Introduction to Automatic Backward Filtering Forward Guiding[1]

[1]

Frank van der Meulen
Delft University of Technology
The Netherlands)

February 16, 2022

Contents

In this document I aim to give an informal treatment of automatic Backward Filtering Forward Guiding, a general algorithm for conditional sampling from a Markov process on a directed acyclic graph. I'll show that the underlying ideas can be understood with a basic background in probability and statistics. The more technical treatment is the paper [2], which I will abbreviate to ABFFG. I specifically assume some background knowledge on likelihood based inference and Bayesian statistics. Section 7 is more demanding: it assumes your are familiar with continuous-time stochastic processes constructed from their infinitesimal generator (see for instance the book by Liggett [3]).

Clearly, all work discussed here is the result of research carried out over the past decade together with various collaborators, most importantly **Moritz Schauer** *(Chalmers University of Technology and University of Gothenburg, Sweden). Section 8 is based on joint work with* **Marcin Mider** *(Trium Analysis Online GmbH, Germany) and* **Frank Schäfer** *(University of Basel, Switzerland) as well.*

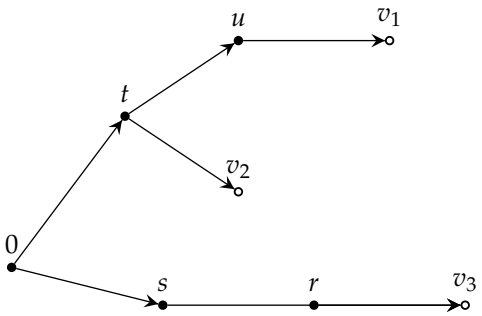
[2] Frank van der Meulen and Moritz Schauer. Automatic backward filtering forward guiding for markov processes and graphical models, 2021

[3] T.M. Liggett. *Continuous Time Markov Processes: An Introduction*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN 9780821884195

Markov processes, and in particular state-space models, are among the most popular probabilistic constructions to model uncertainty in time-evolving data. The statistical problem consists of extracting information about the process using observations from it. For simple settings [4], it is known how to "solve" the statistical problem and the associated methods haven been implemented in mainstream engineering packages such as Matlab. I will start off in Section 1 from the setting of state-space models, as I believe there is some chance of familiarity, which will ease digesting later generalisations. As we will see in Section 5, once the state-space case is well understood, some of these generalisations are almost straightforward.

[4] Most notably linear Gaussian systems, where Kalman filtering has been central for over half a century

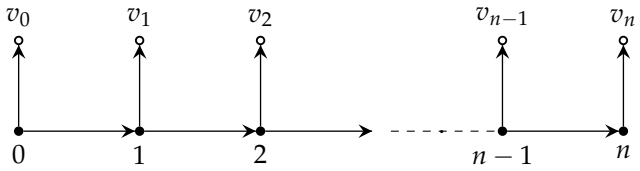
HOWEVER, BEFORE GOING THERE, I'll discuss a visualisation of the general case I aim to deal with. Consider a stochastic process on the tree



Here, the root-vertex is depicted by 0. Along each edge the process evolves according to either one step of a discrete-time Markov chain or a time-span of a continuous-time Markov process. At vertices 0 and t the process splits independently conditional on the values at 0 and t respectively. Observations are at the leaf-vertices v_1, v_2 and v_3. This setting encompasses state-space models (popular for example in signal-processing) and phylogenetic tree models arising in evolutionary biology. The statistical problem I consider consists of inferring the values of the process at the non-leaf vertices (i.e. 0, s, r, t, u). Moreover, if the forward evolution over the edges depends on a parameter θ, we may be interested in estimating θ as well.

