

1 Piecewise Deterministic Processes

Piecewise Deterministic Processes (PDPs) are stochastic processes that jump randomly at an almost surely countable number of random times but otherwise evolve deterministically in continuous time. In this chapter, we give an introduction to this class of models and then give a mathematical representation of this class of models.

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 $\tau_0 < \tau_1 < \tau_2 < \dots$. 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 $0 = t_0 < t_1 < t_2 < \dots$ 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.

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

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

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

where $T_{n,k} := \{(\tau_1, \dots, \tau_k) \in (0, \infty)^k : 0 < \tau_1 < \dots < \tau_k \leq t_n\}$. 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

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

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

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

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 Related Literature

To make inferences on the PDMP models, several particle filters have been proposed in the literature. [Godsill & Vermaak \(2005\)](#) proposed a particle filter that is based on a bootstrap approach. Such a method is termed as *variable rate particle filters* (VRPF). One of the main drawbacks of VRPF is that particles are only proposed based on the information up to a certain time. As a result, later information would not be about to be used to guide the proposals of particles even if they are highly informative. A particle filter based around the Sequential Monte Carlo (SMC) Sampler of [Del Moral et al. \(2006\)](#) has been proposed later by [Whiteley et al. \(2011\)](#). This alleviate the path degeneracy problem of [Godsill & Vermaak \(2005\)](#). However, the way [Whiteley et al. \(2011\)](#) formulates the problem makes the resulting particle filter unsuitable for perform backward simulation, which is an important variance reduction step for SMC samplers. [Finke et al. \(2014\)](#) made modifications and reformulate the particle filter of [Whiteley et al. \(2011\)](#) to make it suitable for backward simulations. A corresponding particle Gibbs sampler was also proposed to make Bayesian inferences on the static parameters

of the PDMP models. However, the particle filters proposed by [Whiteley et al. \(2011\)](#) and [Finke et al. \(2014\)](#) incur an approximation on the posterior distribution of the PDMP and this will become a problem when the PDMP is likely to jump more than once within an observing window. Numerical results in [Finke et al. \(2014\)](#) also shows that such an approximation may become obvious in certain models.

We propose to solve the problem from a different direction. To alleviate the problems existing in [Godsill & Vermaak \(2005\)](#), we formulate a particle filter based on the block sampling strategies of [Doucet et al. \(2006\)](#). At each time, in addition to proposing jumps in the current interval, we also make use of the new information available to make modifications on the jumps proposed in the previous intervals.

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, $(E^{(n)})_{n \in \mathbb{N}}$. Furthermore, we define each distribution π_n^θ as a joint distribution of variables $x_{1:n} := (x_1, x_2, \dots, x_n)$, where $n = 1, 2, \dots, P$. We can also assume, from now on, that only unnormalised versions of $(\pi_n^\theta)_{n \in \mathbb{N}}$ can be evaluated, i.e.

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

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

For each distribution π_n , suppose that samples of $x_{1:n}$ can be obtained from a proposal distribution $q_t(dx_{1:n})$, the Importance Sampling method creates an approximation of $\pi_n(dx_{1:n})$ by a collection of samples $\{X_{1:n}^j, j = 1, 2, \dots, N\}$ and their corresponding normalised weights $\{W_n^j, j = 1, 2, \dots, N\}$. 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 $X_{1:n}^j$, the corresponding unnormalized weight w_n^j can be calculated by

$$w_n^j = \frac{\gamma_n(x_{1:n}^j)}{q_n(x_{1:n}^j)} \quad (4)$$

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

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

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

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 $x_n \sim q_n(\cdot|x_{1:n-1}^j)$ and appending it to $x_{1:n-1}^j$. In this situation, the unnormalized importance weight can be calculated by

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

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)

```

1 for  $n = 1, 2, 3, \dots, P$  do
2   for  $j = 1, 2, \dots, N$  do
3     Sample  $X_{1:n}^j \sim q_n(\cdot|x_{1:n-1}^j)$  ;
4     Set  $X_{1:n}^j := (X_{1:n-1}^j, X_n^j)$  ;
5     Calculate the unnormalized weight by ;
6
7     
$$w_n^j := w_{n-1}^j \frac{\gamma_n(x_{1:n}^j)}{\gamma_{n-1}(x_{1:n-1}^j) q_n(x_n^j|x_{1:n-1}^j)}$$

8   Set the normalized weight by
9
10    
$$W_n^j := w_n^j / \sum_{i=1}^N w_n^i$$

11
12   Approximate  $\pi_n(dx_{1:n})$  by
13
14    
$$\hat{\pi}_n(dx_{1:n}) := \sum_{j=1}^N W_n^j \delta_{x_{1:n}^j}(dx_{1:n})$$


```

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, $\max_{j=1, \dots, N} W_n^j$, 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

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

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

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

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 $X_n^j \sim q_n(\cdot|\tilde{X}_{1:n-1}^j)$ 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}^j . 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, $E\{\mathcal{N}_n^j\}$, will be given by

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

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

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

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 $\mathbf{A}_{n-1} := (A_{n-1}^1, \dots, A_{n-1}^N) \sim r(\cdot|\mathbf{W}_{n-1})$ with $r(\cdot|\mathbf{W}_{n-1})$ being any kernel such that $r(j|\mathbf{W}_{n-1}) = W_{n-1}^j$ and $X_n^j \sim q_n(\cdot|X_{1:n-1}^{A_{n-1}^j})$. 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 $t = n-1, \dots, 1$. Then each particle lineage at step n can be expressed in an alternative way,

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

Algorithm 2: Sequential Importance Resampling (SIR)

1 **for** $n=1$ **do**

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

3 Compute the unnormalized weight by

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

4 Set $W_1^j = w_1^j / \sum_{i=1}^N w_1^i$;
Approximate $\pi_1(dx_1)$ by

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

5 **for** $n = 1, 2, 3, \dots, P$ **do**

6 **for** $j = 1, 2, \dots, N$ **do**

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

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

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

10 Calculate the unnormalized weight by

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

11 Set the normalised weight by

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

12 Approximate $\pi_n(dx_{1:n})$ 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 Block SMC samplers

Suppose that we have already obtained $\{W_{n-1}^i, X_{1:n-1}^i\}_{i=1,\dots,N}$ at step $n-1$. Under BSMC sampling scheme, we propose not only X_n^i but also $X_{n-L:n-1}^i$ for some $L > 1$ according to a proposing density $K_n^\theta(\cdot|x_{1:n-1})$. Hence, $K_n^\theta(x'_{n-L:n}|x_{1:n-1})$ will be the density of the proposed path at step n if the path at step $n-1$ is $x_{1:n-1}$. $X_{1:n}$ 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.

$$X_{1:n} := (X_{1:n-L-1}, X'_{n-L:n}) \quad (10)$$

If we define $p_{n-1}(x_{1:n-1})$ to be the proposed density of the path $X_{1:n-1}$ at step n , then $p_n(x_{1:n})$ can then be obtained by,

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

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

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

We can see that the calculation of 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.

