Journal of Machine Learning Research 26 (2025) 1-51 Submitted 5/25; Revised 12/25; Published 12/25

Backward Filtering Forward Guiding

F. H. van der Meulen F.H.VAN.DER.MEULEN@VU.NL
Department of Mathematics
Vrije Universiteit Amsterdam
De Boelelaan 1111, 1081HV Amsterdam, The Netherlands

S. Sommer SOMMER@DI.KU.DK
Department of Computer Science
University of Copenhagen
Øster Voldgade 3, 1350 København, Denmark

Editor: Anthony Lee

Abstract

We study smoothing for discrete- and continuous-time stochastic processes on directed acyclic graphs (DAGs) when observations are available only at the leaf nodes, a problem common in phylogenetics, epidemiology, and signal processing. We introduce a unified framework built around guiding (also called twisting): a change-of-measure defined by guiding functions that transforms the original process into a guided process whose distribution approximates the smoothing distribution. The Radon-Nikodym derivative quantifies the discrepancy between the two measures. On directed trees, guiding functions are obtained via a backward-filtering step. By isolating backward filtering and forward guiding as elementary operations, we show that the approach extends beyond traditional state-space models and particle filters. We also generalize guiding to edges governed by continuous-time dynamics, using the change-of-measure construction described by Palmowski and Rolski (2002).The versatility of the framework is illustrated with two numerical examples: (i) a diffusion model for shape deformation on a tree, and (ii) inference in a factorial hidden Markov model.

Keywords: Backward information filter, Bayesian network, Branching diffusion process, directed acyclic graph, conditioned Markov process, Doob's h-transform, exponential change of measure, guided process

1. Introduction

Probabilistic inference in structured dynamical systems is a fundamental task in machine learning, with applications ranging from time series modeling to phylogenetics. In many problems, such as evolutionary biology, signal processing, or latent variable modeling, data is observed only at the *leaves* of a structure, while the latent dynamics evolve along a branching (tree-like) structure. The goal is to infer the latent process given these noisy or partial observations; a problem known as *smoothing*.

In this work, we start from the smoothing problem on a directed tree with root vertex r and leaf vertices represented by open circles (see Figure 1). Each branch either evolves according to a continuous-time stochastic process—such as a diffusion process or continuous-time Markov chain—or follows a transition given by a Markov kernel. At branching vertices,

such as 0 and 3, the process splits and evolves conditionally independently, given its value at the vertex. Observations are made at the leaf vertices via emission maps that transform the latent state at the parent into an observed value; for example, vertex 5 might observe a noisy version of vertex 4. The root state x_r is assumed known, and the branch from r to 0 encodes a prior.

This setting generalizes classical *state-space models*, where transitions occur in discrete time, and has received attention in *phylogenetics*, where continuous-time models like Brownian motion or Ornstein-Uhlenbeck processes are tractable and well studied (e.g., Hassler et al. (2023), Ronquist (2004), Zhang et al. (2021)). However, for many models of interest, closed-form transition densities or analytical smoothing solutions are not available.

We consider a general framework to address this challenge based on three key components:

1. *Backward information filtering*, mapping leaf observations into a family of functions g_u on the tree, typically using a simplification of the stochastic process evolving on the tree.

2. *Constructing a guided process* on the tree with distribution obtained by an *exponential change of measure* to the law of the unconditional (forward) process, using the family of functions g_u.

3. *Stochastic simulation*, generating trajectories from the guided process and computing the *weight* of each trajectory.

Step (3) involves simulating a *guided process*—a forward process modified by the exponential change of measure to encourage paths consistent with the observations. This overall strategy, which we call *Backward Filtering Forward Guiding (BFFG)*, yields weighted samples from the smoothing distribution and is compatible with MCMC and particle-based inference methods such as the guided particle filter (Chopin and Papaspiliopoulos, 2020, Chapter 10). This approach allows smoothing in complex continuous-time models on tree structures, even when transition densities are intractable or unknown, significantly broadening the class of models amenable to Bayesian inference.

1.1 Contribution

The contribution of this paper is twofold.

1. **Unified framework.** Our paper introduces a unified framework that brings together several techniques previously treated separately. The idea of *guiding* (or twisting) is established: twisted particle filters were introduced in Whiteley and Lee (2014) and extended in Guarniero et al. (2017) and Heng et al. (2020), though these works focused on state-space models and particle-filtering algorithms. We isolate the essential components into two elementary operations—backward filtering (via pullback and fusion) and forward guiding—and illustrate them with multiple examples, including Ju et al. (2021), highlighting the framework's versatility. Abstracting these operations shows that the methodology extends beyond state-space models and particle filtering. While extending the guided process to a directed tree is straightforward, doing so on a non-tree directed acyclic graph (DAG) requires more care as there exists no

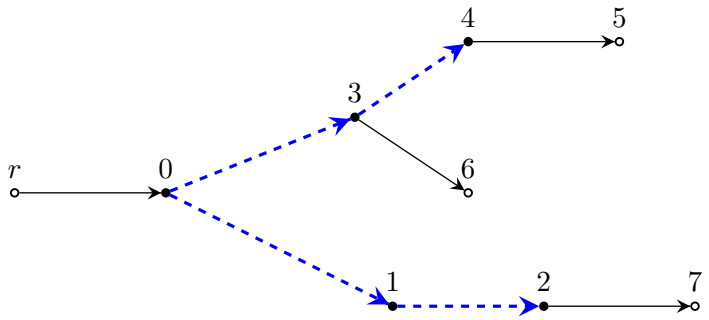

Figure 1: Example of a tree with known root vertex r, with observations at vertices 5, 6 and 7. A continuous time stochastic process evolves on the branches $(0, 3)$, $(3, 4)$, $(0, 1)$ and $(1, 2)$ which are dashed and coloured blue.

conditioned process that follows the same dependency structure as the unconditional process. On a general DAG, we define the guided process so that its dependency structure matches that of the unconditional forward process, building on guiding-function ideas from Lindsten et al. (2018).

2. **Guiding continuous-time transitions** We also generalise guiding to settings where edge transitions arise from evolving a continuous-time process over a fixed interval. This relies on a change-of-measure construction described in Palmowski and Rolski (2002) and earlier sources such as Ethier and Kurtz (1986). To our knowledge, this problem has not been treated in full generality in the literature. As before, we identify the required operations and give detailed examples, including diffusions, continuous-time Markov chains, and the work of Stoltz et al. (2021).

Enforcing the guided process to have the same dependency structure as the unconditional process, enables automatic program transform of the unconditional process to the guided process. This entails that the programme structure of BFFG is the same as that of the forward process, facilitating BFFG to be incorporated into probabilistic programming languages. In a companion paper Schauer et al. (2025) we use category theory to formally prove that BFFG is compositional, a property which we believe to be highly desirable. The literature for inference in graphical models is vast. Nevertheless, as far as we are aware, few works have considered a unified approach to both discrete- and continuous-time processes evolving along the edges of the graph.

1.2 Related Work

In case of a line graph with edges to observation leaves at each vertex (a hidden Markov model) there are two well known cases for computing the smoothing distribution: *(i)* if X is discrete the forward-backward algorithm (Murphy (2012), section 17.4.3), *(ii)* for linear Gaussian systems the Kalman Smoother, also known as Rauch-Tung-Striebel smoother (see for instance chapter 12 in Särkkä and Svensson (2023) or Murphy (2012), section 18.3.2)

for the marginal distributions or its sampling version, where samples from the smoothing distribution are obtained by the forward filtering, backward sampling algorithm, Cf. Carter and Kohn (1994) and Frühwirth-Schnatter (1994). Pearl (1988a) gave an extension of the forward-backward algorithm from chains to trees by an algorithm known as "belief propagation" or "sum-product message passing", either on trees or poly-trees. This algorithm consists of two message passing phases. In the "collect evidence" phase, messages are sent from leaves to the root; the "distribute evidence" phase ensures updating of marginal probabilities or sampling joint probabilities from the root towards the leaves. The algorithm can be applied to junction trees as well, and furthermore, the approximative loopy belief propagation applies belief propagation to sub-trees of the graph. A review is given in Jordan (2004).

Chou et al. (1994) extended the classical Kalman smoothing algorithm for linear Gaussian systems to dyadic trees by using a fine-to-coarse filtering sweep followed by a course-to fine smoothing sweep. This setting arises as a special case of our framework. Extensions of filtering on Triplet Markov Trees and pairwise Markov trees are dealt with in Bardel and Desbouvries (2012) and Desbouvries et al. (2006) respectively.

Particle filters have been employed in related settings, see Doucet and Lee (2018) for an overview. Briers et al. (2010) consider particle methods for state-space models using (an approximation to) the backward-information filter. Twisted particle samplers were introduced in Whiteley and Lee (2014) for Feynman-Kac models. Guarniero et al. (2017) propose an algorithm to approximate the optimal twist iteratively. In controlled Sequential Monte Carlo (Heng et al. (2020)) particle methods are employed for finding an optimal control policy to approximate the backward-information filter. The approximate backward filtering step in this paper builds on the same idea but extends it to a broader context beyond particle filtering and discrete-time models. Lindsten et al. (2016) introduce a new class of SMC algorithms for probabilistic graphical models using a divide-and-conquer tree decomposition. Paige and Wood (2016) train a neural network to approximate the optimal proposal distribution for SMC. This distribution is exactly the twisted kernel (Doob-transformed kernel) characterised in Whiteley and Lee (2014). Thus their method is a learned, amortised approximation of the optimal Doob h-transform in general graphical models. Lawson et al. (2022) introduce SIXO: a method for learning twist functions via density-ratio estimation to reweight filtering SMC toward the smoothing distribution, resulting in tighter variational bounds to improve upon parameter estimation. Lawson et al. (2023) propose "NAS-X", a method that uses smoothing SMC inside a reweighted wake-sleep framework where twist functions are learnt via density-ratio estimation to give low-bias, low-variance gradients for both discrete and continuous sequential latent variable models. Chopin et al. (2023) develop a computational framework to approximate Doob h-transforms for filtering discretely observed diffusions, using backward Kolmogorov equations and neural networks to construct locally optimal particle filters that reduce degeneracy in highly informative observations. It is a practical implementation of the twisted particle filter / guided proposal idea for continuous-time diffusions. A recent application of learning twist functions to large language models is Zhao et al. (2024).

For variational inference, Ambrogioni et al. (2021) propose an approach which, similar to ours, preserves the Markovian structure of the target by learning local approximations to the conditional dynamics. Lindsten et al. (2018) consider a sequential Monte Carlo

(SMC) algorithm for general probabilistic graphical models which can leverage the output from deterministic inference methods. It shares the idea of using a substitute for optimal twisting functions but does not cover continuous-time transitions over edges. We exploit ideas from this paper in Section 8. In earlier work (Schauer et al. (2017), Mider et al. (2021), Corstanje and van der Meulen (2025), Pieper-Sethmacher et al. (2025)) continuous-time guided processes were introduced on state-space models for partially observed stochastic (partial) differential equations and chemical reaction networks. These are specific instances of continuous-time guided processes.

1.3 Outline

We start with a short recap of Markov kernels in Section 2. The backward information filter and forward guiding are discussed in sections 3 and 4 respectively. We summarise the key algorithms of BFFG in Section 5. Examples that illustrate BFFG are given in sections 6 and 7 for discrete and continuous-time procesesses respectively. The extension to a DAG is given in Section 8. We illustrate our results in two numerical examples in Section 9. The appendix contains some technical results and proofs.

Sections 3, 4, 5 and 8 contain the core ideas of our approach.

2. Markov Kernels

We first recap some elementary definitions on Markov kernels, as these are of key importance in all that follows.

Let $S = (E, \mathfrak{B})$ and $S' = (E', \mathfrak{B}')$ be Borel measurable spaces. A Markov kernel between S and S' is denoted by $P\colon S \twoheadrightarrow S'$, where S is the "source" and S' the "target". That is, $P\colon E \times \mathfrak{B}' \to [0,1]$, where

(i) for fixed $B \in \mathfrak{B}'$ the map $x \mapsto P(x, B)$ is $\mathfrak{B}$-measurable and

(ii) for fixed $x \in E$, the map $B \mapsto P(x, B)$ is a probability measure on S'.

For readers less familiar with measure theory, in case of a discrete-time Markov chain $\{X_n\}$, one can think of the Markov kernel P as defined by $P(x, B) = \mathbb{P}(X_{n+1} \in B \mid X_n = x)$.

Let $\mathbf{B}(S)$ denote the set of bounded measurable functions on S. The linear continuous operator $P\colon \mathbf{B}(S') \to \mathbf{B}(S)$ defined by

$$(Ph)(\cdot) = \int_E h(y) P(\cdot, \mathrm{d}y), \qquad h \in \mathbf{B}(S'). \tag{1}$$

will be referred to as the *pullback* of h under the kernel P. In case P admits a density k with respect to Lebesgue measure, then $(Ph)(x) = \int_E h(y)k(x,y)\,\mathrm{d}y$. In the discrete case, $(Ph)(x) = \sum_{y \in E} h(y)k(x,y)$.

3. Backward Information Filter

We consider a stochastic process on a tree with vertex set $\mathcal{T}$, where the root vertex r is excluded. At each vertex where the process splits, it evolves conditionally independent towards its children vertices. Let $\mathcal{V}$ denote the set of leaf vertices and define $\mathcal{S} = \mathcal{T} \setminus \mathcal{V}$ (the

set of non-leaf vertices) Set $\mathcal{S}_r = \mathcal{S} \cup \{r\}$. Thus, in Figure 1, $\mathcal{V} = \{5, 6, 7\}$, $\mathcal{S} = \{0, 1, 2, 3, 4\}$, $\mathcal{S}_r = \{r, 0, 1, 2, 3, 4\}$. Let $\mathcal{E}$ denote the set of all edges in the tree. t

For $t \in \mathcal{T}$ the state of the process is given by X_t. For $T \subset \mathcal{T}$ let $X_T = \{X_t,\, t \in T\}$. Denote the (unique) parent vertex of vertex t by $\mathrm{pa}(t)$. We assume that the transition to X_t, conditional on $X_{\mathrm{pa}(t)} = x$ is captured by a Markov kernel $P_t(x, \cdot)$ with source $(E_{\mathrm{pa}(t)}, \mathcal{B}_{\mathrm{pa}(t)})$ and target $(E_t, \mathcal{B}_t)$. On a directed tree, each vertex has a unique parent. In this case we can identify a directed edge by the vertex it is pointing to. To reduce notation, we will then write P_t rather than $P_{\mathrm{pa}(t),t}$.

We are interested in the distribution of $X_{\mathcal{S}}$ conditional on the event $\{X_{\mathcal{V}} = x_{\mathcal{V}}\}$. We assume that for each $t \in \mathcal{V}$ there exists a dominating measure λ_t such that

$$\frac{\mathrm{d}P_t(x,\, \mathrm{d}y)}{\lambda_t(\mathrm{d}y)}(x) = k_{\mathrm{pa}(t),t}(x, y). \tag{2}$$

3.1 Discrete Edges

We first assume that the probabilistic evolution of the process X is over discrete edges.

Definition 1 *We call an edge $e = (s, t)$ discrete if it is assumed that on e the probabilistic evolution is governed by the Markov kernel P_t.*

The process X conditioned on the event $\{X_{\mathcal{V}} = x_{\mathcal{V}}\}$ follows the same dependency structure as the unconditional process with transition kernels given by

$$P_t^\star(x, A) := \frac{\int_A P_t(x,\, \mathrm{d}y) h_t(y)}{\int P_t(x,\, \mathrm{d}y) h_t(y)} = \frac{(P_t h_t \mathbf{1}_A)(x)}{(P_t h_t)(x)}, \qquad t \in \mathcal{S}.$$

Here, if $\mathcal{V}_t$ denotes the set of leaves descending from vertex t, $h_t(x)$ is the density of $X_{\mathcal{V}_t}$, conditional on $X_t = x$. In Figure 1 for example, h_3 is the density of (X_5, X_6) conditional on $X_3 = x$. The transform of the kernel P_t to $P_t^\star$ is known as *Doob's h-transform*.

3.1.1 Backward Information Filter

It is well known how the functions $\{h_t,\, t \in \mathcal{S}\}$ can be computed recursively starting from the leaves back to the root. These recursive relations have reappeared in many papers, see for instance Felsenstein (1981), Briers et al. (2010) and, in case of state-space models, Guarniero et al. (2017) and Heng et al. (2020). This recursive computation is known as the *Backward Information Filter (BIF)*. Firstly, for any leaf vertex we define

$$h_{\mathrm{pa}(v),v}(x) := k_{\mathrm{pa}(v),v}(x, x_v) \qquad v \in \mathcal{V}. \tag{3}$$

For other vertices t for which h_t has already been computed, set (recalling that P_t acts as an operator according to Equation (1))

$$h_{\mathrm{pa}(t),t} := P_t h_t, \qquad t \in \mathcal{S}. \tag{4}$$

Note that h_t denotes h at *vertex* t, while $h_{\mathrm{pa}(t),t}$ is a different function, corresponding to the *edge* $(\mathrm{pa}(t), t)$. For a given vertex $t \in \mathcal{S}_r$, once $h_{t,u}$ has been computed for all children

vertices u of t, which set we denote by $\mathrm{ch}(t)$, we get by the Markov property (i.e. the process evolves conditionally independent when it branches)

$$h_t(x) = \prod_{u \in \mathrm{ch}(t)} h_{t,u}(x), \qquad t \in \mathcal{S}_r. \tag{5}$$

This can be interpreted as *fusion*, collecting all *messages* $h_{t,u}$ from children at vertex t. Summarising, the BIF provides a recursive algorithm that we use to compute $\{h_t,\, t \in \mathcal{S}\}$. The conditioned process evolves according to the kernels $P_t^\star$, which can be viewed as applying a change of measure to the kernels P_t.

3.2 Continuous Edges

Up to this point, we have assumed edges to represent "discrete time" Markov-transitions. The probabilistic evolution over a single edge is then captured by a Markov kernel P. Now suppose the process transitions over a continuous edge. In this subsection, we fix one such edge.

Definition 2 *We call an edge $e = (s,t)$ continuous if it is assumed that on e a continuous-time process evolves over the time interval $[0, \tau_e]$. The process is assumed to be defined on the filtered probability space $(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$ where $\mathbb{F} = \{\mathcal{F}_u,\, u \in [0,\tau_e]\}$. The process $X^e := (X_u,\, u \in [0,\tau_e])$ is assumed to be a right-continuous, $\mathbb{F}$-adapted Markov process taking values in a metric space E. We identify time 0 with node s and similarly time τ_e with node t. That is, $X_0^e = X_s$ and $X_{\tau_e}^e = X_t$. We denote the infinitesimal generator of the space-time process $((u, X_u),\, u \in [0,\tau_e])$ by $\mathcal{A}$ and its domain by $\mathcal{D}_\mathcal{A}$.*

Fix a continuous edge e. Assume $X_0 = x_s$ and existence of transition kernels $P_{s,t}$ such that

$$\mathbb{P}(X_u \in \mathrm{d}y \mid X_{u'} = x) = P_{u',u}(x, \mathrm{d}y), \qquad 0 < u' < u < \tau_e.$$

Suppose h_t is given and set $h_{\tau_e} := h_t$. For $u \in [0, \tau_e)$ define the function $(u,x) \mapsto h_u(x)$ as the pullback of h_{τ_e} under P_{u,τ_e}:

$$h_u(x) = (P_{u,\tau_e} h_{\tau_e})(x) = \int h_{\tau_e}(y) P_{u,\tau_e}(x, \mathrm{d}y). \tag{6}$$

For convenience, we interchangeably write $h_u(x)$ and $h(u,x)$. For fixed u, the map h_u is defined by $x \mapsto h_u(x)$. It is well known that $(u,x) \mapsto h(u,x)$ satisfies the Kolmogorov backward equation (Bass (2011), Chapter 37, in particular (37.5))

$$\mathcal{A} h_u = 0, \qquad \text{for } u \in [0,\tau_e], \quad \text{subject to } h_{\tau_e}. \tag{7}$$

Put differently, h is space-time harmonic. This provides the BIF when an edge is governed by a continuous-time process evolving. Finally, to blend this into the description given in the previous section, we have the following definition:

Definition 3 *For a continuous edge $e = (s,t)$, we define $h_{s,t} = h_0$, where h_0 is the solution to (7) at time 0.*

Next, we identify the dynamics of the conditioned process. Let $\mathbb{P}_u$ denote the restriction of $\mathbb{P}$ to $\mathcal{F}_u$. Define

$$Z_u = \frac{h_u(X_u)}{h_0(x_s)}.$$

By (6), we can write $h_u(X_u) = \mathbb{E}[A \mid \mathcal{F}_u]$, where $A := h_{\tau_e}(X_{\tau_e})$. It follows that $u \mapsto h_u(X_u)$ is a martingale, provided that A has finite expectation. In the following we assume this to be the case, sufficient conditions being either boundedness of $x \mapsto h_{\tau_e}(x)$ or square integrability of A. For $u \in [0, \tau_e)$ define the measures $\mathbb{P}^\star_u$ by $\mathrm{d}\mathbb{P}^\star_u = Z_u \,\mathrm{d}\mathbb{P}_u$ and note that $\mathbb{P}^\star_u$ is the law of the process $(X_u,\, u \in [0, \tau_e])$, restricted to $\mathcal{F}_u$, that is conditioned on all observations at leaves descending from vertex s. It follows from formula (3.2) in Palmowski and Rolski (2002) that the infinitesimal generator $\mathcal{A}^\star$ of the process under $\mathbb{P}^\star_t$ can be expressed in terms of $\mathcal{A}$ and h in the following way

$$\mathcal{A}^\star f = h^{-1}\mathcal{A}(fh) - h^{-1} f \mathcal{A} h \tag{8}$$

(see also Chetrite and Touchette (2015), Section 4.1). Here, the second term on the right-hand-side must be understood as the composition of three operators: multiplication by h^{-1}, multiplication by f and multiplication by the operator $\mathcal{A}$ applied to h (the first term on the right-hand-side can be similarly understood). Equation (8) is useful for deriving the dynamics of the conditioned process.

