Particle MCMC With Poisson Resampling: Parallelization and Continuous Time Models

Tomasz Cąkała , Błażej Miasojedow & Wojciech Niemiro

To cite this article: Tomasz Cąkała , Błażej Miasojedow & Wojciech Niemiro (2020): Particle MCMC With Poisson Resampling: Parallelization and Continuous Time Models, Journal of Computational and Graphical Statistics, DOI: 10.1080/10618600.2020.1840998

To link to this article: https://doi.org/10.1080/10618600.2020.1840998

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2020, VOL. 00, NO. 0, 1–14
https://doi.org/10.1080/10618600.2020.1840998

Particle MCMC With Poisson Resampling: Parallelization and Continuous Time Models

Tomasz Cąkała[a], Błażej Miasojedow[a], and Wojciech Niemiro[a,b]

[a]Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, Warsaw, Poland; [b]Nicolaus Copernicus University, Torun, Poland

ABSTRACT
We introduce a new version of particle filter in which the number of "children" of a particle at a given time has a Poisson distribution. As a result, the number of particles is random and varies with time. An advantage of this scheme is that descendants of different particles can evolve independently. It makes easy to parallelize computations. Moreover, particle filter with Poisson resampling is readily adapted to the case when a hidden process is a continuous time, piecewise deterministic semi-Markov process. We show that the basic techniques of particle MCMC, namely particle independent Metropolis-Hastings, particle Gibbs sampler and its version with ancestor sampling, work under our Poisson resampling scheme. Our version of particle Gibbs sampler is uniformly ergodic under the same assumptions as its standard counterpart. We present simulation results which indicate that our algorithms can compete with the existing methods. Supplemental materials for this article are available online.

ARTICLE HISTORY
Received August 2019
Revised October 2020

KEYWORDS
Ancestor sampling; Gibbs sampler; Hidden Markov model; Independent Metropolis–Hastings algorithm; Piecewise deterministic semi-Markov process; Pseudo-marginal; Sequential Monte Carlo

1. Introduction

Particle filters (PF) and more generally sequential Monte Carlo methods (SMC) (Gordon, Salmond, and Smith 1993; Doucet, Freitas, and Gordon 2001; Moral, Doucet, and Jasra 2006) are general framework for statistical inference for state-space models. SMC methods have proven effective in various scenarios covering: object tracking, time series analysis in non-Gaussian models (Gordon, Salmond, and Smith 1993; Doucet, Freitas, and Gordon 2001), graphical models (Naesseth, Lindsten, and Schn 2014), rare events estimation (Crou et al. 2011), phylogenetic inference (Bouchard-Ct, Sankararaman, and Jordan 2012) and model selection (Schafer and Chopin 2011). The seminal paper Andrieu, Doucet, and Holenstein (2010) introduced particle Markov chain Monte Carlo methods (PMCMC), which combine strengths of MCMC and SMC algorithms. We provide a brief introduction to SMC in the Supplementary Material, Section A.

Most of the research on SMC methods and their extensions is focused on discrete time models. However, SMC methods for continuous time models are also very active area of research. The general theoretical framework of particle filter methods in continuous setting, together with stability results can be found in Rousset (2006) and Del Moral and Penev (2017). Recently, Arnaudon and del Moral (2020) extends unbiasedness properties of Feynman-Kac models to continuous time settings. This allows to properly define and analyze the stability properties of a particle Gibbs-Glauber sampler over the path space in the general setting. The practical implementations of SMC methods are available for diffusion processes (Fearnhead, Papaspiliopoulos, and Roberts 2008; Gloaguen, Étienne, and Le Corff 2018). In the case of semi-Markov processes there exist algorithms based on discretisation of time; see, for example (Golightly and Wilkinson 2011) or by introducing rather complex birth-death moves (Finke, Johansen, and Span 2014). The main limitation of the existing SMC methods for semi-Markov processes is that they do not fit in PMCMC algorithms. The main issue is that the standard resampling step requires synchronization of all the paths. This becomes difficult in the continuous-time setting because generations of particles overlap. If some sort of discretisation is required to perform resampling step, it is very challenging to put such an algorithm within PMCMC framework. In this article, we propose a new resampling scheme which overcomes this difficulty.

1.1. Our Contribution

In this article we introduce a unified approach for both discrete and continuous time setting. We propose a new Poisson resampling scheme. In contrast to the standard scheme, where children during resampling step choose their parents, in our approach parents choose their children. This difference allows us to easily design SMC algorithm for continuous time model. Our algorithm PTPF (Poisson Tree PF), similarly to Paige et al. (2014) generates a branching process. The main advantage of our approach, and the difference from Paige et al. (2014), is that PTPF can be directly used within PMCMC methods. Moreover, our framework allows us to perform ancestor sampling in the particle Gibbs algorithm in the spirit of Lindsten, Jordan, and Schon (2014). We prove that our version of Particle Gibbs Sampler (in the discrete time setting) is uniformly ergodic, under the same assumptions as its standard counterpart (Lindsten, Douc, and Moulines 2015). In addition, rather surprisingly, this scheme allows for a straightforward

CONTACT Błażej Miasojedow bmiasojedow@gmail.com Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, ul. Banacha 2 02-097 Warsaw, Poland.

extension of PMCMC to a wide class of piecewise deterministic processes (PDP). They are processes which evolve deterministically in continuous time except for a countable collection of stopping times at which they randomly jump, see (Davis 1984). PDPs have recently attracted much attention because they are most natural models of a lot of phenomena in biology (Rudnicki and Tyran-Kamińska 2017) and in other branches of science.

The standard resampling scheme is challenging to implement in parallel; see, for example, (Paige et al. 2014; Murray, Lee, and Jacob 2016). Our scheme is much easier to parallelize, due to the fact that only partial synchronization is required in Poisson resampling. In addition, using Poisson thinning and superposition techniques we can further reduce synchronization of threads.

Since the Poisson resampling produces a random (and varying) number of particles, it is essential to control their population size. To make our algorithms practically applicable, we have to introduce some sort of synchronization of particles. (Even though synchronization is *not* necessary to ensure convergence to the target.) The inherently parallel structure of Poisson resampling has to be reconciled with (partial) synchronization. We have designed some rules (recipes for choosing control parameters in PTPF) which keep the population of particles approximately constant.

Apart from theoretical considerations, we demonstrate our method on a few challenging examples. Our simulations indicate that in the discrete time setting our algorithms yield results as good as standard PMCMC. Parallel implementation of our algorithms leads to significant gains in efficiency. For continuous time models our algorithms can compete with the existing methods, and in some examples outperform them.

The article is organized as follows. Section 2 introduces a general class of semi-Markov state–space models including discrete time state–space models and piece-wise deterministic hidden Markov processes. Next Section 3 presents our basic particle filter (PTPF). Section 4 shows how to construct PMCMC methods based on PTPF. Section 5 introduces specific rules designed to control the size of the population of particles and we define our version of ancestor resampling. Finally, Section 6 presents numerical simulations.

2. Semi-Markov State-Space Models

We consider a rather general family of state-space semi-Markov models. We first consider continuous time models. Assume that $\{\Xi(t), t \geq t_{\min}\}$ is a *piece-wise deterministic process* with values in a Polish space $\mathcal{X}$ and with càdlàg trajectories. The process jumps randomly at a countable set of random times $T_1 < \cdots < T_k < \cdots$. Between jumps it evolves according to a deterministic law. We consider a finite time horizon $t_{\max}$ and write $M = \min\{k : T_k > t_{\max}\}$. Assume that process $\Xi = \{\Xi(t), t_{\min} \leq t \leq t_{\max}\}$ is uniquely determined by its space-time *skeleton* $(X_{1:M}, T_{1:M})$, where

$$X_k = \Xi(T_k-), \qquad k = 1, \dots, M. \tag{2.1}$$

To fix ideas, let us consider the basic special case of deterministic dynamics, namely an invertible flow. Assume that $\{G(\cdot, t), t \in \mathbb{R}\}$ is an abelian group of transformations $\mathcal{X} \to \mathcal{X}$ such that maps $t \mapsto G(x,t)$ are continuous. Then we have $\Xi(t) = G(X_k, t - T_k)$ for $T_{k-1} \leq t < T_k$. (The trajectory of Ξ in the time interval $[T_{k-1}, T_k[$ can be identified by its end location as well as by its initial location. We have chosen the former way to facilitate our construction of ancestor sampling in Section 5. In fact, the assumptions on the deterministic dynamics can be much weaker. To avoid technicalities, details are relegated to the Supplementary Material, Appendix A.) We assume that random variable M is almost surely finite. Our basic assumption is that the skeleton is Markovian, governed by a space-time stochastic transition kernel K. The (prior) probability distribution of the skeleton can be written in a concise form

$$\begin{aligned}\pi_{\text{prior}}(\mathrm{d}x_{1:m}, \mathrm{d}t_{1:m}) &= \Pr(X_1 \in \mathrm{d}x_1, T_1 \in \mathrm{d}t_1, \dots, X_m \in \mathrm{d}x_m, T_m \in \mathrm{d}t_m) \\ &= \prod_{k=1}^{m} K(x_{k-1}, t_{k-1}; \mathrm{d}x_k, \mathrm{d}t_k),\end{aligned} \tag{2.2}$$

if we adopt a convention explained below. The initial distribution is written as $\Pr(X_1 \in \mathrm{d}x_1, T_1 \in \mathrm{d}t_1) = K(x_0, t_0; \mathrm{d}x_1, \mathrm{d}t_1)$, where (x_0, t_0) is a fictitious state with $t_0 = t_{\min}$. Note also that the last point of the skeleton (x_m, t_m) falls beyond the time horizon $(t_m > t_{\max})$. In the sequel, ξ denotes a sample path of Ξ and we write $\xi_{[t',t''[} = \{\xi(t), t \leq t < t'\}$. Some more details and an explicit construction of Ξ are in the Supplementary Material.

The setup described above encompasses continuous time *piece-wise deterministic Markov processes* (Davis 1984), in particular pure jump Markov processes, and a wide class of piece-wise deterministic non-Markovian processes (Whiteley, Johansen, and Godsill 2011; Finke, Johansen, and Span 2014).

The process Ξ is hidden and thus (2.2) plays the role of the prior. Let Υ be an observed random process which depends on Ξ. The *target probability distribution* is the posterior of Ξ given $\Upsilon = y$. Since y is fixed, it will not be explicitly indicated. We just assume that we have a family of functions $\xi_{[t,t'[} \mapsto \ell(\xi_{[t,t'[})$, where we think of $\ell(\xi_{[t,t'[})$ as the conditional probability of observing trajectory $y_{[t,t'[}$ given hidden trajectory $\xi_{[t,t'[}$. The full likelihood of ξ is $\ell(\xi_{[t_{\min},t_{\max}]})$. We require that the following condition holds:

$$\ell(\xi_{[t,t''[}) = \ell(\xi_{[t,t'[})\ell(\xi_{[t',t''[}), \tag{2.3}$$

for $t < t' < t''$ (by convention, $\ell(\xi_{[t,t'[})$ is understood as $\ell(\xi_{[t,t_{\max}]})$ whenever $t' > t_{\max}$). In most applications (2.3) is satisfied. A typical situation is when $\Upsilon = (Y_1, \dots, Y_p)$ is just a sequence of "noisy measurements" on ξ at discrete "observation times", say $t_{\min} \leq {t_{\text{obs}}}^1 < \cdots < {t_{\text{obs}}}^p \leq t_{\max}$. We assume that each Y_r is sampled independently from a (possibly time-dependent) probability density $\ell_r(\cdot|\xi({t_{\text{obs}}}^r))$. The likelihood functions in this model are given by

$$\ell(\xi_{[t,t'[}) = \prod_{t \leq {t_{\text{obs}}}^r < t'} \ell_r(y_r|\xi({t_{\text{obs}}}^r))$$

and clearly fulfil the assumption (2.3). Another example of a likelihood satisfying (2.3) is when Υ a fully observed continuous time random process $\{\Upsilon(t) : t_{\min} \leq t \leq t_{\max}\}$ and $\ell(\xi_{[t,t'[})$ corresponds to $y_{[t,t'[}$. The details are described in Appendix A in Supplementary Material.