Define $X_n(j)$ to be the j th 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,

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

This allows us to transform between the X 's and Z 's by Equation (11)

$$Z_{n,k} := X_{n-k+1}(k) \quad (11)$$

$$X_n(j) := Z_{n+j-1,j} \quad (12)$$

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

$$\begin{aligned} p_n(z_{1:n}) &= K_1^\theta(z_1) \prod_{l=2}^n K_l^\theta(z_l | z_{1:l-1}) \\ &= K_1^\theta(x_1(1)) \prod_{l=2}^L K_l^\theta(x_1(l), \dots, x_l(1) | x_{1:l-1}) \prod_{l=L+1}^n K_l^\theta(x_{l-L}(L+1), \dots, x_l(1) | x_{1:l-1}) \end{aligned}$$

where the conditional path $x_{1:l-1}$ is propagated and obtained by (10). To be compatible with the importance density, the extended target should also include all the variables up to step n . To achieve this, we define

$$\begin{aligned} \tilde{\pi}_n^\theta(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) &:= \gamma_n^\theta(x_1(L+1), \dots, x_{n-L}(L+1), x_{n-L+1}(L), \dots, x_n(1)) \times \\ &\quad \prod_{l=2}^n \lambda_l^\theta(x_{n-L}(L), x_{n-L+1}(L-1), \dots, x_{n-1}(1) | x_{1:n}) \end{aligned} \quad (13)$$

Moreover, the extended target density incorporates the actual target as a marginal. Also, by targeting this extended density, we can get rid of the integral appeared in the previous set up and now the importance weight will become

$$\begin{aligned} W_n &\propto \frac{\tilde{\gamma}_n^\theta(z_{1:n})}{p_n(z_{1:n})} = \frac{\tilde{\gamma}_n^\theta(z_{1:n}) p_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}^\theta(z_{1:n-1}) p_n(z_{1:n})} \times \frac{\tilde{\gamma}_{n-1}^\theta(z_{1:n-1})}{p_{n-1}(z_{1:n-1})} \\ &= W_{n-1} \frac{\gamma_n^\theta(x_{1:n-L}(L+1), x_{n-L+1}(L), \dots, x_n(1)) \lambda_n(x_{n-L}(L), \dots, x_{n-1}(1) | x_{1:n})}{\underbrace{\gamma_{n-1}^\theta(x_{1:n-L-1}(L+1), x_{n-L}(L), \dots, x_{n-1}(1)) K_n^\theta(x_{n-L}(L+1), \dots, x_n(1) | x_{1:n-1})}_{G_n^\theta(z_{1:n})}} \end{aligned} \quad (14)$$

If 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 will be listed in **Algorithm 3**.

Algorithm 3(Block SMC)

Algorithm 3: Block SMC

```

1 for  $n = 1$  do
2   Sample  $X_1^i(1) \sim K_1^\theta(dx_1)$ ;
3   Set  $Z_1^i := X_1^i(1)$ ;
4   Calculate and normalise the weights

```

$$W_1^i \propto \frac{\tilde{\gamma}_1^\theta(z_1^i)}{K_1^\theta(z_1^i)}$$

1. For $n = 1$,

- Sample $X_1^i(1) \sim K_1^\theta(\cdot)$
- Set $Z_1^i := X_1^i(1)$
- Calculate and normalise the weights

$$W_1^i \propto \frac{\tilde{\gamma}_1^\theta(z_1^i)}{K_1^\theta(z_1^i)}$$

2. For $n = 2, 3, \dots$

- Sample $\mathbf{A}_{n-1} \sim \mathbf{r}_{n-1}(\cdot | \mathbf{W}_{n-1})$
- Sample $Z_n^i := (X_n(1), \dots, X_1(n)) \sim K_n^\theta(\cdot | Z_{1:n-1}^{A_{n-1}^i})$
- Set $Z_{1:n}^i := (Z_{1:n-1}^{A_{n-1}^i}, Z_n^i)$
- Calculate and normalise the weights

$$W_n^i \propto G_n^\theta(z_{1:n}^i)$$

where $G_n^\theta(z_{1:n}^i)$ is defined in (14)

4 SMC filter for piecewise deterministic processes

4.1 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 $0 = t_0 < t_1 < t_2 < \dots$, 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

$$X_1 := (k_1, \tau_{1,1:k_1}, \phi_{1,1:k_1}, \phi_0) \quad (15a)$$

$$X_n := (k_n, \tau_{n,1:k_n}, \phi_{n,1:k_n}) \quad (15b)$$

These 'states' will take values in a subset of

$$E_1 := \bigcup_{k_1=0}^{\infty} (\{k_1\} \times T_{(0,t_1],k_1} \times \Phi^{k_1+1}) \quad (16a)$$

$$E_n := \bigcup_{k_n=0}^{\infty} (\{k_n\} \times T_{(t_{n-1},t_n],k_n} \times \Phi^{k_n}) \quad (16b)$$

Define $\mathcal{J}_n := (\hat{k}_n, \hat{\tau}_{n,1:K_n}, \hat{\phi}_{n,0:K_n})$ 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, $(\hat{\tau}_{n,1:\hat{k}_n}, \hat{\phi}_{n,1:\hat{k}_n})$ will be given by

$$\hat{\tau}_{n,1:\hat{k}_n} := \bigcup_{j=1}^n \{\tau_{j,1:k_j}\} \quad (17a)$$

$$\hat{\phi}_{n,0:\hat{k}_n} := \{\phi_0\} \bigcup \left[\bigcup_{j=1}^n \{\phi_{j,1:k_j}\} \right] \quad (17b)$$

with the conventions that $\{\tau_{j,1:k_j}\} := \emptyset$ and $\{\phi_{j,1:k_j}\} := \emptyset$ 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

$$\begin{aligned} \gamma_n(x_{1:n}) &:= S(\hat{\tau}_{n,\hat{k}_n}, t_n) \times q(\hat{\phi}_0) \times g(y_{(0,t_n]} | \mathcal{J}_n) \\ &\quad \times \prod_{j=1}^{\hat{k}_n} \left\{ f(\hat{\tau}_j | \hat{\tau}_{j-1}) q(\hat{\phi}_j | \hat{\phi}_{j-1}, \hat{\tau}_j, \tau_{j-1}^{\wedge}) \right\} \end{aligned} \quad (18)$$

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 $K_n(dx_n | X_{1:n-1})$, where

$$K_n(x_n | x_{1:n-1}) = K_{n,1}(k_n | x_{1:n-1}) K_{n,2}(\tau_{n,1:k_n}, \phi_{n,1:k_n} | k_n, x_{1:n-1}) \quad (19)$$

and the corresponding incremental weight is given by

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

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

$$G_n(x_{1:n}) := \frac{S(\hat{\tau}_n, \hat{k}_n, t_n)}{S(\hat{\tau}_n, \hat{k}_n, t_{n-1})} \times \frac{g(y_{(t_{n-1}, t_n]} | \mathcal{J}_n)}{K_{n,1}(0 | x_{1:n-1})} \quad (20)$$