Example 1 *Consider Figure 1 and observations $\{x_5, x_6, x_7\}$ at leaf vertices. The BIF consists of the following steps:*

- *Initialise from the leaves: set $h_{4,5}(x) = k_{4,5}(x, x_5)$, $h_{3,6}(x) = k_{3,6}(x, x_6)$ and $h_{2,7}(x) = k_{2,7}(x, x_7)$.*
- *Collect incoming messages at vertices 2 and 4: $h_2 = h_{2,7}$ and $h_4 = h_{4,5}$.*
- *Solve (7) on edge $(3, 4)$ (with end value h_4) to get $h_{3,4}$.*
  Solve (7) on edge $(1, 2)$ (with end value h_2) to get $h_{1,2}$.
- *Collect incoming messages at vertices 1 and 3: $h_1 = h_{1,2}$ and $h_3 = h_{3,4}h_{3,6}$.*
- *Solve (7) on edge $(0, 3)$ (with end value h_3) to get $h_{0,3}$.*
  Solve (7) on edge $(0, 1)$ (with end value h_1) to get $h_{0,1}$.
- *Collect incoming messages at vertex 0: $h_0 = h_{0,3}h_{0,1}$.*

If a prior measure $\Pi = P_{r,0}$ is specified on the value at vertex 0, then $h_r(x_r) = \int h_0(x)\Pi(\,\mathrm{d}x)$. We can see that each edge passes a message to the parent vertex of that edge.

Remark 4 *If transitions over edges depend on an unknown parameter θ then h_t will depend on θ in general. For notational convenience, we have suppressed this dependency from the notation. The likelihood for θ is, by definition, given by $L(\theta, x_\mathcal{V}) := h_r(x_r)$.*

4. Guiding

Example 1 shows that the BIF consists of 3 types of operation: *(i)* solving Equation (7), *(ii)* pullback under a Markov kernel as defined in (1), *(iii)* pointwise multiplication of functions. Unfortunately, few cases exist where this can be done in closed form. We propose to resolve this by working with an approximation g of h, for example by choosing a Gaussian approximation to the dynamics of X and computing the h-transform for that approximation. Using g rather than h to change the dynamics of the forward process leads to the notion of a guided process. Contrary to the true conditioned process, the guided process is tractable. In particular, we can simulate the process. This in turn can be used with stochastic simulation to correct for the discrepancy induced by using g instead of h.

4.1 Discrete Case

We start with the definition of the guided process in the discrete case. Recall g_s serves as a deterministic approximation to h_s.

Definition 5 *Assume the maps $x \mapsto g_s(x)$ are specified for each $s \in \mathcal{S}$. We define the* guided process X° *as the process starting in $X^\circ_r = x_r$ and from the root onwards evolving on $\mathcal{S}_r$ according to transition kernel*

$$P^\circ_s(x, \mathrm{d}y) = \frac{g_s(y)P_s(x, \mathrm{d}y)}{\int g_s(y)P_s(x, \mathrm{d}y)} = \frac{g_s(y)P_s(x, \mathrm{d}y)}{(P_s g_s)(x)}, \qquad s \in \mathcal{S}.$$

Here, implicitly, we assume that the dominator is strictly positive and finite. When $g_s = h_s$, then $P^\circ_s = P^\star_s$. Define the measure $\mathbb{P}^\circ$ by

$$\frac{\mathrm{d}\mathbb{P}^\circ}{\mathrm{d}\mathbb{P}}(x_{\mathcal{S}}) = \prod_{s\in\mathcal{S}} \frac{P^\circ_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}{P_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}$$

Similarly, define the measure $\mathbb{P}^\star$ by

$$\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}}(x_{\mathcal{S}}) = \prod_{s\in\mathcal{S}} \frac{P^\star_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}{P_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}$$

Lemma 6

$$\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}}(x_{\mathcal{S}}) = \frac{\prod_{v\in\mathcal{V}} h_{\mathrm{pa}(v),v}(x_{\mathrm{pa}(v)})}{h_r(x_r)}.$$

Proof For $s \in \mathcal{S}$,

$$P^\star_s(x, \mathrm{d}y) = \frac{h_s(y)}{h_{\mathrm{pa}(s),s}(x)} P_s(x, \mathrm{d}y). \tag{9}$$

Thus, using that h_s is defined by fusion,

$$\prod_{s\in\mathcal{S}} \frac{h_s(y)}{h_{\mathrm{pa}(s),s}(x)} = \frac{\prod_{s\in\mathcal{S}} \prod_{t\in\mathrm{ch}(s)} h_{s,t}(X_s)}{\prod_{s\in\mathcal{S}} h_{\mathrm{pa}(s),s}(X_{\mathrm{pa}(s)})} = \frac{\prod_{v\in\mathcal{V}} h_{\mathrm{pa}(v),v}(X_{\mathrm{pa}(v)})}{h_r(x_r)} \tag{10}$$

which follows by cancellation of terms (note that the numerator is a product over all edges except those originating from the root node, whereas the denominator is a product over all

edges except those ending in a leaf node). The result follows upon taking the product over all $s \in \mathcal{S}$ in (9). ■

Theorem 7 *Assume maps $x \mapsto g_t(x)$ are specified for all $t \in \mathcal{S}_r$. If we define $\bar{w}_t(x)$ by*

$$g_{\mathrm{pa}(t)}(x)\bar{w}_t(x) = \begin{cases} (P_t g_t)(x) & \text{if} \quad t \in \mathcal{S} \\ h_{\mathrm{pa}(t),t}(x) & \text{if} \quad t \in \mathcal{V} \end{cases}, \tag{11}$$

then

$$\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^\circ}(x_{\mathcal{S}}) = \frac{g_r(x_r)}{h_r(x_r)} \prod_{t \in \mathcal{T}} \bar{w}_t(x_{\mathrm{pa}(t)}).$$

Proof For $s \in \mathcal{S}$,

$$P_s^\circ(x, \mathrm{d}y) = \frac{g_s(y)}{g_{\mathrm{pa}(s)}(x)} \frac{g_{\mathrm{pa}(s)}(x)}{(P_s g_s)(x)} P_s(x, \mathrm{d}y) = \frac{g_s(y)}{g_{\mathrm{pa}(s)}(x)} \frac{1}{\bar{w}_s(x)} P_s(x, \mathrm{d}y).$$

Therefore

$$\prod_{s \in \mathcal{S}} \frac{P_s^\circ(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}{P_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)} = \prod_{s \in \mathcal{S}} \frac{g_s(x_s)}{g_{\mathrm{pa}(s)}(x_{\mathrm{pa}(s)})} \frac{1}{\bar{w}_s(x_{\mathrm{pa}(s)})} = \frac{\prod_{v \in \mathcal{V}} g_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})}{g_r(x_r)} \prod_{s \in \mathcal{S}} \frac{1}{\bar{w}_s(x_{\mathrm{pa}(s)})}.$$

Using Lemma 6, this gives

$$\begin{aligned} \frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^\circ}(x_{\mathcal{S}}) &= \prod_{s \in \mathcal{S}} \frac{P^\star(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}{P^\circ(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)} = \frac{g_r(x_r)}{h_r(x_r)} \prod_{v \in \mathcal{V}} \frac{h_{\mathrm{pa}(v),v}(x_{\mathrm{pa}(v)})}{g_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})} \prod_{s \in \mathcal{S}} \bar{w}_s(x_{\mathrm{pa}(s)}) \\ &= \frac{g_r(x_r)}{h_r(x_r)} \prod_{t \in \mathcal{S} \cup \mathcal{V}} \bar{w}_t(x_{\mathrm{pa}(t)}). \end{aligned}$$

■

We see that each edge contributes a weight, which is evaluated at the starting value of the branch. In fact, this theorem is valid on a directed acyclic graph. Rather than defining g_t one can also start from defining $g_{s,t}$ for all edges (s,t).

Theorem 8 *Assume maps $x \mapsto g_{s,t}(x)$ are specified for each edge $e = (s,t) \in \mathcal{E}$ and define g_s by fusion, i.e.*

$$g_s(x) = \prod_{t \in \mathrm{ch}(s)} g_{s,t}(x), \qquad s \in \mathcal{S}_r. \tag{12}$$

If we define $w_t(x)$ by

$$g_{\mathrm{pa}(t),t}(x) w_t(x) = \begin{cases} (P_t g_t)(x) & \text{if} \quad t \in \mathcal{S} \\ h_{\mathrm{pa}(t),t}(x) & \text{if} \quad t \in \mathcal{V} \end{cases}, \tag{13}$$

then

$$\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}^\circ}(x_{\mathcal{S}}) = \frac{g_r(x_r)}{h_r(x_r)} \prod_{t \in \mathcal{T}} w_t(x_{\mathrm{pa}(t)}).$$

Proof For $s \in \mathcal{S}$,

$$P_s^\circ(x, \mathrm{d}y) = \frac{g_s(y)}{g_{\mathrm{pa}(s),s}(x)} \frac{g_{\mathrm{pa}(s),s}(x)}{(P_s g_s)(x)} P_s(x, \mathrm{d}y) = \frac{g_s(y)}{g_{\mathrm{pa}(s),s}(x)} \frac{1}{w_s(x)} P_s(x, \mathrm{d}y).$$

Thus, using that g_s is defined by fusion,

$$\prod_{s\in\mathcal{S}} \frac{g_s(y)}{g_{\mathrm{pa}(s),s}(x)} = \frac{\prod_{s\in\mathcal{S}} \prod_{t\in\mathrm{ch}(s)} g_{s,t}(X_s)}{\prod_{s\in\mathcal{S}} g_{\mathrm{pa}(s),s}(X_{\mathrm{pa}(s)})} = \frac{\prod_{v\in\mathcal{V}} g_{\mathrm{pa}(v),v}(X_{\mathrm{pa}(v)})}{g_r(x_r)} \tag{14}$$

which follows by cancellation of terms as in the proof of Lemma 6. This gives

$$\prod_{s\in\mathcal{S}} \frac{P_s^\circ(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)}{P_s(x_{\mathrm{pa}(s)}, \mathrm{d}x_s)} = \frac{\prod_{v\in\mathcal{V}} g_{\mathrm{pa}(v)}(x_{\mathrm{pa}(v)})}{g_r(x_r)} \prod_{s\in\mathcal{S}} \frac{1}{w_s(x_{\mathrm{pa}(s)})}.$$

The remainder of the proof is the same as that of Theorem 7. ■

The maps $h_{\mathrm{pa}(v),v}$ ($v \in \mathcal{V}$) can typically be assumed to be tractable. Recall that at vertex s, $h_s(x)$ is the likelihood in the subtree with root note s with known value x. As a corollary, we get that the likelihood can be expressed as an expectation over the guided process.

Corollary 9

$$h_r(x_r) = g_r(x_r)\, \mathbb{E}^\circ\Big[\prod_{t\in\mathcal{T}} w_t(X_{\mathrm{pa}(t)})\Big].$$

It stands to reason that good choices for g_s approximate h_s. The definition leaves open the choice of $g_{s,t}$. Given data at the leaf nodes, we need to be able to algorithmically and efficiently identify a good choice of $g_{s,t}$ by deterministic calculations. One way of defining it, is to attach to each kernel P_s a kernel $\tilde{P}_s$ (with both kernels having the same source and target) and setting $g_{\mathrm{pa}(s),s}(x) = \int g_s(y)\tilde{P}_s(x, \mathrm{d}y)$. Then the choice of $\tilde{P}$ should be such that calculations in the BIF get simpler. This idea is explored in more detail in concrete examples.

Remark 10 *Within the language of Feynman-Kac models (Chapter 5, Chopin and Papaspiliopoulos (2020)), the kernels P_s° are mutation kernels that can be used in a guided particle filter. Often, these are denoted by M in existing literature.*

4.2 Continuous Case: Exponential Change of Measure

Suppose e is a continuous edge. The maps h_u for $u \in [0, \tau_e]$ can only be computed in closed form in highly specific settings. For this reason, we propose to replace those maps by substitutes g_u. Assume $(u, x) \mapsto g_u(x)$ is in the domain of $\mathcal{A}$. Under weak conditions on $\{g_u,\, u \in [0, \tau_e]\}$ we get that

$$Z_u^g := \frac{g_u(X_u)}{g_0(x_s)} \exp\left(-\int_0^u \frac{(\mathcal{A}g_\tau)(X_\tau)}{g_\tau(X_\tau)}\, \mathrm{d}\tau\right)$$

is a mean-one $\mathcal{F}_u$-local martingale. Corollary 32 in the appendix provides a sufficient condition for this.

A function $g\colon [0,\tau_e] \times E \to \mathbb{R}$ for which Z_u is actually a martingale is called a *good* function, the terminology being borrowed from Palmowski and Rolski (2002). Sufficient conditions for g to be good are given in Proposition 33. Assume we are given a good function g, postpone the question on how to find such functions in specific settings. For $u \in [0,\tau_e)$ we can define the probability measure $\mathbb{P}^\circ_u$ by $\mathrm{d}\mathbb{P}^\circ_u = Z^g_u \,\mathrm{d}\mathbb{P}_u$. Similar to (8), under $\mathbb{P}^\circ_t$ the process (u, X_u) has infinitesimal generator $\mathcal{A}^\circ$ satisfying

$$\mathcal{A}^\circ f = g^{-1}\mathcal{A}(fg) - g^{-1} f \mathcal{A} g \tag{15}$$

from which the dynamics of the process under $\mathbb{P}^\circ_u$ can be derived.

Definition 11 *The process $X = (X_u,\, u \in [0,\tau_e])$, with $X_0 = x_s$, under the law $\mathbb{P}^\circ_{\tau_e}$ is denoted by X° and referred to as the* guided process induced by g on the edge e.

Proposition 12 *Suppose X evolves on the continuous edge $e = (s,t)$. If X° is the guided process induced by g on e, where g is a good function, then Theorem 8 remains valid upon defining*

$$g_{s,t}(x) := g_0(x) \qquad \textit{and} \qquad w_t\left(X^\circ\right) = \exp\left(\int_0^{\tau_e} \frac{\mathcal{A}g}{g}(u, X^\circ_u)\,\mathrm{d}u\right).$$

Using a tractable approximation g to h we aim to approximate the information brought by future observations, and let this approximation guide the process in a natural way. In absence of information from observations, the process evolves just as the unconditional one.

4.2.1 Discrete versus Continuous Edges

It is interesting to see the structure in discrete versus continuous edges. On a discrete edge $e = (0,T)$ (with *vertices* denoted by 0 and T) we have

$$\frac{P^\circ(x_0,\,\mathrm{d}x_T)}{P(x_0,\,\mathrm{d}x_T)} = \frac{g_T(x_T)}{g_{0,T}(x_0)} \frac{g_{0,T}(x_0)}{(Pg_T)(x_T)} \tag{16}$$

while on a continuous edge where *time* evolves from 0 to T we have

$$\frac{\mathrm{d}\mathbb{P}^\circ_T}{\mathrm{d}\mathbb{P}_T}(x) = \frac{g_T(x_T)}{g_0(x_0)} \exp\left(-\int_0^T \frac{\mathcal{A}g_u}{g_u}(x_u)\,\mathrm{d}u\right). \tag{17}$$

In (16) and (17) the first term on the right-hand-side plays exactly the some role, while the weight is defined to be the inverse of the second term. Once the Radon-Nikodym derivative of the law of the conditioned process with respect to the guided process has been defined on each edge, we can define the change of measure on the tree itself. We present this construction for continuous edges in Appendix B.

4.2.2 Choice of g

To forward simulate the guided process, the maps g_u need to be specified along the graph. Ideally, g_u should be like h_u solving $\mathcal{A}h_u = 0$. The operator $\mathcal{A}$ can be written as $\partial_u + \mathcal{L}$, where $\mathcal{L}$ is the infinitesimal generator of the process $(X_u,\, u \in [0,\tau_e])$, applied to functions where the "time"-variable u is considered fixed.

One option for defining g_u consists of replacing $\mathcal{L}$ by $\tilde{\mathcal{L}}$, chosen to be the generator of another continuous-time Markov process. That is, g solves

$$(\tilde{\mathcal{L}} + \partial_u) g_u = 0, \quad u \in [0, \tau_e] \quad \text{subject to} \quad g_{\tau_e}. \tag{18}$$

In this case $\mathcal{A}g = (\mathcal{L} + \partial_u)g = (\mathcal{L} - \tilde{\mathcal{L}})g$. Sections 7.1 and 7.2 provide examples of this approach.

Another choice consists of imposing a truncated series expansion of the ansatz $g(t,x) = \sum_{k=1}^{K} \alpha_k(t)\psi_k(x)$ and solving

$$\partial_t g + \Pi_K(\mathcal{L}g) = 0.$$

Here $\Pi_K f$ projects the function f onto $\{g\colon g(t,x) = \sum_{k=1}^{K} \alpha_k(t)\psi_k(x)\}$. In Section 7.3 we show how this approach is utilised in Stoltz et al. (2021).

5. Backward Filtering Forward Guiding

Application of Theorem 8 requires to compute $\{g_s,\, s \in \mathcal{S}\}$, from the leaves back to the root vertex, hence traversing the tree backwards. Once this computation is performed, the guided process can be forward simulated according to Definition 5. In simulating the guided process, we traverse the tree once again, in reverse direction. For this reason, we call the joint operation *Backward Filtering Forward Guiding (BFFG)*. Note that the dependency structure of the guided process is inherited from the (unconditional) forward process.

If we simulate multiple guided processes, then Theorem 8 reveals that the relative weight of path x is given by

$$g_r(x_r) \prod_{t \in \mathcal{T}} w_t(x_{\mathrm{pa}(t)}).$$

Moreover, it forms a positive unbiased estimator for the likelihood $h_r(x_r)$ (which follows from Corollary 9). For multiple algorithms, such as Sequential Monte Carlo (SMC), estimation of the weight by a positive unbiased estimator suffices.

Denote the collection of g-functions that appear in the definition of $\mathbb{P}^\circ$ by $\mathcal{G} := \{g_{s,t},\, e = (s,t) \in \mathcal{E}\}$. Here, implicitly, for a continuous edge e this includes $\{g_u,\, u \in [0, \tau_e]\}$. In a companion paper Schauer et al. (2025) we study the compositional structure of BFFG in the language of category theory. While we do not go into these details here, we do want to identify the main requirements.

1. $\mathcal{G}$ can be computed in closed form or a "good" choice can be specified right away.

2. Sampling under $\mathbb{P}^\circ$ is tractable. Recall that $\mathbb{P}^\circ$ is obtained by a change of measure of $\mathbb{P}$ using $\mathcal{G}$. It is the law of the guided process.

3. The weights showing up in either Theorem 8 or Proposition 12 (in case of a continuous edge) can be evaluated.

For a discrete edge, one generic approach for (1) consists of pairing each kernel P_t with a kernel $\tilde{P}_t$ for which the backward information filter can easily be computed in closed form. In Section 4.2.2 we already listed some strategies for approaching (1) in case of a continuous edge.

