d}z)\\ &= \int f\circ\psi^{-1}(z)g(z)\upsilon(\mathrm{d}z).\end{aligned}$$

We have

$$\begin{aligned}\mu(\mathrm{d}z')\delta_{\psi^{-1}(z')}(\mathrm{d}z) &= \mu(z')\upsilon(\mathrm{d}z')\delta_{\psi^{-1}(z')}(\mathrm{d}z)\\ &= \mu(z')\upsilon(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z')\\ &= \mu\circ\psi(z)\upsilon(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z')\\ &= \frac{\mu\circ\psi(z)}{\mu(z)}\mu(z)\upsilon(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z')\\ &= \frac{\mu\circ\psi(z)}{\mu(z)}\mu(\mathrm{d}z)\delta_{\psi(z)}(\mathrm{d}z')\end{aligned}$$

□

Corollary 5.6.1.1. *With the assumptions above for* $(n,k)\in[\![0,P]\!]\times[\![0,T]\!]$

$$w_{n,k}(\mathsf{z},\mathsf{z}') = \frac{\mu_n(\mathrm{d}\mathsf{z}')(\Psi^*)^{-k}(\mathsf{z}',\mathrm{d}\mathsf{z})}{\mu_n(\mathrm{d}\mathsf{z})\Psi^k(\mathsf{z},\mathrm{d}\mathsf{z}')} = \frac{\mu_n\circ\psi_n^k(z_0)}{\mu_n(z_0)}\,,$$

which justifies the simplified notation $w_{n,k}(z_0)$ *used in Section 5.2. and for a* $\mu-$*invariant Markov kernel, for example* $R_n(z,\mathrm{d}z') = \delta_x(\mathrm{d}x')\varpi_{n-1}(\mathrm{d}v')$*, from Lemma 5.4.3. From (5.9) we have*

$$\begin{aligned}\bar{w}_n(\mathsf{z},\mathsf{z}') = \frac{\mu_n(\mathrm{d}z_0')}{\mu_{n-1}(\mathrm{d}z_0')}\frac{1}{T+1}\sum_{k=0}^{T} w_{n-1,k}(\mathsf{z},\mathsf{z}')\\ \frac{\mu_n(\mathrm{d}z_0')}{\mu_{n-1}(\mathrm{d}z_0')}\frac{1}{T+1}\sum_{k=0}^{T} w_{n-1,k}(z_0)\,,\end{aligned}$$

therefore justifying the simplified notation $\bar{w}_n(z)$ *used in Section 5.2. This together with the results of Section 5.5 finish the justification of correctness of Algorithm 5.3 and hence Algorithm 5.2.*

Corollary 5.6.1.2. *Taking $\bar{L} = \bar{M}$ and assuming that for any $z \in \mathsf{Z}$, $\upsilon \gg R(z, \cdot)$ we obtain*

$$\begin{aligned}\frac{\bar{\mu}(\mathrm{d}\mathsf{z}')\bar{M}(\mathsf{z}', \mathrm{d}\mathsf{z})}{\bar{\mu}(\mathrm{d}\mathsf{z})\bar{M}(\mathsf{z}, \mathrm{d}\mathsf{z}')} &= \frac{\mu(\mathrm{d}z_0') \sum_{l=0}^{T} w_l(\mathsf{z}')R(z_l', \mathrm{d}z_0)}{\mu(\mathrm{d}z_0) \sum_{l=0}^{T} w_l(\mathsf{z})R(z_l, \mathrm{d}z_0')} \cdot \\ &= \frac{\sum_{l=0}^{T} \mu \circ \psi^l(z'_0)R(z_l', z_0)}{\sum_{l=0}^{T} \mu \circ \psi^l(z_0)R(z_l, z_0')}\end{aligned}$$

So we could use an MH algorithm targeting $\bar{\mu}$ and proposal $\bar{M}$, of acceptance ratio where we need a density for R on the last line. The acceptance ratio therefore compares the 'densities' of the contours. Note that when we use the kernel $R(z, \mathrm{d}z') = \delta_x(\mathrm{d}x') \otimes \varpi(\mathrm{d}v')$ then this has yet another expression. In the limit, or assume that T and ε are properly adjusted, it seems that we would be targeting the distribution of the energies $-\log \mu(z)$ by using this approach i.e. a particle coincides with a contour and the update is in effect a contour change.

