Contents

1 Introduction

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 on the discretely observed piecewise deterministic processes (PDPs). They 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 descriptions on the examples we consider in 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. 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 also have a sequence of known discrete times $0 = t_0 < t_1 < t_2 < ...$ and define $k_n := v_{t_n}$ to be the number of jumps before time t_n 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 $\zeta_{[0,t_n]}$ can be completely determined by $(K_n, \tau_{1:K_n}, \phi_{1:K_n}, \phi_0)$ and the deterministic function F^θ. For simplicity, we propose a Markovian prior on the jump times and jump values in the interval $[0, t_n]$, i.e.

$$p_n^\theta(k_n, \tau_{1:k_n}, \phi_{1:k_n}, \phi_0) = S^\theta(\tau_{k_n}, t_n)q_0^\theta(\phi_0)\mathbb{I}_{(0,t_n]}(\tau_{k_n})\mathbb{I}(0 < \tau_1 < \tau_2 < .. < \tau_{k_n} < t_n) \times \prod_{j=1}^{n} f^\theta(\tau_j|\tau_{j-1})g^\theta(\phi_j|\tau_j, \tau_{j-1}, \phi_{j-1}), \quad (1)$$

where $S^\theta(\tau_{k_n}, t_n) := 1 - \int_{\tau_{k_n}}^{t_n} f^\theta(s|\tau_{k_n})\, ds$ denotes the probability that no jump occurring in the interval $(\tau_{k_n}, t_n]$ 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_n, the number of jumps up to t_n in the posterior π_n. The reason why we included it in the posterior is that π_n can therefore be defined on the increasing subsets of the same space

$$\tilde{E}_n = \bigcup_{k=0}^{\infty} \big\{\{k\} \times T_{n,k} \times \Phi^{k+1}\big\}$$

where $T_{n,k} := \big\{(\tau_1, ..., \tau_k) \in (0, \infty)^k : 0 < \tau_1 < ... < \tau_k <= tn\big\}$.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 the real settings, 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

$$p^\theta(y_{(0,t_n]}|\zeta_{(0,t_n]}) = p^\theta(y_{(\tau_{k_n},t_n]}|\tau_{k_n},\phi_{k_n})\prod_{j=1}^{k_n} p^\theta(y_{(\tau_{j-1},\tau_j]}|\tau_{j-1},\phi_{j-1}) \tag{2}$$

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

$$\begin{aligned}\pi_n^\theta(\zeta_{(0,t_n]}|y_{(0,t_n]}) &= \pi_n^\theta(k_n,\tau_{1:k_n},\phi_{0:k_n}|y_{(0,t_n]}) = \gamma_n^\theta(k_n,\tau_{1:k_n},\phi_{0:k_n}|y_{(0,t_n]})/Z_n^\theta \\ &= p_n^\theta(k_n,\tau_{1:k_n},\phi_{1:k_n},\phi_0)p^\theta(y_{(0,t_n]}|\zeta_{(0,t_n]})/Z_n^\theta\end{aligned} \tag{3}$$

where Z_n^θ is the normalising constant, which is typically unknown. We will refer this posterior distribution as π_n in the future discussion when θ is assumed to be known. In practice, we are often interested in the Bayesian inferences based on the posterior distributions of the jumps defined in (3) with known static model parameters θ. It is a more common problem to make Bayesian inference on the static parameter θ. In this case, we assign a prior $\pi(\theta)$ to the parameters and try to find the posterior distribution of θ

$$\pi(\theta|y_{(0,t_n]}) = \int \pi(\theta)\pi_n^\theta(dk_n, d\tau_{1:k_n}, d\phi_{0:k_n}|y_{(0,t_n]})$$

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

2.1 Elementary Change-point Model

The first example we consider is the elementary change-point model. We assume that the inter-jump times are distributed according to a *Gamma* distribution with shape and scale parameter 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 1 shows the artificial data we simulated from the model with $\alpha = 4, \beta = 10$ and also $(\rho,\sigma_\phi^2,\sigma_y^2) := (0.9, 1.0, 0.5)$. Moreover, we take $\Delta := 1$ and $T := 1,0000$ for the simulated data.

2.2 Shot-noise Cox Model

Another model we considered is the *Short-noise Cox* model which was introduced by Cox and Isham(1980). It plays a central role at modelling the claims arrivals in the insurance market. The *Short-noice Cox* model can be treated as a generalised Poisson point process with more

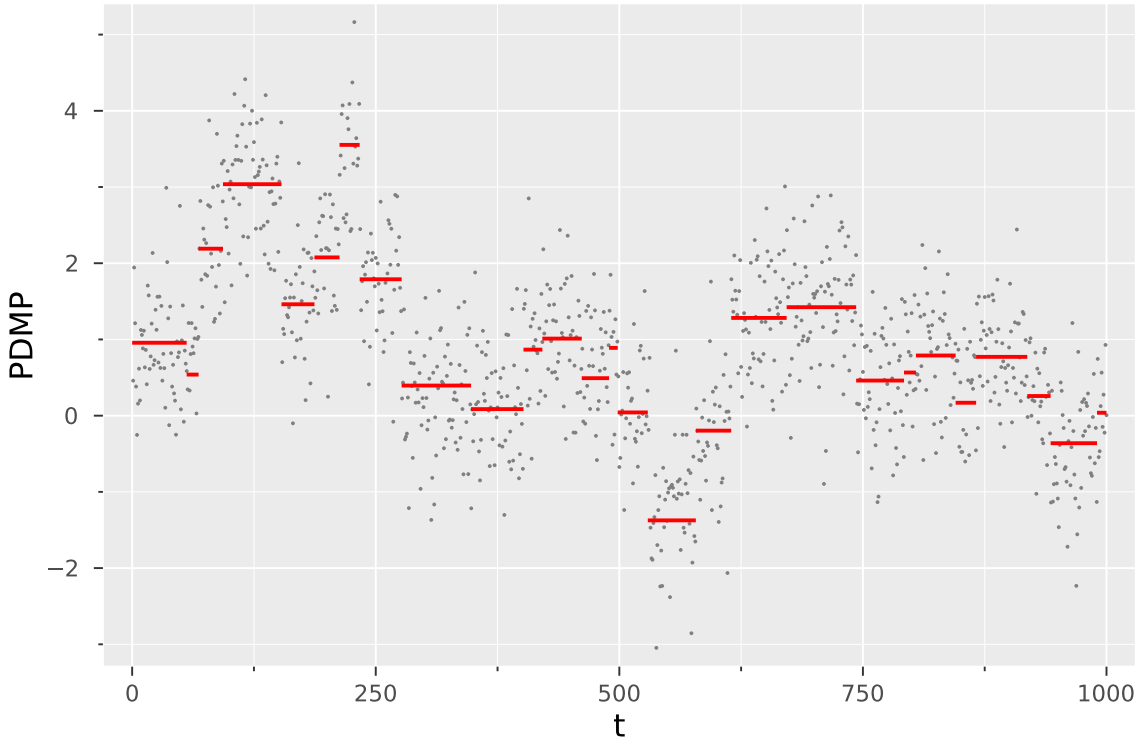

Figure 1: Simulated data of the change-point model described in this section The static parameters take value $(\rho, \sigma_\phi^2, \sigma_y^2, \alpha, \beta) := (0.9, 1.0, 0.5, 4.0, 10.0)$. Grey points are the noisy observations recorded at every $\Delta := 1$. The red lines represent the actual PDMP

flexibility on the intensity, allowing it to be a stochastic process as well. Let ζ_t be a short-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 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 τ_n, 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 shows an example of the intensity process and the corresponding observations simulated from the short-noise Cox model with $\lambda_\tau = 1/40, \lambda_\phi = 2/3$ and $\kappa = 1/100$. Particle filtering is then applied to infer the jump times and jump values embed in the intensity process.

3 Sequential Monte Carlo Methods

3.1 Standard Sequential Monte Carlo (SMC) samplers

Sequential Monte Carlo (SMC) methods is 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.

Our filtering method for PDPs is based on the standard Sequential Monte Carlo methods that can be used to approximate 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 unnomarlised 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, can be evaluated.