In sections 6 and 7 we provide examples of discrete- and continuous time guided processes respectively. In each of these examples the map g can be identified from a parameter ζ. In that case we write $g \sim \mathcal{P}(\zeta)$ to denote that g is parametrised by ζ. Backward filtering is tractable when

- if $g \sim \mathcal{P}(\zeta)$, then $\tilde{P}g \sim \mathcal{P}(\zeta')$ for some ζ' (this corresponds to the pullback step for a discrete edge);
- if $g_{\tau_e} \sim \mathcal{P}(\zeta)$, then for all $u \in [0, \tau_e]$, the solution to (18) satisfies $g_u \sim \mathcal{P}(\zeta_u)$ for some ζ_u (this corresponds to the pullback step for a continuous edge e);
- if $g_i \sim \mathcal{P}(\zeta_i)$, then $\prod_i g_i \sim \mathcal{P}(\zeta')$ for some ζ' (this corresponds to the fusion step).

For each of the examples that follow, the results are written in the following form:

1. *Computing $\mathcal{G}$:*
   - *Pullback for $\tilde{P}$:*
   - *Fusion:*
   - *Initialisation from leaves:*
2. *Sampling under $\mathbb{P}^\circ$:*
3. *Computation of the weight:*

Here, in case of a continuous edge e, "pullback for $\tilde{P}$" refers to the backward filtering step on the edge e under simplified dynamics of the forward process.

Remark 13 *The guiding functions $g_{s,t}$ may be constructed either* a priori *(before the forward pass) or* iteratively*. The latter approach parallels the Iterated Extended Kalman Smoother methodology: a forward pass generates particles using current guides, a backward pass refines the guides using these particles, and the procedure iterates until convergence. For example, the iterated auxiliary particle filter (Guarniero et al. (2017)) is obtained by iteratively running a twisted auxiliary particle filter to approximate the h-transform.*

While an apriori construction is simpler and computationally cheaper per iteration, iterative refinement can yield superior approximations, particularly in high-dimensional or strongly nonlinear settings where initial guiding distributions are poor. Hybrid approaches—applying iterative refinement selectively where effective sample size indicates approximation deficiencies—may offer the best practical trade-off within SMC.

6. Examples of Guided Processes for Discrete Edges

6.1 Nonlinear Gaussian Kernels

Consider the stochastic process on $\mathcal{G}$ with transitions defined by

$$X_t \mid X_s = x \sim N(\mu_t(x), Q_t(x)), \qquad t \in \mathrm{ch}(s). \tag{19}$$

Unfortunately, for this class of models (parametrised by (μ_t, Q_t)), only in very special cases Doob's h-transform is tractable. For that reason, we look for a tractable h-transform to define a guided process X°. This is obtained in the specific case where

$$\tilde{X}_t \mid \tilde{X}_s = x \sim N(\Phi_t x + \beta_t, Q_t), \qquad t \in \mathrm{ch}(s).$$

Below we show that backward filtering, sampling from the forward map and computing the weights are all fully tractable. In the following, we write $\varphi(x; \mu, \Sigma)$ for the density of the $N(\mu, \Sigma)$-distribution, evaluated at x. Similarly, we write $\varphi^{\mathrm{can}}(x; F, H)$ for the density of canonical Normal distribution with potential $F = \Sigma^{-1}\mu$ and precision matrix $H = \Sigma^{-1}$, evaluated at x.

The g functions can be parametrised by the triple (c, F, H):

$$\begin{aligned} g(y) &= \exp\left(c + y'F - \frac{1}{2}y'Hy\right) \\ &= \varpi(c, F, H)\varphi^{\mathrm{can}}(y; F, H) = \varpi(c, F, H)\varphi(y; H^{-1}F, H^{-1}), \end{aligned} \tag{20}$$

where $\log \varpi(c, F, H) = c - \log \varphi^{\mathrm{can}}(0, F, H)$. We write $g \sim \mathcal{P}(c, F, H)$ for g and parameters as in (20).

In the following result we write $P \equiv P_s$, $\tilde{P} \equiv \tilde{P}_s$ and $w \equiv w_s$.

Theorem 14 *Let $P(x, \mathrm{d}y) = \varphi(y; \mu(x), Q(x))\,\mathrm{d}y$ and $\tilde{P}(x, \mathrm{d}y) = \varphi(y; \Phi x + \beta, Q)\,\mathrm{d}y$. Assume that $g \sim \mathcal{P}(c, F, H)$ with invertible H.*

1. *Computing $\mathcal{G}$:*

   - *Pullback for $\tilde{P}$: With $C = Q + H^{-1}$ invertible,*

   $$\tilde{P}g \sim \mathcal{P}\left(\bar{c}, \bar{F}, \bar{H}\right) \quad \textit{where} \quad \begin{cases} \bar{H} = \Phi' C^{-1} \Phi \\ \bar{F} = \Phi' C^{-1}(H^{-1}F - \beta) \\ \bar{c} = c - \log \varphi^{\mathrm{can}}(0, F, H) + \log \varphi(\beta; H^{-1}F, C) \end{cases}.$$

   - *Fusion: if $g_i \sim \mathcal{P}(c_i, F_i, h_i)$, $i = 1, \ldots, k$, then*

   $$\prod_{i=1}^{k} g_i \sim \mathcal{P}\left(\sum_{i=1}^{k} c_i, \sum_{i=1}^{k} F_i, \sum_{i=1}^{k} H_i\right).$$

   - *Initialisation from leaves: if $v \sim N(\Phi x + \beta, Q)$ is observed at a leaf, then $g_{\mathrm{pa}(v),v} \sim \mathcal{P}(c, F, H)$ where*

   $$c = \log \varphi(\beta; v, Q) \qquad F = \Phi' Q^{-1}(v - \beta) \qquad H = \Phi' Q^{-1} \Phi.$$

2. *Sampling under $\mathbb{P}^\circ$:*

$$X^\circ_s \mid X^\circ_{\mathrm{pa}(s)} = x \sim N^{\mathrm{can}}\left(F + Q(x)^{-1}\mu(x), H + Q(x)^{-1}\right).$$

3. *Computation of the weight: if* $C(x) = Q(x) + H^{-1}$ *is invertible, then* $(Pg)(x) = \varpi(c, F, H)\varphi(H^{-1}F; \mu(x), C(x))$ *and* $w(x)$ *is obtained from*

$$w(x) = \frac{(Pg)(x)}{(\tilde{P}g)(x)}.$$

Proof The proof is elementary. For the pullback, similar results have appeared in the literature, for example Chapter 7 in Chopin and Papaspiliopoulos (2020), Wilkinson and Yeung (2002) and Chapter 5 in Cappé et al. (2005).

First note that

$$\begin{aligned}(Pg)(x) &= \int g(y)P(x, \mathrm{d}y) = \int \varpi(c, F, H)\varphi(y; H^{-1}F, H^{-1})\varphi(y; \mu(x), Q(x))\,\mathrm{d}y \\ &= \varpi(c, F, H)\varphi(H^{-1}F; \mu(x), Q(x) + H^{-1}).\end{aligned}$$

By using the specific form of $\mu(x)$ and $Q(x)$ for the kernel $\tilde{P}$ the expression for the weight follows.

To find the parametrisation of $\tilde{P}h$, let $C = Q + H^{-1}$. We have

$$\begin{aligned}(\tilde{P}g)(x) &= \varpi(c, F, H)\varphi(H^{-1}F; \Phi x + \beta, Q + H^{-1}) \\ &= \varpi(c, F, H)(2\pi)^{-d/2}|C|^{-1/2} \\ &\quad \times \exp\left(-\frac{1}{2}x'\Phi' C^{-1}\Phi x + (H^{-1}F - \beta)'C^{-1}\Phi x - \frac{1}{2}(H^{-1}F - \beta)'C^{-1}(H^{-1}F - \beta)\right).\end{aligned}$$

The result follows upon collecting terms. If Φ is invertible, then

$$\bar{F}'\bar{H}\bar{F} = (H^{-1}F - \beta)'C^{-1}(H^{-1}F - \beta).$$

A bit of algebra then gives the stated result. The derivation of the parametrisation of the fusion step is trivial. To derive forward simulation under $\mathbb{P}^\circ$, we write $\propto$ to denote proportionality with respect to y

$$\begin{aligned}\frac{g(y)P(x, \mathrm{d}y)}{\mathrm{d}y} &\propto \exp\left(-\frac{1}{2}y'Hy + y'F\right)\exp\left(-\frac{1}{2}(y - \mu(x))'Q(x)^{-1}(y - \mu(x))\right) \\ &\propto \exp\left(-\frac{1}{2}y'(H + Q(x)^{-1})y + y'(F + Q(x)^{-1}\mu(x))\right) \\ &\propto \varphi^{\mathrm{can}}(y; F + Q(x)^{-1}\mu(x), H + Q(x)^{-1})\end{aligned}$$

which suffices to be shown. ■

Regarding the pullback for $\tilde{P}$, in computing $\bar{H}$ and $\bar{F}$ we can avoid inverting the matrices H and C. To see this, $C^{-1}H^{-1} = (Q+H^{-1})^{-1}H^{-1} = (QH+I)^{-1}$ and $C^{-1} = (Q+H^{-1})^{-1} = H - H(H + Q^{-1})^{-1}H$. While backward filtering for a linear Gaussian process on a tree is well known (see e.g. Chou et al. (1994), section 3), results presented in the literature often don't state update formulas for the constant ϖ (or equivalently c). Of course, the guided process is unaffected by multiplying g by a scalar. However, if $g \sim \mathcal{P}(c, F, H)$ with $c = 0$,

then $Pg \sim \mathcal{P}\left(\bar{c}, \bar{F}, \bar{H}\right)$ with $\bar{c} \neq 0$. To maintain a consistent presentation of all examples in this section we therefore present our results for the full triplet, including c. In case the dynamics of X itself are linear, then one can take $\tilde{P} = P$ which implies $w(x) = 1$ for all x. Moreover, if additionally the process lives on a "line graph with attached observation leaves", where each non-leaf vertex has one child in $\mathcal{S}$ and at most one child in $\mathcal{V}$, BFFG is essentially equivalent to Forward Filtering Backward Sampling (FFBS, Carter and Kohn (1994) and Frühwirth-Schnatter (1994)), the difference being that our procedure applies in time-reversed order. On a general directed tree however the ordering cannot be changed and the filtering steps must be done backwards, as we propose, just like in message passing algorithms in general.

Even if the true forward dynamics are linear, i.e. $x \mapsto \mu_t(x)$ is linear and the covariance matrix $Q_t(x)$ does in fact not depend on x, it can be computationally efficient to backward filter using an approximation. For example, in spatial models Vecchia approximation (Vecchia (1988), Katzfuss (2017)) can be used for constructing a sparse representation of Q_t^{-1}.

6.2 Discrete State-Space Markov Chains and Particle Systems

6.2.1 Branching Particle on a Tree

Assume a "particle" takes values in a finite state space $E = \{1, \ldots, R\}$ according to the $R \times R$ transition matrix K. At each vertex, the particle is allowed to copy itself and branch.

The algorithmic elements can easily be identified. The forward kernel $P(x, \mathrm{d}y)$ can be identified with the matrix K. Furthermore, the map $x \mapsto g(x)$ can be identified with the column vector $\boldsymbol{g} = [g_1, \ldots, g_R]'$, where $g_i = g(i)$. Hence g is parametrised by $\boldsymbol{g}$ and we write $g \sim \mathcal{P}(\boldsymbol{g})$.

Let $\boldsymbol{e}_k$ denote the k-th standard basis-vector in $\mathbb{R}^R$ and denote the j-th element of the vector a by $\langle a \rangle_j$.

Theorem 15 *Let $P(x, \mathrm{d}y)$ and $\tilde{P}(x, \mathrm{d}y)$ be represented by the stochastic matrices K and $\tilde{K}$ respectively. Assume that $h \sim \mathcal{P}(\boldsymbol{g})$.*

1. *Computing $\mathcal{G}$:*
   - *Pullback for $\tilde{P}$: $\tilde{P}g \sim \mathcal{P}\left(\tilde{K}\boldsymbol{g}\right)$*
   - *Fusion: if $g_i \sim \mathcal{P}(\boldsymbol{g}_i)$, $i = 1, \ldots, k$, then*
   $$\prod_{i=1}^{k} g_i \sim \mathcal{P}\left(\bigcirc_{i=1}^{k} \boldsymbol{g}_i\right),$$
   where $\bigcirc$ denotes the Hadamard (entrywise) product.
   - *Initialisation from leaves: at a leaf vertex v with observation $k \in E$, set $\boldsymbol{g}_{\mathrm{pa}(v),v} = \boldsymbol{e}_k$.*
2. *Sampling under $\mathbb{P}^\circ$:*
$$\mathbb{P}^\circ(X_s = k \mid X_{\mathrm{pa}(s)} = \ell) = \frac{\langle \boldsymbol{z}_\ell \bigcirc \boldsymbol{g}_s \rangle_k}{\boldsymbol{z}_\ell' \boldsymbol{g}_s},$$

where $\boldsymbol{z}_\ell = K'_{\mathrm{pa}(s),s}\boldsymbol{e}_\ell$ *is the* ℓ*-th row of* $K_{\mathrm{pa}(s),s}$.

3. *Computation of the weight: The weight at* $x \in E$ *is given by*

$$w(x) = \frac{(Pg)(x)}{(\tilde{P}g)(x)} = \frac{\langle K\boldsymbol{g}\rangle_x}{\langle \tilde{K}\boldsymbol{g}\rangle_x}$$

While we can simply take $\tilde{P} = P$ on a tree, yielding unit weights, in case R is very large, it can nevertheless be advantageous to use a different (simpler) map $\tilde{P}$. To see, this, note that we only need to compute one element of the matrix vector product $K\boldsymbol{g}$, but need to compute the full vector $\tilde{K}\boldsymbol{g}$ in when backward filtering. Hence, choosing $\tilde{K}$ sparse can give computational advantages.

6.2.2 Interacting Particles —Cellular Automata— Agent Based Models

Now consider a discrete time interacting particle process, say with n particles, where each particle takes values in $\{1, \ldots, R\}$. Hence, a particle configuration x at a particular time-instant takes values in $E = \{1, \ldots, R\}^n$.

A simple example consists of particles with state $E = \{1 \equiv \mathbf{S}, 2 \equiv \mathbf{I}, 3 \equiv \mathbf{R}\}$ that represent the health status of individuals that are either **S**usceptible, **I**nfected or **R**ecovered. Then the probability to transition from state **S** to **I** may depend on the number of nearby particles (individuals) that are infected, hence the particles are "interacting". A similar example is obtained from time-discretisation of the contact process (cf. Liggett (2005)). Note that each particle has multiple parents defined by a local neighbourhood and that a model like this is sometimes referred to as a cellular automaton.

Let $x \in E$. Without additional assumptions, the forward transition kernel can be represented by a $R^n \times R^n$ transition matrix. With a large number of particles such a model is not tractable computationally. To turn this into a more tractable form, we make the simplifying assumption that conditional on x each particle transitions independently. This implies that the forward evolution kernel $P(x, \mathrm{d}y)$ can be represented by $(K_1(x), \ldots, K_n(x))$, where $K_i(x)$ is the $R \times R$ transition matrix for the i-th particle. Note that the particles interact because the transition kernel for the ith particle may depend on the state of *all* particles.

Due to interactions, it will computationally be very expensive to use P for backward filtering. Instead, in the backward map we propose to ignore/neglect all interactions between particles. This means that effectively we backward filter upon assuming all particles move independently, and the evolution of the i-th particle depends only on x_i (not x, as in the forward kernel). Put differently, in backward filtering, we simplify the graphical model to n line graphs, one for each particle.

This choice implies that $\tilde{P}$ can be represented by $(\tilde{K}_1, \ldots, \tilde{K}_n)$ and the map $x \mapsto g(x) = \prod_{i=1}^n g_i(x)$ can be represented by $(\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n)$. We write $g \sim \mathcal{P}\left(\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n\right)$. The following result is a straightforward consequence of Theorem 15.

Theorem 16 *Assume* P *and* $\tilde{P}$ *are represented by* $R \times R$ *stochastic matrices* $K_1, \ldots, K_n$ *and* $\tilde{K}_1, \ldots, \tilde{K}_n$ *respectively. Assume that* $g \sim \mathcal{P}\left(\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n\right)$.

1. *Computing* $\mathcal{G}$*:*

- *Pullback for $\tilde{P}$:* $\tilde{P}g \sim \mathcal{P}\left(\tilde{K}_1\boldsymbol{g}^1, \ldots, \tilde{K}_n\boldsymbol{g}^n\right)$
- *Fusion: if* $g_i \sim \mathcal{P}\left(\boldsymbol{g}_1^i, \ldots, \boldsymbol{g}_n^i\right)$*, then*

$$\prod_{i=1}^{k} g_i \sim \mathcal{P}\left(\bigcirc_{i=1}^k \boldsymbol{g}_1^i, \ldots, \bigcirc_{i=1}^k \boldsymbol{g}_n^i\right).$$

- *Initialisation from leaves: for observation* $v \in E$ *set* $\boldsymbol{g}_{\mathrm{pa}(v),v}^i = \boldsymbol{e}_{\langle v\rangle_i}$ *for* $i \in \{1, \ldots, n\}$.

2. *Sampling under* $\mathbb{P}^\circ$*: if the forward transition on the edge* (s,t) *is represented by* $(K_1(x), \ldots, K_n(x))$*, then with* $g_t = \sim \mathcal{P}\left(\boldsymbol{g}^1, \ldots, \boldsymbol{g}^n\right)$

$$\begin{aligned}\mathbb{P}^\circ(X_t = y \mid X_s = x) &= \prod_{i=1}^{n} \mathbb{P}^\circ\left(\langle X_t\rangle_i = \langle y\rangle_i \mid X_s = x\right) \\ &= \prod_{i=1}^{n} \frac{\langle \boldsymbol{z}_\ell^i \bigcirc \boldsymbol{g}^i\rangle_{\langle y\rangle_i}}{(\boldsymbol{z}_\ell^i)'\boldsymbol{g}^i},\end{aligned}$$

   where $\boldsymbol{z}_\ell^i = K_i(x)'\boldsymbol{e}_\ell$ *is the* ℓ*-th row of* $K_i(x)$.

3. *Computation of the weight: Since* $(Pg)(x) \sim \mathcal{P}\left(K_1(x)\boldsymbol{g}_1, \ldots, K_n(x)\boldsymbol{g}_n\right)$*, the weight at* x *is given by*

$$w(x) = \frac{(Pg)(x)}{(\tilde{P}g)(x)} = \prod_{i=1}^{n} \frac{\langle K_i(x)\boldsymbol{g}_i\rangle_{x_i}}{\langle \tilde{K}_i\boldsymbol{g}_i\rangle_{x_i}}.$$

6.2.3 Backward Diagonalisation

The example of the previous section shows a generic way to deal with interacting particles systems. Here, conditional on the state of all particles at a particular "time", all particles transition independently, though with transition probabilities that may depend on the state of *all* particles. The backward filtering is however done on separate line graphs, thereby fully bypassing the need of a tractable fusion step. We call this backward diagonalisation.

6.3 Line Graph with Independent Gamma Increments

In this example, we consider a process with Gamma distributed increments. We consider a line graph with a single observational leaf v. We write $Z \sim \mathrm{Gamma}(\alpha, \beta)$ if Z has density $\psi(x; \alpha, \beta) = \beta^\alpha \Gamma(\alpha)^{-1} x^{\alpha-1} e^{-\beta x} \mathbf{1}_{(0,\infty)}(x)$.

Choose mappings α_t, $\beta_t \colon (0,\infty) \to [1/C, C]$ for some $C > 0$. We define a Markov process with Gamma increments on $\mathcal{G}$ by

$$X_t - X_s \mid X_s = x \sim \mathrm{Gamma}(\alpha_t, \beta_t(x)), \quad \text{if } s = \mathrm{pa}(t), \qquad X_0 = x_0.$$

This implies

$$h_{\mathrm{pa}(t),t}(x, y) = \psi(y - x; \alpha_t, \beta_t(x)).$$

Note that X can be thought of as a time discretised version of the SDE $\mathrm{d}X_t = \beta^{-1}(X_t)\,\mathrm{d}L_t$ driven by a Gamma process (L_t) with scale parameter 1, observed at final time T. See

Belomestny et al. (2019) for a continuous time perspective on this problem. A statistical application will typically involve multiple observations and henceforth multiple line graphs. Simulation on each line graph corresponds to conditional (bridge) simulation for the process X.

