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 on this class of models and we will follow to 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 $\tau_0 < \tau_1 < \tau_2 < ....$ 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. It is clear that the process $\zeta_{[0,t_n]}$ can be completely determined by $(K_n, \tau_{1:K_n}, \phi_{1:K_n}, \phi_0)$. Moreover, we also propose Markovian prior on the jump times and jump values in the interval $[0, t_n]$, i.e.

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

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 $(\tau, t]$ and f^θ, g^θ represents the conditional probability of the jump times and the associated jump values. 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 $\tau_n - \tau_{n-1}$.

For most of the time, 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

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

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. Note that we did not include 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 be defined on the increasing subsets of the same space

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

where $T_{n,k} := \left\{ (\tau_1, ..., \tau_k) \in (0, \infty)^k : 0 < \tau_1 < ... < \tau_k <= tn \right\}$

If all the static parameters used in the model are known, we are able to make inferences on the jump times and jump values using Sequential Monte Carlo (SMC) methods. In the following section, we will introduce the standard SMC samplers as well as the Block SMC samplers, which is a modified version of the standard SMC incorporating with block sampling methods.

2 Sequential Monte Carlo Methods

2.1 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.

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

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

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

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

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 **??**.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 **?**. As a result, resampling particles can be viewed as sampling indices $\mathbf{A}_{n-1} := \left(A_{n-1}^1, ..., A_{n-1}^N\right) \sim r\left(\cdot|\mathbf{W}_{n-1}\right)$ with $r\left(\cdot|\mathbf{W}_{n-1}\right)$ being any kernel such that $r\left(j|\mathbf{W}_{n-1}\right) = W_{n-1}^j$ and $X_n^j \sim q_n\left(\cdot|X_{1:n-1}^{A_{n-1}^j}\right)$. One can also keep track of the ancestor lineage of the particles at time n by defining $B_{n|n}^i := i$ and

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

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

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

SMC method is sometimes also referred 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 (**??**) 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.

2.2 Block SMC samplers

Suppose that we have already obtained $\left\{W_{n-1}^i, X_{1:n-1}^i\right\}_{i=1,\ldots,N}$ 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.

$$X_{1:n} := (X_{1:n-L-1}, X_{n-L:n}^{'}) \tag{10}$$

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 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,

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

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

$$\begin{aligned} Z_{n,k} &:= X_{n-k+1}(k) \\ X_n(j) &:= Z_{n+j-1,j} \end{aligned} \tag{11}$$

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

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

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

$$\begin{aligned} \tilde{\pi}_n^\theta(z_{1:n}) \propto \tilde{\gamma}_n(z_{1:n}) :=& \gamma_n^\theta(x_1(L+1), ..., x_{n-L}(L+1), x_{n-L+1}(L), ..., x_n(1)) \times \\ & \prod_{l=2}^{n} \lambda_l^\theta(x_{n-L}(L), x_{n-L+1}(L-1), ..., x_{n-1}(1)|x_{1:n}) \end{aligned} \tag{12}$$

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

$$\begin{aligned} W_n \propto \frac{\tilde{\gamma}_n^\theta(z_{1:n})}{p_n(z_{1:n})} &= \frac{\tilde{\gamma}_n^\theta(z_{1:n}) p_{n-1}(z_{1:n-1})}{\tilde{\gamma}_{n-1}^\theta(z_{1:n-1}) p_n(z_{1:n})} \times \frac{\tilde{\gamma}_{n-1}^\theta(z_{1:n-1})}{p_{n-1}(z_{1:n-1})} \\ &= W_{n-1} \times \underbrace{\frac{\gamma_n^\theta(x_{1:n-L}(L+1), x_{n-L+1}(L), ..., x_n(1)) \lambda_n(x_{n-L}(L), ..., x_{n-1}(1)|x_{1:n})}{\gamma_{n-1}^\theta(x_{1:n-L-1}(L+1), x_{n-L}(L), ..., x_{n-1}(1)) K_n^\theta(x_{n-L}(L+1), ..., x_n(1)|x_{1:n-1})}}_{G_n^\theta(z_{1:n})} \end{aligned} \tag{13}$$

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)

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 $n = 2, 3, ..$
   - Sample $\mathbf{A_{n-1}} \sim \mathbf{r_{n-1}}(\cdot|\mathbf{W_{n-1}})$
   - Sample $Z_n^i := (X_n(1), ..., X_1(n)) \sim K_n^\theta(\cdot|Z_{1:n-1}^{A_{n-1}^i})$
   - Set $Z_{1:n}^i := \left(Z_{1:n-1}^{A_{n-1}^i}, Z_n^i\right)$
   - 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 (**??**)

3 SMC filter for piecewise deterministic processes

3.1 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 and Vermaak (2004). 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

$$X_1 := (k_1, \tau_{1,1:k_1}, \phi_{1,1:k_1}, \phi_0) \tag{14a}$$
$$X_n := (k_n, \tau_{n,1:k_n}, \phi_{1,1:k_n}) \tag{14b}$$