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

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

The corresponding incremental weight is given by

$$G_n(x_{1:n}) := \frac{S(\hat{\tau}_n, \hat{k}_n, t_n)}{S(\hat{\tau}_{n-1}, \hat{k}_{n-1}, t_{n-1})} \times \frac{h(\tau_{n,1:k_n}, \phi_{n,1:k_n} | \mathcal{J}_{n-1}) g(y_{(t_{n-1}, t_n]} | \zeta_{(0, t_n]})}{K_{n,1}(k_n | x_{1:n-1}) K_{n,2}(\tau_{n,1:k_n}, \phi_{n,1:k_n} | k_n, x_{1:n-1})} \quad (21)$$

where

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

The VRPF method is detailed in Algorithm 19. As shown in Whiteley et al. (2011), the VRPF will suffer from severe path degeneracy. This is because the jump times can be only sampled in the interval $(t_{n-1}, t_n]$ at the n -th SMC step. Therefore, if subsequent observations high suggest a jump in the interval $(t_{n-1}, t_n]$, then this can only be recovered through resampling.

To combat against this weakness, our idea is to incorporate a block sampling step at each SMC step so that the jump times and their associated jump values in the interval $(t_{n-1}, t_n]$ can be revised and modified at the n -th SMC step. This results in a modified VRPF algorithm, called the Block variable rate particle filter. We will refer this as *Block-VRPF* in later sections.

4.2 Block Variable Rate Particle Filter (Block-VRPF)

To fight for the path degeneracy of VRPF, Block-VRPF makes modifications of the jump times and their associated jump values in the previous interval at each time. At the n -th SMC step, in addition to sample jump times and values in the interval $(t_{n-1}, t_n]$, represented by X_n , jumps in the interval $(t_{n-2}, t_{n-1}]$, represented by X_{n-1} , will also be modified. We follow Whiteley et al. (2011) to propose two types of modifications on X_{n-1} - a 'birth' move and an 'adjust' move:

Algorithm 4: Variable Rate Particle Filter (VRPF)

```

1 for  $n=1$  do
2   for  $i = 1, 2, \dots, N$  do
3     Sample  $X_1^i := \left(k_1^i, \tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i, \phi_0^i\right) \sim K_1(\cdot)$ ;
4     Set  $\mathcal{J}_1^i := \left(\tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i\right)$ ;
5     Calculate the un-normalised weight
        
$$G_1(X_1^i) := \frac{\gamma_1(X_1^i)}{K_1(X_1^i)}$$

6   for  $i = 1, 2, \dots, N$  do
7     Calculate the normalised weight  $W_1^i$  such that
        
$$W_1^i \propto G_1(X_1^i), \quad \sum_{i=1}^N W_1^i = 1$$

8 for  $n = 2, 3, \dots, P$  do
9   for  $i = 1, 2, \dots, N$  do
10    Sample  $A_{n-1}^i \sim \mathbf{r}(\cdot | \mathbf{W}_{n-1})$ ;
11    Sample  $X_n^i := \left(k_n^i, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right) \sim K_n(\cdot | X_{1:n-1}^{A_{n-1}^i})$  and set
        
$$X_{1:n}^i := \left(X_{1:n-1}^{A_{n-1}^i}, X_n^i\right)$$
;
12    if  $k_n^i = 0$  then
13      Set  $\mathcal{J}_n^i := \mathcal{J}_{n-1}^{A_{n-1}^i}$ ;
14      Calculate the un-normalised weight,  $G_n(X_{1:n}^i)$ , using Equation (20);
15    if  $k_n^i \geq 1$  then
16      Set  $\mathcal{J}_n^i := \left(\hat{k}_{n-1}^{A_{n-1}^i} + k_n^i, \mathcal{J}_{n-1}^{A_{n-1}^i} \setminus \left\{\hat{k}_{n-1}^{A_{n-1}^i}\right\}, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right)$ ;
17      Calculate the un-normalised weight,  $G_n(X_{1:n}^i)$ , using Equation (21);
18    for  $i = 1, 2, \dots, N$  do
19      Calculate the normalised weight  $W_n^i$  such that
        
$$W_n^i \propto G_n(X_{1:n}^i), \quad \sum_{i=1}^N W_n^i = 1$$


```

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 greater of a and b .
2. If an 'adjust' is proposed, the last jump time in X_{n-1} and its associated jump value, $(\tau_{n-1,k_{n-1}}, \phi_{n-1,k_{n-1}})$, 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} contains zero jump, the 'adjust' will keep X_{n-1} unchanged.

Note that the modifications are not constrained to those we discussed above. Other modification proposals and even modifications on more jumps are also possible. See later section for the discussion of a more general set-up. For now, our method is constrained to these two modifications only.

Let $U_{n-1} := (\overset{\circ}{\tau}_{n-1}, \overset{\circ}{\phi}_{n-1})$ 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} with 0 representing an 'adjust' and 1 representing a 'birth'. Moreover, we use $\bar{X}_{n-1} := (\bar{k}_{n-1}, \bar{\tau}_{n-1,1:\bar{k}_{n-1}}, \bar{\phi}_{n-1,1:\bar{k}_{n-1}})$ to represent X_{n-1} after modification, where \bar{K}_{n-1} the number of jumps contained in the \bar{X}_{n-1} .

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

$$\bar{X}_{n-1} := (k_{n-1} + 1, \tau_{n-1,1:k_{n-1}}, \overset{\circ}{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}}, \overset{\circ}{\phi}_{n-1}) \quad (22a)$$

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

$$\bar{X}_{n-1} := (k_{n-1}, \tau_{n-1,1:k_{n-1}-1}, \overset{\circ}{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}-1}, \overset{\circ}{\phi}_{n-1}) \quad (22b)$$

If the number of jump times in X_{n-1} is 0, $\bar{X}_{n-1} = X_{n-1}$. We treat this as a special case of Equation (22b).

After the state, X_{n-1} is modified, the jump times and jump values in the interval $(t_{n-1}, t_n]$ will be sampled conditioned on the modified PDP. Hence, *Block-VRPF* actually samples (M_{n-1}, U_{n-1}, X_n) 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

$$Z_1 := X_1 \quad (23a)$$

$$Z_n := (M_{n-1}, U_{n-1}, X_n) \quad (23b)$$

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

$$E_1 := \bigcup_{k_1=0}^{\infty} (\{k_1\} \times T_{(0,t_1],k} \times \Phi^{k_1+1}) \quad (24a)$$

and

$$E_n := \left[\left(\{a\} \times T_{(\tau_{n-1}, k_{n-1}-1 \vee t_{n-2}, t_{n-1})}^M \times \Phi \right) \cup \left(\{b\} \times T_{(\tau_{n-1}, k_{n-1} \vee t_{n-2}, t_{n-1})}^M \times \Phi \right) \right] \times \bigcup_{k_n=0}^{\infty} (\{k_n\} \times T_{(0, t_n], k_n} \times \Phi^{k_n}) \quad (24b)$$

where $T_{(s, t]}^M := \{\tau | \tau \in (s, t]\}$. Moreover, denote \bar{U}_{n-1} be 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, $\bar{U}_{n-1} := (\tau_{n-1, k_{n-1}}, \phi_{n-1, k_{n-1}})$ 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, $(\bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1})$ and $(X_{n-1}, M_{n-1}, U_{n-1})$ 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

$$\begin{aligned} Z_{1:n} &:= (X_1, M_1, U_1, \dots, X_{n-1}, M_{n-1}, U_{n-1}, X_n) \\ &:= (\bar{X}_1, M_1, \bar{U}_1, \dots, \bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1}, X_n) \end{aligned} \quad (25)$$