As the h-transform is not tractable, we introduce the process $\tilde{X}$ which is defined as the Markov process with Gamma increments induced by $(\alpha_t, \tilde{\beta})$, with $\tilde{\beta}$ a user specified nonnegative constant (which may depend on the vertex t; we drop this from the notation). This process is tractable and is used to define $\mathcal{G}$.

As before, we state our results using the kernels P and $\tilde{P}$. Define for $A, \beta > 0$

$$g(y) = g(y; A, \beta) = \psi(x_v - y; A, \beta) \tag{21}$$

and write $g \sim \mathcal{P}(A, \beta)$.

Definition 17 *Define the exponentially-tilted Beta-distribution with parameters $\gamma_1, \gamma_2 > 0$ and $\lambda \in \mathbb{R}$ as the distribution with density*

$$q_{\gamma_1,\gamma_2,\lambda}(z) \propto z^{\gamma_1 - 1}(1-z)^{\gamma_2 - 1} e^{-\lambda z} \mathbf{1}_{(0,1)}(z). \tag{22}$$

We denote this distribution by $\mathrm{ExpBeta}(\gamma_1, \gamma_2, \lambda)$.

Sampling from this distribution can be accomplished for example using rejection sampling, with importance sampling distribution $\mathrm{Beta}(\gamma_1, \gamma_2)$.

Theorem 18 *Let $P(x, \mathrm{d}y) = \psi(y-x; \alpha, \beta(x))\,\mathrm{d}y$ and $\tilde{P}(x, \mathrm{d}y) = \psi(y-x; \alpha, \tilde{\beta})\,\mathrm{d}y$. Assume that $g \sim \mathcal{P}(A, \beta)$.*

1. *Computing $\mathcal{G}$:*
   - *Pullback for $\tilde{P}$: $\tilde{P}g \sim \mathcal{P}\left(A + \alpha, \tilde{\beta}\right)$.*
   - *Fusion: need not be defined as we consider a line-graph.*
   - *Initialisation from leaves: $g_{\mathrm{pa}(v),v)}(x) = \psi(x_v - x; A, \tilde{\beta})$.*
2. *Sampling under $\mathbb{P}^\circ$:*
   $$X_s^\circ \mid X_{\mathrm{pa}(s)}^\circ = x \sim x + Z(x_v - x),$$
   where $Z \sim \mathrm{ExpBeta}(\alpha, A, \xi(x))$.
3. *Computation of the weight: the weight at x is given by*
   $$w(x) = \frac{(Pg)(x)}{(\tilde{P}g)(x)} = \left(\frac{\beta(x)}{\tilde{\beta}}\right)^{\alpha} \mathbb{E}\, e^{-\xi(x)Z},$$
   where $Z \sim \mathrm{Beta}(\alpha, A)$ and $\xi(x) = (\beta(x) - \tilde{\beta})(x_v - x)$.

Proof We have

$$\begin{aligned}(Pg)(x) &= \int_x^{x_v} \psi(x_v - y; A, \tilde{\beta})\psi(y - x; \alpha, \beta(x))\,\mathrm{d}y \\ &= \frac{\tilde{\beta}^A \beta(x)^\alpha}{\Gamma(A)\Gamma(\alpha)} e^{-\tilde{\beta}x_v + \beta(x)x} \int_x^{x_v} (x_v - y)^{A-1}(y - x)^{\alpha-1} e^{(\beta-\beta(x))y}\,\mathrm{d}y \\ &= \frac{\beta^A \beta(x)^\alpha}{\Gamma(A)\Gamma(\alpha)} e^{-\beta(x_v - x)}(x_v - x)^{A+\alpha-1} \int_0^1 z^{\alpha-1}(1 - z)^{A-1} e^{-z\xi(x)}\,\mathrm{d}z \\ &= \frac{\beta^A \beta(x)^\alpha}{\Gamma(A)\Gamma(\alpha)} e^{-\tilde{\beta}(x_v - x)}(x_v - x)^{A+\alpha-1} B(\alpha, A)\mathbb{E}e^{-\xi(x)Z}\end{aligned}$$

where we made the substitution $y = x + z(x_v - x)$ at the third equality and $\mathrm{B}(\alpha, \beta)$-denotes the Beta-function, evaluated at (α, β). Using the definition of g we obtain

$$(Pg)(x) = g(x; A + \alpha, \tilde{\beta}) \left(\frac{\beta(x)}{\tilde{\beta}}\right) \mathbb{E}e^{-\xi(x)Z}.$$

Upon taking $\xi(x) = \tilde{\xi}$ we get $\tilde{P}g = g(\cdot; A + \alpha, \tilde{\beta})$. This also directly gives the expression for the weight.

Sampling from $\mathbb{P}^\circ$ over an edge entails drawing from a density which is proportional to $g(y)P(x, \mathrm{d}y)$ (proportionality with respect to y). The claim now follows upon inspecting the derivation of $(Pg)(x)$ and keeping all terms under the integral proportional to z.

■

Note that the expression for the weight immediately suggests a method to estimate $w(x)$ unbiasedly. The setting we consider here is restricted to a line graph though, as fusion of g_1 and g_2 of the form (21) does not lead to a fused function of the same form.

The model can be extended to n Markov processes with Gamma increments, which evolve conditionally independent, where it is assumed that the forward evolution consists of composing kernels

$$P(\boldsymbol{x}, \mathrm{d}\boldsymbol{y}) = \prod_{i=1}^n \psi(\boldsymbol{y}_i - \boldsymbol{x}_i; \alpha, \beta(\boldsymbol{x}))\,\mathrm{d}\boldsymbol{y}.$$

The backward kernel $\tilde{P}$ is then taken to be the same, with $\beta(\boldsymbol{x})$ replaced with β. This is an instance of backward diagonalisation.

6.4 Model by Ju et al. (2021)

Consider N particles (agents) taking values in $\{0, 1\}$. Let $x \in E := \{0, 1\}^N$. One of the models put forward in Ju et al. (2021) is a discrete-time hidden Markov model, where the hidden Markov process has transition probabilities given by

$$P(x, y) = \prod_{i=1}^N \alpha_i(x)^{y_i}(1 - \alpha_i(x))^{1-y_i}, \qquad x, y \in E \tag{23}$$

where

$$\alpha_i(x) = (\lambda_i a_i(x))^{1-x_i}\,(1 - \gamma_i)^{x_i}\,, \qquad \lambda_i \in (0, 1),\ \gamma_i \in (0, 1). \tag{24}$$

Here, with $\mathcal{N}_i$ denoting the indices of the neighbours of particle i,

$$\alpha_i(x) = |\mathcal{N}_i|^{-1} \sum_{m \in \mathcal{N}_i} \mathbf{1}\{x_m = 1\}$$

is the fraction of neighbours of particle i that take value 1. The parameters λ_i and γ_i can be further parametrised by a common parameter θ but are considered fixed in our discussion. The transition probabilities can be summarised by the table

	$y_i = 0$	$y_i = 1$
$x_i = 0$	$1 - \lambda_i a_i(x)$	$\lambda_i a_i(x)$
$x_i = 1$	γ_i	$1 - \gamma_i$

A typical application consists of particles being individuals, with 0 and 1 encoding "susceptible" and "infected" respectively. Furthermore, individual i has its own parameters λ_i and γ_i, which may depend on its characteristics. The probability of a susceptible individual becoming infected increases according to the fraction of individuals in its neighbourhood that are infected. Let

$$I(x) = \sum_{i=1}^{N} \mathbf{1}\{x_i = 1\}$$

denote the total number of infected individuals in the population. At time k, $k \in \{0, 1, \ldots, n\}$, we assume to observe a realisation from the random variable $V_k \sim \mathrm{Bin}(I(x), \rho)$, where x is the state at time k and the parameter ρ can be interpreted as a "reporting probability".

As the dimension of E grows exponentially with N, exact backward filtering is not tractable. Following Ju et al. (2021), we now explain how to overcome this problem within the framework of BFFG. We will parametrise the functions $g\colon E \to \mathbb{R}$ by $g = \psi \circ I$ where $\psi\colon \{0, 1, \ldots, N\} \to \mathbb{R}$. As the domain of ψ is finite, we can identify g with the vector $\boldsymbol{\psi} \in \mathbb{R}^{N+1}$, where $\boldsymbol{\psi}_i := \psi(i)$. Thus $g \sim \mathcal{P}(\boldsymbol{\psi})$

The pullback operation is given by $(Pg)(x) = \sum_{y \in \{0,1\}^N} P(x, y) g(y)$. The computational cost grows exponentially in N. For this reason, consider the "simpler" Markov kernel

$$\tilde{P}(x, y) = \prod_{i=1}^{N} \tilde{\alpha}(x_i, I(x))^{y_i} \left(1 - \tilde{\alpha}(x_i, I(x))\right)^{1 - y_i}, \tag{25}$$

where for $s \in \{0, 1, \ldots, N\}$

$$\tilde{\alpha}(0, s) = \tilde{\lambda} N^{-1} s, \qquad \tilde{\alpha}(1, s) = 1 - \bar{\gamma}. \tag{26}$$

Note that under $\tilde{P}$ all particles evolve conditionally independently with either probability $\tilde{\alpha}(0, I(x))$ or $\tilde{\alpha}(1, I(x))$. Crucially for what follows, these probabilities depend on x only via $I(x)$.

Definition 19 *Suppose $U_k \sim Ber(\theta_k)$, $1 \le k \le N$. If $U_1, \ldots, U_n$ are independent, then the random variable $U = \sum_{k=1}^{N} U_k$ is said to have the PoiBin$(\theta_1, \ldots, \theta_N)$ distribution. Its probability mass function is given by*

$$\mathbb{P}(U = \ell) = \sum_{x \in \{0,1\}^N} \mathbf{1}_{\{\sum_{i=1}^N x_i = \ell\}} \prod_{i=1}^{N} \theta_i^{x_i} (1 - \theta_i)^{1 - x_i}, \qquad 0 \le \ell \le N.$$

Theorem 20 *Let P be defined by (23)—(24) and $\tilde{P}$ by (25)—(26).*

1. *Computing $\mathcal{G}$:*

   - *Pullback for $\tilde{P}$: if $g \sim \mathcal{P}(\boldsymbol{\psi}_1)$, then $\tilde{P}g \sim \mathcal{P}(\boldsymbol{\psi}_2)$ with $\boldsymbol{\psi}_2$ determined by*

   $$\psi_2(x) = \mathbb{E}\psi_1(Z(x)),$$

   where $Z(x)$ is distributed as $Z_0(x)+Z_1(x)$, with $Z_0(x) \sim Bin(N-I(x), \tilde{\alpha}(0, I(x)))$ and $Z_1(x) \sim Bin(I(x), \tilde{\alpha}(1, I(x))$. Thus,

   $$\psi_2(x) = \sum_{k=0}^{N} \psi_1(k) \sum_{j=0}^{k} \binom{N - I(x)}{k-j} \tilde{\alpha}(0, I(x))^{k-j} \times (1 - \tilde{\alpha}(0, I(x))^{N-I(x)-k+j} \binom{x}{j} \tilde{\alpha}(1, I(x))^{j} (1 - \tilde{\alpha}(1, I(x))^{I(x)-j}.$$

   This agrees with Equation (32) in Ju et al. (2021).

   - *Fusion: Suppose $g_i \sim \mathcal{P}(\boldsymbol{\psi}_i)$, $i = 1, \ldots, k$. Then*

   $$\prod_{i=1}^{k} g_i = \prod_{i=1}^{k} \psi_i \circ I \sim \mathcal{P}\left(\bigcirc_{i=1}^{k} \boldsymbol{\psi}_i\right),$$

   where $\bigcirc$ denotes the Hadamard (entrywise) product.

   - *Initialisation from leaves: If v is observed at a leaf node (i.e. an observation), then $g_{\mathrm{pa}(v),v}(x) = \psi_v(I(x)) \sim \mathcal{P}(\boldsymbol{\psi}_v)$ with*

   $$\boldsymbol{\psi}_v(i) = \binom{i}{v} \rho^v (1-\rho)^{i-v} \mathbf{1}_{\{v \le i \le N\}}, \qquad i = 0, \ldots, N$$

   Here, if $i < v$, we can define $\binom{i}{v}$ arbitrarily, as the indicator ensures that $\boldsymbol{\psi}_v(i) = 0$ in that case (alternatively, define $\binom{i}{v} = 0$ for $i < v$ and drop the indicator).

2. *Sampling under $\mathbb{P}^\circ$: Given state x, transition kernel P and $g \sim \mathcal{P}(\boldsymbol{\psi})$, sampling from the guided process can be done in two steps:*

   (a) sample $j^\star$ from $\{0, \ldots, N\}$ with probabilities

   $$p(j) = \frac{\sum_{z:\, I(z)=j} P(x,z) \boldsymbol{\psi}_j}{\sum_k \sum_{z:\, I(z)=k} P(x,z) \boldsymbol{\psi}_k},$$

   (b) sample y conditional on $j^\star$ according to

   $$p(y \mid j^\star) = \frac{P(x,y) \mathbf{1}\{I(y) = j^\star\}}{\sum_{z \in E} P(x,z) \mathbf{1}\{I(z) = j^\star\}}, \qquad y \in E.$$

   Here, we have dropped dependence of y and $j^\star$ on x in the notation. This agrees with Equation (38) in Ju et al. (2021).

3. *Computation of the weight: if* $g \sim \mathcal{P}(\boldsymbol{\psi})$*, then the weight at* x *is given by* $w(x) = (Pg)(x)/(\tilde{P}g)(x)$*, where*

$$(Pg)(x) = \mathbb{E}\boldsymbol{\psi}_{U(x)},$$

and $U(x) \sim PoiBin(\alpha_1(x), \ldots, \alpha_N(x))$*. This agrees with Equation (37) in Ju et al. (2021).*

Remark 21 *Note that when initialising from the leaves that if* v *is close to* N*, most* $\boldsymbol{\psi}_i$ *are zero. This sparsity propagates through the backward filtering, as seen from the pullback and fusion steps.*

Remark 22 *Sampling from* $y \mid j^\star$ *entails sampling from independent, non-identically distributed Bernoulli variables, conditional on their sum (this is denoted the CondBer-distribution in Ju et al. (2021).*

Proof We first derive the expression for the pullback. We have

$$\begin{aligned}(\tilde{P}g)(x) &= \sum_{y\in\{0,1\}^N} \tilde{P}(x,y)\psi_1(I(y)) \\ &= \prod_{i=1}^N \tilde{\alpha}(x_i, I(x))^{y_i} \left(1 - \tilde{\alpha}(x_i, I(x))\right)^{1-y_i} \psi_1(I(y)) \\ &= \prod_{i=1}^N \tilde{\alpha}(0, I(x))^{y_i} (1 - \tilde{\alpha}(0, I(x))^{1-y_i} \psi_1(I(y)),\end{aligned}$$

Hence

$$(\tilde{P}g)(x) = \mathbb{E}\,\psi_1(Z(x)) = \sum_{k=0}^N \psi_1(k)\mathbb{P}(Z(x) = k)$$

where $Z(x)$ is the number of successes in N independent Bernoulli trials, where the i-th Bernoulli trial has success probability $A_i := \tilde{\alpha}(x_i, I(x))$. That is, $Z \sim PoisBin\left(\{A_i\}_{i=1}^N\right)$. Now note that A_i can only takes values $\tilde{\alpha}(0, I(x))$ and $\tilde{\alpha}(1, I(x))$. Hence, we consider $I(x)$ Bernoulli random variables with success probability $\tilde{\alpha}(1, I(x))$, and $N - I(x)$ Bernoulli random variables with success probability $\tilde{\alpha}(0, I(x))$. Then indeed $Z(x)$ is distributed as the sum of $Z_0(x)$ and $Z_1(x)$. The final expression simply follows from the probability mass function for the convolution of two discrete random variables.

The fusion operation follows directly.

The expression for the initialisation from the leaves follows directly from $V \sim \mathrm{Bin}(I(x), \rho)$.

To sample the guided process,

$$P^\circ(x,y) \propto P(x,y)g(y) = \frac{P(x,y)\psi(I(y))}{\sum_{z\in E} P(x,z)\psi(I(z))}$$

Then note that (the summation is over $j \in \{0, \ldots, N\}$ and $k \in \{0, \ldots, N\}$),

$$\begin{aligned} p(y) &= \sum_j p(y \mid j)p(j) = \sum_j \frac{P(x,y)\mathbf{1}\{I(y)=j\}}{\sum_{z\in E} P(x,z)\mathbf{1}\{I(z)=j\}} \frac{\sum_{z:\, I(z)=j} P(x,z)\psi(j)}{\sum_k \sum_{z:\, I(z)=k} P(x,z)\psi(k)} \\ &= \frac{1}{\sum_k \sum_{z:\, I(z)=k} P(x,z)\psi(k)} \sum_j \frac{P(x,y)\sum_{z:\, I(z)=j)} P(x,z)\psi(j)\mathbf{1}\{I(y)=j\}}{\sum_{z\in E} P(x,z)\mathbf{1}\{I(z)=j\}} \\ &= \frac{P(x,y)\psi(I(y))}{\sum_z P(x,z)\psi(I(z))} \sum_j \frac{\sum_{z:\, I(z)=j)} P(x,z)\mathbf{1}\{I(y)=j\}}{\sum_{z\in E} P(x,z)\mathbf{1}\{I(z)=j\}}. \end{aligned}$$

Note that the last term on the third line equals 1, since

$$\sum_j \mathbf{1}\{I(y)=j\} \frac{\sum_{z:\, I(z)=j)} P(x,z)}{\sum_{z:\, I(z)=j} P(x,z)} = 1.$$

Therefore, $p(y) = P^\circ(x,y)$.

Finally, we have

$$\begin{aligned} (Pg)(x) &= \sum_{y\in\{0,1\}^N} P(x,y)g(y) \\ &= \sum_{y\in\{0,1\}^N} \prod_{i=1}^N \alpha_i(x)^{y_i}(1-\alpha_i(x))^{1-y_i}\psi(I(y)) \\ &= \sum_{k=0}^N \sum_{y\in\{0,1\}^N} \mathbf{1}_{\{\sum_{i=1}^N y_i = k\}} \prod_{i=1}^N \alpha_i(x)^{y_i}(1-\alpha_i(x))^{1-y_i}\psi\left(\sum_{i=1}^N y_i\right) \\ &= \sum_{k=0}^N \boldsymbol{\psi}_k \sum_{y\in\{0,1\}^N} \mathbf{1}_{\{\sum_{i=1}^N y_i = k\}} \prod_{i=1}^N \alpha_i(x)^{y_i}(1-\alpha_i(x))^{1-y_i} \\ &= \sum_{k=0}^N \mathbb{P}(U(x)=k)\boldsymbol{\psi}_k. \end{aligned}$$

■

We refer to Ju et al. (2021) for methods to efficiently perform the steps in Theorem 20 and application within the context of controlled Sequential Monte Carlo (Heng et al. (2020)).

7. Examples of Guided Processes for Continuous Edges

7.1 Stochastic Differential Equations

Assume a directed tree where on each edge $e = (s,t)$ not pointing to a leaf vertex, the Markov process X is defined as the solution to the stochastic differential equation (SDE)

$$\mathrm{d}X_u = b(u, X_u)\,\mathrm{d}u + \sigma(u, X_u)\,\mathrm{d}W_u, \qquad u \in [0, \tau_e] \tag{27}$$

As transition densities are not known in closed form, computing the BIF is infeasible. However, for the SDE

$$\mathrm{d}\tilde{X}_u = (B(u)\tilde{X}_u + \beta(u))\,\mathrm{d}u + \tilde{\sigma}(u)\,\mathrm{d}W_u \tag{28}$$

this is possible. Hence, suppose that on each edge e we choose the triplet of maps $(B_e, \beta_e, \tilde{\sigma}_e)$. If follows from the results in Mider et al. (2021) that for $u \in (0, \tau_e]$ we have $g_u(x) = \exp\left(c(u) + F(u)'x - \frac{1}{2}x'H(u)x\right)$, for scalar-valued c, vector-valued F and matrix-valued H. As such, we write $g_u \sim \mathcal{P}\left(c(u), F(u), H(u)\right)$.