These 'states' will take values in a subset of

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k_1} \times \Phi^{k_1+1}\right) \tag{15a}$$
$$E_n := \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(t_{n-1},t_n],k_n} \times \Phi^{k_n}\right) \tag{15b}$$

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

$$\hat{\tau}_{n,1:\hat{k}_n} := \bigcup_{j=1}^{n} \left\{\tau_{j,1:k_j}\right\} \tag{16a}$$

$$\hat{\phi}_{n,0:\hat{k}_n} := \{\phi_0\} \bigcup \left[\bigcup_{j=1}^{n} \left\{\phi_{j,1:k_j}\right\}\right] \tag{16b}$$

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

$$\begin{aligned}\gamma_n(x_{1:n}) := S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right) \times & q\left(\hat{\phi}_0\right) \times g(y_{(0,t_n]}|\mathcal{J}_n) \\ & \times \prod_{j=1}^{\hat{k}_n} \left\{f\left(\hat{\tau}_j|\hat{\tau}_{j-1}\right) q\left(\hat{\phi}_j|\hat{\phi}_{j-1}, \hat{\tau}_j, \hat{\tau_{j-1}}\right)\right\}\end{aligned} \tag{17}$$

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

$$K_n\left(x_n|x_{1:n-1}\right) = K_{n,1}\left(k_n|x_{1:n-1}\right) K_{n,2}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, x_{1:n-1}\right) \tag{18}$$

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

$$G_n(x_{1:n}) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n,\hat{k}_n}, t_{n-1}\right)} \times \frac{g\left(y_{(t_{n-1},t_n]}|\mathcal{J}_n\right)}{K_{n,1}(0|x_{1:n-1})} \tag{19}$$

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

$$G_n\left(x_{1:n}\right) := \frac{S\left(\hat{\tau}_{n,\hat{k}_n}, t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}, t_{n-1}\right)} \times \frac{h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n} | \mathcal{J}_{n-1}\right) g\left(y_{(t_{n-1},t_n]} | \zeta_{(0,t_n]}\right)}{K_{n,1}(k_n | x_{1:n-1}) K_{n,2}(\tau_{n,1:k_n}, \phi_{n,1:k_n} | k_n, x_{1:n-1})} \tag{20}$$

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 **??**. 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, out 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.

3.2 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 **?** to propose two types of modifications on X_{n-1} - a 'birth' move and an 'adjust' move:

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, $\left(\tau_{n-1,k_{n-1}}, \phi_{n-1,k_{n-1}}\right)$, 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} 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 $U_{n-1} := \left(\mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$ 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} with 0 representing an 'adjust' and 1 representing a 'birth'. Moreover, we use $\bar{X}_{n-1} := \left(\bar{k}_{n-1}, \bar{\tau}_{n-1,1:\bar{k}_{n-1}}, \bar{\phi}_{n-1,1:\bar{k}_{n-1}}\right)$ to represent X_{n-1} 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,

$$\bar{X}_{n-1} := \left(k_{n-1} + 1, \tau_{n-1,1:k_{n-1}}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}}, \mathring{\phi}_{n-1}\right) \tag{21a}$$

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

$$\bar{X}_{n-1} := \left(k_{n-1}, \tau_{n-1,1:k_{n-1}-1}, \mathring{\tau}_{n-1}, \phi_{n-1,1:k_{n-1}-1}, \mathring{\phi}_{n-1}\right) \tag{21b}$$

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 (**??**).

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

$$Z_1 := X_1 \tag{22a}$$
$$Z_n := (M_{n-1}, U_{n-1}, X_n) \tag{22b}$$

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

$$E_1 := \bigcup_{k_1=0}^{\infty} \left(\{k_1\} \times T_{(0,t_1],k} \times \Phi^{k_1+1}\right) \tag{23a}$$

and

$$\begin{aligned} E_n := &\left[\left(\{a\} \times T^M_{(\tau_{n-1,k_{n-1}-1} \vee t_{n-2}, t_{n-1}]} \times \Phi\right) \bigcup \left(\{b\} \times T^M_{(\tau_{n-1,k_{n-1}} \vee t_{n-2}, t_{n-1}]} \times \Phi\right)\right] \\ &\times \bigcup_{k_n=0}^{\infty} \left(\{k_n\} \times T_{(0,t_n],k_n} \times \Phi^{k_n}\right) \end{aligned} \tag{23b}$$

where $T^M_{(s,t]} := \{\tau | \tau \in (s, t]\}$. 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, $\bar{U}_{n-1} := \left(\tau_{n-1,k_{n-1}}, \phi_{n-1,k_{n-1}}\right)$ 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, $\left(\bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1}\right)$ 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

$$\begin{aligned} Z_{1:n} &:= (X_1, M_1, U_1, ..., X_{n-1}, M_{n-1}, U_{n-1}, X_n) \\ &:= \left(\bar{X}_1, M_1, \bar{U}_1, ..., \bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1}, X_n\right) \end{aligned} \tag{24}$$