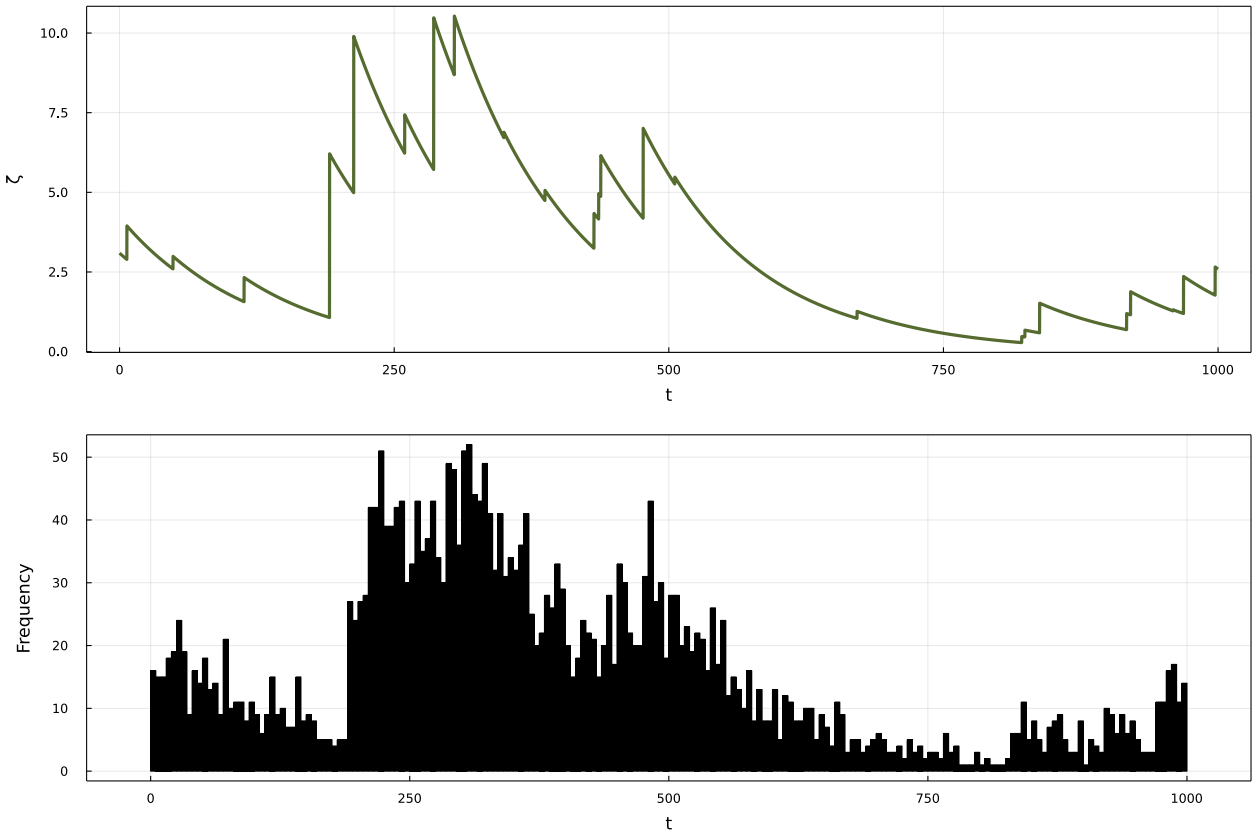

Figure 2: Data simulated from the short-noise Cox model. Static parameters used are $\theta := (\lambda_\tau, \lambda_\phi, \kappa) = (1/40, 2/3, 1/100)$. Top: The intensity process ζ_t of the Cox model. Bottom: histogram of the event times with bin width equal to 2.5.

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 $\left\{X^j_{1:n}, j = 1, 2, ..., N\right\}$ and their corresponding normalised weights $\{W^j_n, j = 1, 2, ..., N\}$. 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 $X^j_{1:n}$, the corresponding unnormalized weight w^j_n can be calculated by

$$w^j_n = \frac{\gamma_n\left(x^j_{1:n}\right)}{q_n\left(x^j_{1:n}\right)} \tag{4}$$

which can be normalized by $W^j_n := w^j_n / \sum_{i=1}^N w^i_n$. Based on the particles and the corresponding

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

$$\hat{\pi}_n(dx_{1:n}) := \sum_{j=1}^{N} W_n^j \delta_{x_{1:n}^j}(dx_{1:n}) \tag{5}$$

However, such a direct implementation of Importance Sampling method is in fact far beyond practical. To obtain good approximation of π_n from Importance Sampling, we should carefully design a proposal q_n that has heavier tail than the target and concentrate on regions of high density of the target. A poorly designed target will make the importance weight have infinite variance or make the method computationally inefficient. However, it is often very hard to design such a good proposal, especially when the target becomes high dimensional with the increase in n.

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 that sort of has 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 type of proposals, we divide the proposal problems 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

$$\begin{aligned} w_n^j &:= \frac{\gamma_n\left(x_{1:n}^j\right)}{q_n\left(x_{1:n}^j\right)} = \frac{\gamma_n\left(x_{1:n}^j\right)}{q_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)} \\ &= \frac{\gamma_{n-1}\left(x_{1:n-1}^j\right)}{q_{n-1}\left(x_{1:n-1}^j\right)} \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)} \\ &= w_{n-1}^j \frac{\gamma_n\left(x_{1:n}^j\right)}{\gamma_{n-1}\left(x_{1:n-1}^j\right) q_n\left(x_n^j | x_{1:n-1}^j\right)} \end{aligned} \tag{6}$$

We summarised Sequential Importance Sampling method in Algorithm 1.Note that at each step, the importance weights are obtained by adding 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.

Algorithm 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

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

However, Sequential Importance Sampling method also suffers from the problem that the estimation variance scales exponentially with the dimension of the problem. As n increases, the variances generally increases exponentially. If we look at the normalized weight at each step, we can see that the maximum unnormalized weight, $\max_{j=1,..,N} W_n^j$, will quickly becomes almost 1, making other weights approach to zero as n increases. As a consequence, 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 step Monte Carlo estimation. This results in the Sequential Monte Carlo methods. At the first step, Sequential Monte Carlo obtains

samples $X_1^{1:N} \sim q_1(dx_1)$ just like the Sequential Importance Sampling does. The importance weight at this iteration will then be given by

$$w_1^j := \frac{\gamma_1\left(x_1^j\right)}{q_1\left(x_1^j\right)} \tag{7}$$

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

$$\hat{\pi}_1\left(dx_1\right) := \sum_{j=1}^N W_1^j \delta_{X_1^j}\left(dx_1\right) \tag{8}$$

At later iterations of SMC, we sample $X_{1:n}^{1:N}$ from different proposals as 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 as a resampling step 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 with 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 being resampled for the j-th particle, $E\left\{\mathcal{N}_n^j\right\}$, 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

$$w_n^j := \frac{\gamma_n\left(\tilde{x}_{1:n-1}^j, x_n^j\right)}{\gamma_{n-1}\left(\tilde{x}_{1:n-1}^j\right) q_n\left(x_n^j|\tilde{x}_{1:n-1}^j\right)} \tag{9}$$

which is normalized by $W_n^j := w_n^j / \sum_{i=1}^N w_n^i$. 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,

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

Algorithm 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 $A_{n-1}^j \sim \mathbf{r}(\cdot|\mathbf{W}_{n-1})$ such that $r(j|\mathbf{W}_{n-1}) = W_{n-1}^j$;

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

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

SMC method is sometimes also referred as Sequential Importance Resampling method. Since resampling will introduce extra variance to the estimation of $\pi_n(dx_{1:n})$ as shown by **?**, it is generally preferable to use (5) 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.

The SMC method solves the problem of weight degeneracy. However, it still suffers from path degeneracy problem. When n increases, the new particles sampled will eventually share the same ancestor.

3.2 Variable rate particle filter (VRPF)

In this section, we describe the SMC filter for PDPs. We first introduce the particle filter named *variable rate particle filter* (VRPF) proposed by Godsill & Vermaak (2005). This is actually a standard SMC algorithm on PDPs with a reparameterised presentation of the PDPs.
Let $0 = t_0 < t_1 < t_2 < ...$, where $t_n, n > 0$, represents the n-th SMC step. 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

$$X_1 := (k_1, \tau_{1,1:k_1}, \phi_{1,1:k_1}, \phi_0) \tag{10a}$$
$$X_n := (k_n, \tau_{n,1:k_n}, \phi_{1,1:k_n}) \tag{10b}$$

These 'states' will take values in a subset of

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k_1} \times \Phi^{k_1+1} \right) \tag{11a}$$
$$E_n := \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(t_{n-1},t_n],k_n} \times \Phi^{k_n} \right) \tag{11b}$$

Define $\mathcal{J}_n := \left(\hat{k}_n, \hat{\tau}_{n,1:K_n}, \hat{\phi}_{n,0:K_n} \right)$ 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

$$\hat{\tau}_{n,1:\hat{k}_n} := \bigcup_{j=1}^{n} \left\{ \tau_{j,1:k_j} \right\} \tag{12a}$$
$$\hat{\phi}_{n,0:\hat{k}_n} := \{\phi_0\} \bigcup \left[\bigcup_{j=1}^{n} \left\{ \phi_{j,1:k_j} \right\} \right] \tag{12b}$$

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 have known that the piecewise deterministic process, $\zeta_{(0,t_n]}$, is completely determined by $\mathcal{J}_n$. Therefore, the target distribution, $\pi_n(x_{1:n}) := \gamma_n(x_{1:n})/Z_n$, defined on $E^n := \prod_{j=1}^n E_j$, is given by

$$\begin{aligned}\gamma_n(x_{1:n}) := S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right) \times & q\left(\hat{\phi}_0\right) \times g(y_{(0,t_n]}|\mathcal{J}_n) \\ & \times \prod_{j=1}^{\hat{k}_n} \left\{ f\left(\hat{\tau}_j|\hat{\tau}_{j-1}\right) q\left(\hat{\phi}_j|\hat{\phi}_{j-1}, \hat{\tau}_j, \hat{\tau_{j-1}}\right)\right\}\end{aligned} \tag{13}$$

Where we assume that $\hat{\tau}_0 := 0$. 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

$$K_n\left(x_n|x_{1:n-1}\right) = K_{n,1}\left(k_n|x_{1:n-1}\right) K_{n,2}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, x_{1:n-1}\right) \tag{14}$$

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 will be 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

$$G_n(x_{1:n}) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n,\hat{k}_n}, t_{n-1}\right)} \times \frac{g\left(y_{(t_{n-1},t_n]}|\mathcal{J}_n\right)}{K_{n,1}(0|x_{1:n-1})} \tag{15}$$

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

$$G_n\left(x_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathcal{J}_{n-1}\right) g\left(y_{(t_{n-1},t_n]}|\zeta_{(0,t_n]}\right)}{K_{n,1}(k_n|x_{1:n-1}) K_{n,2}(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, x_{1:n-1})} \tag{16}$$

where

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

The VRPF method is detailed in Algorithm 3. 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 would be likely to miss it since the observations in the block $(t_{n-1}, t_n]$ would provide little information about such a jump. Even when such jump is samlped, the corresponding sampled jump value would be likely to be inaccurate as well since the information that can be used to is minimal. Moreover, the inaccuracy caused by this problem cannot be corrected by any of the smoothing methods either. Realising this problem inspires us to propose modifications on 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.