Recall that $\xi_{[t_{\min},t_{\max}]}$ is represented by its skeleton $(x_{1:m}, t_{1:m})$. Since the trajectory $\xi_{[t_{k-1},t_k[}$ depends deterministically on $\xi(t_k-) = x_k$, we can write $\ell(\xi_{[t_{k-1},t_k[}) = \ell(x_k; t_{k-1}, t_k)$. The posterior distribution of $(X_{1:M}, T_{1:M})$ is given by

$$\pi_{\text{post}}(\mathrm{d}x_{1:m}, \mathrm{d}t_{1:m}) = \frac{1}{z} \cdot \prod_{k=1}^{m} K(x_{k-1}, t_{k-1}; \mathrm{d}x_k, \mathrm{d}t_k)\ell(x_k; t_{k-1}, t_k), \quad (2.4)$$

where z is a norming constant (the integral of the likelihood with respect to the prior). Our main objects of interest are z and the posterior $\pi = \pi_{\text{post}}$. From now on, we most often drop the subscript "post".

Discrete time hidden Markov models fit in our setup as a special case (identified with piece-wise constant processes). Let $\Xi = (X_1, \dots, X_m)$ be a discrete time Markov chain (in general, inhomogeneous in time) with one-step transition kernels $P_1, \dots, P_{m-1}$. Using a convention explained earlier, the joint (prior) probability distribution is

$$\pi_{\text{prior}}(\mathrm{d}x_{1:m}) = \Pr(X_1 \in \mathrm{d}x_1, \dots, X_m \in \mathrm{d}x_m) = \prod_{t=1}^{m} P_{t-1}(x_{t-1}, \mathrm{d}x_t).$$

The natural assumption about the process of observations in the discrete time setting is that $\Upsilon = (Y_1, \dots, Y_m)$, where Y_t depends only on *one* state X_t of the Markov chain. The likelihood is of the form $\ell_t(x_t)$ and consequently,

$$\pi_{\text{post}}(\mathrm{d}x_{1:m}) = \frac{1}{z} \cdot \prod_{t=1}^{m} P_{t-1}(x_{t-1}, \mathrm{d}x_t)\ell_t(x_t). \quad (2.5)$$

3. Poisson Tree Particle Filter

To define *Poisson tree particle filter* (PTPF) and particle MCMC algorithms based on PTPF we introduce suitable notations. PTPF produces a random structure $\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$.

- $(\mathcal{V}, \mathcal{E})$ is a directed graph with the set $\mathcal{V}$ of nodes and set $\mathcal{E}$ of edges (arrows).
- $\mathbf{X} = \{X_i : i \in \mathcal{V}\}$ is a collection of random variables with values in $\mathcal{X}$.
- $\mathbf{T} = \{T_i : i \in \mathcal{V}\}$ is a collection of random variables with values in $[t_{\min}, \infty[$.
- $S \in \mathcal{V}$ is a (random) node identifying a selected path in the graph.

We will also consider two collections of random variables $\mathbf{\Lambda} = \{\Lambda_i : i \in \mathcal{V}\}$ and $\mathbf{W} = \{W_i : i \in \mathcal{V}\}$, which are functions of $\mathbb{A}$ (and of the fixed observation $\Upsilon = y$).

Graph $(\mathcal{V}, \mathcal{E})$ is a directed forest. Every node has at most one incoming edge. A generic element of $\mathcal{V}$ is denoted by i and a generic element of $\mathcal{E}$ by $i \to j$. If $i \to j$ then we write $i = \mathrm{pa}(j)$ and $j \in \mathrm{ch}(i)$. It is convenient to add a fictitious node 0 to $\mathcal{V}$ and treat the graph as a tree with root 0, adding arrows $0 \to i$ for all nodes i with $\mathrm{pa}(i) = \emptyset$. For any $i \in \mathcal{V}$ there is a unique ancestry line denoted by $\mathrm{an}(i)$. It is a sequence of nodes $(a_1(i), \dots, a_k(i))$ such that $a_k(i) = i$, $a_r(i) = \mathrm{pa}(a_{r+1}(i))$ for $r = 1, \dots, k-1$ and $\mathrm{pa}(a_1(i)) = 0$ (note that $\mathrm{an}(i)$ includes i and does not include the artificial root 0). We also write $X_{\mathrm{an}(i)} = (X_{a_1(i)}, \dots, X_{a_k(i)})$ and $T_{\mathrm{an}(i)} = (T_{a_1(i)}, \dots, T_{a_k(i)})$. To every $i \in \mathcal{V}$ there corresponds a sample path of continuous time process $\Xi_i = \{\Xi_i(t) : t_{\min} \le t < T_i\}$ determined by the space-time skeleton $(X_{\mathrm{an}(i)}, T_{\mathrm{an}(i)})$ (note that Ξ_i is defined on the right open interval $[t_{\min}, T_i[$ and $X_i = \Xi(T_i-)$, in accordance with the conventions introduced in the previous section).

We first describe PTPF informally and explain the role played by all the involved variables. Let us think that node i (or equivalently edge $\mathrm{pa}(i) \to i$) is an identifier of a "particle" which is born at time $T_{\mathrm{pa}(i)}$. Particle i evolves deterministically from its initial location till time T_i and X_i denotes its location immediately prior to T_i. If $T_i > t_{\max}$ then we say i is a terminal node, $i \in \mathcal{V}_{\text{end}}$. Otherwise, i gives birth to a set $\mathrm{ch}(i)$ of children. This is done as follows. First we choose an "intensity parameter" Λ_i (see the paragraph below). We compute the weight W_i which depends on the likelihood in the interval $[T_{\mathrm{pa}(i)}, T_i[$. Then we sample $N_i \sim \mathrm{Poiss}(\Lambda_i W_i)$ and create a set $\mathrm{ch}(i)$ of cardinality N_i (possibly empty) with arrows from i to all $j \in \mathrm{ch}(i)$. For every child $j \in \mathrm{ch}(i)$ we independently sample random pair (X_j, T_j) from a probability distribution $R(X_i, T_i; \cdot, \cdot)$, where R is a "propagation" transition kernel such that "model" kernel K is absolutely continuous with respect to R. Every child $j \in \mathrm{ch}(i)$ immediately jumps to its initial location and evolves deterministically till time T_j. The weight W_j is set to $\ell(X_j; T_i, T_j)\frac{\mathrm{d}K}{\mathrm{d}R}(X_i, T_i, X_j, T_j)$, where ℓ is the likelihood corresponding to the deterministic part of trajectory in the interval $[T_i, T_j[$ with the final location X_j, and $\frac{\mathrm{d}K}{\mathrm{d}R}$ is the Radon-Nikodym derivative: $\frac{\mathrm{d}K}{\mathrm{d}R}(x, t, x', t') = K(x, t, \mathrm{d}x', \mathrm{d}t')/R(x, t, \mathrm{d}x', \mathrm{d}t')$. This procedure is repeated until no "active" nodes are left. A node i is said to be active, $i \in \mathcal{V}_{\text{act}}$, if $T_i \le t_{\max}$ and it has not yet undergone the "propagation procedure" described above. The last stage of PTPF is selecting one node S among nodes i which satisfy $T_i > t_{\max}$. The ancestry line $\mathrm{an}(S)$ identifies a sample path of the hidden process $\{\Xi(t), t \in [t_{\min}, t_{\max}]\}$ which is used as an update in pMCMC algorithms. We also compute an estimate $\hat{Z}$ of the norming constant z.

A few more notations are needed to define PTPF precisely. Assume that for any active node i, the corresponding intensity parameter Λ_i can depend on the history of the whole process before the current time T_i. To avoid vicious circle, at every stage we can pick up (for "propagation") an active node i with the least T_i. (This last rule is introduced to simplify presentation. Later, in Section 5, it will be relaxed.) History up to time T_i, denoted $\mathcal{H}(T_i)$, is defined as a sub tree which includes nodes l, j and arrows $l \to j$ such that $T_l < T_i$, together with the corresponding variables X_l, X_j, T_l, T_j (let us remember that X_j determines the location of a particle born at moment T_l). In other words, $\mathcal{H}(T_i)$ contains information about all the particles j born before T_i and allows us to compute the likelihoods $\ell(\Xi_{j[t',t''[})$ for $t_{\min} \le t' < t'' \le T_j$. Every parameter Λ_i is a function of the history, say $\Lambda_i = \mathbb{L}(\mathcal{H}(T_i))$. Some concrete forms of function $\mathbb{L}$ will be discussed in Section 5. The initial Λ_0 is equal to a constant λ_0 chosen *a priori*. A pseudo-code defining PTPF is the following.

Algorithm PTPF (Poisson Tree Particle Filter)

```
{ Initialize: }
$\mathcal{V} := \mathcal{V}_{\text{act}} := \{0\}$; $\mathcal{E} := \emptyset$; $\mathcal{V}_{\text{end}} := \emptyset$; $X_0 := x_0$; $T_0 := t_0$;
$C_0 := \Lambda_0 := \lambda_0$; $W_0 := 1$
{ Main loop: }
while $\mathcal{V}_{\text{act}} \neq \emptyset$ do
```

```
    Choose $i \in \mathcal{V}_{\text{act}}$ with minimum $T_i$ {This requirement will be relaxed}
    if $i \neq 0$ then
      Compute $\Lambda_i := \mathbb{L}(\mathcal{H}(T_i))$ {This step will be precised later}
      $C_i := C_{\text{pa}(i)}\Lambda_i$
    end if
    Sample $N_i \sim \text{Poiss}(\Lambda_i W_i)$
    if $N_i > 0$ then
      Create set $\text{ch}(i)$ of cardinality $N_i$
      $\mathcal{V} := \mathcal{V} \cup \text{ch}(i), \mathcal{E} := \mathcal{E} \cup \{i \to j : j \in \text{ch}(i)\}$
      for all $j \in \text{ch}(i)$ do
        Sample $(X_j, T_j) \sim R(X_i, T_i; \cdot, \cdot)$ { Propagate}
        Compute $W_j := \frac{dK}{dR}(X_i, T_i, X_j, T_j)\ell(X_j; T_i, T_j)$ { Weigh }
        if $T_j > t_{\max}$ then
          $\mathcal{V}_{\text{end}} := \mathcal{V}_{\text{end}} \cup \{j\}$
        else
          $\mathcal{V}_{\text{act}} := \mathcal{V}_{\text{act}} \cup \{j\}$
        end if
      end for
    end if
    $\mathcal{V}_{\text{act}} := \mathcal{V}_{\text{act}} \setminus \{i\}$
  end while
  { Select $S$: }
  if $\mathcal{V}_{\text{end}} \neq \emptyset$ then
    $\hat{Z} := \sum_{i \in \mathcal{V}_{\text{end}}} W_i / C_{\text{pa}(i)}$
    Select $S \in \mathcal{V}_{\text{end}}$ from the probability distribution $\Pr(S = s) \propto W_s / C_{\text{pa}(s)}$
  else
    $\hat{Z} := 0$
  end if
  Output $\hat{Z}$, $(X_{\text{an}(S)}, T_{\text{an}(S)})$ { Optionally $\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$
  }
```

For discrete time models, with $\Xi = (X_1, \dots, X_m)$ a few details in PTPF become simpler. We can omit $T_{1:m}$ in the input/output. The tree produced by the algorithm is uniquely represented by $(\mathcal{V}, \mathcal{E}, \mathbf{X}, S)$. Kernel $K(x_i, t_i, dx_j, dt_j)$ is reduced to $P_{t-1}(x_i, dx_j)$, where $t_i = t - 1$ and $t_j = t$. The "propagation" kernel is analogously $R_{t-1}(x_i, dx_j)$. It is worth mentioning that R_{t-1} can depend on the "step-ahead likelihood" ℓ_t computed at some "reference point" in the spirit of auxiliary particle filters (Pitt and Shephard 1999). The set of nodes is partitioned into "generations" $\mathcal{V}_t = \{i \in \mathcal{V} : T_i = t\}$, $t = 1, \dots, m$. Nodes belonging to $\mathcal{V}_t$ propagate simultaneously and independently. The set of terminal nodes is $\mathcal{V}_{\text{end}} = \mathcal{V}_m$.