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

3.2.1 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 (**??**), it is the same to express the joint distribution of $Z_{1:n}$ in terms of $(X_1, M_1, U_1, ..., X_{n-1}, M_{n-1}, U_{n-1}, X_n)$ as to express it in terms of $\left(\bar{X}_1, M_1, \bar{U}_1, ..., \bar{X}_{n-1}, M_{n-1}, \bar{U}_{n-1}, X_n\right)$. 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 $\left(\bar{X}_1, \bar{X}_2, ..., \bar{X}_{n-1}, X_n\right)$. 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

$$\begin{aligned} \bar{\gamma}_n\left(Z_{1:n}\right) := \gamma_n\left(\bar{x}_1, ..., \bar{x}_{n-1}, x_n\right) &\mu_n\left(m_{1:n-1} | \bar{x}_1, ..., \bar{x}_{n-1,}, x_n\right) \\ &\times \prod_{j=1}^{n-1} \lambda_j\left(\bar{u}_j | m_{1:j}, \bar{x}_1, ..\bar{x}_j\right) \end{aligned} \tag{25}$$

As in the previous section, 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 defined the PDP, $\zeta_{(0,t_n]}$ at the n-th SMC step. Then, we will have that

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

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + 1 + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \tag{26a}$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1} + k_n, \mathcal{J}_{n-1} \backslash \left\{\hat{k}_{n-1}, \hat{\tau}_{n-1,\hat{k}_{n-1}}, \hat{\phi}_{n-1,\hat{k}_{n-1}}\right\}, \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \tag{26b}$$

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

$$\mathcal{J}_n := \left(\hat{k}_{n-1}+k_n, \mathcal{J}_{n-1}\backslash\left\{\hat{k}_{n-1}\right\}, \tau_{n,1:k_n}, \phi_{n,1:k_n}\right) \tag{26c}$$

Here, we use that convention that if $k_n=0$, then Equation (**??**), (**??**) and (**??**) 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.

3.2.2 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

$$\begin{aligned} K_{n,2}\left(u_{n-1}|m_{n-1}=1, z_{1:n-1}\right) :=& \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,1}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,1}\left(\mathring{\phi}_{n-1}|\mathcal{J}_{n-1}\right) \end{aligned} \tag{27a}$$

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

$$\begin{aligned} K_{n,2}\left(u_{n-1}|a, z_{1:n-1}\right) :=& \mathbb{1}\left(k_{n-1}\geq 1\right) \mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}-1} \vee t_{n-2} < \mathring{\tau}_{n-1} \leq t_{n-1}\right) \\ & \times \alpha_{n-1,0}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right) \beta_{n-1,0}\left(\mathring{\phi}_{n-1}|\mathcal{J}_{n-1}\right) \\ & + \mathbb{1}(k_{n-1}=0) \times \delta_{\emptyset}(u_{n-1}) \end{aligned} \tag{27b}$$

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.

$$\begin{aligned} K_{n,3}\left(x_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) :=& K^1_{n,3}\left(k_n|m_{n-1}, u_{n-1}, z_{1:n-1}\right) \\ & \times K^2_{n,3}\left(\tau_{n,1:k_n}, \phi_{n,1:k_n}|k_n, m_{n-1}, u_{n-1}, z_{1:n-1}\right) \end{aligned} \tag{28}$$

3.2.3 Artificial Densities

For the density $\mu_n\left(m_{1:n-1}|\bar{x}_1,...,\bar{x}_{n-1},x_n\right)$ in (**??**), 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.

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

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

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

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, $\lambda_j\left(\bar{u}_j|m_{1:j},\bar{x}_1,..,\bar{x}_j\right)$, 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 $\lambda_j\left(\bar{u}_j|m_{1:j},\bar{x}_1,...,\bar{x}_j\right)$ becomes degenerate and equals to 1 in these cases. When $M_j=0$ and $k_j\geq 1$, $\lambda_j\left(\bar{u}_j|m_{1:j},\bar{x}_1,...,\bar{x}_j\right):=\lambda_j\left(\hat{\tau}_{j,\hat{k}_j},\hat{\phi}_{j,\hat{k}_j}|m_{1:j},\bar{x}_1,...,\bar{x}_j\right)$ 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. An optimal choice of this density will be discussed later.