Theorem 23 *Assume a directed tree with leaf observatrions $x_v \mid x_{\mathrm{pa}(v)} \sim N(L_v x_{\mathrm{pa}(v)}, \Sigma_v)$. On each edge e, the process evolves according to* (27)*.*

1. *Computing $\mathcal{G}$:*

   - *Pullback for $\tilde{P}$: if on an edge $e = (s,t)$ we are given $g_t \sim \mathcal{P}\left(c(t), F(t), H(t)\right)$, then for $u \in (0, \tau_e]$, $g_u \sim \mathcal{P}\left(c(u), F(u), H(u)\right)$, where $(c(u), F(u), H(u))$ are solved backwards from*

$$\begin{aligned} \mathrm{d}H(u) &= \left(-B(u)'H(u) - H(u)B(u) + H(u)\tilde{a}(u)H(u)\right)\,\mathrm{d}u, \\ \mathrm{d}F(u) &= \left(-B(u)'F(u) + H(u)\tilde{a}(u)F(u) + H(u)\beta(u)\right)\,\mathrm{d}u, \\ \mathrm{d}c(u) &= \left(\beta(u)'F(u) + \frac{1}{2}F(u)'\tilde{a}(u)F(u) - \frac{1}{2}\mathrm{tr}\left(H(u)\tilde{a}(u)\right)\right)\,\mathrm{d}u, \end{aligned} \tag{29}$$

     subject to $(c(t), F(t), H(t))$. Here, $\tilde{a} = \tilde{\sigma}\tilde{\sigma}'$.

   - *Fusion: if $g_i \sim \mathcal{P}\left(c_i, F_i, H_i\right)$, then $\prod_i g_i \sim \mathcal{P}\left(\sum_i c_i, \sum_i F_i, \sum_i H_i\right)$.*

   - *Initialisation from leaves:*

$$g_{\mathrm{pa}(v),v} \sim \mathcal{P}\left(\log \varphi(v; 0, \Sigma_v), v'\Sigma_v^{-1}L_v, L_v'\Sigma_v^{-1}L_v\right). \tag{30}$$

2. *Sampling under $\mathbb{P}^\circ$: on the edge $e = (s,t)$ the guided process evolves according to the SDE*

$$\mathrm{d}X_u^\circ = \left(b(u, X_u^\circ) + a(u, X_u^\circ)(F(u) - H(u)X_u^\circ)\right)\,\mathrm{d}u + \sigma(u, X_u^\circ)\,\mathrm{d}W_u, \qquad u \in [0, \tau_e], \tag{31}$$

   subject to $X_0^\circ = x$, with x the state of the process at vertex s. Here, $a = \sigma\sigma'$.

3. *Computation of the weight:*

$$\log w_t(X^\circ) = \int_0^{\tau_e} \frac{(\mathcal{L} - \tilde{\mathcal{L}})g}{g}(u, X_u^\circ)\,\mathrm{d}u. \tag{32}$$

Proof This is a consequence of Theorem 2.5 in Mider et al. (2021). ■

Remark 24 *The weight is obtained by integrating*

$$\frac{(\mathcal{L}-\tilde{\mathcal{L}})g}{g} = \sum_i (b_i - \tilde{b}_i)\frac{\partial_i g}{g} + \frac{1}{2}\sum_{i,j}(a_{ij}-\tilde{a}_{ij})\frac{\partial^2_{ij} g}{g}$$

over $[0, \tau_e]$. *Here, we have omitted arguments* (u, X°_u) *from the functions and* ∂_i *denotes the partial derivative with respect to* x_i. *If we set* $r_i = (\partial_i g)/g$, *then* $(\partial_{ij} g)/g = \partial_j r_i + r_i r_j$ *and therefore*

$$\frac{(\mathcal{L}-\tilde{\mathcal{L}})g}{g} = \sum_i (b_i - \tilde{b}_i) r_i + \frac{1}{2}\sum_{i,j}(a_{ij}-\tilde{a}_{ij})\left(\partial_j r_i + r_i r_j\right).$$

This expression coincides with the expression for the likelihood given in Proposition 1 of Schauer et al. (2017), but the proof given here is much shorter.

Remark 25 *Note that the solving the ODEs in the pullback operation scales cubically in the dimension of the diffusion process. Improved scaling can be obtained by introducing sparsity in* B *and/or* $\tilde{\sigma}$.

Remark 26 *The literature on filtering and smoothing of partially observed processes defined by stochastic differential equations is vast, see for instance Delyon and Hu (2006), Särkkä and Sottinen (2008), Chopin et al. (2023), Mider et al. (2021) and references therein.*

To relate guide processes along edges as used here to this literature, we make the connection to Särkkä and Sottinen (2008) explicit. For stochastic differential equations, the weight in (32) *can also be obtained from Girsanov's theorem. Let* $r(u,x) = F(u) - H(u)x$. *The derivation in Schauer et al. (2017) applies Girsanov's theorem directly and obtains*

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\mathbb{P}^\circ}(X^\circ) = \exp\left(-\int_0^{\tau_e} r_u' \sigma_u \,\mathrm{d}W^\circ_u - \frac{1}{2}\int_0^{\tau_e} r_u' a_u r_u \,\mathrm{d}u\right),$$

where W° *is a* $\mathbb{P}^\circ$*-Wiener process and the subscript* u *denotes "evaluated in* (u, X°_u)*". Särkkä and Sottinen (2008) likewise rely on Girsanov's theorem to obtain the weight used in their Algorithm 2.1. Their expression becomes identical to the one above upon identifying their* f, g, Q, L, B *with* b, $b + ar$, I, σ, σ *respectively. This ensures their proposal process* $s^\star$ *is alike* X° *defined here. Under these identifications, the process* θ *used in the proof of Theorem 3 in their appendix becomes simply* $-\sigma' r$. *The result follows since their weight equals the stochastic exponential of* $\int \theta(s)\beta(s)$, *with (their notation)* β *denoting Brownian motion. Note that contrary to our result, the derivations in Särkkä and Sottinen (2008) assume that* σ *does not depend on the state. Then again, Särkkä and Sottinen (2008) do not rely on the specific choice of drift function* $b + ar$ *in their importance sampling process.*

7.2 Continuous-time Markov Chains with Countable State Space

Let X denote a continuous time Markov process taking values in the countable set E. Let $\mathcal{Q} = (q(x,y),\, x, y \in E)$ be a Q-matrix, i.e. its elements $q(x,y)$ satisfy $q(x,y) \geq 0$ for all $x \neq y$ and $\sum_y q(x,y) = 0$. Define $c(x) = -q(x,x)$. If $\sup_x c(x) < \infty$, then $\mathcal{Q}$ uniquely defines

a continuous-time Markov chain with values in E. Its transition probabilities $p_u(x,y)$ are continuously differentiable in u for all x and y and satisfy Kolmogorov's backward equations:

$$\frac{\partial}{\partial u}p_u(x,y) = \sum_x q(x,z)p_u(z,y)$$

(cf. Liggett (2010), Chapter 2, in particular Corollary 2.34). The infinitesimal generator acts upon functions $f\colon E \to \mathbb{R}$ by

$$\mathcal{L}f(x) = \sum_{y\in E} q(x,y)\,(f(y)-f(x)) = \sum_{y\in E} q(x,y)f(y), \qquad u\in[0,\infty), \quad x\in E. \tag{33}$$

In the specific setting where E is finite with states labeled by $1,2,\ldots,R$, $\mathcal{Q}$ is an $R\times R$-matrix. Then, by evaluating $\mathcal{L}f(x)$ for each $x\in E$ we can identify $\mathcal{L}f$ by $\mathcal{Q}\boldsymbol{f}$, where $\boldsymbol{f}$ is the column vector with i-th element $\boldsymbol{f}_i = f(i)$.

Next, we explain how the law of the process X changes under the h-transform. Thus, suppose $h\colon [0,\infty)\times E\to\mathbb{R}$ is specified. This function can be used to define a change of measure as detailed in Section 4.2. It induces a guided process X° for which the generator of the space-time process can be derived by combining Equation (15) and the preceding display. Some simple calculations reveal that the generator of X° acts upon functions $f\colon [0,\infty)\times E\to\mathbb{R}$ as

$$(\mathcal{L}^\circ_u f)(x) = \sum_{y\in E} q(x,y)(f(y)-f(x))\frac{h(u,y)}{h(u,x)}. \tag{34}$$

Therefore, comparing to (33), we see that the guided process X° is a time-inhomogenuous Markov process X° with time dependent generator matrix $\mathcal{Q}^\circ_u = (q^\circ_u(x,y),\, x,y\in E)$, where

$$q^\circ_u(x,y) = q(x,y)\frac{h(u,y)}{h(u,x)} \quad \text{if} \quad y\neq x$$

and $q^\circ_u(x,x) = 1-\sum_{y\neq x} q^\circ_u(x,y)$.

For finite-state Markov chains transition densities can be computed via $(p_u(x,y),\, x,y\in E) = e^{u\mathcal{Q}}$ and this would immediately give h, implying no need for constructing a guided process via an approximation to Doob's h-transform. While strictly speaking this is true, in case $|E|$ is large, numerically approximating the matrix exponential can be computationally demanding. In that case, by simplifying the dynamics of the Markov process, one may choose an approximating continuous time Markov process $\tilde{X}$ (on E) where $\tilde{\mathcal{Q}}$ is block diagonal, as this simplifies evaluation of matrix exponentials.

Hence, let $\tilde{X}$ be a continuous time Markov process on E with Q-matrix $\tilde{\mathcal{Q}} = (\tilde{q}(x,y),\, x,y\in E)$. Then the guided process has Q-matrix $\mathcal{Q}^\circ = (q^\circ(x,y),\, x,y\in E)$, where for $y\neq x$, $q^\circ_u(x,y) = q(x,y)g(u,y)/g(u,x)$. If the guided process is simulated on an edge e, its log-weight equals

$$\int_0^{\tau_e} \frac{(\mathcal{L}-\tilde{\mathcal{L}})g}{g}(u,X^\circ_u)\,\mathrm{d}u = \int_0^{\tau_e} \frac{\sum_{y\in E}(q(X^\circ_u,y)-\tilde{q}(X^\circ_u,y))g(u,y)}{g(u,X^\circ_u)}\,\mathrm{d}u.$$

We summarise the key ingredients of BFFG in case the state space is given by $E = \{1,\ldots,R\}$. Suppose $g\colon E\to\mathbb{R}$. Then we can idenfity $g\sim\mathcal{P}(\boldsymbol{g})$, with $\boldsymbol{g}\in\mathbb{R}^R$ given by $\boldsymbol{g}_i = g(i)$.

Theorem 27 *Assume a directed tree, where each edge not ending in a leave node is a continuous edge. On any such edge $e = (s, t)$, the process evolves as a continuous-time Markov chain with generator matrix $\mathcal{Q}$ over time interval $[0, \tau_e]$.*

1. *Computing $\mathcal{G}$:*
   - *Pullback for $\tilde{P}$: if $g_t \sim \mathcal{P}(\boldsymbol{g}_t)$, then for $u \in (0, \tau_e]$, $g_u \sim \mathcal{P}\left(e^{\tilde{\mathcal{Q}}(\tau_e - u)}\boldsymbol{g}_t\right)$.*
   - *Fusion: if $g_i \sim \mathcal{P}(\boldsymbol{g}_i)$, $i = 1, \ldots, k$, then*
   $$\prod_{i=1}^{k} g_i \sim \mathcal{P}\left(\bigcirc_{i=1}^{k} \boldsymbol{g}_i\right),$$
   where $\bigcirc$ denotes the Hadamard (entrywise) product.
   - *Initialisation from leaves: Upon observing state i at a leaf v set $g_{\mathrm{pa}(v),v} \sim \mathcal{P}(\boldsymbol{e}_i)$, where $\boldsymbol{e}_i$ is the ith unit vector in $\mathbb{R}^R$.*
2. *Sampling under $\mathbb{P}^\circ$: on the edge e the guided process has time dependent generator matrix $\mathcal{Q}_u^\circ = (q_u^\circ(x, y),\, x, y \in E)$, where*
   $$q_u^\circ(x, y) = q(x, y)\frac{g(u, y)}{g(u, x)} \quad \text{if} \quad y \neq x$$
   and $q_u^\circ(x, x) = 1 - \sum_{y \neq x} q_u^\circ(x, y)$.
3. *Computation of the weight:*
   $$\log w_t(X^\circ) = \int_0^{\tau_e} \frac{\sum_{y \in E}(q(X_u^\circ, y) - \tilde{q}(X_u^\circ, y))g(u, y)}{g(u, X_u^\circ)}\,\mathrm{d}u.$$

7.3 Model by Stoltz et al. (2021)

Consider N diploid individuals (assume red/green, or, more typically, alleles a and A). At time i, let X_i denote the number of red alleles. The classical Wright-Fisher model with mutation is given by

$$X_{i+1} \mid X_i \sim Bin\left(2N, \frac{(1-u)X_i}{2N} + \frac{v(1-X_i)}{2N}\right),$$

where u is probability of mutating from red to green and v is the probability of mutating from green to red. A diffusion approximation (using a rescaling of time) gives

$$\mathrm{d}X_u = (\beta_1(1 - X_u) + \beta_2 X_u)\,\mathrm{d}t + \sqrt{X_u(1 - X_u)}\,\mathrm{d}W_u, \tag{35}$$

where $\beta_1 = 2Nu$, $\beta_2 = 2Nv$. Denote $b(x) = \beta_1(1 - x) + \beta_2 x$ and $a(x) = x(1 - x)$.

Stoltz et al. (2021) consider a stochastic process evolving on a tree, where along branches the process evolves according to (35) and at the leaves one observes $v_i \sim Bin(n_i, x_i)$. They explain an efficient procedure to compute the backward information filter. The basic idea is

to expand g at any time instance as a linear combination of shifted Chebyshev polynomials. Hence,

$$g(x) = \sum_{k=0}^{K} \lambda_k \psi_k(x), \qquad x \in [0,1], \tag{36}$$

where $\psi_k(x) = T_k(2x-1)$ with $\{T_k\}$ the Chebyshev polynomials of the first kind. Define $\mathcal{S}_k = \text{span}(\psi_0, \dots, \psi_K)$. If g is expanded as in (36), then we write $g \sim \mathcal{P}(\lambda_0, \dots, \lambda_K)$ Alternatively, we can parametrise by the values of g in Chebyshev-Lobatto points on $[0,1]$, which we denote by $x_0, x_1, \dots, x_K$. In this case we write $g \leftrightarrow \bar{\mathcal{P}}(g(x_0), g(x_1), \dots, g(x_K))$. If $g \sim \mathcal{P}(\lambda_0, \dots, \lambda_K)$ is given, it is trivial to obtain $\leftrightarrow \bar{\mathcal{P}}(g(x_0), g(x_1), \dots, g(x_K))$. The reverse operation can be done in $O(K \log K)$-time using the FFT (details are given in in the appendix of Stoltz et al. (2021)). With these facts, we can fully describe the key steps for backwards filtering.

Theorem 28 *Suppose X is a a process on a directed tree evolving according to the SDE (35) over each edge e for time $[0, \tau_e]$.*

1. *Computing $\mathcal{G}$:*

   - *Pullback for $\tilde{P}$: suppose over the edge (s,t) the process X defined by (35) evolves for time $u \in [0, \tau_e]$. If $g_t \sim \mathcal{P}(\lambda_0, \dots, \lambda_K)$, then*

   $$g_u \sim \mathcal{P}(\lambda_0(u), \dots, \lambda_K(u)), \qquad u \in (0, \tau_e],$$

   where $\boldsymbol{\lambda}(u) = [\lambda_0(u), \dots, \lambda_K(u)]$, $u \in (0, \tau_e]$ satisfies the ordinary differential equation

   $$\partial_t \boldsymbol{\lambda}(u) + Q\boldsymbol{\lambda}(u) = 0, \qquad \boldsymbol{\lambda}(\tau_e) = [\lambda_0, \dots, \lambda_K].$$

   The matrix Q is upper-triangular and determined by (38).

   - *Fusion: if $g_i \leftrightarrow \bar{\mathcal{P}}(g_i(x_0), \dots, g_i(x_K))$, then*

   $$\prod_i g_i \leftrightarrow \bar{\mathcal{P}}\left(\prod_i g_i(x_0), \dots, \prod_i g_i(x_K)\right).$$

   - *Initialisation from leaves: Upon observing at a leaf $v \sim Bin(n,x)$ set $g \leftrightarrow \bar{\mathcal{P}}(o(x_0), o(x_1), \dots, o(x_K))$ with $o(x) = \binom{n}{v} x^v (1-x)^{n-v}$.*

2. *Sampling under $\mathbb{P}^\circ$: not considered in Stoltz et al. (2021).*

3. *Computation of the weight: not considered in Stoltz et al. (2021).*

Proof

The pullback operation consists of solving the Kolmogorov backward equation

$$\mathcal{L}g_u(x) + \partial_u g_u(x) = 0, \qquad g_{\tau_e} = g_T, . \tag{37}$$

where $u \in (0, \tau_e]$. The ansatz is that $g_u(x) = \sum_{k=0}^K \lambda_k(u)\psi_k(x)$. Plugging this expression into (37) gives

$$\sum_{k=0}^K \left[\partial_u \lambda_k(u)\psi_k(x) + b(x)\lambda_k(u)\frac{\partial \psi_k(x)}{\partial x} + \frac{1}{2}a(x)\lambda_k(u)\frac{\partial^2 \psi_k(x)}{\partial x^2}\right] = 0.$$

By choice of the drift and diffusivity in (35), for each k, the map $x \mapsto b(x)\frac{\partial \psi_k(x)}{\partial x} + \frac{1}{2}a(x)\frac{\partial^2 \psi_k(x)}{\partial x^2}$ is in $\mathcal{S}_k$. Therefore, there exist coefficients Q_{ik}, $0 \leq i \leq K$ (with $Q_{ik} = 0$ for $i > k$) such that

$$b(x)\frac{\partial \psi_k(x)}{\partial x} + \frac{1}{2}a(x)\frac{\partial^2 \psi_k(x)}{\partial x^2} = \sum_{i=0}^K Q_{ik}\psi_i(x). \tag{38}$$

This implies

$$\sum_{k=0}^K \partial_u \lambda_k(u)\psi_k(x) + \sum_{i=0}^K \sum_{k=0}^K \lambda_k(u) Q_{ik}\psi_i(x) = 0$$

which can be rewritten to

$$\sum_{k=0}^K \left(\partial_u \lambda_k(u) + (Q\boldsymbol{\lambda}(u))_k\right)\psi_k(x) = 0.$$

The result follows by linear independence of the basis functions $\{\psi_k\}_{k=0}^N$.

■

8. Extension to a Directed Acyclic Graph

In this section, we extend our approach from a directed tree to a true DAG. Whereas on the former each vertex has a unique parent vertex, on a DAG a vertex can have multiple parent vertices. Note that on a DAG, contrary to a directed tree, conditioning on the values at leaf-vertices changes the dependency structure (there is a conditional process $X^\star$ defined on the vertex set, but its dependency graph is different from that of X, in particular there is no meaningful transition $P^\star_{\mathrm{pa}(s),s}$ for all s). Also, there is no BIF-type recursion that enables computing the likelihood $h_r(x_r)$. This poses genuine challenges.

We propose to retain the definition of the guided process as in Definition 5. In particular, unlike the conditioned process, the guided process has the same dependency structure as the (unconditional) forward process. Crucially this means forward simulation of the guided process has a similar complexity as forward simulation of the unconditional process. While this enhances automatic translation of sampling unconditional to conditional processes, it need not necessarily lead to computational gains in specific examples when compared to approaches that break the dependency structure.

On a DAG, we can get a similar statement as Theorem 8 by redefining the denominator of the weights for $s \in \mathcal{S}$.

Theorem 29 *Assume maps $x \mapsto g_{s,t}(x)$ are specified for each edge $e = (s,t) \in \mathcal{E}$ and define g_s by fusion, i.e. $g_s(x) = \prod_{t \in \mathrm{ch}(s)} g_{s,t}(x)$, $s \in \mathcal{S}_r$. On a directed acyclic graph, not necessarily a directed tree, the conclusion of Theorem 8 remains valid provided the weight is defined by*

$$\xi_t(x) w_t(x) = \begin{cases} (P_t g_t)(x) & \text{if} \quad t \in \mathcal{S} \\ h_{\mathrm{pa}(t),t}(x) & \text{if} \quad t \in \mathcal{V} \end{cases}, \tag{39}$$

where for $t \in \mathcal{T}$, $\xi_t(x) = \prod_{u \in \mathrm{pa}(t)} g_{u,t}(x_u)$.