3.1. Extended Probability Distributions

The joint probability distribution of all the random variables in

$$\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$$

is called the *extended proposal*, following the terminology established in the SMC literature. The extended proposal is denoted by $\psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s)$. Values of random variables X_i, T_i and S are denoted by the corresponding small case letters x_i, t_i and s. Analogously, notations λ_i, w_i and $\hat{z}$ will be used for values of random variables Λ_i, W_i and $\hat{Z}$, which are functions of $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T})$. Consequently, in the formulas below we use the following notations.

$$w_0 = 1, \qquad w_i = \ell(x_i; t_{\text{pa}(i)}, t_i)\frac{dK}{dR}(x_{\text{pa}(i)}, t_{\text{pa}(i)}, x_i, t_i),$$
$$\lambda_i = \mathbb{L}(\mathcal{H}(t_i)),$$
$$\mathcal{V}_{\text{end}} = \{j \in \mathcal{V} : t_j > t_{\max}\}, \qquad c_j = \lambda_0 \prod_{i \in \text{an}(j)} \lambda_i.$$

Remark 3.1 (Equivalence classes). The labels given to nodes of the graph $(\mathcal{V}, \mathcal{E})$ are irrelevant to the behaviour of the algorithm. Strictly speaking, we are interested in the *equivalence classes* $[\mathbb{A}] = [(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)]$, where two structures are equivalent if they differ from each other only by labelling of the nodes. (i.e., if there is a one-to-one correspondence between the sets of nodes which preserves the set of arrows, the variables X_i T_i and S.) In a single "propagation" step of PTPF, node i "produces" n_i children with probability

$$\exp[-\lambda_i w_i]\frac{(\lambda_i w_i)^{n_i}}{n_i!}.$$

A child with label j is then assigned a pair (x_j, t_j) drawn from R. There are $n_i!$ equivalent configurations of children. Therefore, if we consider the distribution of the equivalence class, then the factorial in the Poisson probability cancels out. Let us introduce the following convention. From now on, *we work with the equivalence classes* without making explicit the distinction between a class $[\mathbb{A}]$ and its representative $\mathbb{A}$.

Now we are in a position to write a formula for the extended proposal. It is convenient to discern two stages: first the marginal distribution of all the variables except S, and then the conditional distribution of S given the rest. This exactly corresponds to the two stages of PTPF: in the "Main loop" we sample $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T})$ and the last part of the algorithm is "Selecting S". (By a harmless abuse of notation we use the same letter ψ for the joint and the marginal distributions.)

The *extended proposal* is given by

$$\psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}) = \prod_{i \in \mathcal{V}\setminus\mathcal{V}_{\text{end}}} \exp[-\lambda_i w_i]\,(\lambda_i w_i)^{|\text{ch}(i)|} \prod_{j \in \text{ch}(i)} R\left(x_i, t_i; dx_j, dt_j\right); \tag{3.2}$$

$$\psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s) = \psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t})\frac{w_s / c_{\text{pa}(s)}}{\hat{z}},$$

where

$$\hat{z} = \sum_{i \in \mathcal{V}_{\text{end}}} w_i / c_{\text{pa}(i)}.$$

In (3.2) and everywhere else we use the convention that $\prod_{i \in \emptyset} \dots = 1$. If $\mathcal{V}_{\text{end}} = \emptyset$ then s is undefined.

The *extended target* is concentrated on trees with $\mathcal{V}_{\text{end}} \neq \emptyset$ and is given by

$$\begin{aligned}\phi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s) &= \psi(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s)\frac{\hat{z}}{z} \\ &= \pi(dx_{\text{an}(s)}, dt_{\text{an}(s)}) \\ &\quad \cdot \psi_{\text{cond}}(\mathcal{V}, \mathcal{E}, d\mathbf{x}, d\mathbf{t}, s | x_{\text{an}(s)}, t_{\text{an}(s)}),\end{aligned} \tag{3.3}$$

where the *conditional proposal distribution* is

$$\begin{aligned}\psi_{\mathrm{cond}}(\mathcal{V},\mathcal{E},\mathrm{d}\mathbf{x},\mathrm{d}\mathbf{t},s|x_{\mathrm{an}(s)},t_{\mathrm{an}(s)}) &= \prod_{i\in\mathcal{V}\setminus\mathcal{V}_{\mathrm{end}}\setminus\mathrm{an}(s)} \exp\left[-\lambda_i w_i\right](\lambda_i w_i)^{|\mathrm{ch}(i)|} \prod_{j\in\mathrm{ch}(i)} R(x_i,t_i;\mathrm{d}x_j,\mathrm{d}t_j)\\ &\times \prod_{i\in\mathrm{an}(s)\setminus\{s\}} \exp\left[-\lambda_i w_i\right](\lambda_i w_i)^{|\mathrm{ch}(i)|-1} \prod_{j\in\mathrm{ch}(i)\setminus\mathrm{an}(s)} R(x_i,t_i;\mathrm{d}x_j,\mathrm{d}t_j).\end{aligned} \tag{3.4}$$

Formula (3.3) plays a crucial role in our article. It relates the result of running PTPF (extended proposal ψ) to the extended target ϕ. Thus ϕ is a probability distribution which, when marginalized to the selected path, yields the target distribution π. It is worth mentioning that (3.3) is an exact analogue of a fact established for filters with deterministic number of particles in (Andrieu, Doucet, and Holenstein 2010, see the sentence which follows Theorem 2). Rather unexpectedly, the same relation is true for PTPF.

To verify that Equations (3.3) and (3.4) are correct, it is enough to rearrange terms in $\psi(\mathcal{V},\mathcal{E},\mathrm{d}\mathbf{x},\mathrm{d}\mathbf{t},s)\hat{z}/z$. First, let us gather the terms corresponding to the selected path. By Equation (2.4) and taking into account the relation $w_j = \frac{\mathrm{d}K}{\mathrm{d}R}(x_{\mathrm{pa}(j)},t_{\mathrm{pa}(j)};x_j,t_j)\ell(x_j;t_{\mathrm{pa}(j)},t_j)$, we obtain

$$\begin{aligned}&\prod_{j\in\mathrm{an}(s)} R(x_{\mathrm{pa}(j)},t_{\mathrm{pa}(j)};\mathrm{d}x_j,\mathrm{d}t_j)w_j/z\\ &\quad= \prod_{j\in\mathrm{an}(s)} K(x_{\mathrm{pa}(j)},t_{\mathrm{pa}(j)};\mathrm{d}x_j,\mathrm{d}t_j)\ell(x_j;t_{\mathrm{pa}(j)},t_j)/z\\ &\quad= \pi(\mathrm{d}x_{\mathrm{an}(s)},\mathrm{d}t_{\mathrm{an}(s)}).\end{aligned}$$

Note that the product on the LHS includes w_s. Now consider the remaining terms. If $i\in\mathrm{an}(s)\setminus\{s\}$ then the exponent in the expression $(\lambda_i w_i)^{|\mathrm{ch}(i)|-1}$ is decreased by one, because one w_i is included in $\pi(x_{\mathrm{an}(s)},t_{\mathrm{an}(s)})$ and one λ_i is present in $c_{\mathrm{pa}(s)}$. In the product of $R(x_i,t_i;\mathrm{d}x_j,\mathrm{d}t_j)$ over the children of i, we drop one term, which corresponds to $j\in\mathrm{an}(s)$, because it is included in $\pi(x_{\mathrm{an}(s)},t_{\mathrm{an}(s)})$. Thus we see that ψ_{cond} in (3.3) is indeed given by (3.4).

Now we can define the conditional PTPF (cPTPF), that is, the algorithm which produces a configuration with the probability distribution ψ_{cond}. cPTPF differs from the basic PTPF only in that the conditioning path $(X_{\mathrm{an}(S)},T_{\mathrm{an}(S)})$ is fixed at the beginning and equal to a given $(\tilde{X}_{1:M},\tilde{T}_{1:M})$ (here and below we use "tildes" to emphasize the distinction between the variables along the conditioning path and the variables indexed by all vertices in $\mathcal{V}$.

Algorithm cPTPF (conditional PTPF)

```
Input $(\tilde{X}_{1:M},\tilde{T}_{1:M})$
{ Initialize: }
$\mathcal{V} := \mathcal{V}_{\mathrm{act}} := \{0\}$; $\mathcal{E} := \emptyset$; $X_0 := x_0$; $T_0 := t_0$; $C_0 := \Lambda_0 := \lambda_0$; $W_0 := 1$
**for** $k := 1$ to $M$ **do**
    $\mathcal{V} := \mathcal{V}\cup\{k\}$; $\mathcal{E} := \mathcal{E}\cup\{k-1\to k\}$
    $(X_k,T_k) := (\tilde{X}_k,\tilde{T}_k)$
**end for**
{ The values $(\tilde{X}_{1:M},\tilde{T}_{1:M})$ taken from the input are henceforth treated as a (fixed) branch of the tree }
$S := M$ { the endpoint $M$ of the conditioning path is treated as the singled out vertex $S$ }
$\mathcal{V}_{\mathrm{act}} := \mathcal{V}\setminus\{M\}$; $\mathcal{V}_{\mathrm{end}} := \{M\}$
{ Main loop: }
......... { the same as in PTPF }
Output $(\mathcal{V},\mathcal{E},\mathbf{X},\mathbf{T},S)$ { Tree with the conditional distribution $\psi_{\mathrm{cond}}$ }
```

4. Particle MCMC Based on PTPF

The two main particle MCMC algorithms are *particle independent Metropolis-Hastings* and *particle Gibbs sampler*. Their versions with Poisson resampling are algorithms PTMH and PTGS defined below (PT stands for *Poisson tree*). We will describe two recipes for simulating a Markov chain $\Xi^{(0)},\Xi^{(1)},\ldots,\Xi^{(n)},\ldots$, where $\Xi^{(n)} = \{\Xi^{(n)}(t) : t_{\min}\le t\le t_{\max}\}$ such that the stationary distribution is the target, that is, the posterior of hidden Ξ given $\Upsilon = y$. As usual, the trajectories are represented by their skeletons, so we actually simulate sequences $(X^{(n)},T^{(n)}) = (X^{(n)}_{1:M^{(n)}},T^{(n)}_{1:M^{(n)}})$, $n = 0,1,\ldots$. The rules of transition from $(\tilde{X},\tilde{T}) = (X^{(n)},T^{(n)})$ to $(X',T') = (X^{(n+1)},T^{(n)})$ are the following.

One step of PTMH (Poisson tree Metropolis-Hastings)

```
Input $\hat{Z}$, $(\tilde{X}_{1:M},\tilde{T}_{1:M})$ { Output of the previous step }
Run PFPF to obtain $(X^*_{1:M^*},T^*_{1:M^*})$ and $\hat{Z}^*$ { Proposal }
Sample $U\sim\mathrm{U}(0,1)$
**if** $U<\hat{Z}^*/\hat{Z}$ **then**
    $(X'_{1:M'},T'_{1:M'}) := (X^*_{1:M^*},T^*_{1:M^*})$; $\hat{Z}' := \hat{Z}^*$ { Accept }
**else**
    $(X'_{1:M'},T'_{1:M'}) := (\tilde{X}_{1:M},\tilde{T}_{1:M})$; $\hat{Z}' := \hat{Z}$ { Reject }
**end if**
Output $\hat{Z}'$, $(X'_{1:M'},T'_{1:M'})$
```

Our particle Gibbs sampler, just as its classical counterpart, can include the additional step of parent sampling. However, we first describe the basic version (without parent sampling).

One step of PTGS (Poisson tree Gibbs sampler)

```
Input $(\tilde{X}_{1:M},\tilde{T}_{1:M})$ { Output of the previous step }
Run cPFPF to obtain $(\mathcal{V},\mathcal{E},\mathbf{X},\mathbf{T},S)$ {Tree with the distribution $\psi_{\mathrm{cond}}$}
{ Forget $S$, select new $S'$: }
Select $S'\in\mathcal{V}_{\mathrm{end}}$ from the probability distribution $\Pr(S'=s')\propto W_{s'}/C_{\mathrm{pa}(s')}$
Output $(X'_{1:M'},T'_{1:M'}) := (X_{\mathrm{an}(S')},T_{\mathrm{an}(S')})$
```

In fact, the main results are straightforward consequences of (3.3).

Proposition 4.1. Let f be a nonnegative function on the space of skeletons $(x_{1:m}, t_{1:m})$ and $\pi(f) = \mathbb{E}_\pi f(X_{1:M}, T_{1:M})$. If the structure $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T})$ is produced by PTPF then the following estimator of $z\pi(f)$ is unbiased:

$$\widehat{z\pi(f)} = \begin{cases} \sum_{i\in\mathcal{V}_{\text{end}}} \frac{W_i}{C_{\text{pa}(i)}} f\left(X_{\text{an}(i)}, T_{\text{an}(i)}\right) & \text{if } \mathcal{V}_{\text{end}} \neq \emptyset; \\ 0 & \text{if } \mathcal{V}_{\text{end}} = \emptyset. \end{cases}$$

In particular, $\hat{Z}$ is an unbiased estimator of z.