3.2.4 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 follow.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 'adjust' is proposed,

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

$$G_n\left(z_{1:n}\right):=\frac{S\left(\hat{\tau}_{n,\hat{k}_n},t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}},t_{n-1}\right)}\times\frac{\mu_{n-1}(m_{n-1}|\bar{x}_{n-1})\lambda_{n-1}(\bar{u}_{n-1}|m_{1:n-1},\bar{x}_{1:n-1})}{K_{n,1}\left(0|\mathcal{J}_{n-1}\right)}\times g\left(y_{(t_{n-1},t_n]}|\mathcal{J}_n\right)$$
$$\times\frac{h\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|\hat{\tau}_{n-1,\hat{k}_{n-1}},\hat{\phi}_{n-1,\hat{k}_{n-1}}\right)}{K^1_{n,3}\left(k_n|0,\emptyset,z_{1:n-1}\right)K^2_{n,3}\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|k_n,0,\emptyset,z_{1:n-1}\right)} \tag{29a}$$

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

$$G_n\left(z_{1:n}\right):=\frac{S\left(\hat{\tau}_{n,\hat{k}_n},t_n\right)}{S\left(\hat{\tau}_{n-1,\hat{k}_{n-1}},t_{n-1}\right)}\times\frac{\mu_{n-1}(m_{n-1}|\bar{x}_{n-1})}{K_{n,1}\left(0|\mathcal{J}_{n-1}\right)}\times\frac{g\left(y_{\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}\wedge\mathring{\tau}_{n-1},t_n\right]}|\mathcal{J}_n\right)}{g\left(y_{\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}\wedge\mathring{\tau}_{n-1},t_{n-1}\right]}|\mathcal{J}_{n-1}\right)}$$
$$\times\frac{\lambda_{n-1}\left(\bar{u}_{n-1}|m_{1:n-1},\bar{x}_{1:n-1}\right)}{\mathbb{1}\left(\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\vee t_{n-2}<\mathring{\tau}_{n-1}\leq t_{n-1}\right)\alpha_{n-1,0}\left(\mathring{\tau}_{n-1}|\mathcal{J}_{n-1}\right)\beta_{n-1,0}\left(\mathring{\phi}_{n-1}|\mathring{\tau}_{n-1},\mathcal{J}_{n-1}\right)}$$
$$\times\frac{f\left(\mathring{\tau}_{n-1}|\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)q\left(\mathring{\phi}_{n-1}|\hat{\phi}_{n-1,\hat{k}_{n-1}-1},\mathring{\tau}_{n-1},\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)}{f\left(\hat{\tau}_{n-1,\hat{k}_{n-1}}|\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)q\left(\hat{\phi}_{n-1,\hat{k}_{n-1}}|\hat{\phi}_{n-1,\hat{k}_{n-1}-1},\tau_{n-1,\hat{k}_{n-1}},\hat{\tau}_{n-1,\hat{k}_{n-1}-1}\right)}$$
$$\times\frac{h\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|\mathring{\tau}_{n-1},\mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|0,u_{n-1},z_{1:n-1})K^2_{n,3}\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|k_n,0,u_{n-1},z_{1:n-1}\right)} \tag{29b}$$

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

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

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

$$\times\frac{h\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|\mathring{\tau}_{n-1},\mathring{\phi}_{n-1}\right)}{K^1_{n,3}(k_n|m_{n-1}=1,u_{n-1},z_{1:n-1})K^2_{n,3}\left(\tau_{n,1:k_n},\phi_{n,1:k_n}|k_n,m_{n-1}=1,u_{n-1},z_{1:n-1}\right)} \tag{30}$$

In Equation (**??**), (**??**) and (**??**), $h\left(\tau_{n,1:k_n}, \phi_{n,1:k_n} | \mathring{\tau}_{n-1}, \mathring{\phi}_{n-1}\right)$ 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 **??**. The sampling schemes are not specifically outlined according to each case. More specifically, U^i_{n-1} in line 12 of Algorithm **??** will become $\emptyset$ when $M^i_{n-1} = 0$ and $k^{A^i_{n-1}}_{n-1} = 0$.

3.3 Some Ideas about the BlockVRPF method

In this previous section, we define $Z_n := (U_{n-1}, M_{n-1}, X_n)$ for $n > 1$ and derive an SMC algorithm treating the Z_n as the state for the n-th SMC step. One potential drawback of this formulation is that the modification (M_{n-1}, U_{n-1}) in Z_n may come with bad proposal jump times and values in the interval $(t_{n-1}, t_n]$, which is denoted by X_n. In this case, the way we calculate the incremental weights will also assign (M_{n-1}, U_{n-1}) with a small weight, even when they are in fact strong modifications. One possible way to alleviate this problem is to separate Z_n into two individual steps. More specifically, we can consider an SMC algorithm that alternatively perform the generation of jump times and values in the new time interval and the modification of jump times and values in the previous blocks. Hence, we can define $Z_1 := X_1, Z_2 := X_2, Z_3 := (M_1, U_1), Z_4 := X_3$, etc.

4 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.