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 (25), it is the same to express the joint distribution of $Z_{1:n}$ in terms of $(X_1, M_1, U_1, \dots, X_{n-1}, M_{n-1}, U_{n-1}, X_n)$ as to express it in terms of $(\bar{X}_1, M_1, \bar{U}_1, \dots, \bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1}, X_n)$. 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 $(\bar{X}_1, \bar{X}_2, \dots, \bar{X}_{n-1}, X_n)$. Hence, it is clearer to show that the extended target density incorporates the correct distribution as a marginal using the later parameterization. The extended target distribution of the algorithm, $\bar{\pi}_n(Z_{1:n}) := \bar{\gamma}_n(Z_{1:n})/\bar{Z}_n$, is given by

$$\begin{aligned} \bar{\gamma}_n(Z_{1:n}) &:= \gamma_n(\bar{x}_1, \dots, \bar{x}_{n-1}, x_n) \mu_n(m_{1:n-1} | \bar{x}_1, \dots, \bar{x}_{n-1}, x_n) \\ &\quad \times \prod_{j=1}^{n-1} \lambda_j(\bar{u}_j | m_{1:j}, \bar{x}_1, \dots, \bar{x}_j) \end{aligned} \quad (26)$$

As in the previous section, denote $\mathcal{J}_n := (\hat{k}_n, \hat{\tau}_{n, 1:\hat{k}_n}, \hat{\phi}_{n, 1:\hat{k}_n})$ be the jump times and values that defined the PDP, $\zeta_{(0, t_n]}$ at the n -th SMC step. Then, we will have that

$$\begin{aligned} \gamma_n(\bar{x}_1, \dots, \bar{x}_{n-1}, x_n) &= \gamma_n(\mathcal{J}_n) \\ &= S(\hat{\tau}_{n, \hat{k}_n}, t_n) q(\hat{\phi}_0) g(y_{(0, t_n]} | \mathcal{J}_n) \\ &\quad \times \prod_{j=1}^{\hat{k}_n} \left\{ f(\hat{\tau}_j | \hat{\tau}_{j-1}) q(\hat{\phi}_j | \hat{\phi}_{j-1}, \hat{\tau}_j, \hat{\tau}_{j-1}) \right\} \end{aligned}$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \setminus \left\{ \hat{k}_{n-1} \right\}, \hat{\tau}_{n-1}, \hat{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n} \right) \quad (27a)$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \setminus \left\{ \hat{k}_{n-1}, \hat{\tau}_{n-1, \hat{k}_{n-1}}, \hat{\phi}_{n-1, \hat{k}_{n-1}} \right\}, \hat{\tau}_{n-1}, \hat{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n} \right) \quad (27b)$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \setminus \left\{ \hat{k}_{n-1} \right\}, \tau_{n,1:k_n}, \phi_{n,1:k_n} \right) \quad (27c)$$

Here, we use that convention that if $k_n = 0$, then Equation (27a), (27b) and (27c) will be simplified to $\left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \setminus \left\{ \hat{k}_{n-1} \right\}, \hat{\tau}_{n-1}, \hat{\phi}_{n-1} \right)$, $\left(\hat{k}_{n-1}, \mathcal{J}_{n-1} \setminus \left\{ \hat{k}_{n-1}, \hat{\tau}_{n-1, \hat{k}_{n-1}}, \hat{\phi}_{n-1, \hat{k}_{n-1}} \right\}, \hat{\tau}_{n-1}^m, \hat{\phi}_{n-1}^m \right)$ and \mathcal{J}_{n-1} respectively.

4.2.2 Proposal Kernel

As for the proposed kernel, we use $K_n(z_n|z_{1:n-1}) := K_{n,1}(m_{n-1}|z_{1:n-1}) K_{n,2}(u_{n-1}|m_{n-1}, z_{1:n-1}) \times K_{n,3}(x_n|z_{1:n-1}, m_{n-1}, u_{n-1})$, in which $K_{n,1}(m_{n-1}|z_{1:n-1}) := K_{n,1}(m_{n-1}|\mathcal{J}_{n-1})$ 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. $K_{n,1}(0|\mathcal{J}_{n-1}) := S\left(\hat{\tau}_{n-1, \hat{k}_{n-1}}, t_{n-1}\right)$. 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

$$\begin{aligned} K_{n,2}(u_{n-1}|m_{n-1} = 1, z_{1:n-1}) &:= \mathbb{1}\left(\hat{\tau}_{n-1, \hat{k}_{n-1}} \vee t_{n-2} < \hat{\tau}_{n-1} \leq t_{n-1}\right) \\ &\times \alpha_{n-1,1}(\hat{\tau}_{n-1}|\mathcal{J}_{n-1}) \beta_{n-1,1}(\hat{\phi}_{n-1}|\mathcal{J}_{n-1}) \end{aligned} \quad (28a)$$

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

$$\begin{aligned} K_{n,2}(u_{n-1}|a, z_{1:n-1}) &:= \mathbb{1}(k_{n-1} \geq 1) \mathbb{1}\left(\hat{\tau}_{n-1, \hat{k}_{n-1}-1} \vee t_{n-2} < \hat{\tau}_{n-1} \leq t_{n-1}\right) \\ &\times \alpha_{n-1,0}(\hat{\tau}_{n-1}|\mathcal{J}_{n-1}) \beta_{n-1,0}(\hat{\phi}_{n-1}|\mathcal{J}_{n-1}) \\ &+ \mathbb{1}(k_{n-1} = 0) \times \delta_\emptyset(u_{n-1}) \end{aligned} \quad (28b)$$

i.e. if X_{n-1} contains no jumps at all and an 'adjust' is proposed, then U_{n-1} will be an empty set.

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.

$$K_{n,3}(x_n | m_{n-1}, u_{n-1}, z_{1:n-1}) := K_{n,3}^1(k_n | m_{n-1}, u_{n-1}, z_{1:n-1}) \times K_{n,3}^2(\tau_{n,1:k_n}, \phi_{n,1:k_n} | k_n, m_{n-1}, u_{n-1}, z_{1:n-1}) \quad (29)$$

4.2.3 Auxiliary Densities

For the density $\mu_n(m_{1:n-1} | \bar{x}_1, \dots, \bar{x}_{n-1}, x_n)$ in (26), we need to make sure that the support of it should be included in the support of the proposed kernel. To make it simple, we propose a factorizable density, i.e.

$$\mu_n(m_{1:n-1} | \bar{x}_1, \dots, \bar{x}_{n-1}, x_n) := \prod_{j=1}^{n-1} \mu_n^j(m_j | \bar{x}_j)$$

Furthermore, we propose a uniform distribution over 'adjust' and 'birth' on m_j if \bar{x}_j contains at least one jump, i.e.