1 *Likelihood computation for a state-space model*

Recall a Markov process is a (time-evolving) memoryless process. This means that given the present state, the past state is irrelevant for its forward evolution. A state-space model can be depicted by the following diagram



Here both black dotted dots and big open dots represent vertices of a graph. At each vertex resides a random quantity, which is either observed (open dot), or latent/non-observed (black filled dot). The arrow describes the probabilistic evolution over an edge connecting two vertices. The arrows connecting the black dots constitute a graphical model for a latent (unobserved) Markov process. If x_s denotes the random quantity at vertex s, then the probability density of "moving" from x_s to x_t is denoted by $p(x_t \mid x_s)$, this is an instance of Bayesian notation [5]. As we number the black vertices by $0, 1, \ldots, n$, we have

$$p(x_0, \ldots, x_n) = p(x_0) \prod_{i=1}^{n} p(x_i \mid x_{i-1}),$$

which follows from the Markov property. Each observation, denoted by v_i, depends only on x_i and it follows from the graphical structure that

$$p(v_0, \ldots, v_n \mid x_0, \ldots, x_n) = \prod_{i=0}^{n} p(v_i \mid x_i).$$

Combining the previous two displayed formulas gives

$$\begin{aligned} p(v_0, \ldots, v_n) &= \int p(v_0, \ldots, v_n \mid x_0, \ldots, x_n) p(x_0, \ldots x_n) \, \mathrm{d}x_0 \cdots \mathrm{d}x_n \\ &= \int \left(p(x_0) \prod_{i=1}^{n} p(x_i \mid x_{i-1}) \right) \left(\prod_{i=0}^{n} p(v_i \mid x_i) \right) \mathrm{d}x_0 \cdots \mathrm{d}x_n. \end{aligned} \tag{1}$$

If the densities "p" appearing here depend on an unknown parameter θ, then we can just add this as a subscript everywhere and we obtain a first result: the likelihood for θ based on the observations $\mathcal{V}_n := \{v_0, \ldots, v_n\}$ equals [6]

$$L(\theta; \mathcal{V}_n) = p_\theta(v_0, v_1, \ldots, v_n).$$

Likelihood based inference then appears straightforward from here; depending on your preference for either maximum likelihood or Bayesian inference, "all" is there.[7] Hence, if all we care about is inferring θ and (1) can easily be evaluated, we're good. The issue is of course that (1) requires evaluation of an n-fold integral, which makes it kind of a beast.

One example when the likelihood can be evaluated in closed form is the linear Gaussian state-space model where[8]

[5] If f_Y denotes the density of the random quantity Y, then in fact we are talking about the mapping $y \mapsto f_Y(y)$. Bayesian notation means we simply write $p(y)$ here, omitting the subscript. This comes very handy at times, but one should be careful about $p(y^2)$, which is to be interpreted as $f_{Y^2}(y^2)$. Later on I will denote the Markov kernel connecting vertices s and t by $\kappa_{s \to t}$, rather than $p(x_t \mid x_s)$.

[6] This is really a definition, the likelihood *is* simply defined as the joint density of all observations. It is given a special name when viewed as a function of the parameter θ for fixed observations, rather than the other way around. Note that as a function of θ, the likelihood is just a nonnegative function: it is *not* a density; it even need not be integrable.

[7] Maximum likelihood means you determine $\operatorname{argmax}_{\theta \in \Theta} L(\theta; \mathcal{V}_n)$, Θ denoting the parameter set, and one can numerically carry out the optimisation. Bayesian inference additionally requires specification of a prior on θ and subsequently the likelihood and prior can be fed into a probabilistic programming language to produce samples from the posterior. Well known examples include `STAN` and `Turing`.

[8] For this model, the Kalman filter provides the basis for numerically efficient evaluation of the likelihood.

$$\begin{aligned} v_i \mid x_i &\sim N(L_i v_i, \Sigma_i) \\ x_i \mid x_{i-1} &\sim N(Bx_{i-1} + \beta, \Gamma_i) \end{aligned} \tag{2}$$

This tractability is lost if the second equation would for example read as

$$x_i \mid x_{i-1} \sim N(b(x_{i-1}), \Gamma_i), \tag{3}$$

with $x \mapsto b(x)$ a nonlinear map. If any of the distributions in (2) would be non Gaussian, then the calculation would also break down.