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 newcoming observations. This inspires us to incorporate the block sampling strategy to perform such modifications. In this section, we are going to first introduce the block sampling strategy of Doucet et al. (2006). We will then describe the modifications we made on the VRPF to tackle the aforementioned limitations.

4.1 Block SMC samplers

Suppose that we have already obtained $\left\{W_{n-1}^i, X_{1:n-1}^i\right\}_{i=1,\dots,N}$ at step $n-1$. Under the standard SMC scheme, only $\{X_n^i\}_{i=1,..,N}$ are sampled at step n and the paths $X_{1:n}^i$ are constructed by concatenating X_n^i with $X_{1:n-1}^i$ 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 discrepency. As a consequence, the sampled paths $X_{1:n-1}^i$ 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 ways of alleviating such limitation of standard SMC scheme. Instead only sampling X_n^i's at step n, part of the previous paths are also sampled in light of $\pi_n(x_{1:n})$. Suppose that $X_{n-L:n}^{',i}$ 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, $X_{1:n}$ will then be obtained by discarding $X_{n-L:n-1}$ from $X_{1:n-1}$ and adding $X_{n-L:n}^{'}$ to it, i.e.

$$X_{1:n} := (X_{1:n-L-1}, X_{n-L:n}^{'}) \tag{17}$$

Algorithm 3: 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 (15);
    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 (16);
  for $i = 1, 2, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

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

If we define $\mathcal{Q}_n(x_{1:n})$ to be the proposed density of the path $X_{1:n}$ at step n, then $\mathcal{Q}_n(x_{1:n})$ is given by

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

Therefore, when the path is propagated from $X_{1:n-1}$ to $X_{1:n}$, importance weight will become

$$W_n^i \propto \frac{\gamma_n(x_{1:n})}{p_n(x_{1:n})} = \frac{\gamma_n(x_{1:n})}{\int p_{n-1}(x_{1:n-1})K_n(x'_{n-L:n}|x_{1:n-1})dx_{n-L:n-1}}$$

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 though it is possible 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 the notations at later stage, we also define Z_n to be the variable(s) sampled at step n. Hence,

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

This allows us to transform between the X's and Z's by Equation (18)

$$Z_{n,k} := X_{n-k+1}(k) \qquad X_n(j) := Z_{n+j-1,j} \tag{18}$$

where $Z_{n,k}$ represents the kth element in Z_n. Based on the previous discussion, the proposal distribution of all these variables will be

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

where the conditional path $x_{1:l-1}$ is propagated and obtained by (17). 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

$$\begin{aligned}&\tilde{\pi}_n(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) := \gamma_n(x_1(L+1), ..., x_{n-L}(L+1), x_{n-L+1}(L), ..., x_n(1))\times \\ &\prod_{t=2}^{L}\lambda_t(x_1(t-1), x_2(t-2), ..., x_{t-1}(1)|z_{1:t})\prod_{t=L+1}^{n}\lambda_t(x_{t-L}(L), x_{t-L+1}(L-1), ..., x_{t-1}(1)|z_{1:t})\end{aligned} \tag{19}$$

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

$$\tilde{\pi}_n(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) := \gamma_n(x_1(n), x_2(n-1), ..., x_{n-1}(2), x_n(1)) \times \prod_{t=2}^{n} \lambda_t(x_1(t-1), x_2(t-2), ..., x_{t-1}(1)|x_{1:t}) \quad (20)$$

Moreover, the extended target density incorporates the actual target as a marginal. One can see that by targeting the extended density defined in (19) and (20), we can get rid of the integral appearing in the previous set up and the importance weight will become

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

Hence, the incremental weight for the SMC with block sampling strategy is then given by

$$\mathcal{G}_n(z_{1:n}) := \frac{\tilde{\gamma}_n(z_{1:n})\mathcal{Q}_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}(z_{1:n-1})\mathcal{Q}_n(z_{1:n})} = \frac{\tilde{\gamma}_n(z_{1:n})}{\tilde{\gamma}_{n-1}(z_{1:n-1})K_n(z_n|z_{1:n-1})} \quad (21)$$

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

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

$$\times \frac{\lambda_n(x_{n-1}(1), ..., x_{n-L}(L)|x_{1:n-L}(L+1), x_{n-L+1}(L), .., x_{n-1}(2), x_n(1))}{K_n(x_n(1), .., x_{n-L}(L+1)|x_{1:n-L-1}(L+1), x_{n-L}(L), .., x_{n-2}(2), x_{n-1}(1))} \quad (22)$$

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

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

$$\frac{\lambda_n(x_1(n-1), .., x_{n-1}(1)|x_1(n), x_2(n-1), .., x_{n-1}(2), x_n(1))}{K_n(x_n(1), ..., x_1(n)|x_1(n-1), x_2(n-2), ..x_{n-1}(1))} \quad (23)$$

If the resampling steps are also included in this scheme, the weights at step n will then be equal to the incremental weights only. Details of the Block SMC method are given in Algorithm 4. 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 (22) and (23). Doucet et al. (2006) derived the following two propositions that provide us guides on how to choose these densities.

Algorithm 4: 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 (23);
  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 (22);

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 (22) is given by

$$\lambda_n^{opt}(x_{n-L:n-1} | x_{1:n-L-1}, x'_{n-L:n}) = \frac{\gamma_{n-1}(x_{1:n-1}) K_n(x'_{n-L:n} | x_{1:n-1})}{\int \gamma_{n-1}(x_{1:n-1}) K_n(x'_{n-L:n} | x_{1:n-1}) \, dx_{n-L:n-1}} \tag{24}$$

where we use represent $(x_{1:n-L-1}(L+1), x_{n-L}(L), .., x_{n-2}(2), x_{n-1}(1)))$ and $(x_n(1), x_{n-1}(2), ..., x_{n-L}(L+1))$ as $x_{1:n-1}$ and $x'_{n-L:n}$ respectively for the ease of presentation. The optimal conditional distribution that minimizes the incremental weight of (23) will be given in similar form.

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

$$K_n^{opt}(x'_{n-L:n} | x_{1:n-1}) = \pi_n(x'_{n-L:n} | x_{1:n-L-1}) \tag{25}$$

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 would need to choose λ_n that approximates (24). Moreover, the optimal proposal distribution defined in (25) 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.

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 block 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 the jumps that are already sampled. 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 the ease of representation, we will constrain ourselves to 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. Hence, we also limit our presentation to the resampling of one block only.

Let $u_{n-1} := \left(\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$ be the modified jump time and its 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 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 $X_{n-1}(2) :=$

$\left(\bar{k}_{n-1}, \bar{\tau}_{n-1,1:\bar{k}_{n-1}}, \bar{\phi}_{n-1,1:\bar{k}_{n-1}}\right)$ to represent $X_{n-1}(1)$ after modification, where $\bar{k}_{n-1}$ the number of jumps contained in the $X_{n-1}(2)$.

Therefore, it is clear to see that, according to the modifications we are proposing,

$$X_{n-1}(2) := \left(k_{n-1}+1, \tau_{n-1,1:k_{n-1}}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}}, \mathring{\phi}_{n-1}\right) \tag{26a}$$

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

$$X_{n-1}(2) := \left(k_{n-1}, \tau_{n-1,1:k_{n-1}-1}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}-1}, \mathring{\phi}_{n-1}\right) \tag{26b}$$

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 (26b).