Proof. By Equation (3.3), if $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S) \sim \phi$ then the marginal distribution of $(X_{\text{an}(S)}, T_{\text{an}(S)})$ is π. Therefore $\mathbb{E}_\phi f(X_{\text{an}(S)}, T_{\text{an}(S)})) = \pi(f)$. Again using Equation (3.3), we see that

$$\psi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}) \frac{w_s}{c_{\text{pa}(s)}} = \hat{z}\psi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s) = z\phi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s),$$

where $(w_s/c_{\text{pa}(s)})/\hat{z} = \Pr_\psi(S = s|\mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{t})$. Now it is enough to multiply both sides of the last display by $f(x_{\text{an}(s)}, t_{\text{an}(s)})$, integrate over $(\mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t})$ and sum over $s \in \mathcal{V}_{\text{end}}$ to obtain the result. Unbiasedness of $\hat{Z}$ follows if we put $f \equiv 1$. □

Theorem 4.2. Markov chains generated by algorithms PTMH and PTGS have the equilibrium distribution equal to the target $\pi = \pi_{\text{post}}$ given by Equation (2.4).

Proof. The line of argument is almost the same as for the classical PMCMC algorithms with multinomial resampling. The crucial point is Equation (3.3).

For PTMH, we use Equation (3.3) to infer that

$$\frac{\hat{z}^*}{\hat{z}} = \frac{\phi(\mathcal{V}^*, \mathcal{E}^*, \mathrm{d}\mathbf{x}^*, \mathrm{d}\mathbf{t}^*, s^*)\psi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s)}{\phi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s)\psi(\mathcal{V}^*, \mathcal{E}^*, \mathrm{d}\mathbf{x}^*, \mathrm{d}\mathbf{t}^*, s^*)},$$

where $\hat{z}^*, \mathcal{V}^*, \mathcal{E}^*, \mathbf{x}^*, \mathbf{t}^*, s^*$ are new values produced by running PTMH, while $\hat{z}, \mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{t}, s$ are values from the previous step. It follows that this algorithm is a proper Metropolis-Hastings procedure with the proposal distribution $\psi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s)$ and the target $\phi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s)$ on the space of configurations. The second equation in (3.3) shows that the distribution ϕ preserved by PTMH has the right marginal distribution $\pi(\mathrm{d}x_{\text{an}(s)}, \mathrm{d}t_{\text{an}(s)})$.

For PTGS, (3.3) shows that by running cPTPF we sample a configuration with the conditional distribution $\psi_{\text{cond}}(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s|x_{\text{an}(s)} = x_{1:m}, t_{\text{an}(s)} = t_{1:m})$. If, at the input, $(X_{1:m}, T_{1:m}) \sim \pi$ then configuration $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T})$ obtained by PTPF has the distribution ϕ marginalised with respect to S. Consequently, after new S' has been chosen, we obtain $(\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S') \sim \phi$ with the marginal $(X_{\text{an}(S')}, T_{\text{an}(S')}) \sim \pi$ at the output. □

Remark 4.3. Algorithms presented in this section sample a trajectory of the hidden process, given static parameters of the model. Bayesian inference on static parameters could be done in the same way as in standard PMCMC methods, for details we refer to Andrieu, Doucet, and Holenstein (2010).

5. Variants and Extensions

In this section we present several variants and extensions of the basic algorithms. In particular we introduce the additional step of ancestor sampling in our particle Gibbs algorithm. The discussion is focused on two closely related issues. First is choosing the intensity parameters Λ_i. Second is parallelization of computations.

In our description of algorithm PTPF, the step of choosing Λ_is was left unspecified. We only assumed that $\Lambda_i = \mathbb{L}(\mathcal{H}(T_i))$, without any conditions on function $\mathbb{L}$. This assumption is sufficient to ensure that our algorithms are correct, that is, the results in Section 4 and their proofs are valid. However, the efficiency of the algorithms crucially depends on the choice of Λ_is. The intensity parameters control the size of the population of particles. It is equally undesirable to allow for an uncontrolled increase and for a rapid decrease (or even extinction) of the population.

One of our objectives is to construct algorithms in which computations are performed in a parallel way. In principle, perfectly parallel versions of PTPF, PTMH, and PTGS are simple. If every parameter Λ_i depends only on an(i), that is, if we set $\mathbb{L}(\mathcal{H}(T_i)) = \mathbb{L}(X_{\text{an}(i)}, T_{\text{an}(i)})$ then the descendants of i evolve completely independently of other nodes not belonging to an(i). However, this scenario is unrealistic, because it makes the number of particles impossible to control.

Another scenario is in some sense at the opposite extreme. Suppose that to control the number of particles, we allow Λ_i to depend on all the particles existing immediately before T_i, that is, $\{j \in \mathcal{V} : T_{\text{pa}(j)} < T_i \leq T_j\}$. We apply this scenario to discrete time models and we show that it allows for at least partial parallelization of computations. For continuous time models, we propose more complicated rules for choosing Λ_is. We begin with the easier case of discrete time models.

5.1. Discrete Time Models

If time is discrete ($t = 1, \ldots, m$) then $\mathcal{V}_t = \{i \in \mathcal{V} : T_i = t\}$ is tth "generation" of particles and all $i \in \mathcal{V}_t$ propagate simultaneously. It is natural to choose a common value, $\Lambda_i = \Lambda_t$, for all $i \in \mathcal{V}_t$. An obvious way to stabilize the number of particles is to choose

$$\Lambda_t = \frac{\lambda_0}{\sum_{j\in\mathcal{V}_t} W_j}, \tag{5.1}$$

because then $\mathbb{E}|\mathcal{V}_{t+1}| = \sum_{i\in\mathcal{V}_t} \Lambda_i W_i = \lambda_0$. Our simulations show that the rule (5.1) well stabilizes not only the expected number but also the actual number of particles, see the results presented in Section 6. Moreover, PTGS with the rule (5.1) is uniformly ergodic, under the same assumptions as for the standard Particle GS. Theorem 5.2 below and the method of proof are similar to Lindsten, Douc, and Moulines (2015). We verify a Doeblin condition for one step transition of discrete time PTGS. The proof of Theorem 5.2 is given in the Supplementary Material.

Recall that in a single step, PTGS takes a trajectory $\tilde{X}_{1:m}$ and outputs a new trajectory $X'_{1:m} = X_{\text{an}(S')}$. The target distribution $\pi = \pi_{\text{prior}}$ is given by (2.5). Symbol Pr refers to the the transition probability of PTGS.

Theorem 5.2. Consider discrete time PTGS with the rule (5.1). If the likelihood functions are uniformly bounded, that is, $\|\ell_t\|_\infty = \sup_{x_t\in\mathcal{X}} \ell_t(x_t) \le c < \infty$ then the following minorisation condition holds. For every measurable subset $\mathcal{D}$ of $\mathcal{X}^m$ and every $x_{1:m} \in \mathcal{X}^m$ we have

$$\Pr(X'_{1:m} \in \mathcal{D} | \tilde{X}_{1:m} = x_{1:m}) \ge \varepsilon\pi(\mathcal{D}),$$

for some constant $\varepsilon > 0$.

This theoretical result confirms that under (5.1), PTGS is as efficient as its classical counterpart. Let us remark that Theorem 5.2 remains valid (with the same proof) also for PTGAS, the version of PTGS with ancestor sampling to be introduced in the next subsection.

Some special properties of the Poisson distribution offer a way to reconcile (5.1) with the parallel structure of computations. Well-known techniques of "thinning" and "superposition" can be used in sampling the Poisson tree $\mathbb{A}$. We can use some preliminary approximation of $\sum_{j\in\mathcal{V}_t} W_j$ to compute "tentative" value of Λ_t at every time t. Then, in the next stage, the tree can be adjusted by sampling additional children and their descendants (superposition) or removing some children and their descendants (thinning). Another method is to use only a random sample of existing particles to determine Λ_t.

5.2. Continuous Time Models

Choosing the intensity parameters is more difficult in the case of continuous time models. We have $W_i = \frac{\mathrm{d}K}{\mathrm{d}R}(X_{\mathrm{pa}(i)}, T_{\mathrm{pa}(i)}, X_i, T_i)\ell(X_i; T_{\mathrm{pa}(i)}, T_i)$, thus W_i depends on the sample path Ξ_i in the time interval $[T_{\mathrm{pa}(i)}, T_i[$. It is not reasonable to compare likelihoods which correspond to different time intervals, so a formula analogous to (5.1) would make little sense. The solution we propose is in a sense a compromise between the two "extreme" scenarios sketched in the first part of this section. Roughly speaking, we partition the interval $[t_{\min}, t_{\max}]$ into sub-intervals or "strips." The particles within every strip evolve independently. At the end of the strip we synchronize the particles and compute some statistic which is used to determine Λ_is in the next strip.

We proceed to details. The points of partition (arbitrarily chosen) are

$$t_{\min} = t_{\mathrm{syn}}{}^0 < t_{\mathrm{syn}}{}^1 < \cdots < t_{\mathrm{syn}}{}^r < \cdots < t_{\mathrm{syn}}{}^q = t_{\max}$$

(t_{syn} standing for "synchronization time"). Let

$$\mathcal{F}^r = \{i : t_{\mathrm{syn}}{}^r \le T_i < t_{\mathrm{syn}}{}^{r+1}\}.$$

If $i \in \mathcal{F}^r$ then we say that particle i is in rth strip, that is, has a chance to propagate in the interval $[t_{\mathrm{syn}}{}^r, t_{\mathrm{syn}}{}^{r+1}[$. Let

$$\mathcal{F}^r_\circ = \{i : T_{\mathrm{pa}(i)} < t_{\mathrm{syn}}{}^r \le T_i < t_{\mathrm{syn}}{}^{r+1}\} \quad \text{and}$$
$$\mathcal{G}^r = \{i : T_{\mathrm{pa}(i)} < t_{\mathrm{syn}}{}^r, T_i \ge t_{\mathrm{syn}}{}^{r+1}\}.$$

Note that $\mathcal{F}^r \setminus \mathcal{F}^r_\circ$ is the set of nodes in $\mathcal{F}^r$ whose parents are also in $\mathcal{F}^r$. We illustrate above notation in Figure 1. The number of particles that exist immediately before time $t_{\mathrm{syn}}{}^r$ is $|\mathcal{F}^r_\circ \cup \mathcal{G}^r|$. For every $i \in \mathcal{F}^r_\circ$, let

$$L^r_i = \ell(\Xi_{i[t_{\mathrm{syn}}{}^{r-1}, t_{\mathrm{syn}}{}^r[})$$

be the partial likelihood corresponding to the path Ξ_i in the *previous* strip.

Let

$$L^r_\circ = \sum_{i\in\mathcal{F}^r_\circ} L^r_i.$$

Let us emphasise that the likelihoods for different paths are computed for the same time interval $[t_{\mathrm{syn}}{}^{r-1}, t_{\mathrm{syn}}{}^r[$. Now, we propose the following rule of computing Λ_is in rth strip. Choose a non-decreasing function $b :]-\infty, \infty[\to [0, \infty]$. Assume that $\mathcal{F}^r_\circ \ne \emptyset$ and put

$$\Lambda_i = \begin{cases} \dfrac{1}{W_i}\dfrac{L^r_i}{L^r_\circ} b(\lambda_0 - |\mathcal{G}^r|) & \text{for } i \in \mathcal{F}^r_\circ; \\ \\ \dfrac{1}{W_i} & \text{for } i \in \mathcal{F}^r \setminus \mathcal{F}^r_\circ. \end{cases} \tag{5.3}$$

The idea behind this seemingly complicated formula is simple. To begin with, λ_0 is the expected *initial* number of particles. We would like to keep the number of particles as close to λ_0 as possible, in the course of building the tree. To this end, we try to control the Poisson intensities $\Lambda_i W_i$. Note that under Equation (5.3) we obtain

$$\sum_{i\in\mathcal{F}^r_\circ} \Lambda_i W_i = b\left(\lambda_0 - |\mathcal{G}^r|\right).$$

This expression is the expected number of children of nodes in $\mathcal{F}^r_\circ$ (conditioned on the history of the process before $t_{\mathrm{syn}}{}^r$). The second line in Equation (5.3) implies that for every node in $\mathcal{F}^r \setminus \mathcal{F}^r_\circ$, the expected number of children is one. Particles corresponding to $\mathcal{G}^r$ "pass through" the strip $[t_{\mathrm{syn}}{}^r, t_{\mathrm{syn}}{}^{r+1}[$ unchanged. Putting this together, we see that the (conditional) expected number of particles that exist immediately before $t_{\mathrm{syn}}{}^{r+1}$ is

$$b\left(\lambda_0 - |\mathcal{G}^r|\right) + |\mathcal{G}^r|.$$

If we put $b(l) = \max(l, 0)$ then the expected number of particles immediately before $t_{\mathrm{syn}}{}^{r+1}$ would be equal to $\max(\lambda_0, |\mathcal{G}^r|)$.