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

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, $\lambda_j(\bar{u}_j | m_{1:j}, \bar{x}_1, \dots, \bar{x}_j)$, 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 $\lambda_j(\bar{u}_j | m_{1:j}, \bar{x}_1, \dots, \bar{x}_j)$ becomes degenerate and equals to 1 in these cases. When $M_j = 0$ and $k_j \geq 1$, $\lambda_j(\bar{u}_j | m_{1:j}, \bar{x}_1, \dots, \bar{x}_j) := \lambda_j(\hat{\tau}_{j,\hat{k}_j}, \hat{\phi}_{j,\hat{k}_j} | m_{1:j}, \bar{x}_1, \dots, \bar{x}_j)$ 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, $G_n(z_{1:n}) := \hat{\gamma}_n(z_{1:n}) / [\hat{\gamma}_{n-1}(z_{1:n-1}) K_n(z_n | z_{1:n-1})]$ will be calculated as follow. In this section, we set

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

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

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

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

$$\times g(y_{(t_{n-1}, t_n]}|\mathcal{J}_n) \times \frac{h(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}})}{K_{n,3}^1(k_n|0, \emptyset, z_{1:n-1}) K_{n,3}^2(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 0, \emptyset, z_{1:n-1})} \quad (30a)$$

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

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

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

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

$$\times \frac{h(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\hat{\tau}_{n-1}, \hat{\phi}_{n-1})}{K_{n,3}^1(k_n|0, u_{n-1}, z_{1:n-1}) K_{n,3}^2(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 0, u_{n-1}, z_{1:n-1})} \quad (30b)$$

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

$$\begin{aligned}
G_n(z_{1:n}) &:= \frac{S(\hat{\tau}_{n,\hat{k}_n}, t_n)}{S(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1})} \times \frac{\mu_{n-1}(m_{n-1}|\bar{x}_{n-1})}{K_{n,1}(m_{n-1}|\mathcal{J}_{n-1})} \times \frac{g(y(\hat{\tau}_{n-1}, t_n]|\mathcal{J}_n)}{g(y(\hat{\tau}_{n-1}, t_{n-1}]|\mathcal{J}_{n-1})} \\
&\times \frac{f(\hat{\tau}_{n-1}|\hat{\tau}_{n-1,\hat{k}_{n-1}}) q(\hat{\phi}_{n-1}|\hat{\phi}_{n-1,\hat{k}_{n-1}}, \hat{\tau}_n, \hat{\tau}_{n-1,\hat{k}_{n-1}}) \lambda_{n-1}(\bar{u}_{n-1}|m_{1:n-1}, \bar{x}_{1:n-1})}{\mathbb{1}(\hat{\tau}_{n-1,\hat{k}_{n-1}} \vee t_{n-2} < \hat{\tau}_{n-1} \leq t_{n-1}) \alpha_{n-1,1}(\hat{\tau}_{n-1}|\mathcal{J}_{n-1}) \beta_{n-1,1}(\hat{\phi}_{n-1}|\mathcal{J}_{n-1})} \\
&\times \frac{h(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\hat{\tau}_{n-1}, \hat{\phi}_{n-1})}{K_{n,3}^1(k_n|m_{n-1}=1, u_{n-1}, z_{1:n-1}) K_{n,3}^2(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1}=1, u_{n-1}, z_{1:n-1})} \quad (31)
\end{aligned}$$

In Equation (30a), (30b) and (31), $h(\tau_{n,1:k_n}, \phi_{n,1:k_n}|\hat{\tau}_{n-1}, \hat{\phi}_{n-1})$ and $K_{n,3}^2(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, 1, u_{n-1}, z_{1:n-1})$ 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_{n-1}^i in line 12 of Algorithm 5 will become \emptyset when $M_{n-1}^i = 0$ and $k_{n-1}^{A_{n-1}^i} = 0$.

5 Generic Formulation of BlockVRPF Samplers

We discussed a specific way of modifying the jump times and values in each time interval. Such modifications are similar to Finke et al. (2014) and Whiteley et al. (2011). In this section, we argue that the BlockVRPF sampler are suitable for more general modifications.

6 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.

Algorithm 5: Block Variable Rate Particle Filter (BlockVRPF)

1 **for** $n=1$ **do**

2 **for** $i = 1, 2, \dots, N$ **do**

3 Sample $X_1^i := \left(k_1^i, \tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i, \phi_0^i\right) \sim K_1(\cdot)$;

4 Set $Z_1^i := X_1^i$ and $\mathcal{J}_1^i := \left(\tau_{1,1:k_1^i}^i, \phi_{1,1:k_1^i}^i\right)$;

5 Calculate the un-normalised weight

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

6 **for** $i = 1, 2, \dots, N$ **do**

7 Calculate the normalised weight W_1^i such that

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

8 **for** $n = 2, 3, \dots, P$ **do**

9 **for** $i = 1, 2, \dots, N$ **do**

10 Sample $A_{n-1}^i \sim \mathbf{r}(\cdot | \mathbf{W}_{n-1})$;

11 Sample $M_{n-1}^i \sim K_{n,1}(\cdot | \mathcal{J}_{n-1}^{A_{n-1}^i})$;

12 Sample $U_{n-1}^i := \left(\tau_{n-1}^i, \phi_{n-1}^i\right) \sim K_{n,2}(\cdot | m_{n-1}^i, z_{1:n-1}^{A_{n-1}^i})$;

13 Sample $X_n^i := \left(k_n^i, \tau_{n,1:k_n^i}^i, \phi_{n,1:k_n^i}^i\right) \sim$
 $K_{n,3}^1(\cdot | m_{n-1}^i, u_{n-1}^i, z_{1:n-1}^{A_{n-1}^i}) K_{n,3}^2(\cdot | k_n^i, m_{n-1}^i, u_{n-1}^i, z_{1:n-1}^{A_{n-1}^i})$;

14 Set $Z_n^i := (M_{n-1}^i, U_{n-1}^i, X_n^i)$ and $Z_{1:n}^i := (Z_{1:n-1}^{A_{n-1}^i}, Z_n^i)$;

15 Define \mathcal{J}_n^i according to (27a), (27b) or (27c) based on $\mathcal{J}_{n-1}^{A_{n-1}^i}$;

16 Calculate the incremental weight $G_n(z_{1:n}^i)$ according to (30a), (30b) or (31);

17 **for** $n = 1, \dots, N$ **do**

18 Calculate the normalised weight W_n^i such that

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

6.1 Particle Markov Chain Monte Carlo (PMCMC) Methods

The Particle Markov Chain Monte Carlo (PMCMC) is a class of Bayesian inference methods that combine SMC with MCMC to approximately generate samples from the posterior distribution of the parameters for models whose likelihood is intractable due to the presense 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

$$p(y_{1:P}|\theta) = \int p(x_{1:P}, y_{1:P}|\theta) dx_{1:P} = \int p(x_{1:P}|\theta) p(y_{1:P}|x_{1:P}, \theta) dx_{1:P} \quad (32)$$

which is intractable in most scenario. This hinders 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 targetting 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 unnormalised 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 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 using the empirical distribution obtained by running an SMC algorithm targeting $\pi_P(x_{1:P}|\theta', y_{1:P})$ with N particles 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

$$\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P}) := \sum_{n=1}^N W_P^n \delta_{x_{1:P}^n}(dx_{1:P}) \quad (33)$$