Besides parameter estimation we may also be interested in recovering the latent states $\boldsymbol{x} := (x_0, \ldots, x_n)$. For example, when $v_0, v_1, \ldots, v_n$ is a noisy version of an underlying signal $x_0, x_1, \ldots, x_n$, or when the observations v_i only measure part of the signal x_i. I will take the Bayesian point of view here, which means that I view $\prod_{i=0}^n p(v_i \mid x_i)$ as the likelihood (it comes from the observation equation in the state-space model) and $p(x_0)\prod_{i=1}^n p(x_i \mid x_{i-1})$ as the prior density of $\boldsymbol{x}$ (it comes from the state equation in the state-space model). Then, we wish to find the posterior density

$$p^\star(x_0, \ldots, x_n) := p(x_0, \ldots, x_n \mid v_0, \ldots, v_n) = \frac{p(x_0)\prod_{i=1}^n p(x_i \mid x_{i-1}) \times \prod_{i=0}^n p(v_i \mid x_i)}{p(x_0)\int \prod_{i=1}^n p(x_i \mid x_{i-1}) \times \prod_{i=0}^n p(v_i \mid x_i)\,\mathrm{d}x_i}.$$

Note that $p(x_0)$ cancels out. The great thing about Markov Chain Monte Carlo methods is that we don't need to evaluate the denominator. So if there is no unknown parameter, this looks good.

Finally, think about the actual setting we often encounter in practice: the parameter is unknown and we wish to infer both $\boldsymbol{x}$ and θ. What to do? And yes, keep in mind that later I wish to extend to the setting where the arrows on the edges correspond to evolving a continuous-time Markov process for some time interval.

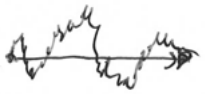


Figure 1: Transition over an edge according to a continuous-time Markov process with "diffusion"-behaviour.

2 *Backward Information Filter (BIF)*

As said, evaluation of (1) is not trivial (in fact, impossible for most models). To deal with this problem, the first thing to notice is that there is an efficient recursive way to compute it. This may remind you of dynamic programming, what I explain here is a simple version of the product-sum algorithm which is well explained in Chapter 8 of [9] for example. The idea is to compute the h-function [10]

$$h(x_i) = p(v_i, \ldots, v_n \mid x_i). \tag{4}$$

For $i = n$ this is simple: $h(x_n) = p(v_n \mid x_n)$.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007. ISBN 978-0-387-31073-2

[10] The terminology h-function is non-standard. We borrow it from the much related concept of Doob's h-transform.

Now note the following recursive relation

$$\begin{aligned} p(v_n, v_{n-1} \mid x_{n-1}) &= \int p(v_n, v_{n-1}, x_n \mid x_{n-1}) \,\mathrm{d}x_n \\ &= \int p(v_n, v_{n-1} \mid x_n, x_{n-1}) p(x_n \mid x_{n-1}) \,\mathrm{d}x_n \\ &= p(v_{n-1} \mid x_{n-1}) \int p(v_n \mid x_n) p(x_n \mid x_{n-1}) \,\mathrm{d}x_n. \end{aligned}$$

Denoting the left-hand-side by $h(x_{n-1})$ this reads

$$h(x_{n-1}) = p(v_{n-1} \mid x_{n-1}) \int h(x_n) p(x_n \mid x_{n-1}) \,\mathrm{d}x_n. \tag{5}$$

[11] The BIF can be applied more generally on a directed tree and, with some adaptation, also on a Directed Acyclic Graph (DAG).

The "n" can in fact be replaced by i and this recursion is known as the Backward Information Filter (BIF).[11]

$$\underbrace{p(v_n \mid x_n)}_{h(x_n)} \longrightarrow \underbrace{p(v_n, v_{n-1} \mid x_{n-1})}_{h(x_{n-1})} \longrightarrow \cdots \longrightarrow \underbrace{p(v_n, \ldots, v_0 \mid x_0)}_{h(x_0)}$$

The notation I use here is rather informal [12]. Equation (5) can be viewed as follows: at time $n-1$ there are two children vertices: the observation at time $n-1$ and the vertex corresponding to x_n. The leaf vertex gives as contribution $p(v_{n-1} \mid x_{n-1})$ while the vertex for x_n gives contribution $\int h(x_n) p(x_n \mid x_{n-1}) \,\mathrm{d}x_n$. Further ahead we will call the latter the pullback of h along $p(x_n \mid x_{n-1})$. Finally, both child contributions are multiplied to arrive at (5).