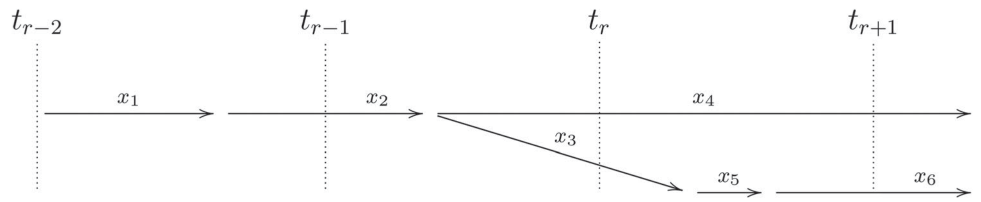

Figure 1. Example of a simple Poisson tree. Edge x_4 belongs to $\mathcal{G}^r \cap \mathcal{F}^{r+1}_\circ$. Edge x_3 belongs to $\mathcal{F}^r_\circ$ while x_5 is an element of $\mathcal{F}^r \setminus \mathcal{F}^r_\circ$.

However, if $|\mathcal{G}^r| \geq \lambda_0$ then the particles in $\mathcal{F}^r$ would have zero chance to propagate. Therefore, a reasonable strategy is to choose e.g. $b(l) = \max(l, b_0)$ for some small constant $b_0 > 0$.

Equation (5.3) involves, apart from the quantities specific to node i, only sets $\mathcal{F}^r_\circ$, $\mathcal{G}^r$ and $L^r_\circ$. This means that we have to identify all the particles which exist at moment ${t_{\text{syn}}}^r$, know their lifespans and locations before we proceed to processing nodes in $\mathcal{F}^r$. On the other hand, *we need not sort T_is for $i \in \mathcal{F}^r_\circ$*. This fact is important for efficient implementation of the algorithm. Once we create children of nodes in $\mathcal{F}^r_\circ$, we compute their lifespans and identify nodes in $\mathcal{F}^r \setminus \mathcal{F}^r_\circ$. Descendants of every node in $\mathcal{F}^r_\circ$ evolve independently until ${t_{\text{syn}}}^{r+1}$, the next moment of synchronization.

5.3. Ancestor Sampling

Although algorithm PTGS does preserve π, its mixing properties are poor because of the well-known phenomenon of path-degeneration (as for the classical particle Gibbs Sampler). A remedy is to additionally resample parents, that is, change those arrows in $\mathcal{E}$ which lead to nodes in the (old) selected path. We adapt the method proposed in Lindsten, Jordan, and Schon (2014) to our Poisson tree setting.

Before we define *Poisson tree Gibbbs sampler with ancestor sampling* (PTGAS), let us revisit PTGS. This algorithm takes a trajectory $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$ and builds a tree $\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, S)$ with conditional distribution ψ_{cond}. If the conditioning trajectory is distributed according to the target π, then $\mathbb{A}$ has the extended target distribution ϕ. Now observe that $\mathbb{A}$ can be built recurrently (epoch by epoch). If we consider continuous time, the consecutive epochs correspond to the "strips" defined by $t_{\min} = {t_{\text{syn}}}^0 < {t_{\text{syn}}}^1 < \cdots < {t_{\text{syn}}}^r < \cdots < {t_{\text{syn}}}^q = t_{\max}$. Let $\mathbb{A}^r = (\mathcal{V}^r, \mathcal{E}^r, \mathbf{X}^r, \mathbf{T}^r, S^r)$, with $\mathbf{X}^r = \{X_i : i \in \mathcal{V}^r\}$ and $\mathbf{T}^r = \{T_i : i \in \mathcal{V}^r\}$, be the (partial) tree constructed in epoch $r-1$. Roughly speaking, we think of $\mathbb{A}^r$ as a tree built by PTGS restricted to the time interval $[t_{\min}, {t_{\text{syn}}}^r[$ (instead of $[t_{\min}, t_{\max}]$). Since S^r plays the role of the selected node, we assume that $T_{S^r} \geq {t_{\text{syn}}}^r$ and $(X_{S^r}, T_{S^r}) = (\tilde{X}_{M^r}, \tilde{T}_{M^r})$, where $M^r = \min\{k : \tilde{T}_k \geq {t_{\text{syn}}}^r\}$ ("tildes" indicate the values taken from the input path). In rth epoch we extend $\mathbb{A}^r$ to $\mathbb{A}^{r+1}$. We identify M^{r+1}. The segment $(\tilde{X}_{M^r+1:M^{r+1}}, \tilde{T}_{M^r+1:M^{r+1}})$ of the conditioning trajectory (possibly empty) is rewritten as a branch of the tree $\mathbb{A}^{r+1}$ with the last node S^{r+1}. Now we allow the nodes in the $\mathcal{F}^r$ to propagate in the usual manner, stopping as soon as a new node lands outside $\mathcal{F}^r$ (the nodes $i \in \mathcal{V}^r$ such that $T_i < {t_{\text{syn}}}^r$ are already inactive). It is clear that extending the tree in this way for $r = 0, \ldots, q-1$, we finally obtain the output equivalent to that of PTGS.

PTGAS is obtained from PTGS if additionally, after every epoch r, we change the arrow leading to S^{r+1}. We sample $\text{pa}(S^{r+1})$ from all the nodes belonging to $\mathcal{F}^r$ (lying in the current strip). The new parent of S^{r+1} is chosen with probability

$$\Pr(\text{pa}(S^{r+1}) = i) \tag{5.4}$$
$$= \frac{\frac{W_i}{C_{\text{pa}(i)}} k(X_i, T_i; X_{S^{r+1}}, T_{S^{r+1}}) \ell(X_{S^{r+1}}; T_i, T_{S^{r+1}})}{\sum_{i' \in \mathcal{F}^r} \frac{W_{i'}}{C_{\text{pa}(i')}} k(X_{i'}, T_{i'}; X_{S^{r+1}}, T_{S^{r+1}}) \ell(X_{S^{r+1}}; T_{i'}, T_{S^{r+1}})},$$
for $i \in \mathcal{F}^r$,

where we assume that the transition kernel $K(x, t, \cdot, \cdot)$ is represented by transition density k, and ℓ is the likelihood.

For discrete time models, PTGAS simplifies. "Generations of nodes" corresponding to discrete time moments $t = 1, \ldots, m$ take over the role of "strips" and "epochs." We replace $\mathcal{F}^t$ by $\mathcal{V}_t = \{i : T_i = t\}$. Tree $\mathbb{A}^{t+1}$ is obtained from $\mathbb{A}^t$ by sampling and adding vertices in $\mathcal{V}_{t+1}$ (with incoming arrows). In the ancestor sampling step, we choose a parent of a currently singled out node S^{t+1} among nodes in $\mathcal{V}_t$. Moreover, the probability of sampling $\text{pa}(S^{t+1})$ becomes simpler, too. Recall that for discrete time models, the posterior is given by Equation (2.5). Therefore the weights are $W_i = \frac{\mathrm{d}P_{t-1}}{\mathrm{d}R_{t-1}}(X_{\text{pa}(i)}, X_i)\ell_t(X_i)$ for $i \in \mathcal{V}_t$. Assume that the intensity parameters are given by Equation (5.1). In formula (5.4), the likelihood terms cancel out, because the likelihood at time $t+1$ depends only on the location of a particle at $t+1$, so it is not altered by a change of parent. We also have $c_{\text{pa}(i)} = c_{\text{pa}(i')}$ whenever i and i' belong to the same "generation", in view of (5.1). Consequently in a discrete time version of PTGAS,

$$\Pr(\text{pa}(S^{t+1}) = i) = \frac{W_i p_t(X_i; X_{S^{t+1}})}{\sum_{i' \in \mathcal{V}_t} W_{i'} p_t(X_{i'}; X_{S^{t+1}})}, \text{ for } i \in \mathcal{V}_t,$$

where we assume that the transition kernels $P_t(x_t, \mathrm{d}x_{t+1})$ are represented by transition *densities* $p_t(x_t, x_{t+1})$. Summing up, PTGAS in discrete time is as simple as the classical ancestor sampling in Lindsten, Jordan, and Schon (2014), with exactly the same probability of sampling parents.

The pseudocode below describes a general version of PTGAS.

One step of PTGAS (Poisson Tree Gibbs with Ancestor Sampling)

```
Input $(\tilde{X}_{1:M}, \tilde{T}_{1:M})$   { Conditioning path }
$\mathcal{V} := \mathcal{V}_{\text{act}} := \{0\}$; $\mathcal{V}_{\text{end}} := \emptyset$; $\mathcal{E} := \emptyset$; $X_0 := x_0$; $T_0 := t_0$;
$C_0 := \Lambda_0 := \lambda_0$; $W_0 := 1$
$\mathbb{A}^0 := (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{T}, \text{null})$
for $r := 0$ to $q-1$ do
  { Update tree $\mathbb{A}^r$ to $\mathbb{A}^{r+1}$ }
  $\mathbb{A} := \mathbb{A}^r = (\mathcal{V}^r, \mathcal{E}^r, \mathbf{X}^r, \mathbf{T}^r, S^r)$
  $\mathcal{V}_{\text{act}} := \mathcal{V}^r_{\text{end}} \cap \mathcal{F}^r$
  { Nodes belonging to the strip $\mathcal{F}^r$ become active }
  if $M^{r+1} > M^r + 1$ then
    { Include a segment of the conditioning path in the tree
    $\mathbb{A}^{r+1}$ }
    for $k := M^r + 1$ to $M^{r+1} - 1$ do
      $\mathcal{V}_{\text{act}} := \mathcal{V}_{\text{act}} \cup \{k\}$; $\mathcal{V} := \mathcal{V} \cup \{k\}$;
      $\mathcal{E} := \mathcal{E} \cup \{k-1 \to k\}$
      $(X_k, T_k) := (\tilde{X}_k, \tilde{T}_k)$
    end for
    ......... { Execute "Main loop" as in Algorithm PTPF,
    with $t_{\max}$ replaced by ${t_{\text{syn}}}^{r+1}-$ }
  end if
  $S := M^{r+1}$; $\mathcal{V}_{\text{end}} := \mathcal{V}_{\text{end}} \cup \{S\}$; $\mathcal{V} := \mathcal{V} \cup \{S\}$
  $(X_S, T_S) := (\tilde{X}_{M^{r+1}}, \tilde{T}_{M^{r+1}})$
  { Sampling "new" parent of $S$: }
  if $\tilde{T}_{M^r} < {t_{\text{syn}}}^{r+1}$ then
    { This condition ensures that the "old parent" belongs to
    $\mathcal{F}^r$: }
```

```
            Sample $\mathrm{pa}(S) = i$ from $\mathcal{F}^r$ according to the probability
            distribution
                $\Pr(i) \propto \frac{W_i}{C_{\mathrm{pa}(i)}} k(X_i, T_i; X_S, T_S)\ell(X_S; T_i, T_S)$,
            $\mathcal{E} := \mathcal{E} \cup \{i \to S\}$
            $W_S := \frac{dK}{dR}(X_i, T_i, X_S, T_S)\ell(X_S; T_i, T_S)$   { Recompute
            weight $W_S$ }
        end if
        $\mathbb{A}^{r+1} := \mathbb{A}$
    end for
    { Select new $S'$: }
    Select $S' \in \mathcal{V}_{\mathrm{end}}$ according to the probability distribution
    $\Pr(S' = s') \propto W_{s'}/C_{\mathrm{pa}'(s')}$
    Output $(X'_{1:M'}, T'_{1:M'}) := (X_{\mathrm{an}'(S')}, T_{\mathrm{an}'(S')})$
```

The following theorem shows that the ancestor sampling step is correct.

Theorem 5.5. Markov chain generated by algorithm PTGAS has the equilibrium distribution equal to the target π.