4.1 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

$$p(y_{1:P}|\theta) = \int p(x_{1:P}, y_{1:P}|\theta)dx_{1:P} = \int p(x_{1:P}|\theta)p(y_{1:P}|x_{1:P}, \theta)dx_{1:P} \tag{31}$$

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 unnomarlised density $\pi_P(\theta, x_{1:P}, y_{1:P})$ can be evaluated point-wise, one can also replace *Step 1* with a Matropolis-Hastings step when direct sampling is not possible. On the other hand, sampling from the denstiy in *Step 2* is in general intractable except for some specific scenarios such as linear Gaussian models and finite state Hidden Markov models (**?**). Hence, prcatically one should replace *Step 2* with a Metropolis-Hastings update. In order ensure good performance of the MH update in *Step 2*, ond 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 emprical 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

$$\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P}) := \sum_{n=1}^{N} W_P^n \delta_{x_{1:P}^n}(dx_{1:P}) \tag{32}$$

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 (**??**) 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

$$q(x_{1:P}) := \mathbb{E}_{W_{1:P}^{1:N}, x_{1:P}^{1:N}} \left[\hat{\pi}_P(dx_{1:P}|\theta', y_{1:P})\right] \tag{33}$$

where the expectation is taken with respect to all the random variables generated by the SMC algorithm (**?**). 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 algorihtm 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

$$\psi_P^{N,\theta}(\mathbf{x}_1, ..., \mathbf{x}_P, \mathbf{a}_1, ..., \mathbf{a}_{P-1}) := \left\{\prod_{j=1}^{N} K_1\left(x_1^j\right)\right\} \prod_{n=2}^{P} \left\{r^\theta\left(\mathbf{a}_{n-1}|\mathbf{W}_{n-1}\right) \prod_{j=1}^{N} K_n\left(x_n^j | x_{1:n-1}^{a_{n-1}^j}\right)\right\} \tag{34}$$

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 algorihtm 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 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 fullfilled 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

$$\tilde{\pi}_P(b_{1:P}, \theta, \mathbf{x}_{1:P}, \mathbf{a}_{1:P-1}) = \frac{\pi_P(\theta, x_{1:P}^{b_{1:P}}, b_{1:P})}{N^P} \psi_P^{N,\theta}(\mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k} || \theta, x_{1:P}^{b_{1:P}}, b_{1:P}) \tag{35}$$

where we define $\mathbf{x}_{1:P}^{-k} := \mathbf{x}_{1:P} \backslash x_{1:P}^{b_{1:P}^k}$ and similarly $\mathbf{a}_{1:P-1}^{-k} := \mathbf{a}_{1:P-1} \backslash a_{1:P-1}^{b_{2:P}^k}$.

Hence, we can define the joint distribution of $\mathbf{X}_{1:P}^{-k}$ and $\mathbf{A}_{1:P-1}^{-k}$ will be given by

$$\begin{aligned}\psi_P^N\left(\mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k} || x_{1:P}^k, b_{1:P}^k\right) &:= \frac{\psi_P^N(\mathbf{x}_1, ..., \mathbf{x}_P, \mathbf{a}_1, ..., \mathbf{a}_{P-1})}{K_1\left(x_1^{b_1^k}\right)\prod_{n=2}^P\left\{r\left(b_{n-1}^k|\mathbf{W}_{n-1}\right)K_n\left(x_n^{b_n^k}|x_{1:n-1}^{b_{1:n-1}^k}\right)\right\}} \\ &= \left\{\prod_{1\leq j\leq N, j\neq b_1^k} K_1\left(x_1^j\right)\right\}\prod_{n=2}^P r\left(\mathbf{a}_{n-1}^{-b_n^k}|\mathbf{W}_{n-1}, a_{n-1}^{b_n^k}\right) \\ &\times \prod_{1\leq j\leq N, j\neq b_n^k} K\left(x_n^j|x_{1:n-1}^{a_{n-1}^j}\right)\end{aligned} \tag{36}$$

The PMCMC methods actually target an extended distribution given by

$$\bar{\pi}_P^N\left(\theta, k, b_{1:P}^k, x_{1:P}^k, \mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k}\right) = \frac{\pi_P(\theta, x_{1:P}^k)}{N^P}\psi_P^N\left(\mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k} || x_{1:P}^k, b_{1:P}^k\right) \tag{37}$$

Note that Equation (**??**) is included in the extended distribution and this is actually the conditional distribution of $\left(\mathbf{X}_{1:P}^{-k}, \mathbf{A}_{1:P-1}^{-k}\right)$ given the path $\left(k, X_{1:P}^k, B_{1:P}^k\right)$ in $\bar{\pi}_P^N$. This gives actaully inspires the idea of partible Gibbs sampler described in Andrieu et al. (2010). Therefore, at each iteration, given we have obtained $\left(\theta, k, b_{1:P}^k, x_{1:P}^k, \mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k}\right)$, we can do the following to obtain a new set of random variables