[12] Here, Bayesian notation starts to break-down, also as I apply it to h, so $h(x_n)$ is in fact $h_n(x_n)$ and similarly $h(x_{n-1})$ is $h_{n-1}(x_{n-1})$.

The terminology "Backward Information Filter" is perhaps only partially appropriate. It is an algorithm with steps running backwards in time taking the data ("information") into account, so calling it "Backward Information" seems appropriate. "Filter" may be a bit confusing, because commonly the filtering density of state of x_i (say) is defined by $p(x_i \mid v_0, \ldots v_i)$. The BIF is about computing $p(v_i, \ldots, v_n \mid x_i)$ though.

Now suppose h_i has been computed (suppose we can actually do this for now). Define

$$p^\star(x_i \mid x_{i-1}) = \frac{p(x_i \mid x_{i-1}) h(x_i)}{\int p(x_i \mid x_{i-1}) h(x_i) \,\mathrm{d}x_i}. \tag{6}$$

What is this density reflecting? Assume at time $i-1$ you know x_{i-1} but can also peak into the future and see $v_i, \ldots, v_n$ (this is the case: these are part of the observed data). Then $p^\star(x_i \mid x_{i-1})$ is the density of moving to x_i in view of this information. The $\star$ reminds us of conditioning on $v_i, \ldots, v_n$. Plugging the parameter θ back into the notation, and assuming prior distribution $p(\theta)$ for the parameter, we can sample from $\theta, \boldsymbol{x} \mid \mathcal{V}_n$ by the following iterative scheme[13]

- sample $\boldsymbol{x} \mid \theta, \mathcal{V}_n$; the "target" density being proportional to $\prod_{i=0}^n p^\star_\theta(x_i \mid x_{i-1})$;

Note that $p^\star$ is obtained by a change of measure on p using h. This transform is known as Doob's h-transform.

[13] This is the Gibbs sampler, in this setting also known as data-augmentation. The algorithm requires initialisation of θ or, if the first step consists of sampling θ, $\boldsymbol{x}$.

- sample $\theta \mid \boldsymbol{x}, \mathcal{V}_n$; the "target" density being proportional to $p(\theta)\prod_{i=0}^{n} p^\star_\theta(x_i \mid x_{i-1})$.

Here, for $i = 0$, $p(x_0 \mid x_{-1})$ is simply meant to be $p(x_0)$, simplifying notation.

WHAT DID WE OBTAIN SO FAR? We recursively compute h as in (4) and derived a two-step sampling procedure to sample from the joint distribution of hidden states $\boldsymbol{x}$ and parameter θ. All of this works, provided we can actually compute h.

3 *Forward guiding*

There are few cases where h can be computed in closed form, the easy cases include

1. the discrete setting, where $x_i, v_i \in E$ and E can be represented by the set of labels $E = \{1, \dots, R\}$;
2. the linear Gaussian setting, where $x_i \mid x_{i-1} \sim N(B_i x_{i-1} + \beta_i, \Gamma_i)$ and $v_i \mid x_i \sim N(L_i x_i, \Sigma_i)$.

Now imagine $p(x_i \mid x_{i-1}) \approx \tilde{p}(x_i \mid x_{i-1})$ and $p(v_i \mid x_i) \approx \tilde{p}(v_i \mid x_i)$, where $\tilde{p}$ falls in one of the two enumerated settings. An initial thought could be: "Ok, let's use the approximation $\tilde{p}$ then, with a bit of luck this is not too bad.". In fact, we can (and should) do better. [14] What we rather propose to do, is performing the BIF with $\tilde{p}$, yielding maps $\tilde{h}$ (this is tractable, by choice of $\tilde{p}$) and defining

$$p^\circ(x_i \mid x_{i-1}) = \frac{p(x_i \mid x_{i-1})\tilde{h}(x_i)}{\int p(x_i \mid x_{i-1})\tilde{h}(x_i)\,\mathrm{d}x_i}. \tag{7}$$

Note that this resembles the definition of $p^\star$ in (6). Whereas h in (6) ensures correct conditioning, $\tilde{h}$ in (7) ensures guiding to take the observations into account. [15] Note that p is still in the expression for p°!