Proof. To begin with, consider the extended target distribution ϕ given by formula (3.3) together with Equation (3.2). For the tree $\mathbb{A} = (\mathcal{V}, \mathcal{E}, \mathbf{x}, \mathbf{t}, s)$ we have

$$\phi(\mathbb{A}) = \phi(\mathcal{V}, \mathcal{E}, \mathrm{d}\mathbf{x}, \mathrm{d}\mathbf{t}, s) = \prod_{i \in \mathcal{V} \setminus \mathcal{V}_{\mathrm{end}}} \exp\left[-\lambda_i w_i\right] (\lambda_i w_i)^{|\mathrm{ch}(i)|}$$

$$\prod_{j \in \mathrm{ch}(i)} R\left(x_i, t_i; \mathrm{d}x_j, \mathrm{d}t_j\right) \frac{w_s/c_{\mathrm{pa}(s)}}{z}.$$

Now consider a modified tree $\mathbb{A}'$ in which the arrow leading to s is altered (and everything else, including s itself, remains unchanged). If $i = \mathrm{pa}(s)$ and $i' \in \mathcal{V}$ then $\mathbb{A}' = (\mathcal{V}, \mathcal{E}', \mathbf{x}, \mathbf{t}, s)$ with $(\mathcal{E} \setminus \{i \to s\}) \cup \{i' \to s\}$. The weights w_is are obviously the same in $\mathbb{A}$ and $\mathbb{A}'$, for $i \neq s$. However, the weight w_s is altered (we have to bear in mind that every weight w_i depends not only on i but also on $\mathrm{pa}(i)$, although it is not explicit in our notation). To avoid misunderstanding in the formula below we write $w_{\mathrm{pa}(s) \to s}$ instead of w_s. Let us additionally *assume* that all λ_is are the same in $\mathbb{A}$ and in $\mathbb{A}'$. Then we obtain the formula:

$$\frac{\phi(\mathbb{A}')}{\phi(\mathbb{A})} = \frac{R(x_{i'}, t_{i'}, \mathrm{d}x_s, \mathrm{d}t_s)}{R(x_i, t_i, \mathrm{d}x_s, \mathrm{d}t_s)} \cdot \frac{\lambda_{i'} w_{i'}}{\lambda_i w_i} \cdot \frac{w_{i' \to s}/c_{i'}}{w_{i \to s}/c_i}.$$

Indeed, it is enough to compare the formulas for $\phi(\mathbb{A}')$ with $\phi(\mathbb{A})$ and identify the items which are different in these formulas:

- Quotient $R(x_{i'}, t_{i'}; \mathrm{d}x_s, \mathrm{d}t_s) / R(x_i, t_i; \mathrm{d}x_s, \mathrm{d}t_s)$ appears in the ratio r $\phi(\mathbb{A}')/\phi(\mathbb{A})$ for the obvious reason.
- Expression $\lambda_{i'} w_{i'} / \lambda_i w_i$ appears in the ratio, because $|\mathrm{ch}(i')|$ is increased by one and $|\mathrm{ch}(i)|$ is decreased by one.
- Expression $w_s/c_{\mathrm{pa}(s)}$ is altered.

Taking into account the identities $R(x_i, t_i, \mathrm{d}x_s, \mathrm{d}t_s) w_{i \to s} = K(x_i, t_i, \mathrm{d}x_s, \mathrm{d}t_s)\ell(x_s; t_i, t_s)$ and $c_i/\lambda_i = c_{\mathrm{pa}(i)}$ we obtain

$$\frac{\phi(\mathbb{A}')}{\phi(\mathbb{A})} = \frac{K(x_{i'}, t_{i'}, \mathrm{d}x_s, \mathrm{d}t_s)}{K(x_i, t_i, \mathrm{d}x_s, \mathrm{d}t_s)} \cdot \frac{\ell(x_s; t_i, t_s)}{\ell(x_s; t_i, t_s)} \cdot \frac{w_{i'}/c_{\mathrm{pa}(i')}}{w_i/c_{\mathrm{pa}(i)}} \qquad (5.6)$$

From (5.6) it follows that ϕ is preserved if $\mathrm{pa}(s)$ is sampled with probability $\propto k(x_i, t_i, x_s, t_s)\ell(x_s; t_i, t_s) w_i / c_{\mathrm{pa}(i)}$.

The rest of the proof proceeds by induction with respect to the epochs. Consider updating $\mathbb{A}^r$ to $\mathbb{A}^{r+1}$. Assume that $\mathbb{A}^r$ has the probability distribution ϕ^r, that is the extended target defined as in formulas (3.3) and (3.2) but with the time horizon restricted to $[t_{\min}, t_{\mathrm{syn}}{}^r[$ instead of $[t_{\min}, t_{\max}]$. If we updated $\mathbb{A}^r$ to $\mathbb{A}^{r+1}$ without parent sampling, that is if we put $\mathrm{pa}(S^{r+1}) = M^{r+1} - 1$ then the resulting $\mathbb{A}^{r+1}$ would be distributed according to ϕ^{r+1}. Sampling of a new parent of S^{r+1} with probabilities given by (5.4) preserves ϕ^{r+1}, *provided that λ_is are unchanged.* We have established this fact in the fist part of the proof. The argument used for the whole tree $\mathbb{A}$ applies also to $\mathbb{A}^{r+1}$.

The crucial element of our algorithm PTGAS is the restriction of ancestor sampling to the current strip $\mathcal{F}^r$. We sample $\mathrm{pa}(S^{r+1})$ from the nodes belonging to $\mathcal{F}^r$ under the assumption that the "old parent" $M^{r+1} - 1$ belongs to $\mathcal{F}^r$, too (this is ensured by the condition $\tilde{T}_{M^r} < t_{\mathrm{syn}}{}^{r+1}$; if this condition is not fulfilled then we do nothing). Our rules for choosing λ_is described earlier in this section are such that for $i \in \mathcal{F}^r$, λ_i depends only on the behavior of the trajectory ξ_i in the previous strip, $\mathcal{F}^{r-1}$. Our ancestor sampling does not alter these trajectories, so the λ_is are unchanged. This observation completes the proof. □

6. Simulation Results

In this section we examine PTGS's properties in a series of numerical evaluations. We begin with discussion on choice of synchronization strips. To assess the overall correctness of PTGS scheme in discrete time setting we compare our implementation with particle Gibbs with ancestor sampling sampler—PGAS—from Lindsten, Jordan, and Schon (2014). In subsequent sections we investigate properties of continuous time PTGS and implementation details. Computer code used in this section can be found in the supplementary material. Data from experiments is provided in designated GitHub repository [1].

6.1. Choice of Strip Size

Adjustment of strip size is of crucial importance for PTGAS, among others it affects:

1. Ancestor sampling performance.
2. Filtering effectiveness.
3. Numerical properties of the algorithm.

Number of children of a given particle depends only on its relative fitness in the most recent strip. Hence, if a particle spans multiple strips, only part of its trajectory will be compared with observations during filtering. To increase efficiency of filtering, we would ideally want all of the particles which have been born in $[t_r, r_{r+1}[$ to die in $[t_{r+1}, t_{r+1}[$ – this ensures that every part of trajectory will be confronted with observed process.

Recall that probability of choosing given trajectory for the next iteration is proportional to $W_s/C_{\mathrm{pa}(s)}$. For every particle $\Lambda = \frac{1}{W}\Lambda'$ where W is particle's weight and Λ' its Poisson parameter. Each Λ' which is not equal to 1 corresponds to

[1] *https://github.com/tc360950/PoissonTreeData.git*

ancestor particle which is born and dies in different strips (let us denote the set of such particles by $\mathcal{F}_\circ$), thus $W_s/C_{\mathrm{pa}(s)} = \left(W_s \prod_{i\in\mathrm{an}(s)} W_i\right) \cdot \prod_{i\in\mathrm{an}(s)\cap\mathcal{F}_\circ} \frac{1}{\Lambda_i'}$. Left part of the expression is equal to the weight of whole trajectory and does not depend on synchronization strips. Particle s has survived till $t_{\max}$ so most of Λ_i's in the preceding expression will be strictly greater than one with high probability, as a result variability of $\mathcal{F}_\circ$ directly affects weight variance.

The preceding discussion indicates that a viable strategy would be choosing strip size which maximizes expected number of particles which are born and die in neighbouring strips. This problem is intractable in general—especially when jump size is not independent from ancestor value—but can be approximately solved for homogeneous Poisson processes. For proposal density with mean jump equal to λ we use strip size which is optimal for Poisson point process with intensity $\frac{1}{\lambda}$—details of this scheme may be found in the appendix.

Only particles which span at least two different strips are susceptible to ancestor sampling. Therefore, one may find setting strip size to some small value to be a tempting resolution. However, if $\mathrm{pa}(s) \in \mathcal{F}^r$ then a new parent for s can be chosen only from $\mathcal{F}^r$—hence if a strip size becomes too small AS efficiency decreases, example of such phenomenon can be found in the appendix. Strategy from the preceding paragraph is a step in the right direction—our results indicate that it does enhance capabilities of ancestor sampling. Since implementation of this heuristic is fairly straightforward, this is the approach we have chosen for our simulations in the subsequent sections—its assessment may be found in the appendix.

6.2. Discrete Time Models

Here we study differences between PTGAS and classical particle Gibbs sampler with ancestor sampling (PGAS) in the setting of discrete time processes. To this end we examine two state-space models – stochastic volatility model and simple nonlinear model considered in Andrieu, Doucet, and Holenstein (2010). Both algorithms have been run on the same sets of hidden and observed trajectories.

6.2.1. A Nonlinear State–Space Model

We start our study with a simple nonlinear state–space model given by equations:

$$X_k = \frac{X_{k-1}}{2} + 25\frac{X_{k-1}}{1+X_{k-1}^2} + 8\cos(1.2k) + V_k$$

$$Y_k = \frac{X_k^2}{20} + W_k,$$

where $X_1 \sim \mathcal{N}(0,5)$, $V_n \sim \mathcal{N}(0,\sigma_V^2)$, $W_n \sim \mathcal{N}(0,\sigma_W^2)$.

Priors for both parameters have been set to $\mathcal{IG}(0.01, 0.01)$ with $(N, \lambda_0, T) = (300, 300, 300)$. Both algorithms have been run for 10,000 iterations with 3000 burn-in and starting parameters equal to: $\sigma_V^2 = 10$, $\sigma_W^2 = 1$.

6.2.2. Stochastic Volatility Model With Leverage

Next we consider model governed by equations

$$X_{k+1} = \mu(1-\phi) + \phi \cdot X_k + \sigma \cdot \mathcal{N}(0,1)$$

$$Y_k = e^{(-0.5X_k)} \cdot \mathcal{N}(0,1)$$

Priors and sampling method were taken from Kim, Shepherd, and Chib (1998). Observations have been taken from Standard and Poor's (SP) 500 data for the interval 2017-03-10 – 2018-05-17.

Both algorithms have been run for 10 000 iterations with 5000 burn-in and $(N, \lambda_0) = (1000, 1000)$. Trajectory length has been set to 300. Comparison of estimated densities of static parameters may be found in Figure 2, for additional reference we have plotted estimated auto-correlation functions for σ parameter.

For both models we have not found any significant differences between classical particle Gibbs Sampler and the Poisson Tree scheme. The speed and quality of convergence seems to be comparable. Both ancestor sampling schemes seem to provide

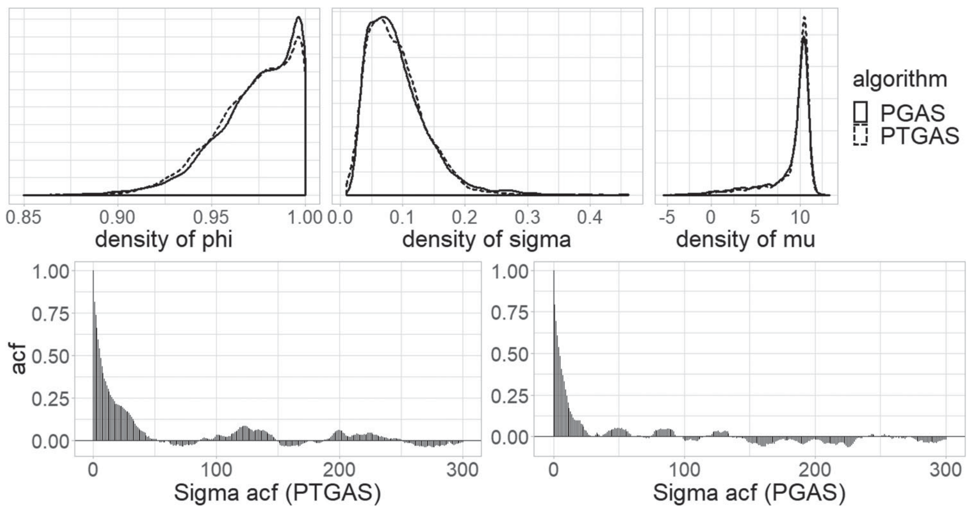

Figure 2. Upper row—comparison of estimated densities of static parameters in stochastic volatility model with leverage. Bottom row—acf plots for σ parameter for PTGAS (left) and PGAS (right).