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, *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 the notation, we define

$$Z_1 := X_1(1) \tag{27a}$$
$$Z_n := (M_{n-1}, U_{n-1}, X_n(1)) \tag{27b}$$

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

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k} \times \Phi^{k_1+1}\right) \tag{28a}$$

and

$$\begin{aligned} E_n := &\left[\left(\{0\} \times T^M_{(\tau_{n-1,k_{n-1}-1} \vee t_{n-2}, t_{n-1}]} \times \Phi\right) \bigcup \left(\{1\} \times T^M_{(\tau_{n-1,k_{n-1}} \vee t_{n-2}, t_{n-1}]} \times \Phi\right)\right] \\ &\times \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(0,t_n],k_n} \times \Phi^{k_n}\right) \end{aligned} \tag{28b}$$

where $T^M_{(s,t]} := \{\tau | \tau \in (s,t]\}$. Moreover, denote $\bar{U}_{n-1}$ the jump time and value in the interval $(t_{n-1}, t_n]$ that are modified at 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

$$\begin{aligned} Z_{1:n} &:= (X_1(1), M_1, U_1, ..., X_{n-1}(1), M_{n-1}, U_{n-1}, X_n(1)) \\ &:= \left(X_1(2), M_1, \bar{U}_1, ..., X_{n-1}(2), M_{n-1}, \bar{U}_{n-1}, X_n(1)\right) \end{aligned} \tag{29}$$

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

4.2.1 Target Density

In the following, we are going to present the extended target distribution of the algorithm. Since we have the one-to-one correspondence in (29), 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 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

$$\bar{\gamma}_n\left(Z_{1:n}\right) := \gamma_n(x_{1:n-1}(2), x_n(1))\mu_n\left(m_{1:n-1}|x_{1:n-1}(2), x_n(1)\right) \times \prod_{j=1}^{n-1} \lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:n-1}(2), x_n(1)\right) \quad (30)$$

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 of integral evaluation, as discussed previous section. Similarly, denote $\mathcal{J}_n := \left(\hat{k}_n, \hat{\tau}_{n,1:\hat{k}_n}, \hat{\phi}_{n,1:\hat{k}_n}\right)$ be the jump times and values that defining 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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \quad (31a)$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}, \hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \quad (31b)$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \quad (31c)$$

Here, we use the convention that if $k_n = 0$, then Equation (31a), (31b) and (31c) 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.

4.2.2 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 to 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 an 'birth' is proposed, then U_{n-1} will be proposed according to

$$\begin{aligned} K_{n,2}\left(u_{n-1}|m_{n-1}=1, z_{1:n-1}\right) :=& \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,1}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,1}\left(\mathring{\phi}_{n-1}|\mathcal{J}_{n-1}\right) \end{aligned} \tag{32a}$$

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

$$\begin{aligned} K_{n,2}\left(u_{n-1}|a, z_{1:n-1}\right) :=& \mathbb{1}\left(k_{n-1} \geq 1\right) \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}-1} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,0}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,0}\left(\mathring{\phi}_{n-1}|\mathcal{J}_{n-1}\right) \\ & + \mathbb{1}(k_{n-1}=0) \times \delta_{\emptyset}(u_{n-1}) \end{aligned} \tag{32b}$$

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.

$$\begin{aligned} K_{n,3}\left(x_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) :=& K^1_{n,3}\left(k_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) \\ & \times K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1}, u_{n-1}, z_{1:n-1}\right) \end{aligned} \tag{33}$$

4.2.3 Auxiliary Densities

As discussed before, the auxiliary densities are included to ensure that the importance weight will have the correct form to incorporate γ_n as marginals. However, when choosing these auxiliary densities, we need to ensure that the support of them are 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 (30), we propose a factorizable 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, $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right)$, is also subject to our choices. 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 $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right)$ becomes degenerate and equals to 1 in these cases. When $M_j = 0$ and $k_j \geq 1$, $\lambda_j\left(\bar{u}_j|m_{1:j}, x_{1:j}(2)\right) := \lambda_j\left(\hat{\tau}_{j,\hat{k}_j}, \hat{\phi}_{j,\hat{k}_j}|m_{1:j}, x_{1:j}(2)\right)$ will 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.

4.2.4 Incremental Weight

The incremental weight, $G_n\left(z_{1:n}\right) := \hat{\gamma}_n\left(z_{1:n}\right) / \left[\hat{\gamma}_{n-1}\left(z_{1:n-1}\right) K_n\left(z_n|z_{1:n-1}\right)\right]$ will be 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 $k_{n-1} = 0$

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

$$\times g\left(y_{(t_{n-1},t_n]}|\mathcal{J}_n\right) \times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right)}{K^1_{n,3}\left(k_n|0, \emptyset, z_{1:n-1}\right) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 0, \emptyset, z_{1:n-1}\right)} \quad (34a)$$

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

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

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

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

$$\times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|0, u_{n-1}, z_{1:n-1}) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 0, u_{n-1}, z_{1:n-1}\right)} \quad (34b)$$

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

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

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

$$\times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|m_{n-1}=1, u_{n-1}, z_{1:n-1}) K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1}=1, u_{n-1}, z_{1:n-1}\right)} \tag{35}$$

In Equation (34a), (34b) and (35), $h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$ and $K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 1, u_{n-1}, z_{1:n-1}\right)$ will all become 1 when $k_n = 0$. We omit the explicit representation of these two situations here since they are actually the special cases of the above representations. The BlockVRPF method is outlined in Algorithm 5. The sampling schemes are not specifically outlined according to each case. More specifically, U^i_{n-1} in line 12 of Algorithm 5 will become $\emptyset$ when $M^i_{n-1} = 0$ and $k^{A^i_{n-1}}_{n-1} = 0$. BlockVRPF sampler we proposed 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 neat the end of a block in the actual process. This is illustrated in Figure 3. One can see that there is a jump occurring near the end of the first time block in the actual PDMP and this is missed by the VRPF sampler completely. One 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).

4.3 Comparison between VRPF and BlockVRPF Samplers

To compare the performances of VRPF and BlockVRPF samplers, we apply both samplers to the simulated data of the elementary change-point model shown in Figure 4, which are the first 500 observations shown in Figure 1. In addition, we deliberately discretize the time so that there are jumps near the end of each time block. The time discretisations are also shown in Figure 4. For both algorithms, the number of jumps in each time block are 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 BlockVRPF algorithm, a *birth* move will be proposed with probability $1 - S(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1})$. Otherwise, an *adjustment* will be proposed instead. For a *birth* move, the

Algorithm 5: 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 (31a), (31b) or (31c) based on $\mathcal{J}_{n-1}^{A_{n-1}^i}$;
    Calculate the incremental weight $G_n\left(z_{1:n}^i\right)$ according to (34a), (34b) or (35);
  for $n = 1, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

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

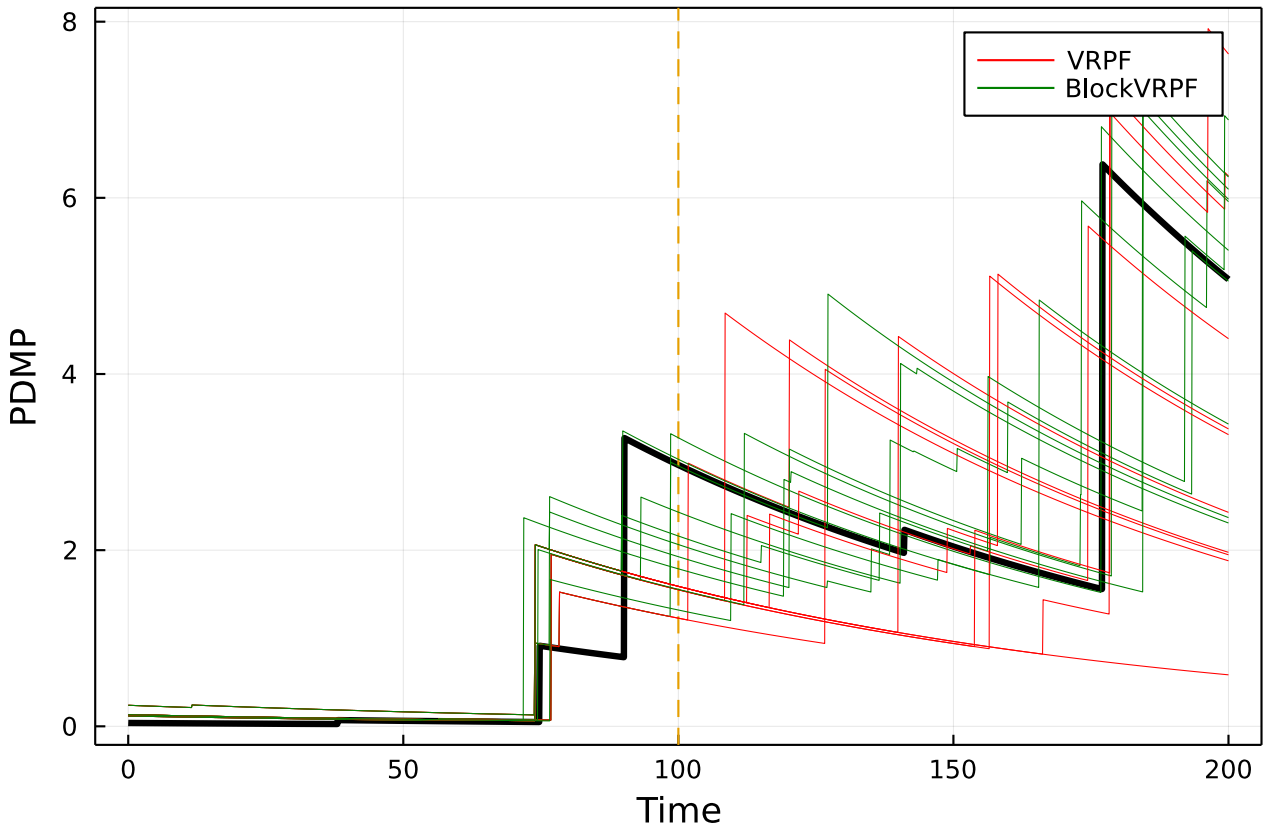

Figure 3: Illustration of the correcting power of BlockVRPF sampler. Black line represents the actual PDMP we are interested. Given the same sampled jumps in the first block (time blocks are splitted by the yellow dashed line), both VRPF and BlockVPRF sampler were used to sample the jumps in the second block. These sampled processes are represented by red lines (VRPF) and green lines (BlockVRPF)

new jump times will be proposed uniformly between $\hat{\tau}_{n-1,\hat{k}_{n-1}}$ and t_{n-1}. For an *adjustment*, a modified jump time will be proposed from a Gaussian distribution centered 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. Resampling is also done 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 method. Figure 5 shows