[14] This is an important point which I have often seen misunderstood. As an example, consider the state-space model where the state evolves according to (3). By linearisation, we may be able to find (B, β) in (2) which would then define $\tilde{p}$.

[15] Put differently, p° is obtained using Doob-h-transform with $\tilde{h}$, just like $p^\star$ is obtained with h.

THE REASON THAT THIS IS USEFUL lies in the fact that we can compute the likelihood ratio between the star and circ densities. Clearly, if we assume x_0 to be known

$$\frac{p^\star(x_1, \dots, x_n)}{p^\circ(x_1, \dots, x_n)} = \prod_{i=1}^{n} \frac{h(x_i)}{\tilde{h}(x_i)} \frac{\int p(x_i \mid x_{i-1})\tilde{h}(x_i)\,\mathrm{d}x_i}{\int p(x_i \mid x_{i-1})h(x_i)\,\mathrm{d}x_i}.$$

Using the recursive relation (5) this can be simplified to

$$\frac{p^\star(x_1, \dots, x_n)}{p^\circ(x_1, \dots, x_n)} = \frac{\tilde{h}(x_0)}{h(x_0)} \prod_{i=0}^{n} \frac{p(v_i \mid x_i)}{\tilde{p}(v_i \mid x_i)}. \tag{8}$$

This is a key identity to what follows. Whereas $p^\star$ is intractable (because h is), we have p° at our disposal and impose the assumption that sampling from $\boldsymbol{x}$ under p° is tractable. The above formula tells us how to correct for the discrepancy between $p^\star$ and p°. In fact, everywhere we encounter $p^\star(x_1, \ldots, x_n)$ we can safely replace it with $p^\circ(x_1, \ldots, x_n)$ times the product on the right-hand-side of (8). The only intractable term $h(x_0)$ fortunately turns out to cancel in Markov Chain Monte Carlo methods!

4 *Backward Filtering Forward Guiding*

This section can be a short: we just combine what we have derived. That is, we use $\tilde{p}$ for the BIF to get $\tilde{h}$. This defines p° via (7). Then we can forward sample $\boldsymbol{x}$ under p° to guide $\boldsymbol{x}$ to the observations and compute a correction by (8). So what we do is backward filtering, followed by forward guiding. [16]

Classical cases, where actually the forward model corresponds to the discrete or linear Gaussian setting, are special cases. In such settings we don't need to use an approximate $\tilde{h}$ (however, it still can be computationally advantegeous). If we don't use the approximation, then $p^\star = p^\circ$, and the right-hand-side of (8) will be 1. Then, if we only care about parameter estimation, there is no need to do forward guiding: the BIF will result in a closed form expression for the likelihood which may subsequently be used in either a Bayesian analysis or for maximum-likelihood estimation. However, as in a general setting it will be impossible to compute the BIF filter efficiently, performing the BIF for a simpler process will be a way out. Let me stress again that the guided process still contains the (possibly complicated) forward transition density p. Note that due to the Markov property we only need to be able to sample one step forward according to p°, the BIF-backward recursion is inherently more difficult.

[16] A natural question is whether one could also do forward filtering, backward guiding. While in certain cases this is indeed possible, forward guiding is more practical, because is shares structure with the unconditional forward dynamcis.