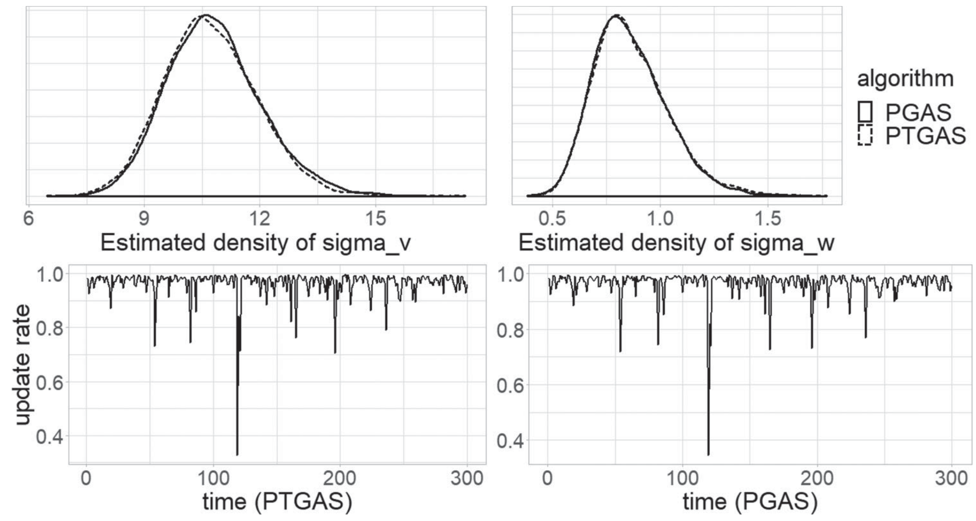

Figure 3. Upper row—comparison of estimated densities of static parameters in nonlinear change point model. Bottom row—comparison of update rates for X_t versus $t \in \{1, \dots, 300\}$ for last 1000 iterations of PGAS (right) and PTGAS (left) in nonlinear change point mode.

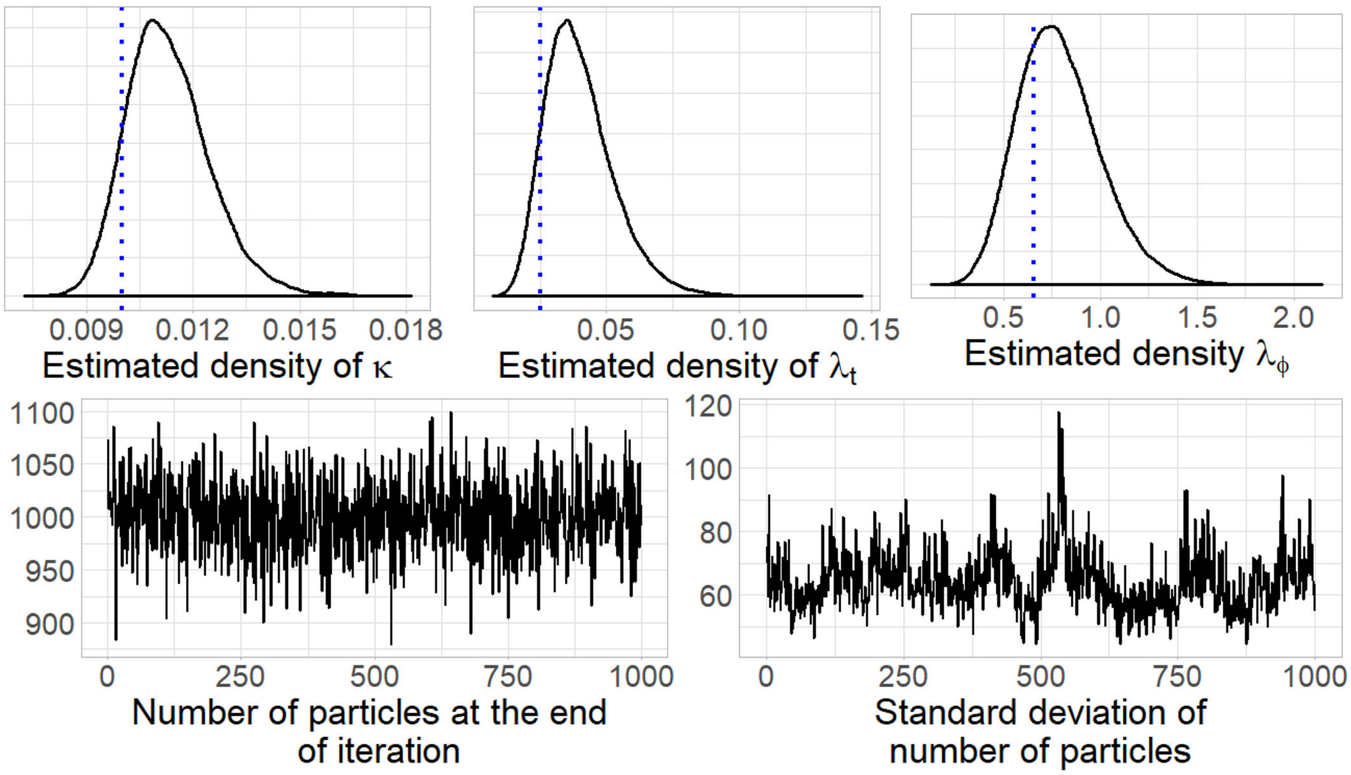

Figure 4. Upper row – densities of static parameters for Shot-noise-Cox-process model estimated from 100,000 iterations of PTGAS. Vertical lines indicate the true values of the parameters. Bottom row – left figure displays number of particles which have survived to $t_{\max}$ for 1000 subsequent iterations of PTGAS, right figure depicts standard deviation of number of particles at the end of each synchronization strip for 1000 subsequent iterations of PTGAS. All figures correspond to the same run of PTGAS.

equivalent improvement in mixing (depicted in bottom row of Figure 3). This is not surprising since, as has been noted in the proof of Theorem 5.2, PTGS can be seen as classical particle Gibbs sampler (with multinomial resampling) but with random change of population size before every propagation step. Bottom row of Figure 4 shows that random fluctuations in population size are small hence properties of both algorithms are similar and differences come down to easier parallelization of PTGS. The obvious disadvantage of PTGS is a little bit more involved implementation—primarily it is harder to vectorize it efficiently.

6.3. Continuous Time Models

Here we illustrate properties of PTGS on two piece-wise deterministic processes which have been considered among others in Finke, Johansen, and Span (2014). In both examples b function—which controls the size of population—has been defined by

$$b(x) = \max(0, x)$$

and dynamic strategy of choosing synchronization strip size from Section 6.1 has been employed. Details of this scheme and rationale for choosing this form of b are given in the appendix.

6.3.1. *Elementary Change-Point Model*

For a first example we use a simple piece-wise deterministic process in which skeleton of Ξ is assumed to be $AR(1)$ process (with coefficient ρ) with unknown variance of noise - σ. Jump times are sampled from gamma distribution with unknown parameters α, β and mean $\frac{\alpha}{\beta}$. Observations are assumed to be taken at ends of fixed time intervals – in our example $[n, n+1[$ for $n \in \mathbb{N}$ – and formed by adding mean zero Gaussian noise with variance σ_y to Ξ.

Data used for simulations has been sampled with static parameters equal to $(\alpha, \beta, \rho, \sigma, \sigma_y) = (2, 0.5, 0.9, 1, 2)$. Parameters have been assigned mean zero Gaussian prior truncated to $\mathbb{R}_+$ with covariance matrix
$diag(10, 10, 1, 10, 10)$.

Figure 5 shows densities of static parameters estimated from 100,000 iterations of PTGAS with 20,000 burn-in and mean population size equal to 500. Posterior distribution has been approximated with 15,000 runs of Gaussian random walk Metropolis–Hastings (one run for each pair (ρ, σ), (α, β) and one for σ_y) with unit kernel variance.

Figure 6 depicts update frequency (calculated every 0.5 time-step) for Poisson Tree Gibbs sampler with (left) and without ancestor sampling (right). It is evident that the ancestor sampling step enhances mixing, though the results look worse than in the discrete time setting. We have found that without ancestor sampling the algorithm tends to get stuck at short trajectories (in the sense of number of jumps) for a few iterations. This phenomenon is presumably caused by the fact that short trajectories have potentially smaller accumulated ancestors' weights and thus their final weights (i.e., weights used to choose a new fixed trajectory) are order of magnitude bigger than that of longer trajectories.

6.4. *Shot-Noise-Cox-Process Model*

The model assumes observations Υ to be an inhomogenous Poisson process with intensity modeled by latent trajectory $(\xi(t), t \in [0, t_{\max}])$ sampled from Ξ.

Ξ is assumed to be a piece-wise deterministic process, governed by kernel with density:

$$k(x_{n-1}, t_{n-1}; x_n, t_n) = \lambda_t \cdot e^{-\lambda_t \cdot (t_n - t_{n-1})} \cdot \mathbb{1}_{(t_n - t_{n-1} > 0)} \cdot \lambda_\phi \cdot e^{-\lambda_\phi \cdot (x_n - \bar{x}_{n-1})} \cdot \mathbb{1}_{(x_n - \bar{x}_{n-1} > 0)}$$

where $\bar{x}_m = x_m \cdot e^{-\kappa \cdot (t_m - t_{m-1})}$

For our simulations we have chosen

$$\kappa = 0.01\,, \lambda_t = \frac{1}{40}\,, \lambda_\phi = \frac{2}{3}\,, t_{\max} = 1000$$

and priors (truncated to $\mathbb{R}_+$)

$$(\kappa, \lambda_t, \lambda_\phi) \sim \mathcal{N}(0, 10) \otimes Exp(1.5) \otimes \mathcal{N}(0, 10).$$

Sampling from the posterior distribution has been approximated by two-step procedure – first we sample λ_t from its posterior then we use 2000 iterations of Gaussian random-walk Metropolis-Hastings algorithm to approximately sample from:

$$(\kappa, \lambda_\phi) \sim \pi(\kappa, \lambda_\phi | \xi, \Upsilon).$$

(where ξ is a trajectory sampled in antecedent run of PTGAS)

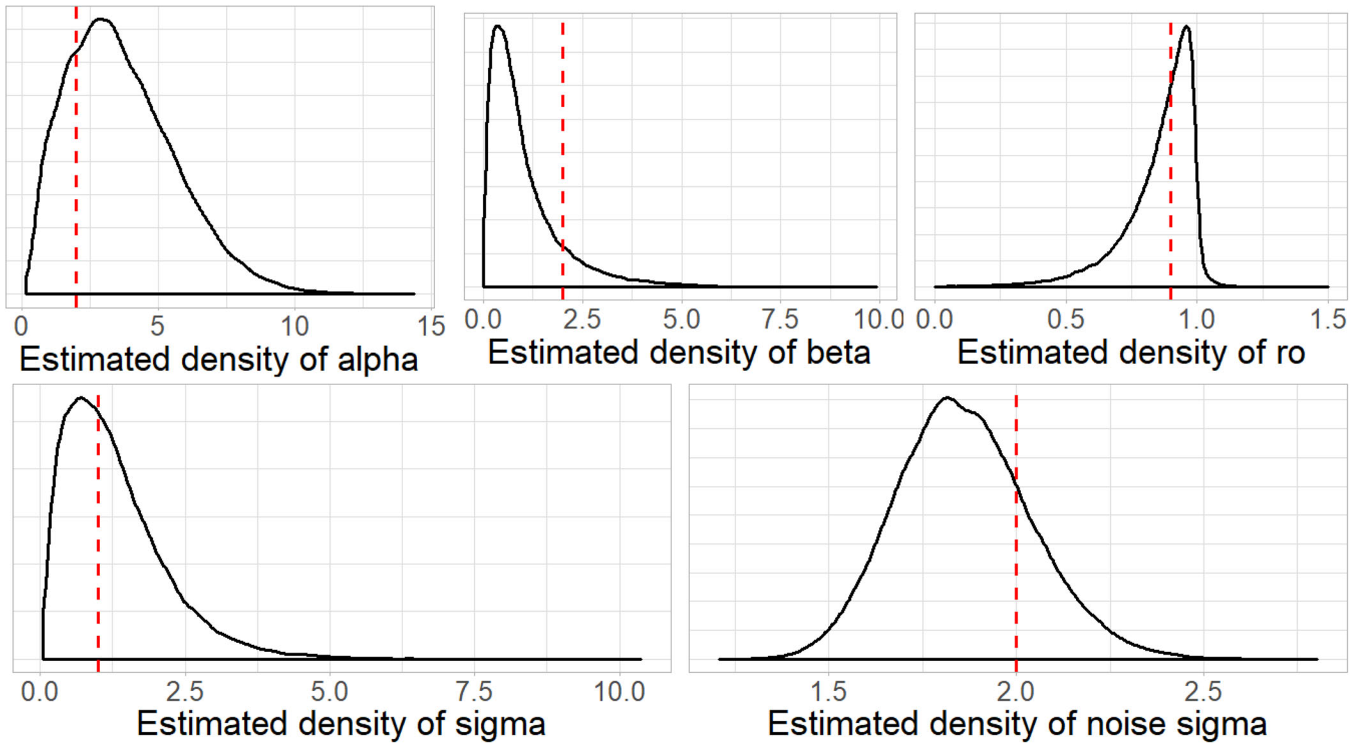

Figure 5. Densities of static parameters in elementary change-point model estimated from 100,000 iterations of PTGAS. Vertical dashed lines indicate the true values of the parameters.