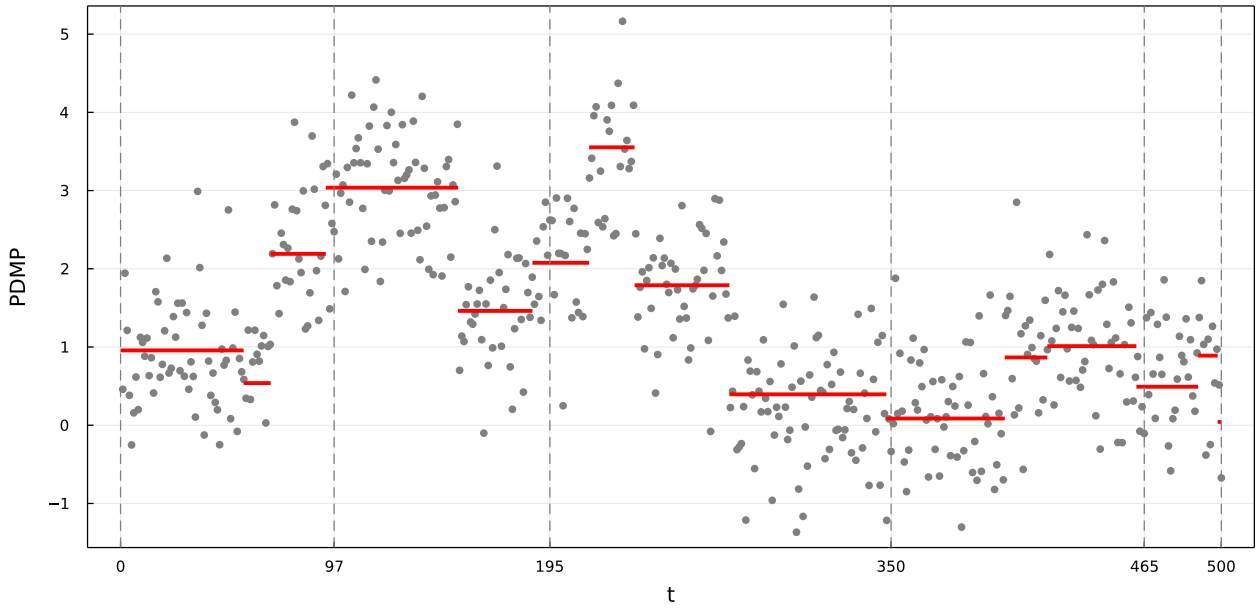

Figure 4: Data used in the simulations discussed in section 5.1. Grey dotted lines represent the time discretisations used in the simulations.

the sampled jump times obtained from both methods just before and after $t = 350$, which is one of the time discretisation point. We can see that 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 5. On the contrary, BlockVRPF successfully detect the jump near $t = 350$ and places higher weights in the region around that jump as well. Figure 6 shows the histograms of the nearest sampled jump times to the actual one obtained from both methods. It is clear to see 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 VRPF method is in a region centered at $t = 400$. This suggests that BlockVRPF are more robust to find and recover 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 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 to find 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 BlockVRPF method is more robust comparing to the VRPF method. However, one may also notice that the calculation of incremental weights for the BlockVRPF sampler involve a likelihood ratio of 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 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.

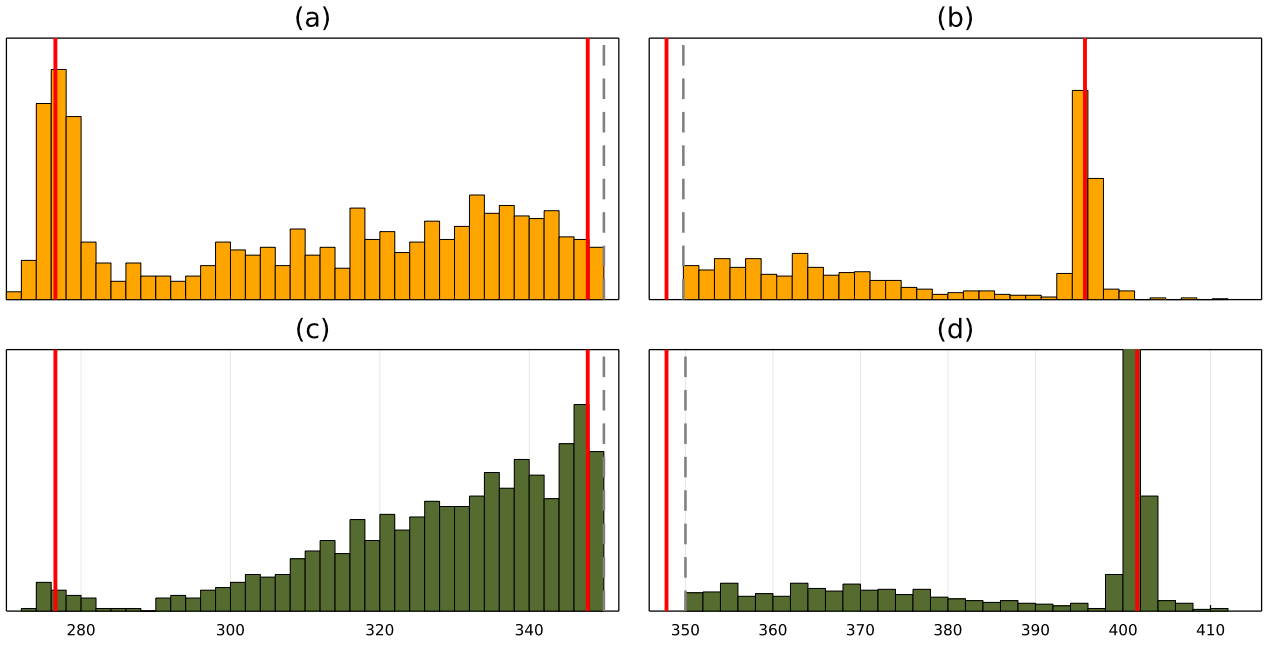

Figure 5: Histograms showing the sampled jump times just before and after a time discretisation point (represented by the grey dashed line). Left: last sampled jump times before the discretisation. Right: first sampled jump times after the discretisation. Results obtained from VRPF and BlockVRPF samplers 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 parameters 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

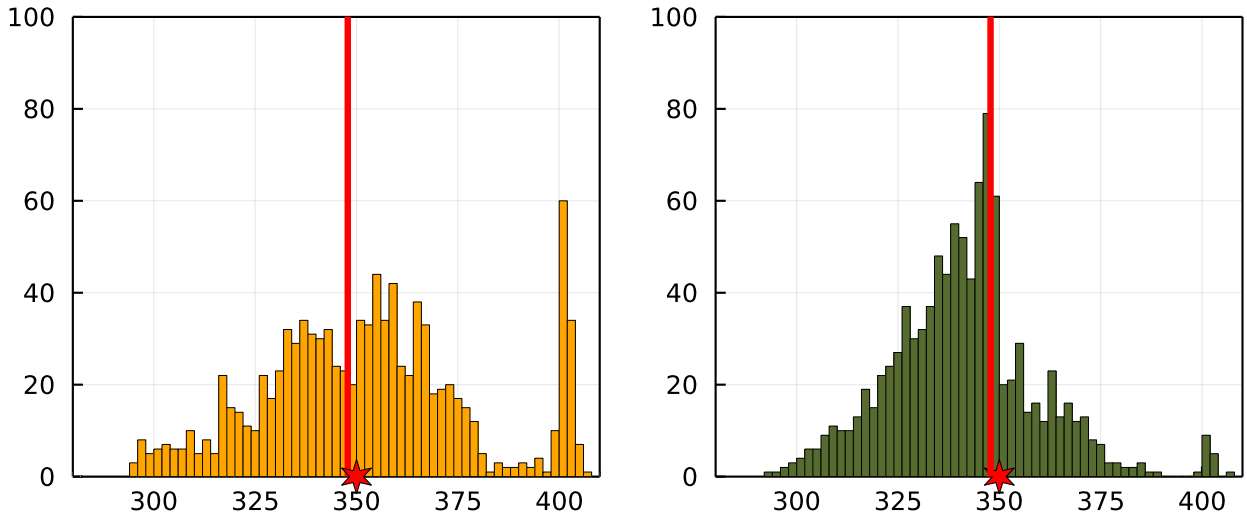

Figure 6: Histograms of the sampled nearest jump times to the jump just before $t = 350$ obtained from both methods. Left: results obtained from VRPF sampler. Right: results obtained from BlockVRPF sampler. Red line represents the actual jump times and red start represents the time discretisation point.