Conclusion

In this thesis, we have introduced improvements and novelties into different areas of the realm of Monte Carlo methods. At the same time, it also opens up several questions that are worth considering in the future:

1. In Chapter 2, we tried to resample the jumps in the previous time block at each SMC iteration. However, we still sample the modification type and modification positions according to the prior. As new information has arrived, it's worth trying to design more efficient proposals based on the new information. Our hypothesis is that with more efficient proposals, the new algorithm proposed in Chapter 2 would be able to produce even better performance. As shown in the simulations, having the block sampling step in fact introduces more variance. Since the main idea of having the block sampling is to recover and modify the jumps that are close to the end of the time block, it is also worth investigating the algorithm that only re-samples part of the previous block.

2. The algorithm introduced in Chapter 4 requires that the number of latents that generate the data should be fixed. However, in some situations, the number of latents within the generator is also random. Hence, a natural extension would be developing a Latent-ABC-SMC algorithm that is able to have variable number of latents or only involves a fraction of the latents.

3. The algorithm developed in Chapter 5 opens up many questions. First of all, simulations have suggested that the stepsize and trajectory lengths do have an effect on the efficiency of the algorithms. Hence, it is worth investigating possible ways of adaptations within the SMC algorithm. The idea of sampling Markov process snippets can also be incorporated with MCMC algorithms, which is another direction that is worth considering. Numerical

results have shown great advantages of the algorithm over other methods. It is also worth trying to explain such an advantage from a theoretical perspective.

References

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., Lee, A. and Livingstone, S. (2020), 'A general perspective on the metropolis-hastings kernel', *arXiv preprint arXiv:2012.14881* .

Andrieu, C. and Roberts, G. O. (2009), 'The pseudo-marginal approach for efficient monte carlo computations', *The Annals of Statistics* **37**(2), 697–725.

Andrieu, C. and Thoms, J. (2008), 'A tutorial on adaptive mcmc', *Statistics and computing* **18**(4), 343–373.

Beaumont, M. A., Zhang, W. and Balding, D. J. (2002), 'Approximate Bayesian computation in population genetics', *Genetics* **162**(4).

Bernton, E., Jacob, P. E., Gerber, M. and Robert, C. P. (2019), 'Approximate bayesian computation with the wasserstein distance', *arXiv preprint arXiv:1905.03747* .

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M. and Stuart, A. (2013), 'Optimal tuning of the hybrid Monte Carlo algorithm', *Bernoulli* **19**(5A), 1501–1534.

Betancourt, M. (2011), Nested sampling with constrained hamiltonian monte carlo, *in* 'AIP Conference Proceedings', Vol. 1305, American Institute of Physics, pp. 165–172.

Bierkens, J. and Roberts, G. (2017), 'A piecewise deterministic scaling limit of lifted metropolis–hastings in the curie–weiss model', *The Annals of Applied Probability* **27**(2), 846–882.

Bouchard-Côté, A., Vollmer, S. J. and Doucet, A. (2018), 'The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method', *Journal of the American Statistical Association* **113**(522), 855–867.

Bunch, P. and Godsill, S. (2013), 'Particle smoothing algorithms for variable rate models', *IEEE Transactions on Signal Processing* **61**(7), 1663–1675.

Calvo, M. P., Sanz-Alonso, D. and Sanz-Serna, J. M. (2021), 'HMC: reducing the number of rejections by not using leapfrog and some results on the acceptance rate', *Journal of Computational Physics* **437**, 110333.

Campos, C. M. and Sanz-Serna, J. M. (2015), 'Extra chance generalized hybrid Monte Carlo', *Journal of Computational Physics* **281**, 365–374.

Centanni, S. and Minozzo, M. (2006*a*), 'Estimation and filtering by reversible jump åcmc for a doubly stochastic poisson model for ultra-high-frequency financial data', *Statistical Modelling* **6**(2), 97–118.

Centanni, S. and Minozzo, M. (2006*b*), 'A monte carlo approach to filtering for a class of marked doubly stochastic poisson processes', *Journal of the American Statistical Association* **101**(476), 1582–1597.

Cérou, F., Del Moral, P., Furon, T. and Guyader, A. (2012), 'Sequential monte carlo for rare event estimation', *Statistics and computing* **22**(3), 795–808.