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 (33) 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

$$q(x_{1:P}) := \mathbb{E}_{W_{1:P}^{1:N}, x_{1:P}^{1:N}} [\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P})] \quad (34)$$

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 $\mathbf{X}_n := (X_n^1, X_n^2, \dots, X_n^N)$ and $\mathbf{A}_{n-1} := (A_{n-1}^1, A_{n-1}^2, \dots, A_{n-1}^N)$ 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 $(\mathbf{X}_1, \dots, \mathbf{X}_P, \mathbf{A}_1, \dots, \mathbf{A}_{P-1})$ and the joint density of these random variables will be given by

$$\psi_P^{N,\theta}(\mathbf{x}_1, \dots, \mathbf{x}_P, \mathbf{a}_1, \dots, \mathbf{a}_{P-1}) := \left\{ \prod_{j=1}^N K_1^\theta(x_1^j) \right\} \prod_{n=2}^P \left\{ r^\theta(\mathbf{a}_{n-1}|\mathbf{W}_{n-1}) \prod_{j=1}^N K_n^\theta(x_n^j|x_{1:n-1}^{a_{n-1}^j}) \right\} \quad (35)$$

Such an SMC sampler will produce N distinct paths, i.e. $X_{1:P}^1, X_{1:P}^2, \dots, X_{1:P}^N$. For a specific path, $X_{1:P}^k$, we can denote it as

$$X_{1:P}^k := X_{1:P}^{B^k} = (X_1^{B_1^k}, X_2^{B_2^k}, \dots, X_P^{B_P^k})$$

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 $(\mathbf{X}_1, \dots, \mathbf{X}_P, \mathbf{A}_1, \dots, \mathbf{A}_{P-1})$. The design of the target distribution should fulfill the following two conditions:

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

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

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

$$\tilde{\pi}_P(\theta, b_{1:P}, \mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) = \frac{\pi_P(\theta, x_{1:P}^{b_{1:P}}, b_{1:P})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}}|\theta, x_{1:P}^{b_{1:P}}, b_{1:P}) \quad (36)$$

where we define $\mathbf{x}_{1:P}^{-b_{1:P}} := \mathbf{x}_{1:P} \setminus x_{1:P}^{b_{1:P}}$ and similarly $\mathbf{a}_{1:P-1}^{-b_{2:P}} := \mathbf{a}_{1:P-1} \setminus a_{1:P-1}^{b_{2:P}}$. The first part of (36) is included to meet the first condition. Practically, one could simply sample an lineage index k with probability W_P^k and set $b_P := k$ and this would implicitly define the values of b_{P-1}, \dots, 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 (36) is designed to meet the second requirement. As we need to define the distribution of the rest particles and ancestor indices and hope to design a distribution that are close to $\psi_P^{N,\theta}$. One way to do this is to define a conditional distribution of the rest 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

$$\begin{aligned} \psi_P^{N,\theta} \left(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}} | x_{1:P}^{b_{1:P}}, b_{1:P} \right) &:= \frac{\psi_P^{N,\theta}(\mathbf{x}_{1:P}, \mathbf{a}_{1:P})}{K_1(x_1^{b_1}) \prod_{n=2}^P \left\{ r(b_{n-1} | \mathbf{W}_{n-1}) K_n(x_n^{b_n} | x_{1:n-1}^{b_{1:n-1}}) \right\}} \\ &= \left\{ \prod_{1 \leq j \leq N, j \neq b_1} K_1(x_1^j) \right\} \prod_{n=2}^P r(\mathbf{a}_{n-1}^{-b_n} | \mathbf{W}_{n-1}, a_{n-1}^{b_n}) \\ &\quad \times \prod_{1 \leq j \leq N, j \neq b_n} K(x_n^j | x_{1:n-1}^{a_{n-1}^j}) \end{aligned} \quad (37)$$

Since Equation (37) is the conditional distribution of $(\mathbf{X}_{1:P}^{-b_{1:P}}, \mathbf{A}_{1:P-1}^{-b_{2:P}})$ given the path $(x_{1:P}^{b_{1:P}}, b_{1:P})$ in $\tilde{\pi}_P^N$. This gives actually inspires the idea of partible Gibbs sampler described in [Andrieu et al. \(2010\)](#). At each iteration, given we have obtained $(\theta, b_{1:P}, x_{1:P}^{b_{1:P}}, \mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}})$, 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 $\psi_P^{N,\theta^*}(\mathbf{x}_{1:P}^{-b_{1:P}}, \mathbf{a}_{1:P-1}^{-b_{2:P}} | x_{1:P}^{b_{1:P}}, b_{1:P})$. 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.

Algorithm 6: Conditional SMC Sampler (cSMC)

```

1  Given a path  $(X_{1:P}^*, B_{1:P})$ ;
2  for  $n=1$  do
3    for  $j = 1, 2, 3, \dots, N$  do
4      if  $j \neq B_1$  then
5        Sample  $X_1^j \sim K_1(\cdot)$ ;
6      if  $j = B_1$  then
7        Set  $X_1^j = X_1^*$ ;
8      Calculate the weight  $w_1^j$  and normalise the weights  $W_1^j \propto w_1^j$ ;
9  for  $n = 2, 3, \dots, P$  do
10   for  $j = 1, 2, \dots, N$  do
11     if  $j \neq B_n$  then
12       Sample  $A_{n-1}^j \sim r(\cdot | \mathbf{W}_{n-1})$ ;
13       Sample  $X_n^j \sim K_n(\cdot | X_{1:n-1}^{A_{n-1}^j})$ ;
14     if  $j = B_n$  then
15       Set  $A_{n-1}^j = B_{n-1}$ ;
16       Set  $X_n^j = X_n^*$ ;
17   Calculate the weights  $w_n^j$  and normalise them  $W_n^j \propto w_n^j$ ;

```

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

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

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

$$\begin{aligned} \pi_P^\theta(x_{1:P}^{b_{1:P}}) &:= w_1^{b_1} K_1(x_1^{b_1}) \prod_{n=2}^P w_n^{b_n} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \\ &= \left\{ \frac{w_1^{b_1}}{\sum_{i=1}^N w_1^i} K_1(x_1^{b_1}) \prod_{n=2}^P \frac{w_n^{b_n}}{\sum_{i=1}^N w_n^i} K_n^\theta(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^P \sum_{i=1}^N w_n^i \right\} \\ &= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^P W_{n-1}^{b_{n-1}} K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^P \sum_{i=1}^N w_n^i \right\} \\ &= W_P^{b_P} \left\{ K_1(x_1^{b_1}) \prod_{n=2}^P r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x_n^{b_n}|x_{1:n-1}^{b_{1:n-1}}) \right\} \left\{ \prod_{n=1}^P \sum_{i=1}^N w_n^i \right\} \end{aligned} \quad (38)$$

If we substitute this into (36), 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 $n = P-1, P-2, \dots, 2, 1$. 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.