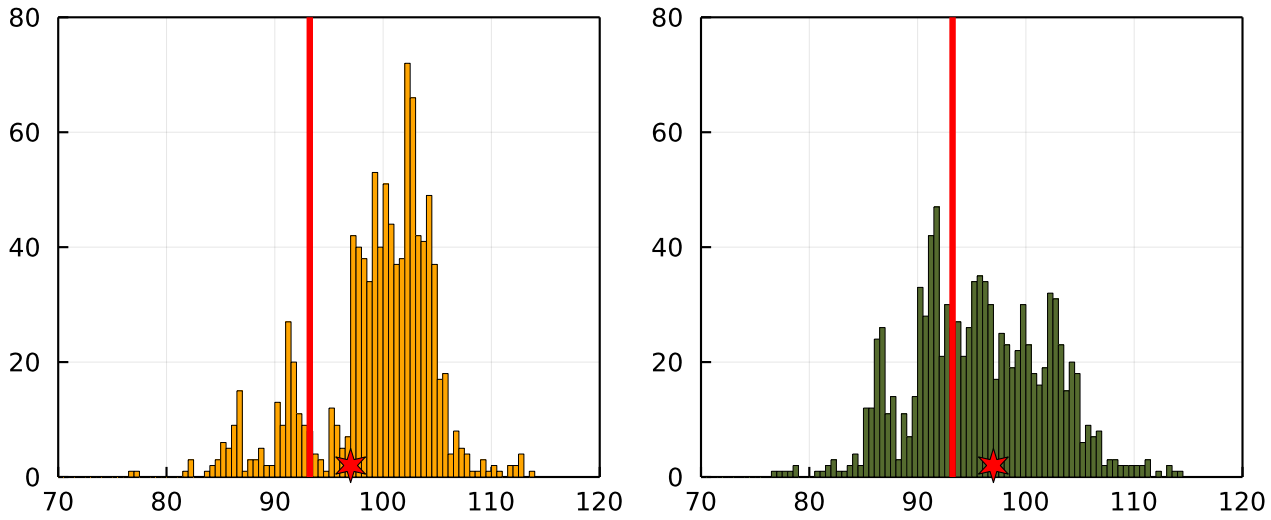

Figure 7: Histograms of the sampled nearest jump times to the jump just before $t = 97$ obtained from both methods. Left: results obtained from VRPF sampler. Right: results obtained from BlockVRPF sampler. Red line represents the actual jump times and red start represents the time discretisation point.

BlockVRPF method to obtain estimations of the posterior distributions of the static parameters for the two models considered in this work.

5 Static Parameter Estimations 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 is known. In this section, we are trying to make inferences on these static parameters in the piecewise deterministic processes.

There has been many Bayesian methods based around the SMC sampler proposed by various authors to estimate the static parameters of Markov models, such as Neal (2001) and Chopin (2002). More recently, Andrieu et al. (2010) and Chopin et al. (2013) showed that SMC methods can be combined with Markov Chain Monte Carlo (MCMC) methods to make estimations on the static parameters in state-space models.

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

$$p(y_{1:P}|\theta) = \int p(x_{1:P}, y_{1:P}|\theta)dx_{1:P} = \int p(x_{1:P}|\theta)p(y_{1:P}|x_{1:P}, \theta)dx_{1:P}, \tag{36}$$

which is intractable in most scenario. 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 a 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 to its target density by

$$\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P}) := \sum_{n=1}^{N} W_P^n \delta_{x_{1:P}^n}(dx_{1:P}), \tag{37}$$

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 (37) 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 acceptance probability of the MH update requires one to compute the proposal density $q(dx_{1:P})$ that is in this case given by

$$q(x_{1:P}) := \mathbb{E}_{W_{1:P}^{1:N}, x_{1:P}^{1:N}} \left[\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P})\right] \tag{38}$$

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

$$\psi_P^{N,\theta}(\mathbf{x}_1, ..., \mathbf{x}_P, \mathbf{a}_1, ..., \mathbf{a}_{P-1}) := \left\{\prod_{j=1}^{N} K_1^\theta\left(x_1^j\right)\right\} \prod_{n=2}^{P} \left\{r^\theta\left(\mathbf{a}_{n-1}|\mathbf{W}_{n-1}\right) \prod_{j=1}^{N} K_n^\theta\left(x_n^j | x_{1:n-1}^{a_{n-1}^j}\right)\right\} \tag{39}$$

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

$$\tilde{\pi}_P(\theta, b_{1:P}, \mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) = \frac{\pi_P(\theta, x_{1:P}^{b_{1:P}}, b_{1:P})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}}|\theta, x_{1:P}^{b_{1:P}}, b_{1:P}) \tag{40}$$

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 (40) 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 (40) 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

$$\begin{aligned}
\psi_P^{N,\theta}\left(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}}|x_{1:P}^{b_{1:P}}, b_{1:P}\right) &:= \frac{\psi_P^{N,\theta}(\mathbf{x}_{1:P}, \mathbf{a}_{1:P})}{K_1\left(x_1^{b_1}\right) \prod_{n=2}^P \left\{r\left(b_{n-1}|\mathbf{W}_{n-1}\right) K_n\left(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}\right)\right\}} \\
&= \left\{\prod_{1\leq j\leq N, j\neq b_1} K_1\left(x_1^j\right)\right\} \prod_{n=2}^P r\left(\mathbf{a}_{n-1}^{-b_n}|\mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \\
&\quad \times \prod_{1\leq j\leq N, j\neq b_n} K\left(x_n^j|x_{1:n-1}^{a_{n-1}^j}\right)
\end{aligned} \tag{41}$$

Since Equation (41) 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 gives actaully inspires the idea of partible 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 'collapsed' Gibbs sampler which was discussed in Liu (2001) and Van Dyk & 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 & 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 conditional SMC (cSMC) sampler and was first introduced in Andrieu et al. (2010). The details of the sampler is listed in Algorithm 6.

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 & 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

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

Algorithm 6: 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 $A_{n-1}^j \sim r(\cdot|\mathbf{W}_{n-1})$;
      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 $W_n^j \propto w_n^j$;

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

$$\begin{aligned}
\pi_P^\theta(x_{1:P}^{b_{1:P}}) &:= w_1^{b_1} K_1(x_1^{b_1}) \prod_{n=2}^{P} w_n^{b_n} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \\
&= \left\{ \frac{w_1^{b_1}}{\sum_{i=1}^N w_1^i} K_1(x_1^{b_1}) \prod_{n=2}^{P} \frac{w_n^{b_n}}{\sum_{i=1}^N w_n^i} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\} \\
&= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^{P} W_{n-1}^{b_{n-1}} K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\} \\
&= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^{P} r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^{P} \sum_{i=1}^{N} w_n^i \right\}
\end{aligned} \tag{42}$$

If we substitute this into (40), 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 completes one sweep of the particle Gibbs sampler and its full implementation is listed in Algorithm 7.

Algorithm 7: 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 6;
  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 $X_{1:P}^{(n)} = \left(X_1^{(n),B_1^{(n)}}, ..., X_P^{(n),B_P^{(n)}}\right)$;

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 likelihood to have mixing issues. In the particle Gibbs sampler we discussed before, we only samples the ancestor index at P according to the importance weight $\mathbf{W}_P$, and traces 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 a 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 longer time and 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 a poor mixing of the Gibbs sampler. One way to solve this problem was proposed by Whiteley (2010) and further explored later by Lindsten & 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

$$\psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}}|x_{1:P}^{b_{1:P}}, b_{1:P}) := \prod_{\substack{j=1 \\ j\neq b_1}}^{N} K_1\left(x_1^j\right) \prod_{n=2}^{t} \left\{ r\left(\mathbf{a}_{n-1}^{-b_n}|\mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \prod_{\substack{j=1 \\ j\neq b_n}}^{N} K_n(x_n^j|x_{1:n-1}^{a_{n-1}^j}) \right\} \tag{43}$$

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

$$\begin{aligned}
&\tilde{\pi}_P^N(b_t^*|\theta^*, \mathbf{x}_{1:t}^{*,-b_{1:t}}, \mathbf{a}_{1:t-1}^{*,-b_{2:t}}, b_{1:t}, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}, b_{t+1:P}^*) \\
&= \frac{\pi_P(\theta^*, b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}}|b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}) \\
&= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})\pi(\theta^*)}{N^P} \prod_{\substack{j=1 \\ j\neq b_1}}^{N} K_1\left(x_1^j\right) \prod_{n=2}^{t} \left\{ r\left(\mathbf{a}_{n-1}^{-b_n}|\mathbf{W}_{n-1}, a_{n-1}^{b_n}\right) \prod_{\substack{j=1 \\ j\neq b_n}}^{N} K_n(x_n^j|x_{1:n-1}^{a_{n-1}^j}) \right\} \\
&= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi(\theta^*)W_t^{b_t}}{N^P} \psi_t^{N,\theta^*}(\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \left\{ \prod_{n=1}^{t} \sum_{i=1}^{N} w_n^i \right\} \propto W_t^{b_t} \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})}
\end{aligned} \tag{44}$$

where the last line is derived in a similar way as we did in (42). 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 8

Algorithm 8: 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 6, 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 PGBS sampler generally has much better than the standard particle Gibbs sampler. Due to the addition of the backward sampling step, we would be 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 would be highly likely significantly different from the previous one. This brings a big improvement on the mixing in the state space, hence the particle Gibbs sampler overall.