Chevallier, A., Pion, S. and Cazals, F. (2018), Hamiltonian Monte Carlo with boundary reflections, and application to polytope volume calculations, PhD thesis, INRIA Sophia Antipolis, France.

Chopin, N. (2004), 'Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference', *The Annals of Statistics* **32**(6), 2385 – 2411.
URL: *https://doi.org/10.1214/009053604000000698*

Chopin, N. and Ridgway, J. (2017), 'Leave pima indians alone: binary regression as a benchmark for bayesian computation', *Statistical Science* **32**(1), 64–87.

Dau, H.-D. and Chopin, N. (2020), 'Waste-free sequential Monte Carlo', *arXiv preprint arXiv:2011.02328* .

Del Moral, P., Doucet, A. and Jasra, A. (2006), 'Sequential monte carlo samplers', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **68**(3), 411–436.

Del Moral, P., Doucet, A. and Jasra, A. (2007), 'Sequential monte carlo for bayesian computation', *Bayesian statistics* **8**(1), 34.

Del Moral, P., Doucet, A. and Jasra, A. (2012), 'An adaptive sequential monte carlo method for approximate bayesian computation', *Statistics and computing* **22**(5), 1009–1020.

Doucet, A., Briers, M. and Sénécal, S. (2006), 'Efficient block sampling strategies for sequential monte carlo methods', *Journal of Computational and Graphical Statistics* **15**(3), 693–711.

Duane, S., Kennedy, A. D., Pendleton, B. J. and Roweth, D. (1987), 'Hybrid monte carlo', *Physics letters B* **195**(2), 216–222.

Fasiolo, M. and Wood, S. N. (2018), Abc in ecological modelling, *in* 'Handbook of approximate Bayesian computation', Chapman and Hall/CRC, pp. 597–622.

Fearnhead, P. and Prangle, D. (2012), 'Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **74**(3), 419–474.

Finke, A., Johansen, A. M. and Spanò, D. (2014), 'Static-parameter estimation in piecewise deterministic processes using particle gibbs samplers', *Annals of the Institute of Statistical Mathematics* **66**(3), 577–609.

Forneron, J.-J. and Ng, S. (2016), A likelihood-free reverse sampler of the posterior distribution, *in* 'Essays in Honor of Aman Ullah', Emerald Group Publishing Limited.

Godsill, S. and Vermaak, J. (2005), Variable rate particle filters for tracking applications, *in* 'IEEE/SP 13th Workshop on Statistical Signal Processing, 2005', IEEE, pp. 1280–1285.

Graham, M. and Storkey, A. (2017), Asymptotically exact inference in differentiable generative models, *in* 'Artificial Intelligence and Statistics', PMLR, pp. 499–508.

Green, P. J. (1995), 'Reversible jump markov chain monte carlo computation and bayesian model determination', *Biometrika* **82**(4), 711–732.

Gryazina, E. and Polyak, B. (2014), 'Random sampling: Billiard walk algorithm', *European Journal of Operational Research* **238**(2), 497–504.

Gustafson, P. (1998), 'A guided walk Metropolis algorithm', *Statistics and computing* **8**(4), 357–364.

Hoffman, M. D. and Sountsov, P. (2022), Tuning-free generalized hamiltonian monte carlo, *in* 'International Conference on Artificial Intelligence and Statistics', PMLR, pp. 7799–7813.

Hoffman, M., Radul, A. and Sountsov, P. (2021), An adaptive-mcmc scheme for setting trajectory lengths in hamiltonian monte carlo, *in* 'International Conference on Artificial Intelligence and Statistics', PMLR, pp. 3907–3915.

Holden, P. B., Edwards, N. R., Hensman, J. and Wilkinson, R. D. (2018), Abc for climate: dealing with expensive simulators, *in* 'Handbook of approximate Bayesian computation', Chapman and Hall/CRC, pp. 569–595.

Liepe, J. and Stumpf, M. P. (2018), Abc in systems biology, *in* 'Handbook of Approximate Bayesian Computation', Chapman and Hall/CRC, pp. 513–539.

Lindsten, F. and Schön, T. B. (2012), On the use of backward simulation in the particle gibbs sampler, *in* '2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 3845–3848.

Lindsten, F. and Schön, T. B. (2013), 'Backward simulation methods for monte carlo statistical inference', *Foundations and Trends® in Machine Learning* **6**(1), 1–143.