6.2 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

Algorithm 7: particle Gibbs sampler

```

1 Initialise at any  $X_{1:P}^{(0)}, k^{(0)}$  and  $B_{1:P}^{(0)}$ ;
2 for  $n = 1, 2, \dots$  do
3   Sample  $\theta^{(n)} \sim \pi_P(\cdot | x_{1:P}^{(n-1)})$ ;
4   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;
5   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
    $m = P-1, \dots, 2, 1$ ;
6   Set  $X_{1:P}^{(n)} = \left( X_1^{(n), B_1^{(n)}}, \dots, X_P^{(n), B_P^{(n)}} \right)$ ;

```

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}, \dots, 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 $b_1^* \sim \tilde{\pi}_P^N(b_1^* | \theta^*, \mathbf{x}_{1:P}^{*, -b_{1:P}}, \mathbf{a}_{1:P-1}^{*, -b_{2:P}}, b_1, x_1^{b_1}, x_{2:P}^{b_{2:P}^*}, b_{2:P}^*)$

From the above scheme, one can see that the conditional distribution 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

$$\psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}} | x_{1:P}^{b_{1:P}}, b_{1:P}) := \prod_{\substack{j=1 \\ j \neq b_1}}^N K_1(x_1^j) \prod_{n=2}^t \left\{ r(\mathbf{a}_{n-1}^{-b_n} | \mathbf{W}_{n-1}, a_{n-1}^{b_n}) \prod_{\substack{j=1 \\ j \neq b_n}}^N K_n(x_n^j | x_{1:n-1}^{a_{n-1}^j}) \right\} \quad (39)$$

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

$$\begin{aligned} & \tilde{\pi}_P^N(b_t^* | \theta^*, \mathbf{x}_{1:t}^{*, -b_{1:t}}, \mathbf{a}_{1:t-1}^{*, -b_{2:t}}, b_{1:t}, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}, b_{t+1:P}^*) \\ &= \frac{\pi_P(\theta^*, b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:t}^{-b_{1:t}}, \mathbf{a}_{1:t-1}^{-b_{2:t}} | b_{1:t}, b_{t+1:P}^*, x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*}) \\ &= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}}) \pi(\theta^*)}{N^P} \prod_{\substack{j=1 \\ j \neq b_1}}^N K_1(x_1^j) \prod_{n=2}^t \left\{ r(\mathbf{a}_{n-1}^{-b_n} | \mathbf{W}_{n-1}, a_{n-1}^{b_n}) \prod_{\substack{j=1 \\ j \neq b_n}}^N K_n(x_n^j | x_{1:n-1}^{a_{n-1}^j}) \right\} \\ &= \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \frac{\pi(\theta^*) W_t^{b_t}}{N^P} \psi_t^{N,\theta^*}(\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \left\{ \prod_{n=1}^t \sum_{i=1}^N w_n^i \right\} \propto W_t^{b_t} \frac{\pi_P^{\theta^*}(x_{1:t}^{b_{1:t}}, x_{t+1:P}^{b_{t+1:P}^*})}{\pi_t^{\theta^*}(x_{1:t}^{b_{1:t}})} \end{aligned} \quad (40)$$

where the last line is derived in a similar way as we did in (38). 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

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

Algorithm 8: particle Gibbs with Backward Simulation

```

1 Start at  $\theta^{(0)}$ ,  $X_{1:P}^{(0)}$  and  $B_{1:P}^{(0)}$  arbitrarily;
2 for  $n=1,2,\dots$  do
3   Sample  $\theta^{(n)} \sim \pi_P \left( \cdot | X_{1:P}^{(n-1)} \right)$  using e.g. a MH kernel;
4   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  $\mathbf{W}_{1:P}^{(n)}$ ;
5   for  $t = P, P-1, \dots, 2, 1$  do
6     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  $b_j = a_j^{(n),b_{j+1}}$ 
7   Set  $x_{1:P}^{(n)} := x_{1:P}^{(n),b_{1:P}^{(n)}}$ 

```

values that define the PDP in the interval (t_{n-1}, t_P) , we then should have that

$$\begin{aligned}
\check{k}_n &:= \sum_{j=n}^P k_j \\
\check{\tau}_{n,1:\check{k}_n} &:= \bigcup_{j=n}^P \{\tau_{j,1:k_j}\} \\
\check{\phi}_{n,1:\check{k}_n} &:= \bigcup_{j=n}^P \{\phi_{j,1:k_j}\}
\end{aligned}$$

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

$$\mathcal{G}_{n|P}(x_{1:n}^{b_{1:n}^*}) := \frac{f(\check{\tau}_{n+1,1}^* | \hat{\tau}_{n,\hat{k}_n}) g(\check{\phi}_{n+1,1}^* | \hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}_{n+1,1}^*) p(y_{(t_n, \check{\tau}_{n+1,1}^*]} | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \quad (41)$$

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

$$\mathcal{G}_{n|P}(x_{1:n}^{b_{1:n}^*}) = \frac{S(\hat{\tau}_{n,\hat{k}_n}, t_P) p(y_{(t_n, t_P]} | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \quad (42)$$

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, \dots, Z_P . Given $Z_{n+1}^{b_{n+1}^*}, \dots, Z_P^{b_P^*}$ and the corresponding $\check{\mathcal{J}}_n^*$, if $M_n^* = 1$, the corresponding backward sampling incremental weight would be given by

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{\mathbb{I}(\hat{\tau}_n^* > \hat{\tau}_{n,\hat{k}_n}) f(\hat{\tau}_n^* | \hat{\tau}_{n,\hat{k}_n}) g(\hat{\phi}_n^* | \hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_n^*, \hat{\tau}_{n,\hat{k}_n}) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n) p(y(\hat{\tau}_n^*, t_n) | \hat{\phi}_{n,\hat{k}_n})} \quad (43)$$

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

$$\mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) := \frac{\mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1}) S(\hat{\tau}_{n,\hat{k}_n}, t_P) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}) p(y(t_n, t_P) | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \quad (44)$$

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

$$\begin{aligned} \mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) &:= \frac{f(\check{\tau}_{n+1,1}^* | \hat{\tau}_{n,\hat{k}_n}) g(\check{\phi}_{n+1,1}^* | \hat{\phi}_{n,\hat{k}_n}, \hat{\tau}_{n,\hat{k}_n}, \check{\tau}_{n+1,1}^*) p(y(t_n, \check{\tau}_{n+1,1}^*) | \hat{\phi}_{n,\hat{k}_n})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \\ &\quad \times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} < t_{n-1}) \mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}) \end{aligned} \quad (45)$$