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

$$\mathcal{G}_{n|P}(x_{1:n}^{b_{1:n}}) := \frac{f(\check{\tau}_{n+1,1}^*|\hat{\tau}_{n,\hat{k}_n}) g(\check{\phi}_{n+1,1}^*|\hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}_{n+1,1}^*) p(y_{(t_n,\check{\tau}_{n+1,1}^*]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \tag{45}$$

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

$$\mathcal{G}_{n|P}(x^{b_{1:n}}_{1:n}) = \frac{S(\hat{\tau}_{n,\hat{k}_n}, t_P)p(y_{(t_n,t_P]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \tag{46}$$

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 $Z^{b^*_{n+1}}_{n+1}, .., Z^{b^*_P}_P$ and the corresponding $\check{\mathcal{J}}^*_n$, if $M^*_n = 1$, the corresponding backward sampling incremental weight would be given by

$$\mathcal{G}_{n|P}(z^{b_{1:n}}_{1:n}) := \frac{\mathbb{I}(\mathring{\tau}^*_n > \hat{\tau}_{n,\hat{k}_n})f(\mathring{\tau}^*_n|\hat{\tau}_{n,\hat{k}_n})g(\mathring{\phi}^P_n*|\hat{\phi}_{n,\hat{k}_n}, \mathring{\tau}^*_n, \hat{\tau}_{n,\hat{k}_n})\mu(M^*_n|z_{1:n}, z^{b^*_{n+1:P}}_{n+1:P})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)p(y_{(\mathring{\tau}^*_n,t_n]}|\hat{\phi}_{n,\hat{k}_n})} \tag{47}$$

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

$$\mathcal{G}_{n|P}(z^{b_{1:n}}_{1:n}) := \frac{\mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1})S(\hat{\tau}_{n,\hat{k}_n}, t_P)\mu(M^*_n|z_{1:n}, z^{b^*_{n+1:P}}_{n+1:P})p(y_{(t_n,t_P]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \tag{48}$$

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

$$\mathcal{G}_{n|P}(z^{b_{1:n}}_{1:n}) := \frac{f(\check{\tau}^*_{n+1,1}|\hat{\tau}_{n,\hat{k}_n})g(\check{\phi}^*_{n+1,1}|\hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}^*_{n+1,1})p(y_{(t_n,\check{\tau}^*_{n+1,1}]}|\hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}$$
$$\times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1})\mu(M^*_n|z_{1:n}, z^{b^*_{n+1:P}}_{n+1:P}) \tag{49}$$

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

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

$$\times \frac{\mu(M^*_n|z_{1:n}, z^{b^*_{n+1:P}}_{n+1:P})\lambda(\hat{\tau}_{n,\hat{k}_n}, \hat{\phi}_{n,\hat{k}_n}|z_{1:n}, z^{b^*_{n+1:P}}_{n+1:P})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} > t_{n-1}, \mathring{\tau}^*_n > \hat{\tau}_{n,\hat{k}_n-1}) \tag{50}$$

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

$$\bar{\gamma}^\theta_t(Z_{1:t}) := \gamma^\theta_t(x_{1:t-1}(2), x_t(1))\mu^\theta_t(m_{1:t-1}|x_{1:t-1}(2), x_t(1))$$
$$\times \prod_{j=1}^{t-1} \lambda^\theta_j(\bar{u}_j|m_{1:j}, x_{1:j}(2), x_{j+1}(1)) \tag{51}$$

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 (51). 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 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 & 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 affecting 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 8.

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 9. 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 (34a), (34b) and (35), 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.

Algorithm 9: 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 $\mathcal{J}_P^{(n)}$

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 make sure that the incremental weights after rejuvenation is not too large. In the numerical studies of this chapter, we propose to use a Gaussian kernel centered 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 on the mixings.

5.4 Numerical Examples

In this section, we are going to apply the particle Gibbs sampler with both VRPF and BlockVRPF method to perform Bayesian inferences on two challenging examples discussed in section 2 - the *Elementary Change-point Model* and the *Short-noise Cox Model*. We follow what we have done in section 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 burnins.

5.4.1 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 $Bernoulli(0.1)$. When $m_n = 0$, the corresponding backward kernel for the jump times and values will be Gaussian kernels centered 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 $Diag(10, 10^2, 10)$, truncated in the region $(0, \infty)^3$. We ran both VRPF-PG and BlockVRPF-PG samplers with 10, 50 and 100 particles. Figure 8 shows the estimated marginal posterior distributions obtained from both algorithms. One can see that the BlockVRPF-PG could 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 estimations. This can be viewed as an advantage of BlockVRPF-PG sampler comparing to the algorithm proposed by Finke et al. (2014), for which discretisations do affect the estimation 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 accuracy of such approximation depends on the time discretisations. This was also shown in the simulation results on the same model presented in Finke et al. (2014). For 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 BlockVRPF-PG sampler is easier.

Figure 9 shows the autocorrelations of the two algorithms. One can see that using the same number of particles, VRPF-PG has smaller auto correlations comparing to BlockVRPF-PG

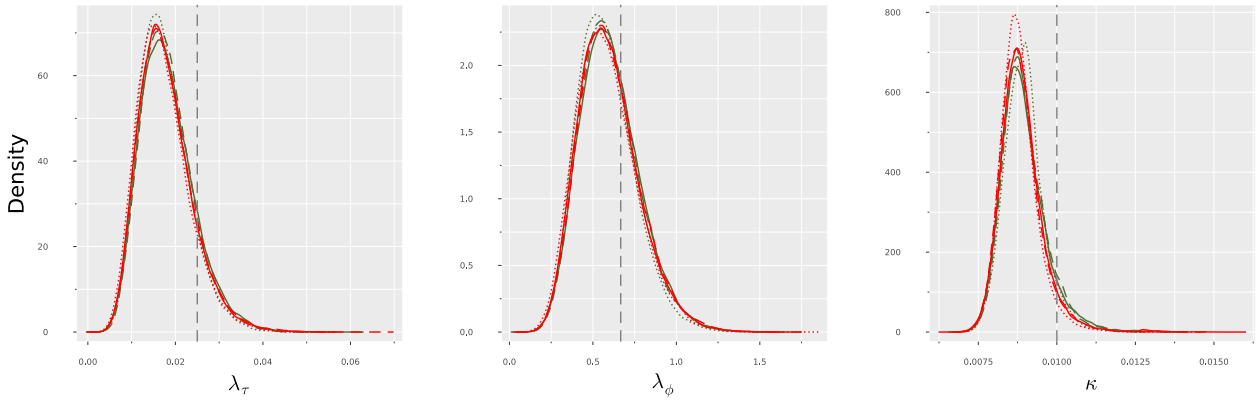

Figure 8: Kernel density estimations of the posteriors of the static parameters in short-noise Cox model. Results obtained from VRPF-PG are in red colour and results obtained from BlockVRPF-PG are in green colour. Results obtained by using 10, 50 and 100 particles are represented by dot, dashed and solid lines respectively. Grey vertical dashed lines represent the true parameter values.

sampler. This is possibly due to the suboptimal choice of backward kernels in the BlockVRPF sampler, as discussed in section 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.

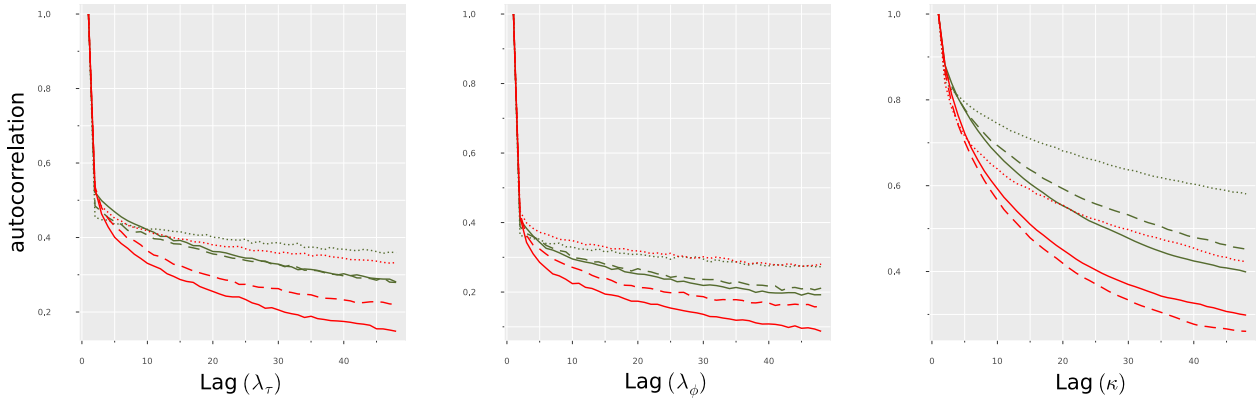

Figure 9: Autocorrelations obtained from both algorithms. Results obtained from VRPF-PG and BlockVRPF-PG samplers are in red and green colour respectively. Dot, dashed and solid lines represent the results obtained by using 10, 50 and 100 particles.