Liu, J. S. (2001), *Monte Carlo strategies in scientific computing*, Vol. 10, Springer.

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.

Marjoram, P., Molitor, J., Plagnol, V. and Tavaré, S. (2003), 'Markov chain monte carlo without likelihoods', *Proceedings of the National Academy of Sciences* **100**(26), 15324–15328.

Martin, J. S., Jasra, A. and McCoy, E. (2013), 'Inference for a class of partially observed point process models', *Annals of the Institute of Statistical Mathematics* **65**(3), 413–437.

Meeds, E. and Welling, M. (2015), 'Optimization monte carlo: Efficient and embarrassingly parallel likelihood-free inference', *arXiv preprint arXiv:1506.03693* .

Mohasel Afshar, H. and Domke, J. (2015), 'Reflection, refraction, and hamiltonian monte carlo', *Advances in neural information processing systems* **28**.

Murray, I. and Graham, M. (2016), Pseudo-marginal slice sampling, *in* 'Artificial Intelligence and Statistics', PMLR, pp. 911–919.

Neal, P. (2012), 'Efficient likelihood-free bayesian computation for household epidemics', *Statistics and Computing* **22**(6), 1239–1256.

Neal, R. M. (2011), *MCMC Using Hamiltonian Dynamics*, CRC Press, chapter 5.

Nsengiyumva, A. C. et al. (2013), 'Numerical simulation of stochastic differential equations'.

Pakman, A. and Paninski, L. (2013), 'Auxiliary-variable exact hamiltonian monte carlo samplers for binary distributions', *Advances in neural information processing systems* **26**.

Pakman, A. and Paninski, L. (2014), 'Exact hamiltonian monte carlo for truncated multivariate gaussians', *Journal of Computational and Graphical Statistics* **23**(2), 518–542.

Papamakarios, G. and Murray, I. (2016), Fast ε-free inference of simulation models with bayesian conditional density estimation, *in* 'Advances in neural information processing systems', pp. 1028–1036.

Polyak, B. T. and Gryazina, E. (2014), 'Billiard walk-a new sampling algorithm for control and optimization', *IFAC Proceedings Volumes* **47**(3), 6123–6128.

Prangle, D. (2017), 'Adapting the abc distance function', *Bayesian Analysis* **12**(1), 289–309.

Prangle, D., Everitt, R. G. and Kypraios, T. (2018), 'A rare event approach to high-dimensional approximate bayesian computation', *Statistics and Computing* **28**(4), 819–834.

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A. and Feldman, M. W. (1999), 'Population growth of human Y chromosomes: A study of y chromosome microsatellites', *Molecular Biology and Evolution* **16**(12).

Rao, V. and Teg, Y. W. (2013), 'Fast mcmc sampling for markov jump processes and extensions.', *Journal of Machine Learning Research* **14**(11).

Rayner, G. D. and MacGillivray, H. L. (2002), 'Numerical maximum likelihood estimation for the g-and-k and generalized g-and-h distributions', *Statistics and Computing* **12**(1), 57–75.

Reeves, R. and Pettitt, A. (2005), A theoretical framework for approximate bayesian computation, *in* 'Proceedings of the 20th International Workshop for Statistical Modelling, Sydney Australia', pp. 393–396.

Ridgway, J. (2016), 'Computation of gaussian orthant probabilities in high dimension', *Statistics and computing* **26**(4), 899–916.

Rotskoff, G. M. and Vanden-Eijnden, E. (2019), 'Dynamical computation of the density of states and bayes factors using nonequilibrium importance sampling', *Physical review letters* **122**(15), 150602.

Sisson, S. A., Fan, Y. and Tanaka, M. M. (2007), 'Sequential monte carlo without likelihoods', *Proceedings of the National Academy of Sciences* **104**(6), 1760–1765.

Stein, E. M. and Shakarchi, R. (2009), *Real analysis*, Princeton University Press.

Tavaré, S., Balding, D. J., Griffiths, R. C. and Donnelly, P. (1997), 'Inferring coalescence times from DNA sequence data'.

Thin, A., Janati El Idrissi, Y., Le Corff, S., Ollion, C., Moulines, E., Doucet, A., Durmus, A. and Robert, C. X. (2021), 'Neo: Non equilibrium sampling on the orbits of a deterministic transform', *Advances in Neural Information Processing Systems* **34**, 17060–17071.