One way to view the combined procedure of backward filtering forward guiding is as follows: we compute $\tilde{h}(x_n)$ and put it on a pile. Next, we compute $\tilde{h}(x_{n-1})$ and put it on top of that pile. We continue until we get $\tilde{h}(x_0)$. In the end, we have a pile with (from top to bottom)

$$[\tilde{h}(x_0), \tilde{h}(x_1), \ldots, \tilde{h}(x_n)]. \tag{9}$$

Next to it, we place the pile with (again from top to bottom)

$$[p(x_0), p(x_1 \mid x_0), \ldots, p(x_n \mid x_{n-1}]. \tag{10}$$

Then we simply pick the top element from both piles, combine the contributions from each pile into p° and simulate from it to get x_0°.

Repeating this procedure until the pile is empty results in the samples

$$[x_0^\circ, x_1^\circ, \ldots, x_n^\circ].$$

5 *Extension to a tree and general DAG*

The state-space model considered so far has a very simple topology. In what follows, I'll generalise the approach to a tree topology. This means that at any vertex, there can be multiple leaf vertices, and that any vertex may "duplicate" followed by conditionally independent evolutions over both duplicates. To explain the setting, consider the typical setting depicted in Figure 2.

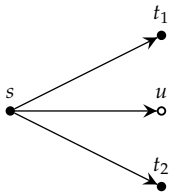


Figure 2: Part of a tree with parent vertex s.

The vertex labeled s has three children: t_1, t_2 and u. As before we assume the Markov property, meaning that x_{t_1}, x_{t_2} and u are independent, conditional on x_s. We then have

$$h(x_s) = p(u \mid x_s) \prod_{i=1}^{2} \int p(x_{t_i} \mid x_s) h(x_{t_i}) \, \mathrm{d}x_{t_i}.$$

This can be viewed as each of the children, t_1, t_2 and u, sending a message to their common parent vertex. After vertex s has received messages from all of its children, the messages get multiplied. Indeed, BFFG can be interpreted as a message passing algorithm with messages (for this specific example)

$$\begin{aligned} m_{t_i}(x_s) &= \int p(x_{t_i} \mid x_s) h(x_{t_i}) \, \mathrm{d}x_{t_i}, \qquad i = 1, 2, \\ m_u(x_s) &= p(u \mid x_s). \end{aligned}$$

For a general Directed Acyclic Graph (DAG) there is one additional ingredient needed. The difficulty lies in the fact that a vertex can have multiple parent vertices.[17] For such a vertex we need to "split" h in the backward filtering step to its parents. Hence, as an example, we need to decompose $h(x_1, x_2)$ into $h_1(x_1)$ and $h_2(x_2)$. A tractable approach for doing this is in the ABFFG-paper. This is a bit of an opposite operation compared to fusion, though whereas fusion is exact, an approximation is made when doing a split operation in backward filtering. Nevertheless, as explained in the paper, we can devise an algorithm for sampling from the exact smoothing distribution.

[17] As an example, suppose at a vertex we compute the sum of the values at the parent indices.

6 *Compositionality*

Reading code not written yourself is often hard. Even pseudo-code, as sometimes seen in scientific papers I find usually hard to digest.

Especially in filtering, there appear so many indices! Older versions of the ABFFG manuscript also contained those indices, but in fact we can get rid of those. Key is compositionality: assembling the bigger, more complex algorithm by piecing together smaller, simpler pieces. That is exactly what we can do here: first we formalise our notation a bit. We assume that each forward transition corresponds to a Markov kernel $\kappa(x, \mathrm{d}y)$ [18]

For a Markov kernel we have the following two linear operators. For a bounded measurable function h we define the pullback by

$$(\kappa h)(x) = \int_E \kappa(x, \mathrm{d}y) h(y). \tag{11}$$

To give this a probabilistic interpretation, note that $(\kappa h)(x) = \mathbb{E}[h(X_{n+1}) \mid X_n = x]$. As an example, if the state space is finite (say $E = \{1, \dots, R\}$), then the preceding display reads $(\kappa h)(x) = \sum_{y=1}^{R} \kappa(x, y) h(y)$ and $\kappa(x, y)$ is the one-step transition probability to go from state x to y.