1. Sample $\theta^* \sim \bar{\pi}_P^N\left(\cdot|b_{1:P}^k, x_{1:P}^k, \mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k}\right)$
2. Sample $\mathbf{X}_{1:P}^{*,-k}, \mathbf{A}_{1:P-1}^{*,-k} \sim \bar{\pi}_P^N\left(\cdot|\theta^*, b_{1:P}^k, x_{1:P}^k\right) = \psi_P^N\left(\mathbf{x}_{1:P}^{-k}, \mathbf{a}_{1:P-1}^{-k} || x_{1:P}^k, b_{1:P}^k\right)$
3. Sample $k^*, b_{1:P}^*, x_{1:P}^{k^*} \sim \bar{\pi}_P^N\left(\cdot|\theta^*, \mathbf{X}_{1:P}^{*,-k}, \mathbf{A}_{1:P-1}^{*,-k}, b_{1:P}^k, x_{1:P}^k\right)$

As described in Liu(2001), step 1 can be simplified to sample $\theta^* \sim \pi_P(\cdot|x_{1:P}^k)$ and this still leaves the target density $\bar{\pi}_P^N$ invariant. This is a special Gibbs sampler known as 'collapsed' Gibbs sampler. For step 2, we need a special type of SMC sampler, named Conditional SMC (CSMC). This was first described in Andrieu (2010) and the main idea of SMC is to run a SMC sampler, keeping a path, $\left(X_{1:P}^k, B_{1:P}^k\right)$ unchanged. Algorithm **??** gives the details of the conditional SMC sampler.

For step 3, a path is chosen from the N distinct paths generated from the SMC sampler. Since each path is associated with a normalised weight W_P^j, we therefore chose the j-th path at step 3 with probability W_P^j. As long as we chosen the j-th path, the ancestor indices $B_{1:P}$ will be set deterministically by the relation

$$B_P = j \qquad B_n = A_n^{B_{n+1}}, \text{for } n = P-1, .., 1$$

Algorithm **??** gives a detailed outline of a particle Gibbs sampler.

5 Particle Gibbs with Backward Sampling

In the previous section, the particle Gibbs sampler only samples a particle at the final time P according to the importance weight, and traces the ancestral lineage back of the particle 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. To do this, instead of only sampling the last particle and trace back to find the path, we sample $B_{1:P}$ one by one from their full conditional distributions given, $\left(\theta^*, \mathbf{X}^*_{1:P}, \mathbf{A}^*_{1:P-1}\right)$. More precisely, step 3 of the original particle Gibbs sampler will be replace by additional P steps:

1. Sample $B^*_{1:P} \sim \bar{\pi}^N_P\left(\cdot|\theta^*, \mathbf{x}^*_{1:P}, \mathbf{a}^*_{1:P-1}\right)$ and keep B^*_P only
2. For $n = P-1, P-2, .., 2, 1$, sample $B^*_{1:n} \sim \bar{\pi}^N_P\left(\cdot|\theta^*, \mathbf{x}^*_{1:P}, \mathbf{a}^*_{1:P-1}, b^*_{n+1}, .., b^*_P\right)$ and keep B^*_n only

Below, we derive the conditional distribution for backward simulation in the particle Gibbs sampler. Also, $B_{1:P}$ will be independent of all the ancestor indices $\mathbf{A}_{1:P-1}$. Therefore, we can do the backward simulation of $B_{1:P}$ from the marginal distribution of $B_{1:P}$ and $\mathbf{X}^*_{1:P}$ relative to $\bar{\pi}^N_P$. By summing over all the choices of $\mathbf{A}_{1:P-1}$, the marginal distribution of $B_{1:P}$ and $\mathbf{X}^*_{1:P}$ will be given by

$$\bar{\pi}^N_P\left(\theta, b_{1:P}, \mathbf{x}_{1:p}\right) = \frac{\pi_P\left(\theta, x^{b_{1:P}}_{1:P}\right)}{N^P} \frac{\left\{\prod_{j=1}^N K_1(x^j_1)\right\} \prod_{n=2}^P \left[\prod_{j=1}^N \left\{\sum_{l=1}^N W^l_{n-1} K_n(x^j_n|x^l_{1:n-1})\right\}\right]}{K_1\left(x^{b_1}_1\right) \prod_{n=2}^P r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x^{b_n}_n|x^{b_{n-1}}_{1:n-1})} \tag{38}$$