5.4.2 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 $\text{Diag}(10^2, 10^2, 10, 10^3, 10^2)$, truncated in the region $\mathbb{R} \times (0, \infty)^4$. Figure

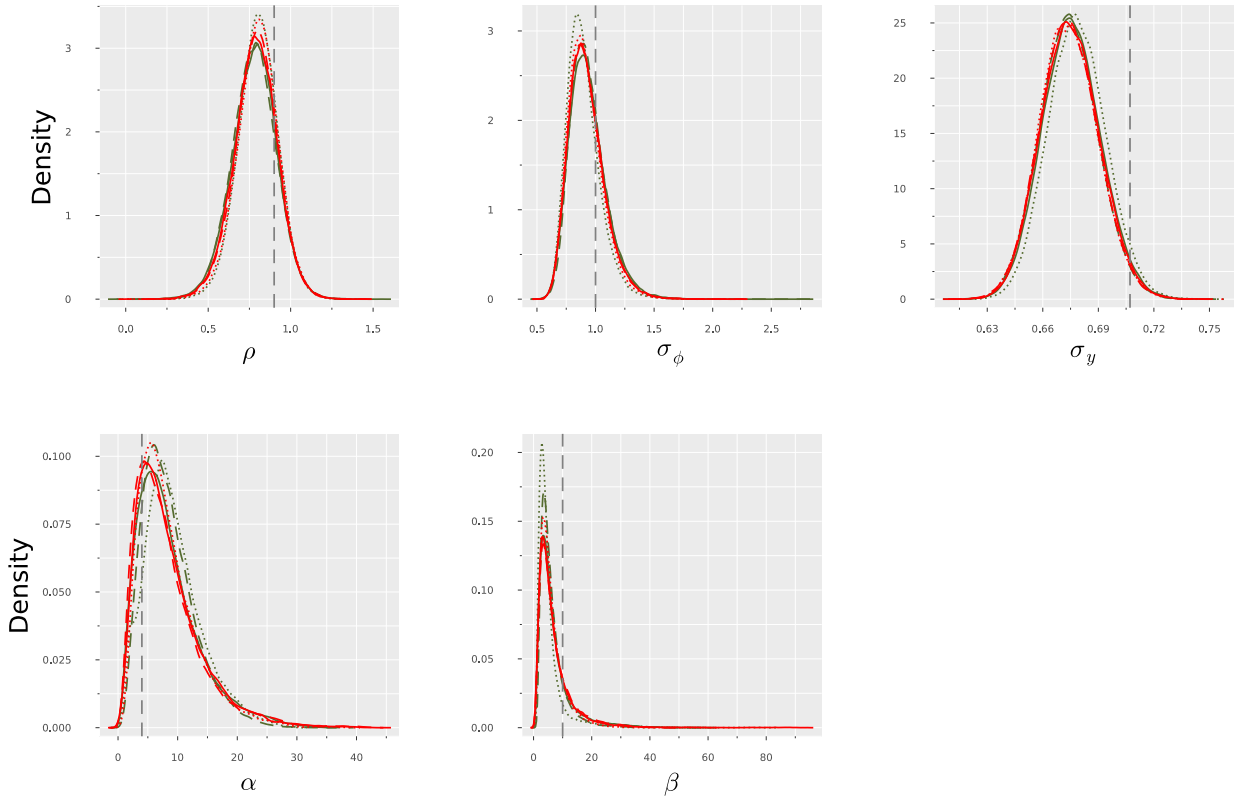

Figure 10: Kernel density estimators of the posterior distributions of parameters in elementary change-point model. The results obtained by using VRPF and BlockVRPF method are represented in red and green colour respectively. Results obtained by using 25, 50 and 100 particles are represented in dot, dashed and solid lines respectively.

10 shows the posterior estimations of the parameters obtained by using both VRPF-PG and BlockVRPF-PG algorithms. One can see that both algorithms yield comparable estimations, regardless 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

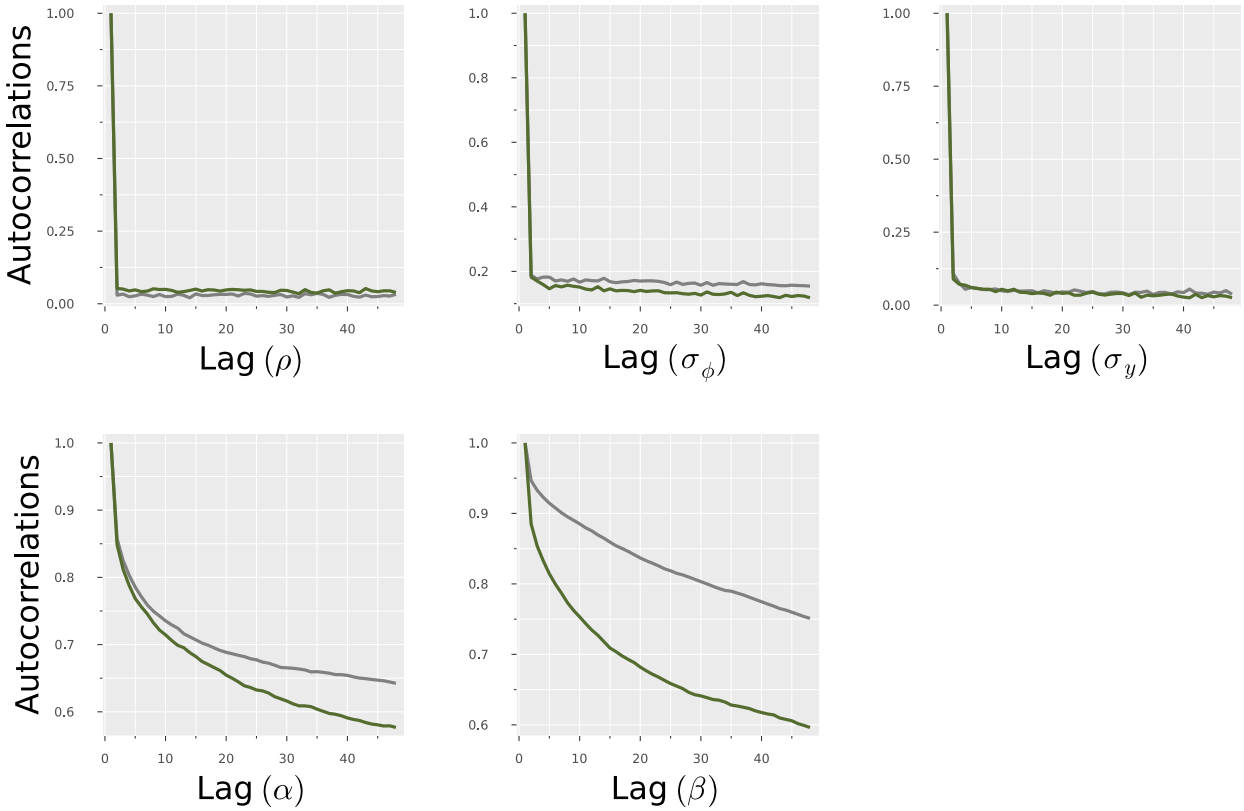

Figure 11: Autocorrelations obtained by running BlockVRPF-PG with 25 particles and different choices of backward kernels. Grey lines represent the results obtained using standard deviations 0.1 and 0.05 while green lines represent results obtained using standard deviations 0.1 and 0.2.

variance introduced to the BlockVRPF method because of the modification move at each SMC step, which results in the sampled PDMP becoming more likely to get stuck at local modes if the backward kernels are badly chosen. In fact, the choices of backward kernels are crucial to the performance of 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 11 shows the autocorrelations obtained from the two runs. One can see that by 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.

6 Conclusion

References

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

Doucet, A., Briers, M., and Sénécal, S. Efficient block sampling strategies for sequential monte carlo methods. *Journal of Computational and Graphical Statistics*, 15(3):693–711, 2006.

Finke, A., Johansen, A. M., and Spanò, D. Static-parameter estimation in piecewise deterministic processes using particle gibbs samplers. *Annals of the Institute of Statistical Mathematics*, 66(3):577–609, 2014.

Godsill, S. and Vermaak, J. Variable rate particle filters for tracking applications. In *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pp. 1280–1285. IEEE, 2005.

Lindsten, F. and Schön, T. B. On the use of backward simulation in the particle gibbs sampler. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3845–3848. IEEE, 2012.

Lindsten, F. and Schön, T. B. Backward simulation methods for monte carlo statistical inference. *Foundations and Trends® in Machine Learning*, 6(1):1–143, 2013.

Liu, J. S. *Monte Carlo strategies in scientific computing*, volume 10. Springer, 2001.

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

Van Dyk, D. A. and Park, T. Partially collapsed gibbs samplers: Theory and methods. *Journal of the American Statistical Association*, 103(482):790–796, 2008.

Whiteley, N. Discussion on particle markov chain monte carlo methods,. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

Whiteley, N., Johansen, A. M., and Godsill, S. Monte carlo filtering of piecewise deterministic processes. *Journal of Computational and Graphical Statistics*, 20(1):119–139, 2011.

A Additional simulation results of section 4.3

In this appendix, we show the sampled nearest jump times to each of the actual jumps in the data.