Van Dyk, D. A. and Park, T. (2008), 'Partially collapsed gibbs samplers: Theory and methods', *Journal of the American Statistical Association* **103**(482), 790–796.

Von Neumann, J. (1963), 'Various techniques used in connection with random digits', *John von Neumann, Collected Works* **5**, 768–770.

Whiteley, N. (2010), 'Discussion on particle markov chain monte carlo methods,', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(3), 269–342.

Whiteley, N., Johansen, A. M. and Godsill, S. (2011), 'Monte carlo filtering of piecewise deterministic processes', *Journal of Computational and Graphical Statistics* **20**(1), 119–139.

Wilkinson, D. J. (2018), *Stochastic modelling for systems biology*, Chapman and Hall/CRC.

Wilkinson, R. D. (2013), 'Approximate bayesian computation (abc) gives exact results under the assumption of model error', *Statistical applications in genetics and molecular biology* **12**(2), 129–141.

Appendix A

MCMC Implementations for g-and-k Distributions

We introduced the implementation of an MCMC sampler targeting the posterior distribution of the g-and-k distribution, which is describe in Bernton et al. (2019) and earlier in Rayner and MacGillivray (2002). Note that the definition of the probability density function of g-and-k distribution relies on the ability of obtaining $z = g_\theta^{-1}(x)$ with given x. Hence, one requires the implementation of a root finding algorithm to solve the equation $g_\theta(z) = x$. In this section, we described the bisection algorithm that can be used to solve the aforementioned equation. The detail of the algorithm is listed in Algorithm A.1 and we can use it to find the corresponding z such that $g_\theta(z) = x$ with given x and parameter θ.

Also, one may notice that

$$\frac{dg_\theta^{-1}(x)}{dx} = \left[\frac{g_\theta(z)}{dz} \Big|_{z=g_\theta^{-1}(x)} \right]^{-1}$$

Hence, one can obtain the numerical value of $\frac{dg_\theta^{-1}(x)}{dx}$ by calculating $g'_\theta(g_\theta^{-1}(x))$ and taking the reciprocal. Since it is easy to differentiate $g_\theta(z)$, this derivative can be easily calculated once we obtained the value of $g_\theta^{-1}(x)$. The leads to a ways of calculating the probability density function of the g-and-k distribution with high numerical precision which is outlined in Algorithm A.2

Algorithm A.1: Bisection Algorithm, $\mathbf{Bisec}(f, a, b, \text{tol})$

Input:

- The function whose root is solved, $f(x)$
- The starting values, $a < b$, with $f(a)f(b) < 0$
- The precision, tol > 0.

Set $x = (a + b)/2$;
while $|f(x)| \geq$ *tol* **do**
  if $f(x)f(b) < 0$ **then**
    Set $a = x$;
    Set $x = (a + b)/2$;
  else if $f(x)f(a) < 0$ **then**
    Set $b = x$;
    Set $x = (a + b)/2$

Output: x, the solution of $f(x) = 0$ with precision tol

Algorithm A.2: Calculation of $l(x|\theta) = f_X(x|\theta)$ with given θ and x

Set $a = 0, b = 0$;
Set $f(z) = g_\theta(z) - x$ **while** $f(a)f(b) > 0$ **do**
  Set $a = a + 1$;
  Set $b = b + 1$;
Calculate $z =$ **Bisec**(f, a, b, tol) ;
return $\phi(z)|g'_\theta(z)|^{-1}$ *where* ϕ *denotes the density function of standard Normal distribution*

Now we are ready to devise the implement an MCMC sampler targeting the posterior distribution $\pi(\theta|\mathbf{x}) \propto \pi(\theta)\prod_{i=1}^{n} l(x_i|\theta) = \pi(\theta)l(\mathbf{x}|\theta)$. Given the current parameter state θ, one first sample $\theta' \sim q(\theta'|\theta)$ where q represents a certain proposal distribution. We then calculate $l(\mathbf{x}|\theta')$ and $l(\mathbf{x}|\theta)$ using Algorithm A.2. With probability

$$\alpha = \left\{1, \frac{l(\mathbf{x}|\theta')q(\theta|\theta')}{l(\mathbf{x}|\theta)q(\theta'|\theta)}\right\}$$

move to the state θ'. Otherwise, stay at the current state θ.