Therefore, the conditional distribution for sampling the ancestor indices will be given by:

$$\begin{aligned}
\bar{\pi}^N_P\left(b_{1:P}|\theta^*, \mathbf{x}^*_{1:P}, \mathbf{a}^*_{1:P-1}\right) &= \bar{\pi}^N_P\left(b_{1:P}|\theta^*, \mathbf{x}^*_{1:P}\right) \propto \bar{\pi}^N_P\left(b_{1:P}, \theta^*, \mathbf{x}^*_{1:P}\right) \\
&\propto \frac{\pi_P\left(\theta, x^{b_{1:P}}_{1:P}\right)}{N^P} \frac{1}{K_1\left(x^{b_1}_1\right) \prod_{n=2}^P r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x^{b_n}_n|x^{b_{n-1}}_{1:n-1})} \\
&\propto \frac{\pi_P\left(\theta, x^{b_{1:P}}_{1:P}\right)}{K_1\left(x^{b_1}_1\right) \prod_{n=2}^P r(b_{n-1}|\mathbf{W}_{n-1}) K_n(x^{b_n}_n|x^{b_{n-1}}_{1:n-1})}
\end{aligned} \tag{39}$$

For Step 1 in the above backward simulation scheme, we sample $B^*_P = k$ with probability proportional to the backward simulation weight

$$W^k_{P|P} \propto \frac{\pi_P\left(\theta^*, x^k_{1:P}\right)}{p\left(x^k_{1:P}\right)} \propto W^k_P \tag{40a}$$

And for $n = P-1,..,1$, we sample $B_n^* = k$ with probability proportional to the backward simulation weight given by

$$W_{n|P}^k \propto W_n^k \frac{\pi_P\left(\theta^*, x_{1:n}^k, x_{n+1}^*, x_{n+2}^*, .., x_P^*\right)}{\pi_n\left(\theta^*, x_{1:n}^k\right)} \tag{40b}$$

We will give details on the expression of Equation (**??**) for both VRPF and BlockVRPF method. For the VRPF method, we have defined $X_n := (k_n, \tau_{n,1:k_n}, \phi_{n,1:k_n})$. Hence, when $n = P-1,..,1$, if $\sum_{j=n+1}^{P} k_j^* = 0$ (i.e. there are no jumps in $X_{n+1}^*, .., X_P^*$), $W_{n|P}^k$ will be given by

Algorithm **??** outlines the detail of particle Gibbs sampler with backward simulation. By incorporating backward simulation, instead of fixing a path deterministically, the algorithm now can explore all the possible combinations of the particles generated from the SMC sampler and hence the probability of a new path being proposed become much higher. Therefore, the particle Gibbs with backward simulation will hugely improve the mixing of the Gibbs sampler, hence reducing the variance for estimating the static-parameters.

6 Metropolis-Hastings within Particle Gibbs

Often, it is impossible to sample a new parameter exactly from the posterior parameter density $\pi_P(\theta|x_{1:P})$. Lindsten et al. (2018) proposed a method that combines Metropolis-Hastings method and Gibbs sampler together. The method replaces Line 3 of Algorithm **??** with a Metropolis-Hastings update. Instead of sampling $\theta^{(n)}$ directly from $\pi_P\left(|x_{1:P}^{(n-1)}\right)$, a new value of θ, θ', is proposed according to a proposal density $q\left(\cdot|\theta^{(n-1)}\right)$. We will then accept this new value with probability

$$\alpha\left(\theta^{(n-1)}, \theta'\right) := \min\left(1, \frac{\pi_P\left(\theta'|x_{1:P}^{(n-1)}\right)}{\pi_P\left(\theta^{(n-1)}|x_{1:P}^{(n-1)}\right)} \frac{q\left(\theta^{(n-1)}|\theta'\right)}{q\left(\theta'|\theta^{(n-1)}\right)}\right) \tag{41}$$

which is a standard Metropolis-Hastings update. For the PDP models, the posterior parameter density is also possible to calculate since

$$\pi_P\left(\theta'|x_{1:P}\right) \propto \pi_P\left(x_{1:P}|\theta', y_{(0,t_P]}\right) p\left(\theta'\right) \propto p\left(y_{(0,t_P]}|x_{1:P}, \theta')p(x_{1:P}|\theta'\right) p\left(\theta'\right)$$

By plugging the Metropolis-Hastings step into particle Gibbs with Backward Simulation scheme, we obtain the Metropolis within particle Gibbs method (MwPG). This is represented in Algorithm

7 Auxiliary Variable Rejuvenation

In this section, we discussed the rejuvenation step introduced by **?** to improve the mixing of the particle Gibbs scheme. For the BlockVRPF scheme, the incremental weights at step n, described in Equation (**??**) and (**??**), 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 $(x_{1,m}, x_{2,m}, .., x_{n-1,m}, x_n)$ and Θ. Therefore, we are only interested in the distribution $\gamma_n(x_{1,m}, .., x_{n-1,m}, x_n)$ in Equation (**??**). Hence, $M_{1:n-1}$ and $U_1^{'}, ..., U_{n-1}^{'}$ 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 **??**

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 Derivation of Full Conditional Densities Jump-value Proposals