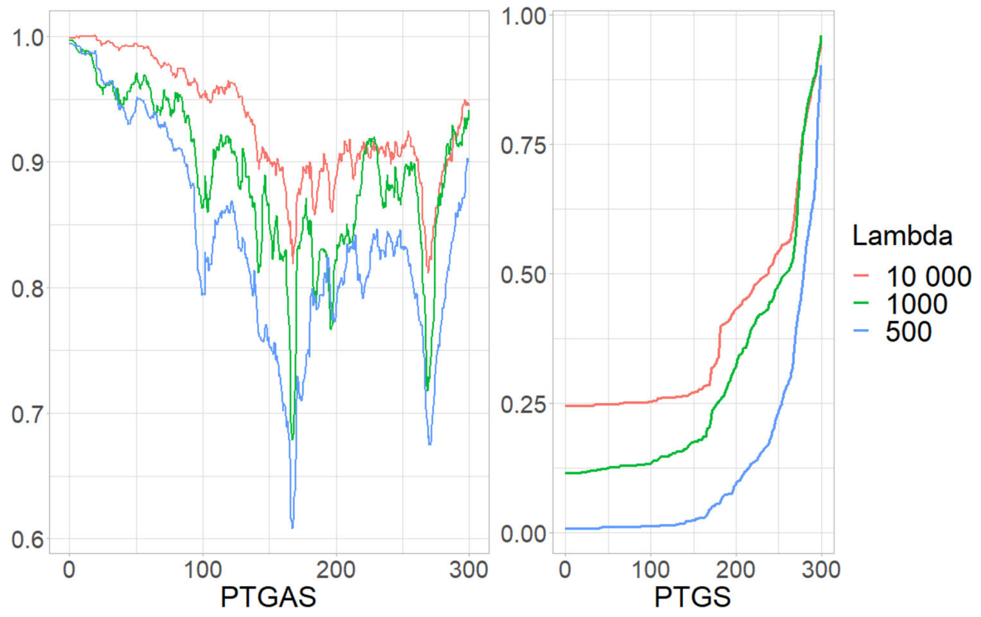

Figure 6. Comparison of update rates for Ξ at $t \in \{1, \ldots, 300\}$ for 1000 iterations of PTGS (right) and PTGAS (left) in elementary change-point model for three choices of $\lambda_0 - \{10{,}000, 1000, 500\}$.

Figure 4 display the result of 100,000 iterations of PTGAS with 20,000 burn-in and $\lambda_0 = 1000$. Initial parameters have been sampled from the prior density but value of κ has been additionally divided by 10 to avoid starting in a region where values at the end of life of particle become too small to handle numerically. Although PTGAS seems to converge quite fast with respect to number of iterations, pre-convergence sampling requires unproportionally long computation time. This is due to high dependence of iteration's computational cost on average jump length.

Bottom row of Figure 4 displays data on the number of surviving particles for 1000 iterations of Shot-noise-Cox-process model. Left figure depicts number of particles which have survived given iteration while right plot shows standard deviations of number of particles at the end of each synchronization strip. It is clear that the employed strategy is effective at controlling size of population.

6.5. Parallelization

A nonlinear state–space model from Section 6.2 has been used for comparison between classical Particle Gibbs sampler and Poisson Tree scheme. The two algorithms have been implemented in C++ and employed utilities from standard library. Time measurements have been obtained from runs on 64 cores, 2.5GHz per each. For both approaches computations were performed by fixed number of threads working in parallel (plus one additional thread in case of PTGS whose sole purpose of existence was to gather sums of weights from each thread, combine them and redistribute the result amongst workers).

Distribution of work between finite number of worker threads is straightforward in the setting of classical Particle Filter but gets more troublesome with introduction of PTGS—one can still divide first population uniformly between threads letting each thread take care of its own batch but we have found that after average batch to thread ratio decreases beyond a certain point (we have empirically observed this threshold to be around 500 particles per thread) some batches perish completely after few dozens of propagations. To counter this erratic behaviour every 50 steps we synchronize all threads and let one chosen thread redistribute surviving particles uniformly. Despite the additional overhead introduced by this operation we have found this modified scheme to be more effective. The detailed comparison of our parallel implementation with approaches presented in Paige et al. (2014); Murray, Lee, and Jacob (2016) is discussed in Supplementary Material.

Table 1. Execution time in milliseconds for parallel implementations of PTGS and PGS (Particle Gibbs Sampler) algorithms for elementary change-point model.

Algorithm / Population size	10,000	50,000	100,000	500,000	1,000,000
PTGS	3211	6182	9994	37668	76737
PGS	3908	21304	42243	242265	452760

Table 1 shows the comparison of mean times of execution (in milliseconds) for 10 iterations, trajectory length equal to 400 and number of threads working in parallel equal to 50. For big population to thread ratio PTGS provides much faster time of execution. However, we have found that its time of execution exhibits larger variance between different runs. Both algorithms scale quite well with increasing number of threads but ordinary Particle Gibbs sampler seems to be more resistant to decrease in number of threads—for instance after reduction to only 10 worker threads execution time for one million is around 50s for PGS and 20s for PTGS. When number of particles is smaller than 3000 parallel Poisson Tree implementation ceases to be practicable. This shows that PTGS should be preferred to PGS for models where ancestor sampling is not feasible—for instance discrete models—and thus large number of particles is necessary.

Implementation of the algorithm for continuous time is analogous (with thread synchronization after every synchronization strip) but quantity of work assigned to every thread in a given synchronization strip has a much higher variance. What is more, expected cost of kernel's iteration changes with every change of strip size. Additional bookkeeping is necessary—for every synchronization strip $|\mathcal{G}^r|$, $b(\lambda_0 - |\mathcal{G}^r|)$ must be recorded. Particles which have not propagated are kept on stacks—one for every synchronisation strip. Particles after propagation are stored in vectors (analogously one for every synchronisation strip). This enforces much greater movement of data from one place to another (which is essentially constant in discrete time setting). Performance of implementation increases with size of synchronization strips, since synchronization of threads happens less often. However, strip size adjustment greatly affects filtering and ancestor sampling effectiveness and we advise to prioritize algorithm's performance over computational considerations.

Supplementary Material

Supplementary Material: Document containing all postponed proofs and detailed discussions omitted in main text. (pdf)

Codes.zip: Archive containing all codes used to obtain simulation results.

Acknowledgments

The paper is partially supported by Polish National Science Center grant: NCN UMO-2018/31/B/ST1/00253.

ORCID

Błażej Miasojedow http://orcid.org/0000-0002-3691-9372
Wojciech Niemiro http://orcid.org/0000-0002-7076-8838

References

Andrieu, C., Doucet, A., and Holenstein, R. (2010), "Particle Markov chain Monte Carlo Methods," *Journal of the Royal Statistical Society*, Series B, 72, 269–342. [1,5,6,10]

Arnaudon, M., and del Moral, P. (2020), "A Duality Formula and a Particle Gibbs Sampler for Continuous Time Feynman-Kac Measures on Path Spaces," Available at *http://arxiv.org/abs/1805.05044v4;http://arxiv.org/pdf/1805.05044v4*. [1]

Bouchard-Ct, A., Sankararaman, S., and Jordan, M. I. (2012), "Phylogenetic Inference Via Sequential Monte Carlo," *Systematic Biology*, 61, 579–593. [1]

Crou, F., Moral, P. D., Furon, T., and Guyader, A. (2011), "Sequential Monte Carlo for Rare Event Estimation," *Statistics and Computing*, 22, 795–808. [1]

Davis, M. H. A. (1984), "Piecewise-deterministic Markov Processes: A General Class of Nondiffusion Stochastic Models," *Journal of Royal Statistical. Society*, Series B, 46, 353–388. [2]

Del Moral, P., and Penev, S. (2017), *Stochastic Processes: From Applications to Theory*. Boca Raton, FL: CRC Press. Chapman & Hall/CRC Texts in Statistical Science Series. [1]

Doucet, A., Freitas, N., and Gordon, N. editors. (2001), *Sequential Monte Carlo Methods in Practice*. New York: Springer. [1]

Fearnhead, P., Papaspiliopoulos, O., and Roberts, G. O. (2008), "Particle Filters for Partially Observed Diffusions," *Journal of the Royal Statistical Society*, Series B, 70, 755–777. [1]

Finke, A., Johansen, A. M., and Span, D. (2014), "Static-parameter Estimation in Piecewise Deterministic Processes Using Particle Gibbs Samplers," *Annals of the Institute of Statistical Mathematics*, 66, 577–609. [1,2,11]

Gloaguen, P., Étienne, M.-P., and Le Corff, S. (2018), "Online Sequential Monte Carlo Smoother for Partially Observed Diffusion Processes," *EURASIP Journal on Advances in Signal Processing*, 1687–6180. [1]

Golightly, A., and Wilkinson, D. J. (2011), "Bayesian Parameter Inference for Stochastic Biochemical Network Models Using Particle Markov Chain Monte Carlo," *Interface Focus*, 1, 807–820, [1]

Gordon, N., Salmond, D., and Smith, A. (1993), "Novel Approach to Nonlinear/Non-gaussian Bayesian State Estimation," *IEE Proceedings F Radar and Signal Processing*, 140, 107–113. [1]

Kim, S., Shepherd, N., and Chib, S. (1998), "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models," *Review of Economic Studies*, 65, 361–393. [10]

Lindsten, F., Douc, R., and Moulines, E. (2015), "Uniform Ergodicity of the Particle Gibbs Sampler," *Scand. J. Stat.*, 42, 775–797. [1,6]

Lindsten, F., Jordan, M. I., and Schon, T. B. (2014), "Particle Gibbs with Ancestor Sampling," *J. Mach. Learn. Res.*, 15, 2145–2184. [1,8,9]

Moral, P. D., Doucet, A., and Jasra, A. (2006), "Sequential Monte Carlo Samplers," *J. R. Statist. Soc*, Series B, 68, 411–436. [1]

Murray, L. M., Lee, A., and Jacob, P. E. (2016), Parallel resampling in the particle filter. *J. Comput. Graph. Stat.*, 25, 789–805. [2,13]

Naesseth, C. A., Lindsten, F., and Schn, T. B. (2014), "Sequential Monte Carlo for Graphical Models," In *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems*, eds., Z Ghahramani, M Welling and C Cortes, vol. 2, Cambridge, MA: MIT Press. [1]

Paige, B., Wood, F., Doucet, A., and Teh, Y. W. (2014), "Asynchronous Anytime Sequential Monte Carlo," in *Advances in Neural Information Processing Systems*, eds., Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, vol. 27, pp. 3410–3418. Cambridge, MA: Curran Associates, Inc. [1,2,13]

Pitt, M. K., and Shephard, N. (1999), "Filtering Via Simulation: Auxiliary Particle Filters," *J. Amer. Stat. Assoc.*, 94, 590–599. ISSN 01621459. [4]

Rousset, M., "On the Control of an Interacting Particle Estimation of Schrdinger Ground States," *SIAM J. Math. Anal.*, 38, 824–844. [1]

Rudnicki, R., and Tyran-Kamińska, M. *Piecewise Deterministic Processes in Biological Models*. Springer International Publishing, 2017. [2]

Schafer, C., and Chopin, N. (2011), "Sequential Monte Carlo on Large Binary Sampling Spaces," *Statistics and Computing*, 23, 163–184. [1]

Whiteley, N., Johansen, A. M., and Godsill, S. (2011), "Monte Carlo Filtering of Piecewise Deterministic Processes," *J. Comput. Graph. Stat.*, 20, 119–139. [2]