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

$$\begin{aligned} \mathcal{G}_{n|P}(z_{1:n}^{b_{1:n}}) &:= \frac{f(\hat{\tau}_n^* | \hat{\tau}_{n,\hat{k}_n-1}) g(\hat{\phi}_n^* | \hat{\phi}_{n,\hat{k}_n-1}, \hat{\tau}_{n,\hat{k}_n-1}, \hat{\tau}_n^*)}{f(\hat{\tau}_{n,\hat{k}_n} | \hat{\tau}_{n,\hat{k}_n-1}) g(\hat{\phi}_{n,\hat{k}_n} | \hat{\phi}_{n,\hat{k}_n-1}, \hat{\tau}_{n,\hat{k}_n-1}, \hat{\tau}_n^*)} \times \frac{p(y(\hat{\tau}_{n,\hat{k}_n} \wedge \hat{\tau}_n^*, \hat{\tau}_n^*) | \hat{\phi}_{n,\hat{k}_n-1})}{p(y(\hat{\tau}_{n,\hat{k}_n} \wedge \hat{\tau}_n^*, t_n) | \hat{\phi}_{n,\hat{k}_n-1}, \hat{\phi}_{n,\hat{k}_n})} \\ &\quad \times \frac{\mu(M_n^* | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*}) \lambda(\hat{\tau}_{n,\hat{k}_n}, \hat{\phi}_{n,\hat{k}_n} | z_{1:n}, z_{n+1:P}^{b_{n+1:P}^*})}{S(\hat{\tau}_{n,\hat{k}_n}, t_n)} \times \mathbb{I}(\hat{\tau}_{n,\hat{k}_n} > t_{n-1}, \hat{\tau}_n^* > \hat{\tau}_{n,\hat{k}_n-1}) \end{aligned} \quad (46)$$

7 Auxiliary Variable Rejuvenation

In this section, we discussed the rejuvenation step introduced by [Finke et al. \(2014\)](#) to improve the mixing of the particle Gibbs scheme. For the BlockVRPF scheme, the incremental weights at step n , described in Equation (30a), (30b) and (31), involve a likelihood ratio of the observations given the original PDP and the modified PDP. Hence, if a modification (i.e. a birth move or a birth move) on a particular particle moves the PDP from a wrong position to the correct position, the likelihood ratio will become so large that this specific particle will be dominating over all the others. In this case, the particle Gibbs sampler will be stuck in this specific path and it will be very hard for a new path to be proposed, even though it is not a path that fits the observations well.

The rejuvenation step proposed by [Finke et al. \(2014\)](#) can be used to solve this problem. This step is suitable whenever the SMC filtering scheme involves auxiliary variables. In the Block-VRPF scheme, what we are really interested is the distribution of variables $(x_{1,m}, x_{2,m}, \dots, x_{n-1,m}, x_n)$ and Θ . Therefore, we are only interested in the distribution $\gamma_n(x_{1,m}, \dots, x_{n-1,m}, x_n)$ in Equation (26). Hence, $M_{1:n-1}$ and U'_1, \dots, U'_{n-1} are the auxiliary variables appeared in the target. In the standard particle Gibbs scheme, we propose a new value of Θ conditioning on both the interested and auxiliary variables. The rejuvenation step, instead, samples Θ conditioning on the interested variables only. Then, all the auxiliary variables will be re-sampled conditioning on the interested variable. Hence, if the sampled path $X_{1:P}$ can be splitted into the interested variables X_I and auxiliary variables X_A (i.e. $X_{1:P} := (X_I, X_A)$) and the target density $\pi_P(\theta, x_{1:P}) := \tilde{\pi}_P(\theta, x_I) \times L(x_A|x_I)$, we will split the first step at each particle Gibbs sweep. This method will be outlined in Algorithm 8

Algorithm 9: particle Gibbs with rejuvenation step

```

1 Initialise  $\theta^{(0)}$ ,  $X_{1:P}^{(0)}$  and  $B_{1:P}^{(0)}$  arbitrarily;
2 for  $n=1, 2, \dots$  do
3   Sample  $\theta^{(n)} \sim \tilde{\pi}_P(\cdot | x_I^{(n-1)})$ ;
4   Sample  $x_A^* \sim L(\cdot | x_I^{(n-1)})$ ;
5   Set  $X_{1:P}^{(n-1)} := (x_I^{(n-1)}, x_A^*)$ ;
6   Run a cSMC algorithm conditional on  $X_{1:P}^{(n-1)}, B_{1:P}^{(n-1)}$ , outlined in Algorithm 6, to
      get  $\mathbf{X}_{1:P}^{(n)}$  and  $\mathbf{A}_{1:P-1}^{(n)}$ ;
7   Run a backward simulation to get  $B_{1:P}^{(n)}$  using the same method as in Algorithm 8;
8   Set  $X_{1:P}^{(n)} = \left( X_1^{(n), B_1^{(n)}}, \dots, X_P^{(n), B_P^{(n)}} \right)$ ;

```

Note that the rejuvenation step re-samples the auxiliary variables, which are the modification moves as well as the jump times and values being modified at each SMC step. Hence, this gives the path a chance to have another modification move or another jump times and values to be modified. As such, the likelihood ratio of this specific particle will possibly be decreased, giving other particles from the SMC step an opportunity to be chosen at each sweep. Therefore, such a rejuvenation step will help the particle Gibbs scheme to escape from the stuck path, hence improving the mixing of the chain.

8 Numerical Examples

8.1 Generic Setup

In this section, we are going to show the correctness of the BlockVRPF sampler within particle Gibbs through two challenging problems - the *elementary change-point model* and the *short-noise Cox model*. As a comparison, we also ran the particle Gibbs with VRPF sampler and for each method, we also perform the backward simulation step described in section 6.2. In the particle Gibbs sampler, static parameters θ are updated through a Metropolis within particle Gibbs (MwPG) step with new parameters set being proposed through a centered multivariate Gaussian kernel using an appropriately chosen covariance matrix that reflects the structure of the joint posterior density of θ . We ran a pilot particle Gibbs sampler following the adaptive tuning strategies of [Andrieu & Thoms \(2008\)](#) to obtain the suitable scale factor λ and covariance matrix Σ for the proposals. Hence, in both examples, proposal parameter sets θ' are sampled from $\mathcal{N}(\theta, \lambda^2 \Sigma)$ where θ is the previous parameter state. In the actual particle Gibbs sampler, given a PDMP obtained from the cSMC algorithm, we run the MH algorithm for 500 times for faster mixing of the particle Gibbs sampler.

8.2 Elementary Change Point Model

References

- Andrieu, C. and Thoms, J. A tutorial on adaptive mcmc. *Statistics and computing*, 18(4): 343–373, 2008.
- 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.
- Del Moral, P., Doucet, A., and Jasra, A. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.
- 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.
- 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.