Proof By Bayes' theorem

$$\frac{\mathrm{d}\mathbb{P}^\star}{\mathrm{d}\mathbb{P}}(x_{\mathcal{S}}) = \frac{\prod_{v \in \mathcal{V}} h_{\mathrm{pa}(v),v}(x_{\mathrm{pa}(v)})}{h_r(x_r)}.$$

The remainder of the proof is similar to the proof of Theorem 8, except that (14) takes the form

$$\begin{aligned} \prod_{s \in \mathcal{S}} \frac{g_s(x_s)}{\xi_s(x_{\mathrm{pa}(s)})} &= \frac{\prod_{s \in \mathcal{S}} \prod_{t \in \mathrm{ch}(s)} g_{s,t}(x_s)}{\prod_{s \in \mathcal{S}} \prod_{u \in \mathrm{pa}(s)} g_{u,s}(x_u)} = \frac{\prod_{v \in \mathcal{V}} \prod_{u \in \mathrm{pa}(v)} g_{u,v}(x_u)}{\prod_{s \in \mathrm{ch}(r)} g_{r,s}(x_r)} \\ &= \frac{\prod_{v \in \mathcal{V}} \prod_{u \in \mathrm{pa}(v)} g_{u,v}(x_u)}{g_r(x_r)} = \frac{\prod_{v \in \mathcal{V}} \xi_v(x_{\mathrm{pa}(v)})}{g_r(x_r)}. \end{aligned}$$

■

It remains to define g_e for all edges e. When the kernel $P_{\mathrm{pa}(s),s}$ has multiple parent nodes, the function

$$g(x_{\mathrm{pa}(s)}) := (\tilde{P}_{\mathrm{pa}(s),s} g)(x_{\mathrm{pa}(s)})$$

does not, in general, factorize as $\prod_{u \in \mathrm{pa}(s)} g_u(x_u)$. A straightforward, albeit crude, approximation is to define

$$g_u(x_u) = \mathbb{E}\big[\, g(X_{\mathrm{pa}(s)}) \mid X_u = x_u \,\big],$$

although evaluating this expectation is typically intractable.

An alternative is to employ deterministic approximations, as proposed in Lindsten et al. (2018), where SMC methods are developed for general Bayesian networks. In that framework, guiding functions are constructed from deterministic approximations such as those obtained via loopy belief propagation Pearl (1988b) or expectation propagation Opper and Winther (2000); Minka (2001). While neither approach guarantees good performance universally, both have been found effective in many practical settings.

In the present work we adopt the former approach. In contrast to Lindsten et al. (2018), we restrict attention to directed acyclic graphs and fix the dependency structure of the guided process to coincide with that of the forward process. This choice provides a natural means of defining $g_{s,t}$ for each edge (s,t). We first review shortly the message-passing equations for loopy-belief propagation. To simplify notation, we adopt Bayesian notation.

8.1 Loopy-belief Propagation

On a DAG the joint density factorises as $p(x_1, \dots, x_n) = \prod_i p(x_i \mid \mathrm{pa}(x_i))$. Each conditional term $p(x_i \mid \mathrm{pa}(x_i))$ defines a *factor* f_i and associated *potential* ψ_{f_i} by

$$\psi_{f_i}(x_i, \mathrm{pa}(x_i)) := p(x_i \mid \mathrm{pa}(x_i)). \tag{40}$$

Let $\mathcal{F}$ denote the set of factor nodes. For each factor $f \in \mathcal{F}$ its scope is defined as the set of variables it depends on, $\mathrm{scope}(f) \subseteq \mathcal{T}$, where $\mathcal{T}$ denotes the set of variable nodes. Clearly, $\mathrm{scope}(f_i) = \{x_i, \mathrm{pa}(x_i)\}$.

Loopy-belief propagation is an iterative message passing algorithm where it is assumed that the joint density (up to normalisation) factorises as

$$p(x) \propto \prod_{f \in \mathcal{F}} \psi_f(x_{\mathrm{scope}(f)}),$$

which is clearly the case on a DAG. For $t \in \mathcal{T}$, let $\mathrm{ne}(t) = \{f \in \mathcal{F} \colon t \in \mathrm{scope}(f)\}$ be the set of factors connected to variable t. Loopy-belief propagation consists of two types of message passing equations:

Variable-to-Factor Messages: The message from a variable node t to a neighbouring factor node f is the product of all incoming messages from other neighbouring factors:

$$m_{t \to f}(x_t) = \prod_{f' \in \mathrm{ne}(t) \setminus \{f\}} m_{f' \to t}(x_t).$$

Factor-to-Variable Messages: The message from a factor node f to a neighbouring variable node t is defined as a marginalisation over all other variables in the factor's scope:

$$m_{f \to t}(x_t) = \sum_{x_{\mathrm{scope}(f) \setminus \{t\}}} \psi_f(x_{\mathrm{scope}(f)}) \prod_{s \in \mathrm{scope}(f) \setminus \{t\}} m_{s \to f}(x_s).$$

8.2 Guiding Functions based on Loopy-belief Propagation

In the factor graph, each f_i is connected to the child variable x_i and each of its parents $x_j \in \mathrm{pa}(x_i)$. Therefore, an edge in the DAG connecting from x_j to x_i corresponds exactly to a factor-variable edge (f_i, x_j) in the factor graph. There is exactly one message $m_{f_i \to x_j}$ along that edge in the factor graph. This message carries the "backward" information from the child factor f_i to its parent variable x_j. Hence, every edge connecting x_j to x_i in the DAG supports a *unique* factor-to-variable message $m_{f_i \to x_j}$. This motivates the following definition.

Definition 30 *Suppose e is the edge connecting variable x_j to variable x_i. Let $m_{f_i \to x_j}$ be the unique corresponding factor-to-variable message. We define $g_e = m_{f_i \to x_j}$.*

9. Numerical Examples

We illustrate the backward filtering, forward guiding approach with three numerical examples.

9.1 Diffusion Process on a Tree

We consider directed trees where a along each edge a diffusion process generated by an stochastic differential equation (SDE) evolves. As the transition densities of the process are intractable, we adopt the approach we have outlined:

1. on each of the edges we define a function g by backward filtering;
2. the process X° is forward simulated.

We illustrate the described methods using two examples of SDEs on directed trees. The backward filtering and forward guiding is based on Theorem 23. The experiments and figures were produced using the Hyperiax framework (`https://github.com/computationalevolutionarymorphometry/hyperiax/`).

9.2 An MCMC Algorithm for Parameter Estimation

We start with a low dimensional example where the process evolves on each branch according to the SDE

$$\mathrm{d}X_t = \tanh\cdot\left(\begin{bmatrix}-\theta_0 & \theta_0\\ \theta_1 & -\theta_1\end{bmatrix} X_t\right)\mathrm{d}t + \begin{bmatrix}\sigma_0 & 0\\ 0 & \sigma_1\end{bmatrix}\mathrm{d}W_t, \tag{41}$$

where the dot appearing in the drift means that the tanh function is applied coordinatewise. Here $\boldsymbol{\theta} := (\theta_0, \theta_1, \sigma_0, \sigma_1) \in (0,\infty)^4$. Note that θ_0 is the force at which the first component is drawn towards the second component (and θ_1 vice versa). We forward simulated from the model with parameters $(\theta_0, \theta_1, \sigma_0, \sigma_1) = (0.0, 0.65, 0.1, 0.4)$ on a tree with 5 levels and 121 nodes of which 81 are leaf nodes. The edge lengths are uniform random sampled in the interval $[1.2, 2.2]$. The resulting sampled values for both coordinates are visualised in Figure 2.

We assume, as throughout, that only the values at the leaf-vertices are observed. The leaf observations happens with added uncorrelated Gaussian noise in each component with variance $1e-3$. This is incorporated into the backwards filter by setting $\Sigma_v = 1e-3$ in Equation (30). The tree-structure itself is assumed known though. We aim to estimate the parameters $\boldsymbol{\theta}$. We employ flat priors and use an MCMC-algorithm that iteratively updates the unobserved paths conditional on $\boldsymbol{\theta}$ and the observations, and $\boldsymbol{\theta}$ conditional on the unobserved paths. Elements of $\boldsymbol{\theta}$ were updated using random-walk Metropolis-Hastings steps. The missing paths were updated using the BFFG-algorithm, where $\tilde{X}$ is chosen as in (28), with

$$B = \begin{bmatrix}-\theta_1 & \theta_1\\ \theta_2 & -\theta_2\end{bmatrix} \qquad \beta = \begin{bmatrix}0\\ 0\end{bmatrix} \qquad \tilde{\sigma} = \begin{bmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{bmatrix}.$$

Rather than the missing paths, we updated the innovations driving the SDE using a pre-conditioned Crank-Nicolson (pCN) update—defined in (48)— with $\lambda = 0.9$, for otherwise the resulting chain would be reducible (the pCN updates are just as in Mider et al. (2021), see Roberts and Stramer (2001) for the necessity of updating innovations rather than paths directly). A detailed description of the algorithm used is given in the appendix in Section C. Traceplots after running the algorithm for 20 000 iterations are shown in Figure 3 and corresponding density plots after a burn in of 2000 iterations are shown in Figure 4. The acceptance rate was 0.58. It is remarkable that especially θ_0 and σ_0 can be recovered quite

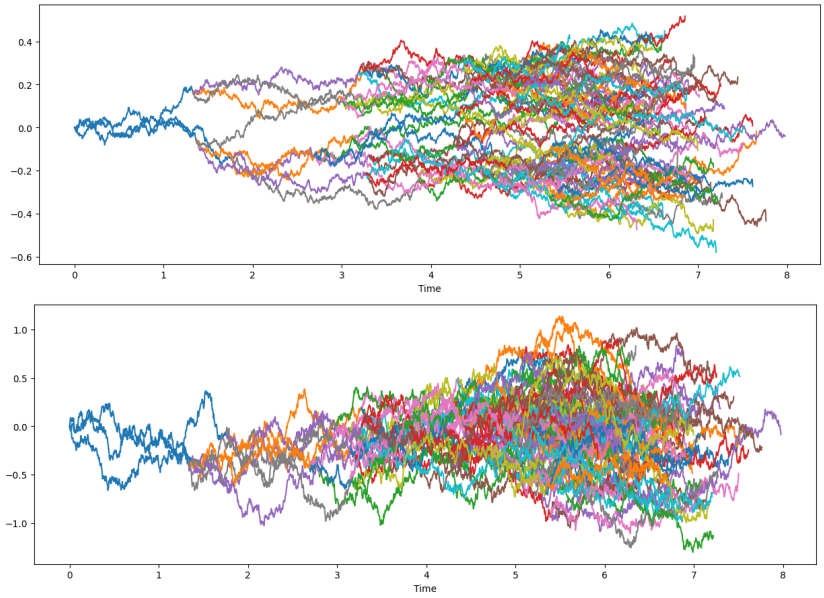

Figure 2: Forward simulated paths from (41) with $(\theta_0, \theta_1, \sigma_0, \sigma_1) = (0.0, 0.65, 0.1, 0.4)$ on a 5 level tree with 121 nodes of which 81 are leaf nodes. Only the values at the leaf nodes are observed.

well. Note that this example serves to illustrates BFFG; the chosen MCMC-sampler for updating the parameter θ is about the simplest choice possible and more efficient proposals and samplers can be exploited.

9.3 Application to shape analysis of butterfly wings

We then proceed to a higher dimensional example where we model evolution of butterfly wing shapes perturbed by a Kunita-like flow. The use of Kunita flows in evolutionary models and application of BFFG for biological data is further explored in Stroustrup et al. (2025), and the theoretical foundation for use of Kunita flows for shape stochastic is further detailed in Sommer et al. (2025). In evolutionary biology, trait evolution is commonly modelled with Brownian motion (Felsenstein (1985)), traditionally for low dimensional data, i.e. small numbers of indiviual traits. Kunita flows provide a way of moving from this case to shape data that posses large numbers of points with high correlation between points. The Kunita flow model can thus be seen as a Brownian motion equivalent for morphological data. Given observations of the morphology at current time, i.e. at the leafs of the phylogenetic tree, we aim to estimate properties of the evolutionary process at prior times. In particular, parameters for the inter-point covariance and node values at prior times, e.g. the morphology at the start of the evolution at the root of the tree. To exemplify this, we consider the outline of butterfly wings represented by point configurations $x = (x_1, \ldots, x_n)$, $x_i \in \mathbb{R}^d$, $d = 2$, and

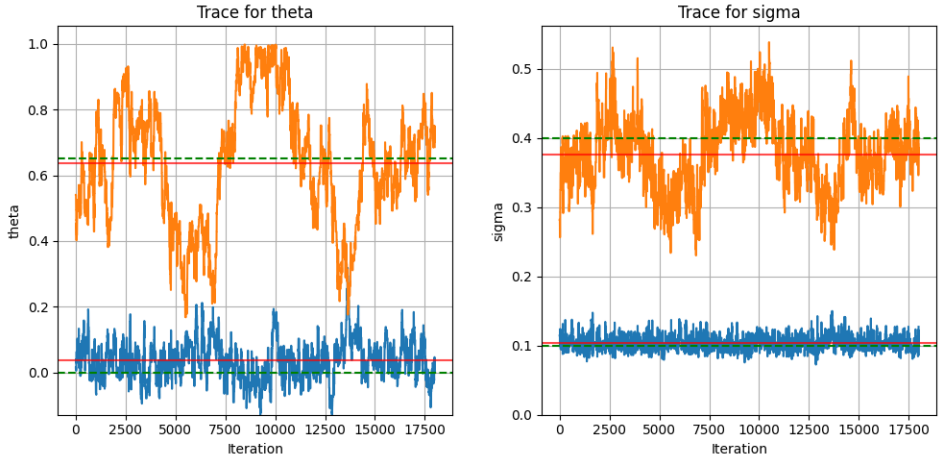

Figure 3: Traceplots for the parameters $\theta_0, \theta_1, \sigma_0, \sigma_1$. Traceplots for θ_0 and σ_0 are in blue; traceplots for θ_1 and σ_1 are in orange. Green horizontal lines indicate true valeus; red horizontal lines show posterior mean after removing burnin samples.

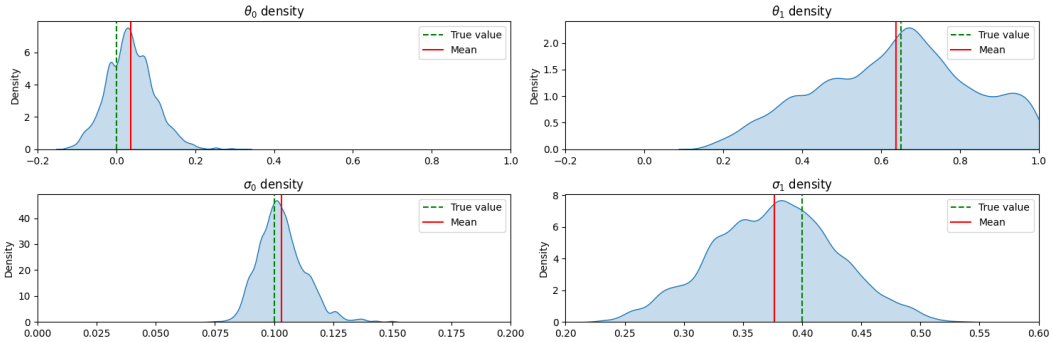

Figure 4: Densities after removing the first 2000 iterations which are considered burnin samples.

the Kunita flow

$$\mathrm{d}x_t^i = \int_{\mathbb{R}^d} k_{\boldsymbol{\theta}}(x_t^i, \zeta)\,\mathrm{d}W_t(\zeta)\,\mathrm{d}\zeta \tag{42}$$

where $k_{\boldsymbol{\theta}}$ is a kernel that describes the diffusivity operator of the Kunita flow in integral form. We let the forward model be a finite dimensional version of this with the SDE

$$\mathrm{d}x_t = K_{\boldsymbol{\theta}}(x_t)\,\mathrm{d}W_t \tag{43}$$

where $K_{\boldsymbol{\theta}}$ is the matrix $[K_{\boldsymbol{\theta}}(x)]_{ij} = k_{\boldsymbol{\theta}}(x_i, x_j)\mathrm{Id}_d$, and W_t is an nd-dimensional Wiener process. We take the kernel $k_{\boldsymbol{\theta}}(x_i, x_j) = \bar{k}_{\boldsymbol{\theta}}(\|x_i - x_j\|)$ where $\bar{k}_{\boldsymbol{\theta}}(r) = \alpha 4(3+3r+r^2)\exp(-r/\sigma)$. Conditioned on the leaf values, the shapes are observed with independent Gaussian noise with variance ϵ on each coordinate so that the parameter vector is $\boldsymbol{\theta} = (\alpha, \sigma, \epsilon)$.

Backwards filtering: For the auxiliary process (28), we backwards filter on each edge $e = (s, t)$ by taking $B \equiv 0$ and $\beta \equiv 0$. On each edge, we redefine $\tilde{\sigma}_e$ iteratively. Let the choice of $\tilde{\sigma}_e$ on edge e in iteration i be denoted by $\tilde{\sigma}_e^{(i)}$, $i \geq 1$. Let $v(u) = H^{-1}(u)F(u)$ $(0 \leq u \leq \tau_e)$ and denote its value in iteration i by $v^{(i)}(u)$. We set $\tilde{\sigma}_e^{(1)} = K\left(v^{(1)}(\tau_e)\right)$ and for $i \geq 1$

$$\tilde{\sigma}^{(i+1)}(u) = \frac{u}{\tau_e} K\left(v^{(i+1)}(\tau_e)\right) + \left(1 - \frac{u}{\tau_e}\right) K\left(v^{(i)}(0)\right).$$

In this way, we aim to iteratively refine the backwards filter.

The tree and observed leaf-node shapes are shown in Figure 5. We then run an MCMC chain to estimate the parameters $\boldsymbol{\theta}$ using Algorithm 1 in Appendix C. The "full conditional" for updating the parameter ϵ is proportional to

$$(2\pi\varepsilon)^{-|\mathcal{V}|/2} \exp\left(-\frac{1}{2\varepsilon}\sum_{v\in\mathcal{V}} \|x_v - x_{\mathrm{pa}(v)}\|^2\right)\pi(\epsilon)$$

Assuming the inverse-Gamma prior with parameter $(A_\varepsilon, B_\varepsilon)$ on ϵ, it is easily seen that the full conditional distribution of ϵ is inverse-Gamma with parameter

$$\left(A_\epsilon + \frac{|\mathcal{V}|}{2}, B_\epsilon + \sum_{v\in\mathcal{V}} \|x_v - x_{\mathrm{pa}(v)}\|^2\right).$$

We took $A_\epsilon = 2$ and $B_\epsilon = 0.005$. Trace plots for $\boldsymbol{\theta}$ are shown in Figure 6. In Figure 7, we show samples from the posterior root of the tree together with the original leaf values, and posterior samples of the parent of the leaves representing butterflies papilio xuthus and papilio zelicaon (two rightmost leaves of the tree in Figure 5).

9.4 Factorial Hidden Markov Model

We consider factorial hidden Markov models (Ghahramani and Jordan (1997)). By exploiting the dependence structure of such a model, it provides an example of a non-tree DAG.

Assume for $i \in \{1, 2, \ldots, n\}$ independent latent Markov chains $\{x_{i,t}\}$, $t \in \{0, 1, \ldots, T\}$. At each time t, assume that y_t is observed conditionally on $\boldsymbol{x}_t := (x_{1,t}, \ldots, x_{n,t})$ with density

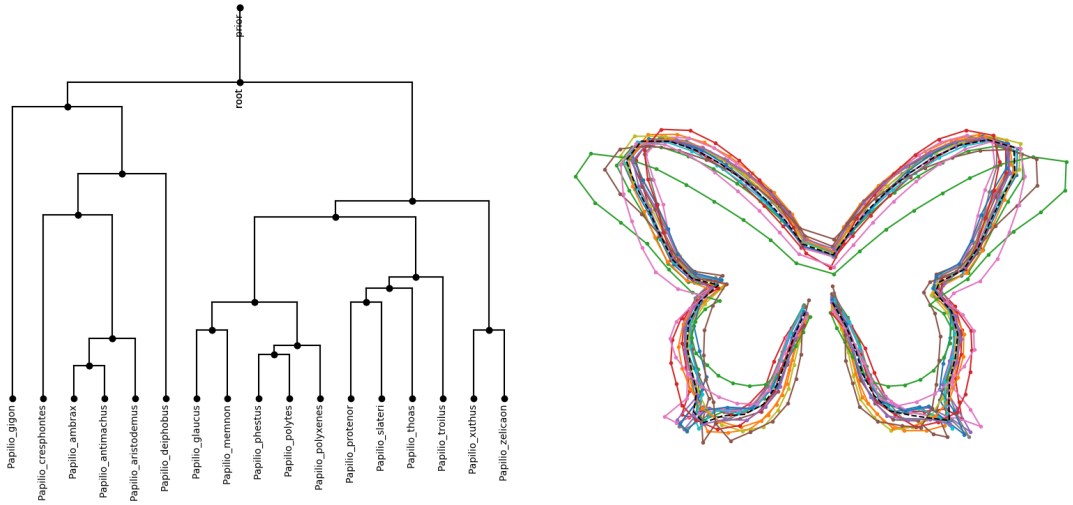

Figure 5: Tree and observed shapes.