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

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

Algorithm .3: Variable Rate Particle Filter (VRPF)

for *n=1* **do**
  for $i = 1, 2, ..., N$ **do**
    Sample $X_1^i := \left(k_1^i, \tau^i_{1,1:k_1^i}, \phi^i_{1,1:k_1^i}, \phi_0^i\right) \sim K_1(\cdot)$;
    Set $\mathcal{J}_1^i := \left(\tau^i_{1,1:k_1^i}, \phi^i_{1,1:k_1^i}\right)$;
    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 $X_n^i := \left(k_n^i, \tau^i_{n,1:k_n^i}, \phi^i_{n,1:k_n^i}\right) \sim K_n\left(\cdot|X_{1:n-1}^{A_{n-1}^i}\right)$ 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 (**??**);
    if $k_n^i \geq 1$ **then**
      Set $\mathcal{J}_n^i := \left(\hat{k}_{n-1}^{A_{n-1}^i} + k_n^i, \mathcal{J}_{n-1}^{A_{n-1}^i} \backslash \left\{\hat{k}_{n-1}^{A_{n-1}^i}\right\}, \tau^i_{n,1:k_n^i}, \phi^i_{n,1:k_n^i}\right)$;
      Calculate the un-normalised weight, $G_n(X_{1:n}^i)$, using Equation (**??**);
  for $i = 1, 2, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

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

Algorithm .4: 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 (**??**), (**??**) or (**??**) based on $\mathcal{J}_{n-1}^{A_{n-1}^i}$;
    Calculate the incremental weight $G_n\left(z_{1:n}^i\right)$ according to (**??**), (**??**) or (**??**);
  for $n = 1, ..., N$ **do**
    Calculate the normalised weight W_n^i such that

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

Algorithm .5: Conditional SMC Sampler (CSMC)

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

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

Algorithm .7: 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 $\theta^{(n)} \sim \pi_P\left(\cdot|X_{1:P}^{(n-1)}\right)$;
  Run a cSMC algorithm conditional on $X_{1:P}^{(n-1)}, B_{1:P}^{(n-1)}$, outlined in Algorithm **??**, to get $\mathbf{X}_{1:P}^{(n)}$ and $\mathbf{A}_{1:P-1}^{(n)}$;
  Run a backward simulation to get $B_{1:P}^{(n)}$:;
  for $n = P$ **do**
    Sample $B_P^{(n)} = k$ with probability proportional to $W_{P|P}^k$, described in (**??**);
  for $n = P-1, .., 1$ **do**
    Sample $B_n^{(n)} = k$ with probability proportional to $W_{n|P}^k$, described in (**??**);
  Set $X_{1:P}^{(n)} = \left(X_1^{(n),B_1^{(n)}}, ..., X_P^{(n),B_P^{(n)}}\right)$;

Algorithm .8: Metropolis within particle Gibbs (MwPG)

Initialise $\theta^{(0)}$,$X_{1:P}^{(0)}$ and $B_{1:P}^{(0)}$ arbitrarily;
for *n = 1,2,...* **do**
  Perform a Metropolis-Hastings step to obtain a new value of θ;
  - Propose $\theta' \sim q(\cdot|\theta^{(n-1)})$
  - Set $\theta^{(n)} := \theta'$ with probability given in Equation (**??**)
  - Otherwise, set $\theta^{(n)} := \theta^{(n-1)}$

  Run a cSMC algorithm conditional on $X_{1:P}^{(n-1)}, B_{1:P}^{(n-1)}$, outlined in Algorithm **??**, to get $\mathbf{X}_{1:P}^{(n)}$ and $\mathbf{A}_{1:P-1}^{(n)}$;
  Run a backward simulation to get $B_{1:P}^{(n)}$ using the same method as in Algorithm **??**;
  Set $X_{1:P}^{(n)} = \left(X_1^{(n),B_1^{(n)}}, ..., X_P^{(n),B_P^{(n)}}\right)$;

Algorithm .9: particle Gibbs with rejuvenation step

Initialise $\theta^{(0)}$, $X_{1:P}^{(0)}$ and $B_{1:P}^{(0)}$ arbitrarily;

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

  Sample $\theta^{(n)} \sim \tilde{\pi}_P\left(\cdot|x_I^{(n-1)}\right)$;

  Sample $x_A^* \sim L\left(\cdot|x_I^{(n-1)}\right)$;

  Set $X_{1:P}^{(n-1)} := \left(x_I^{(n-1)}, x_A^*\right)$;

  Run a cSMC algorithm conditional on $X_{1:P}^{(n-1)}, B_{1:P}^{(n-1)}$, outlined in Algorithm **??**, to get $\mathbf{X}_{1:P}^{(n)}$ and $\mathbf{A}_{1:P-1}^{(n)}$;

  Run a backward simulation to get $B_{1:P}^{(n)}$ using the same method as in Algorithm **??**;

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