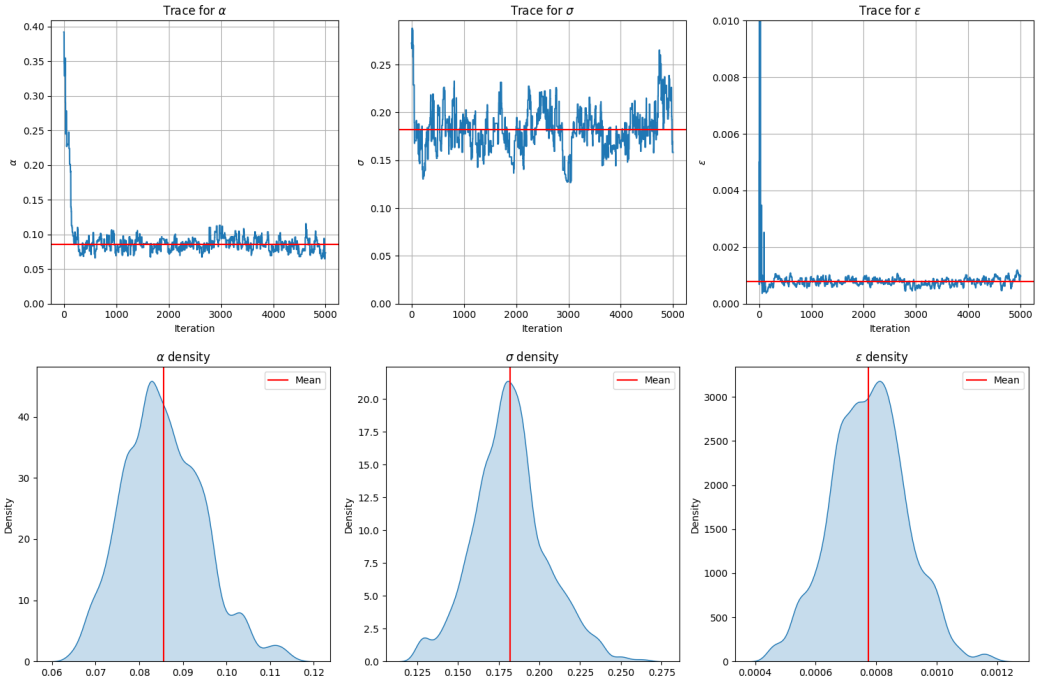

Figure 6: Trace and density plots for the parameters α, σ for the kernel used in the Kunita flow and the observation noise variance ϵ. Red lines show the mean values of the samples.

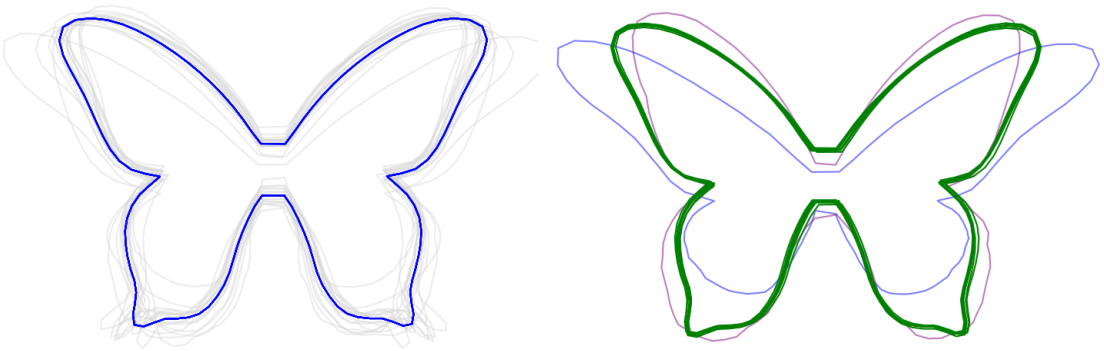

Figure 7: (left) Posterior samples of the root of the tree (leaf values in the background). (right) Posterior samples of the parent of the leaves representing butterflies papilio xuthus and papilio zelicaon (continuous lines).

$p(y_t \mid \boldsymbol{x}_t)$. We have

$$p(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_T \mid y_0, \ldots, y_T) \propto \prod_{t=0}^{T} \left[p(y_t \mid \boldsymbol{x}_t) \prod_{i=1}^{n} p(x_{i,t} \mid x_{i,t-1}) \right],$$

where $p(x_{i,0} \mid x_{i,-1}) \equiv p(x_{i,0})$. We define guiding functions using loopy-belief propagation. For that purpose, denote the vertex corresponding to x_t^i by (i,t). For $(i,t) \in \{1,\ldots,n\} \times \{0,\ldots,T\}$ define the factor $f_{i,t}$ with associated potential

$$\psi_{f_{i,t}}(x_{t-1}^i, x_t^i) = p(x_{i,t} \mid x_{i,t-1}).$$

Define the factors o_t with associated potential

$$\psi_{o_t}(\boldsymbol{x}_t) = p(y_t \mid \boldsymbol{x}_t).$$

It follows that $\text{scope}(f_{i,t}) = \{(i,t-1),(i,t)\}$ and $\text{scope}(o_t) = \cup_{i=1}^n \{(i,t)\}$. Now $\text{ne}((i,t)) = \{f_{i,t}, f_{i,t+1}, o_t\}$. With this notation, we can give the message passing equations from loopy-belief propagation. Note that all messages below, either from or to the variable node (i,t), are functions of $x_{i,t}$.
Messages from (i,t)*:* let $\odot$ denote pointwise multiplication of functions.

$$\begin{aligned} m_{(i,t)\to o_t} &= m_{f_{i,t}\to(i,t)} \odot m_{f_{i,t+1}\to(i,t)} \\ m_{(i,t)\to f_{i,t}} &= m_{o_t\to(i,t)} \odot m_{f_{i,t+1}\to(i,t)} \\ m_{(i,t)\to f_{i,t+1}} &= m_{f_{i,t}\to(i,t)} \odot m_{o_t\to(i,t)} \end{aligned}$$

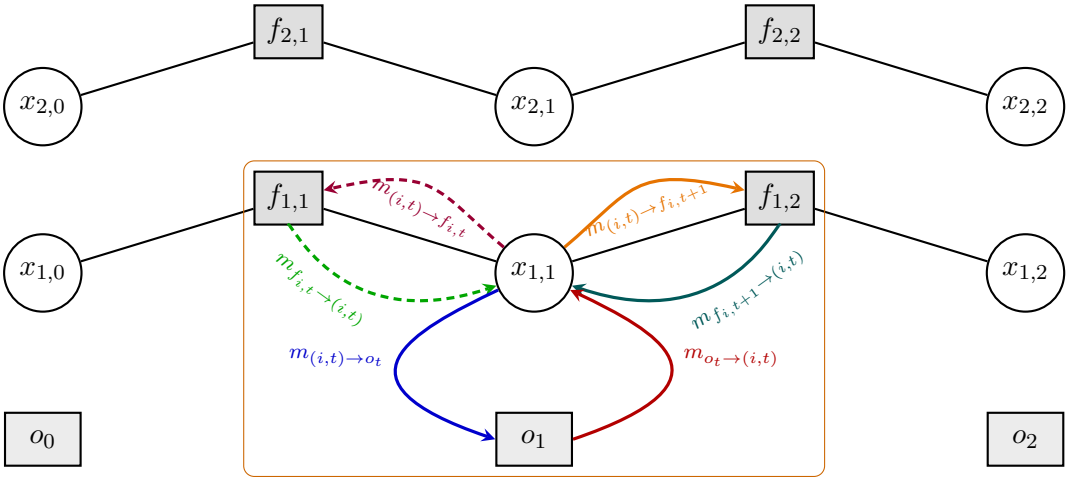

Figure 8: Factorial HMM factor graph (two chains, three time slices) with transition factors $f_{i,t}$ (grey squares) and observation factors o_t (light grey squares). Message directions around $x_{1,1}$ highlight the forward (orange/teal), backward (purple/green), and observation (blue/red) passes used in loopy belief propagation.

Messages to (i,t)*:* let $\boldsymbol{x}_{-i,t} := \boldsymbol{x}_t \setminus \{x_{i,t}\}$.

$$m_{o_t\to(i,t)}(x_{i,t}) = \int p(y_t \mid \boldsymbol{x}_t) \prod_{j\neq i} m_{(j,t)\to o_t}(x_{j,t})\, \mathrm{d}\boldsymbol{x}_{-i,t} \tag{44}$$

$$m_{f_{i,t}\to(i,t)}(x_{i,t}) = \int p(x_{i,t} \mid x_{i,t-1}) m_{(i,t-1)\to f_{i,t}}(x^i_{t-1})\, \mathrm{d}x_{i,t-1} \tag{45}$$

$$m_{f_{i,t+1}\to(i,t)}(x_{i,t}) = \int p(x_{i,t+1} \mid x_{i,t}) m_{(i,t+1)\to f_{i,t+1}}(x_{i,t+1})\, \mathrm{d}x_{i,t+1} \tag{46}$$

Figure 8 illustrates the model and messages.

Guiding functions: denote an edge connecting $x_{i,t}$ and $x_{i,t+1}$ by $(i,t),(i,t+1)$ and an artificially created edge connecting $x_{i,t}$ and y_t by $(i,t),y_t$. We propose the following messages to be sent to the vertex corresponding to x^i_t:

- message from $f_{i,t+1}$: $g_{(i(t),(i,t+1)} = m_{f_{i,t+1}\to(i,t)}$.
- message from o_t: $g_{(i,t),y_t} = m_{o_t\to(i,t)}$.

Note that evaluating $m_{o_t\to(i,t)}(x_{i,t})$ is expensive for large n. In that case, mean-field approximations may be employed to reduce computational complexity. Naturally, there exist other methods for inference in factorial hidden Markov models, see for instance (Ghahramani and Jordan (1997) and Rimella and Whiteley (2022)).

9.5 Electricity Usage Example

As a specific example of the factorial hidden Markov models, we consider an electricity usage example motivated by Kolter and Jaakkola (2012). Assume $x_{i,t} \in \{0,1\}$ and let $t \mapsto x_{i,t}$

model electricity usage for the i-th customer which is either on or off. Assume initial state $x_{i,0} = 0$ and that $p(x_{i,t} \mid x_{i,t-1})$ is governed by the transition matrix

$$\begin{bmatrix} \eta_0 & 1-\eta_0 \\ 1-\eta_1 & \eta_1 \end{bmatrix}$$

At each time instance t we observed a noisy realisation of the accumulated usage:

$$y_t \mid \boldsymbol{x}_t \sim N\left(\sum_{i=1}^{n} \theta_i x_{i,t}, \sigma^2\right).$$

Here, θ_i is the usage of customer i over one time instance, for simplicity assumed to be constant over time. Messages are simply vectors in $\mathbb{R}^2$. Computation of messages from a variable corresponds to entrywise multiplication of two-dimensional vectors. The computations in (45) and (46) are simply matrix-vector multiplications. Thus, all steps are cheap, except for computing $m_{o_t \to (i,t)}$ in (44).

We will assume $\theta_1, \ldots, \theta_n$ to be known and $\eta_0 = \eta_1 =: \eta \in (0,1)$ unknown. Upon employing a uniform prior on η, we aim to estimate η and reconstruct latent paths $t \mapsto x_t^i$.

Computational complexity: running the forward-backward algorithm (aggregating the n latent states at each time instance) results in computational complexity $\mathcal{O}(TK^2)$, where $K = 2^n$. The main computational challenge in applying BFFG with loopy-belief propagation consists of computing messages $m_{o_t \to (i,t)}$ as in (44). Using gray-codes, this can be done with computational complexity $\mathcal{O}(TK)$.

We run a pseudo-marginal Metropolis-Hastings scheme, where the likelihood estimator is obtained via independent importance sampling using samples from the guided process.

To evaluate the performance of the inference algorithm for the coupling parameter η, we conducted a simulation study with three true parameter values ($\eta = 0.2,\ 0.5,\ 0.7$) representing weak, moderate, and strong coupling between the latent chains. For each true value, we generated 10 independent data sets and ran MCMC inference for 20,000 iterations with a burn-in period of 5,000 iterations. The model consisted of $N = 3$ factorial chains of length $T = 50$, with observation noise $\sigma = 5.0$ and $\theta = 10.0,\ 15.0,\ 12.0$. Figure 9 shows the aggregated posterior distributions across all data sets for each true η value. The posteriors are centred around the true parameter values, demonstrating parameter recovery across different coupling strengths. Figure 10 displays representative MCMC trace plots for each scenario

Acknowledgments

We thank Richard Kraaij for discussions on the topic of h-transforms in the early stage of this work and valuable feedback from Marc Corstanje. The data for the butterfly example was kindly provided and preprocessed by Michael Baand Severinsen. We thank Frank Schäfer for preliminary simulation experiments in Julia language (`MitosisStochasticDiffEq.jl`-package) for the first experiment in Section 9.1. We are grateful to two anonymous referees for detailed reading and providing feedback that helped improve the paper considerably. Parts of this paper grew out of a working paper called "Automatic Backward Filtering

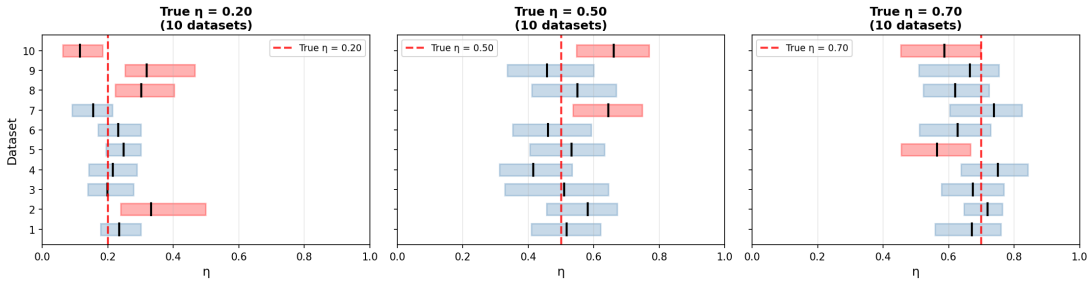

Figure 9: 95% credibility intervals for the coupling parameter η across simulation runs. Each panel shows results for different true parameter values ($\eta = 0.2, 0.5, 0.7$), with horizontal boxes representing the credibility intervals for each of the 10 independent data sets. The black vertical lines within each box indicate posterior means; the red dashed lines mark the true parameter values. Blue boxes indicate intervals that contain the true value, while red boxes represent cases where the true value falls outside the 95% credible interval.

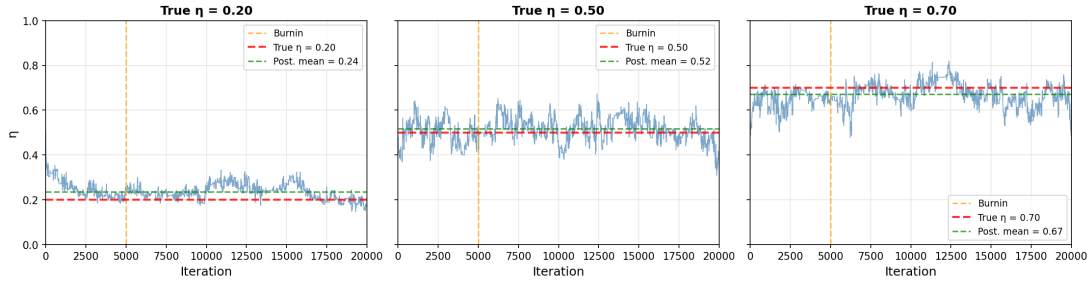

Figure 10: Representative MCMC trace plots for each true η value, showing convergence of the sampler. Each panel displays the full trajectory of 20,000 iterations with a burn-in period of 5,000 iterations (marked by orange dashed line). The red dashed line indicates the true parameter value, and the green dashed line shows the posterior mean.

Forward Guiding for Markov processes and graphical models" written by the first author jointly with Moritz Schauer, whose contributions are acknowledged.

The second author is supported by a research grant (VIL40582) from VILLUM FONDEN and the Novo Nordisk Foundation grants NNF18OC0052000, NNF24OC0093490, NNF24OC0089608.

Appendix A. Auxiliary Results on Exponential Change of Measure

Theorem 31 (Corollary 3.3 of Chapter 2 in Ethier and Kurtz (1986)) *Let U and Y be real-valued, right-continuous, $\mathcal{F}_t$-adapted processes on the probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t,\, t \geq 0\}, \mathbb{P})$. Suppose that for each t, $\inf_{s\leq t} U_s > 0$. Then*

$$M_1(t) = U_t - \int_0^t Y_s \,\mathrm{d}s$$

is an $\mathcal{F}_t$-local martingale if and only if

$$M_2(t) = U_t \exp\left(-\int_0^t \frac{Y_s}{U_s}\,\mathrm{d}s\right)$$

is an $\mathcal{F}_t$-local martingale

For a function f in the domain of the infinitesimal generator $\mathcal{L}$, we have that

$$M_t = f(X_t) - f(X_0) - \int_0^t (\mathcal{L}f)(X_s)\,\mathrm{d}s$$

is an $\mathcal{F}_t$-martingale (Dynkin's martingale). We can extend this to the space-time process: for f in the domain of $\mathcal{A} = \mathcal{L} + \partial_t$ the process

$$\bar{M}_t = f(t, X_t) - f(0, X_0) - \int_0^t (\mathcal{A}f)(s, X_s)\,\mathrm{d}s$$

is an $\mathcal{F}_t$-martingale.

Corollary 32 *Suppose X takes values in a metric space E and $g\colon [0,T]\times E \to \mathbb{R}$. If $\inf_{s\leq t} g(s, X_s) > 0$, then*

$$g(t, X_t)\exp\left(-\int_0^t \frac{(\mathcal{A}g)(s, X_s)}{g(s, X_s)}\,\mathrm{d}s\right)$$

is an $\mathcal{F}_t$-local martingale

Proof Apply Theorem 31 with $U_s = g(s, X_s)$ and $Y_s = (\mathcal{A}g)(s, X_s)$. ■

Proposition 33 (Proposition 3.2 in Palmowski and Rolski (2002)) *Suppose g is a positive function that is in the domain of $\mathcal{A}$. Then g is a good function if either of the following conditions holds:*

- *$g \in \mathbf{B}(E)$ and $(\mathcal{A}g)/g \in \mathbf{B}(E)$;*
- *$g, \mathcal{A}g \in \mathbf{B}(E)$ and $\inf_{t,x} g_t(x) > 0$.*

Appendix B. Radon-Nikodym Derivative on a Tree with Continuous Edges

Consider a directed tree consisting exclusively of continuous edges. Denote the process evolving on the tree by X and its law on $(\Omega, \mathcal{F})$ by $\mathbb{P}$. Expectation under $\mathbb{P}$ is denoted by $\mathbb{E}$. Let the depth of vertex t be the number of edges from the root vertex r to vertex t and denote this by $|t|$. Any edge $(\mathrm{pa}(t), t)$ can be uniquely indexed by t. Denote the length of such an edge by τ_t and set $X^t := \{X^t_u, 0 \leq u \leq \tau_t\}$. Let $\mathcal{T}_0 = \{r\}$ and let $\mathcal{T}_n = \{t \in \mathcal{T} \colon |t| \leq n\}$ denote the set of edges with depth less equal than n. Introduce the "depth"-filtration

$$\mathcal{F}_0 = \{\emptyset, \Omega\} \quad \text{and for } n \geq 1 \quad \mathcal{F}_n = \sigma\left(X^t_u \colon t \in \mathcal{T}_n, 0 \leq u \leq \tau_t\right).$$

Note that for $t \in \mathcal{T}_n \setminus \{r\}$, $X^t_0 = X^{\mathrm{pa}(t)}_{\tau_{\mathrm{pa}(t)}} \in \mathcal{F}_{n-1}$.

Let $\mathbb{P}^t_x$ denote the law of the process on edge t, started at x. Assume the change of measure from $\mathbb{P}^t_x$ to $\mathbb{Q}^t_x$ defined by $\mathbb{Q}^t_x(A) = \mathbb{E}[\mathbf{1}_A Z^t_x]$. For a subtree of level n, define the Radon-Nikodym derivate D_n by

$$D_n = \prod_{k=1}^{n} \prod_{t \in \mathcal{T}_n \setminus \mathcal{T}_{n-1}} Z^t_{X^t_0}.$$

Each factor $Z^t_{X^t_0}$ is $\mathcal{F}_t$-measurable, hence D_n is $\mathcal{F}_n$-measurable. We have

$$\begin{aligned}
\mathbb{E}[D_n \mid \mathcal{F}_{n-1}] &= D_{n-1}\mathbb{E}\left[\prod_{t \in \mathcal{T}_n \setminus \mathcal{T}_{n-1}} Z^t_{X^t_0} \mid \mathcal{F}_{n-1}\right] \\
&= D_{n-1}\mathbb{E}\left[\prod_{\{t \colon |t|=n-1\}} \prod_{s \in \mathrm{ch}(t)} Z^s_{X^s_0} \mid \mathcal{F}_{n-1}\right] \\
&= D_{n-1} \prod_{\{t \colon |t|=n-1\}} \mathbb{E}\left[\prod_{s \in \mathrm{ch}(t)} Z^s_{X^s_0} \mid \mathcal{F}_{n-1}\right].
\end{aligned}$$

The term in brackets equals 1. This can be seen from

$$\mathbb{E}\left[\prod_{s \in \mathrm{ch}(t)} Z^s_{X^s_0} \mid \mathcal{F}_{n-1}\right] = \mathbb{E}\left[\prod_{s \in \mathrm{ch}(t)} Z^s_{X^s_0} \mid X^t_{\tau_t}\right] = \prod_{s \in \mathrm{ch}(t)} \mathbb{E}\left[Z^s_{X^t_{\tau_t}} \mid X^t_{\tau_t}\right] = 1.$$

Here, the first equality follows from the Markov property, the second equality follows from the process evolving conditionally independent when it branches and the final equality follows from Z^s_x being a Radon-Nikodym derivative of two probability measures. Therefore, we have shown that $\{D_n, \mathcal{F}_n\}$ is a mean-one martingale. Define for $A \in \mathcal{F}_n$

$$\mathbb{Q}_n(A) = \mathbb{E}[\mathbf{1}_A D_n].$$

We can verify projective consistency: for $m < n$ and $A \in \mathcal{F}_m$

$$\mathbb{Q}_n(A) = \mathbb{E}[\mathbf{1}_A D_n] = \mathbb{E}\left[\mathbb{E}[\mathbf{1}_A D_n \mid \mathcal{F}_m]\right] = \mathbb{E}[\mathbf{1}_A D_m] = \mathbb{Q}_m(A).$$

By the Kolmogorov extension theorem there exists a unique measure $\mathbb{Q}$ on $\sigma\left(\bigcup_n \mathcal{F}_n\right)$ such that its restriction to $\mathcal{F}_n$ coincides with $\mathbb{Q}_n$.

Next, we apply this result with $\mathbb{Q}$ and $\mathbb{P}$ the laws of the conditioned process ($\mathbb{P}^h$) and guided process ($\mathbb{P}^g$) on a branch respectively. In this case

$$Z_{t,x} = \frac{g(0,x)}{h(0,x)} \frac{h(\tau_t, X_{\tau_t})}{g(\tau_t, X_{\tau_t})} \exp\left(\int_0^{\tau_t} \frac{\mathcal{A}g}{g}(u, X_u)\,\mathrm{d}u\right).$$

Therefore, by cancellation of terms similar to Equation (14), the likelihood ratio on the tree is given by

$$D_n = \frac{g(r,x_r)}{h(r,x_r)} \exp\left(\sum_{s\in\mathcal{S}} \int_0^{\tau_s} \frac{\mathcal{A}g}{g}(u, X_u)\,\mathrm{d}u\right) \prod_{v\in\mathcal{V}} \frac{h_{\mathrm{pa}(v),v}(X_{\mathrm{pa}(v)})}{g_{\mathrm{pa}(v),v}(X_{\mathrm{pa}(v)})}.$$

Appendix C. Algorithm for Numerical Results of Section 9.1

Assume a directed tree with inner edges that are all continuous edges. Denote the collection of all such edges by $\mathcal{E}$. On edge e, assume the continuous-time process evolving on it is parametrised by the parameter θ and that there exists a map $\mathcal{F}_\theta$ and random process Z^e such that $X^{\circ,e} = (X^\circ_u,\, u \in [0,\tau_e]) = \mathcal{F}_\theta(x_0, Z^e)$, assuming $X^{\circ,e}_0 = x_0$. In the following, we assume the value of the root vertex, denoted by x_r is known. Let $\mathcal{X}^\circ = (X^{\circ,e},\, e\in\mathcal{E})$, $\mathcal{Z} = (Z^e,\, e\in\mathcal{E})$ and write (with abuse of notation) $\mathcal{X}^\circ = \mathcal{F}_\theta(x_r, \mathcal{Z})$. Let

$$\Psi(\mathcal{X}^\circ;\theta) = g_r(x_r;\theta) \exp\left(\sum_{e\in\mathcal{E}} \int_0^{\tau_e} \frac{\mathcal{A}g}{g}(u, X^\circ_u)\,\mathrm{d}u\right). \tag{47}$$

Algorithm 1 defines a Markov chain that has the posterior distribution of $(\theta, \mathcal{Z})$ as invariant distribution. It is an MCMC algorithms that updates θ and $\mathcal{Z}$ from their conditional distributions.

Let Π denote the distribution of $\mathcal{Z}$ and $Q(\mathcal{Z},\,\mathrm{d}\mathcal{Z}^\circ)$ the Markov kernel that samples $\mathcal{Z}^\circ$ conditional on $\mathcal{Z}$. Define

$$w(\mathcal{Z},\mathcal{Z}^\circ) = \frac{\Pi(\,\mathrm{d}\mathcal{Z}^\circ)Q(\mathcal{Z}^\circ,\,\mathrm{d}\mathcal{Z})}{\Pi(\,\mathrm{d}\mathcal{Z})Q(\mathcal{Z},\,\mathrm{d}\mathcal{Z}^\circ)}.$$

Let $\overline{\Pi}$ denote the distribution of θ and $\overline{Q}(\theta,\,\mathrm{d}\theta^\circ)$ the Markov kernel that samples θ° conditional on θ. Define

$$\bar{w}(\theta,\theta^\circ) = \frac{\overline{\Pi}(\,\mathrm{d}\theta^\circ)\overline{Q}(\theta^\circ,\,\mathrm{d}\theta)}{\overline{\Pi}(\,\mathrm{d}\theta)\overline{Q}(\theta,\,\mathrm{d}\theta^\circ)}.$$

Finally, let N denote the total number of iterations that the chain is simulated.

C.1 Special Case: Stochastic Differential Equation

If $X^{\circ,e}$ solves (31), Z^e is the driving Wiener process of the stochastic differential equation on edge e. Here, we assume a strong solution to the SDE exists. In that case, one choice for Q is the preconditioned Crank-Nicolson scheme, that proposes Z°_e conditional on Z^e as follows

$$Z^{\circ,e} := \lambda Z^e + \sqrt{1-\lambda^2}\bar{W}^e, \tag{48}$$

Algorithm 1 MCMC-implementation of BFFG with parameter estimation.

Initialize: Sample $(\mathcal{Z}, \theta) \sim \Pi \otimes \overline{\Pi}$. Backward filter on the tree with θ. Set $\mathcal{X} = \mathcal{F}_\theta(x_r, \mathcal{Z})$.
for $i = 0$ to N **do**
    Update path
    Sample $\mathcal{Z}^\circ \sim Q(\mathcal{Z}, \cdot)$. Set $\mathcal{X}^\circ = \mathcal{F}_\theta(x_r, \mathcal{Z}^\circ)$. Accept $\mathcal{Z}^\circ$ with probability $A \wedge 1$ where

$$A = \frac{\Psi(\mathcal{X}^\circ; \theta)}{\Psi(\mathcal{X}; \theta)} w(\mathcal{Z}, \mathcal{Z}^\circ),$$

    If accepted, set $\mathcal{Z} := \mathcal{Z}^\circ$ and $\mathcal{X} := \mathcal{X}^\circ$.
    Update parameter θ
    Sample θ° from $\overline{Q}(\theta, \cdot)$.
    Backward filter on the tree with θ°.
    Set $\mathcal{X}^\circ = \mathcal{F}_{\theta^\circ}(x_r, \mathcal{Z})$.
    Accept θ° with probability $\bar{A} \wedge 1$, where

$$\bar{A} = \frac{\Psi(\mathcal{X}^\circ; \theta^\circ)}{\Psi(\mathcal{X}; \theta)} \bar{w}(\theta, \theta^\circ).$$

    If accepted, set $\theta := \theta^\circ$ and $\mathcal{X} := \mathcal{X}^\circ$.
end for

where W^e is a Wiener process on edge e that is independent of Z^e and $\lambda \in [0, 1)$ is a tuning parameter. For this choice $w(\mathcal{Z}, \mathcal{Z}^\circ) = 1$.

References

Luca Ambrogioni, Kate Lin, Emily Fertig, Sharad Vikram, Max Hinne, Dave Moore, and Marcel van Gerven. Automatic structured variational inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130, pages 676–684. PMLR, 2021.

Noémie Bardel and François Desbouvries. Exact Bayesian prediction in a class of Markov-switching models. *Methodol. Comput. Appl. Probab.*, 14(1):125–134, 2012. ISSN 1387-5841. doi: 10.1007/s11009-010-9189-4. URL `https://doi.org/10.1007/s11009-010-9189-4`.

Richard F Bass. *Stochastic processes*, volume 33. Cambridge University Press, 2011.

Denis Belomestny, Shota Gugushvili, Moritz Schauer, and Peter Spreij. Nonparametric Bayesian inference for Gamma-type Lévy subordinators. *Communications in Mathematical Sciences*, 17(3):781–816, 2019. doi: 10.4310/CMS.2019.v17.n3.a8. URL `http://dx.doi.org/10.4310/CMS.2019.v17.n3.a8`.

Mark Briers, Arnaud Doucet, and Simon Maskell. Smoothing algorithms for state–space models. *Annals of the Institute of Statistical Mathematics*, 62(1):61, 2010.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer, New York, 2005. ISBN 9780387402642.

Chris K. Carter and Robert Kohn. On gibbs sampling for state space models. *Biometrika*, 81(3):541–553, 1994. doi: 10.1093/biomet/81.3.541. URL https://academic.oup.com/biomet/article-abstract/81/3/541/257082.

Raphaël Chetrite and Hugo Touchette. Nonequilibrium markov processes conditioned on large deviations. In *Annales Henri Poincaré*, volume 16, pages 2005–2057. Springer, 2015.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer series in statistics. Springer International Publishing, 2020. ISBN 9783030478452. URL https://books.google.nl/books?id=ZZEAEAAAQBAJ.

Nicolas Chopin, Andras Fulop, Jeremy Heng, and Alexandre H. Thiery. Computational doob h-transforms for online filtering of discretely observed diffusions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5904–5923. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chopin23a.html.

Kenneth C. Chou, Alan S. Willsky, and Albert Benveniste. Multiscale recursive estimation, data fusion, and regularization. *IEEE Trans. Automat. Control*, 39(3):464–478, 1994. ISSN 0018-9286. doi: 10.1109/9.280746. URL https://doi.org/10.1109/9.280746.

Marc Corstanje and Frank van der Meulen. Guided simulation of conditioned chemical reaction networks. *Statistical Inference for Stochastic Processes*, 28(8), 2025. doi: 10.1007/s11203-025-09326-9. URL https://link.springer.com/article/10.1007/s11203-025-09326-9.

Bernard Delyon and Ying Hu. Simulation of conditioned diffusion and application to parameter estimation. *Stochastic Processes and their Applications*, 116(11):1660 – 1675, 2006. ISSN 0304-4149. doi: https://doi.org/10.1016/j.spa.2006.04.004. URL http://www.sciencedirect.com/science/article/pii/S0304414906000469.

François Desbouvries, Jean Lecomte, and Wojciech Pieczynski. Kalman filtering in pairwise Markov trees. *Signal Processing*, 86:1049–1054, 2006.

Arnaud Doucet and Anthony Lee. Sequential Monte Carlo methods. *Handbook of Graphical Models*, pages 165–189, 2018.

Stewart N. Ethier and Thomas G. Kurtz, editors. *Markov Processes*. John Wiley & Sons, Inc., 1986. doi: 10.1002/9780470316658. URL https://doi.org/10.1002/9780470316658.

Joseph Felsenstein. Evolutionary trees from dna sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6), 1981. doi: https://doi.org/10.1007/BF01734359.

Joseph Felsenstein. Phylogenies and the Comparative Method. *The American Naturalist*, 125(1):1–15, January 1985. ISSN 0003-0147. doi: 10.1086/284325. URL https://www.journals.uchicago.edu/doi/abs/10.1086/284325.

Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15(2):183–202, 1994. doi: 10.1111/j.1467-9892.1994.tb00184.x.

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273, 1997. doi: 10.1023/A:1007425814087.

Pieralberto Guarniero, Adam M. Johansen, and Anthony Lee. The iterated auxiliary particle filter. *Journal of the American Statistical Association*, 112(520):1636–1647, 2017. doi: 10.1080/01621459.2016.1222291. URL https://doi.org/10.1080/01621459.2016.1222291.

Gabriel W Hassler, Andrew F Magee, Zhenyu Zhang, Guy Baele, Philippe Lemey, Xiang Ji, Mathieu Fourment, and Marc A Suchard. Data integration in bayesian phylogenetics. *Annual Review of Statistics and Its Application*, 10:353–377, 2023.

Jeremy Heng, Adrian N. Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904 – 2929, 2020. doi: 10.1214/19-AOS1914. URL https://doi.org/10.1214/19-AOS1914.

Michael I. Jordan. Graphical models. *Statist. Sci.*, 19(1):140–155, 02 2004. doi: 10.1214/088342304000000026. URL https://doi.org/10.1214/088342304000000026.

Nianqiao Ju, Jeremy Heng, and Pierre E. Jacob. Sequential monte carlo algorithms for agent-based models of disease transmission. *arXiv preprint arXiv:2101.12156*, 2021. URL https://arxiv.org/abs/2101.12156.

Matthias Katzfuss. A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112(517):201–214, 2017.

Zico Kolter and Tommi Jaakkola. Approximate inference in additive factorial hmms with application to energy disaggregation. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 22 of *Proceedings of Machine Learning Research*, pages 1472–1482, La Palma, Canary Islands, 2012. URL https://proceedings.mlr.press/v22/zico12.html.

Dieterich Lawson, Allan Raventós, Andrew Warrington, and Scott Linderman. Sixo: Smoothing inference with twisted objectives. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38844–38858. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/fddc79681b2df2734c01444f9bc2a17e-Paper-Conference.pdf.

Dieterich Lawson, Michael Li, and Scott Linderman. NAS-X: Neural adaptive smoothing via twisting. In Amir Globerson, Moritz Hardt, Sergey Levine, and Kate Saenko, editors, *Advances in Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=A9mHph8GJk.

Thomas M. Liggett. *Interacting particle systems*. Classics in Mathematics. Springer-Verlag, Berlin, 2005. ISBN 3-540-22617-6. doi: 10.1007/b138374. URL https://doi.org/10.1007/b138374. Reprint of the 1985 original.

Thomas M. Liggett. *Continuous Time Markov Processes: An Introduction*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN 9780821884195. URL https://books.google.nl/books?id=ClW_cKYSON0C.

Fredrik Lindsten, Adam M. Johansen, Christian A. Naesseth, Bonnie Kirkpatrick, Thomas B. Schön, John A. D. Aston, and Alexandre Bouchard-Côté. Divide-and-conquer with sequential monte carlo. *Journal of Computational and Graphical Statistics*, 26(2): 445–458, 2016. doi: 10.1080/10618600.2016.1237363.

Fredrik Lindsten, Jouni Helske, and Matti Vihola. Graphical model inference: Sequential monte carlo meets deterministic approximations. *Advances in Neural Information Processing Systems*, 31, 2018.

Marcin Mider, Moritz Schauer, and Frank Van der Meulen. Continuous-discrete smoothing of diffusions. *Electronic Journal of Statistics*, 15(2):4295–4342, 2021.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 362–369. Morgan Kaufmann, 2001.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive computation and machine learning)*. Massachusetts Institute of Technology, 2012.

Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

Brooks Paige and Frank Wood. Inference networks for sequential monte carlo in graphical models. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 3040–3049. PMLR, 2016.

Zbigniew Palmowski and Tomasz Rolski. A technique for exponential change of measure for Markov processes. *Bernoulli*, 8(6):767–785, 2002.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann series in representation and reasoning. Elsevier Science, 1988a. ISBN 9781558604797. URL https://books.google.co.uk/books?id=AvNID7LyMusC.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988b.

Thorben Pieper-Sethmacher, Daniele Avitabile, and Frank van der Meulen. Guided filtering and smoothing for infinite-dimensional diffusions, 2025. URL https://arxiv.org/abs/2507.06786.

Lorenzo Rimella and Nick Whiteley. Exploiting locality in high-dimensional factorial hidden markov models. *Journal of Machine Learning Research*, 23(4):1–34, 2022. URL `https://jmlr.org/papers/v23/19-267.html`. Submitted Apr 2019; Revised Oct 2021; Published Jan 2022.

Gareth O Roberts and Osnat Stramer. On inference for partially observed nonlinear diffusion models using the metropolis–hastings algorithm. *Biometrika*, 88(3):603–621, 2001.

Fredrik Ronquist. Bayesian inference of character evolution. *Trends in ecology & evolution*, 19(9):475–481, 2004.

Simo Särkkä and Tommi Sottinen. Application of Girsanov theorem to particle filtering of discretely observed continuous-time non-linear systems. *Bayesian Anal.*, 3(3):555–584, 2008. ISSN 1936-0975,1931-6690. doi: 10.1214/08-BA322. URL `https://doi.org/10.1214/08-BA322`.

Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*, volume 17 of *Institute of Mathematical Statistics Textbooks*. Cambridge University Press, 2 edition, 2023. ISBN 9781108926645. doi: 10.1017/9781108917407.

Moritz Schauer, Frank van der Meulen, and Harry van Zanten. Guided proposals for simulating multi-dimensional diffusion bridges. *Bernoulli*, 23(4A):2917–2950, 2017. doi: 10.3150/16-bej833. URL `https://doi.org/10.3150/16-bej833`.

Moritz Schauer, Frank van der Meulen, and Andi Q. Wang. Compositionality in algorithms for smoothing. *arXiv preprint arXiv:2303.13865*, 2025.

Stefan Sommer, Gefan Yang, and Elizabeth Louise Baker. Stochastics of shapes and Kunita flows. *arXiv preprint arXiv.2512.11676*, December 2025. doi: 10.48550/arXiv.2512.11676.

Marnus Stoltz, Boris Baeumer, Remco Bouckaert, Colin Fox, Gordon Hiscott, and David Bryant. Bayesian inference of species trees using diffusion models. *Systematic Biology*, 70(1):145–161, 2021.

Sofia Stroustrup, Morten Pedersen, Frank van der Meulen, Stefan Sommer, and Rasmus Nielsen. Stochastic phylogenetic models of shape. *submitted*, 2025.

Aldo V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):297–312, 1988.

Nick Whiteley and Anthony Lee. Twisted particle filters. *Annals of Statistics*, 42(1):115–141, 2014. doi: 10.1214/13-AOS1167.

Darren J Wilkinson and Stephen KH Yeung. Conditional simulation from highly structured gaussian systems, with application to blocking-mcmc for the bayesian analysis of very large linear models. *Statistics and Computing*, 12(3):287–300, 2002.

Zhenyu Zhang, Akihiko Nishimura, Paul Bastide, Xiang Ji, Rebecca P Payne, Philip Goulder, Philippe Lemey, and Marc A Suchard. Large-scale inference of correlation among mixed-type biological traits with phylogenetic multivariate probit models. 2021.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *URL https://arxiv.org/abs/2404.17546*, 2024.