For a measure μ we define

$$(\mu\kappa)(\mathrm{d}y) = \int \mu(\mathrm{d}x) \kappa(x, \mathrm{d}y).$$

This is the pushforward of the measure μ. The interpretation is as follows: suppose at time n we sample x_n from the measure μ and subsequently evolve the Markov chain from x_n to x_{n+1} according to the Markov kernel κ. Then $\mu\kappa$ is the distribution of x_{n+1} [19]. In the finite-state setting we have that for $x \in E$, $(\mu\kappa)(x) = \sum_{x=1}^{R} \mu(x) \kappa(x, y)$.

Recall in the description of BFFG the analogy of having the two piles (9) and (10). This analogy can be formalised as viewing one step of BFFG as applying a backward map together with a forward map.

Recall that in each step of the BIF we take a function $h(x_i)$ and do two things:

- we put it on top of the "h-pile" (9);
- we compute $h(x_{i-1})$ as in (5) (note that part of this computation is indeed the pullback as defined in (11)).

We interpret dropping $h(x_i)$ on the h-pile as sending a message m which is used later in forward sampling (guiding). Viewed a bit more abstractly, each step in the BIF takes a function h, produces a new function h' and sends a message m. Once all backward steps of the BIF have been completed, we have the pile of messages and we can combine it with the pile of forward evolutions, alike (10). More formally we will shortly define a forward map for this.

Before entering the definitions, let's look at a small visualisation: We start from the right, where h serves as input to a backward map $\mathcal{B}$. This map produces $h' := \kappa h$, but also a message m, which is used

[18] This means that for a (measurable) set B, the mapping $x \mapsto \kappa(x, B)$ is measurable and that for fixed x, $B \mapsto \kappa(x, B)$ ia a probability measure. The idea is that if at time i the process is at x, then the state at time $i + 1$ is drawn from the measure $\kappa(x, \cdot)$. It the state-space is finite, this simply boils down to sampling the state from a (finite) probability vector.

[19] We first compute the joint distribution of (x_n, x_{n+1}) and then integrate out x_n.

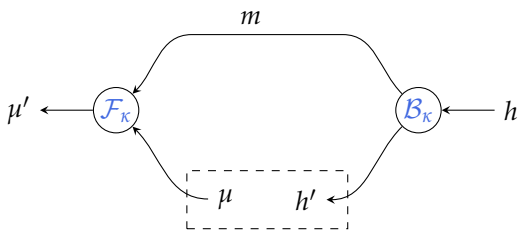


Figure 3: One step of BFFG. Read from right to left. For Definition 3, $\mathcal{B}_\kappa$ needs to be replaced by $\mathcal{B}_{\tilde{\kappa}}$.

in the forward map $\mathcal{F}$. The latter pushes forward the measure μ using the message m.

Definition 1. For a Markov kernel κ and function h define the backward map $\mathcal{B}_\kappa$ by [20]

$$\mathcal{B}_\kappa(h) = (m, \kappa h), \quad \text{where} \quad m(x,y) = \frac{h(y)}{(\kappa h)(x)}. \tag{12}$$

This map returns both the pullback κh and an appropriate message m for the map $\mathcal{F}_\kappa$ specified in the following definition.

Definition 2. For a Markov kernel κ, message m (as defined in (12)) and measure μ define the forward map $\mathcal{F}_\kappa$ by

$$\mathcal{F}_\kappa(m,\mu) = \nu, \quad \nu(\mathrm{d}y) = \int m(x,y)\mu(\mathrm{d}x)\kappa(x,\mathrm{d}y). \tag{13}$$

[21]

In case κh is intractable or computationally demanding, we propose to replace the kernels κ in the backward map by simpler kernels $\tilde{\kappa}$. [22]

If μ is a probability measure and $\mathcal{B}_\kappa$ sends the message m, then $\mathcal{F}_\kappa(m,\mu)$ is again a probability measure. If the BIF is intractable, we replace κ in the backward map by the kernel $\tilde{\kappa}$, where $\tilde{\kappa}$ is chosen such that the BIF *is* tractable.[23] In that case $\mathcal{F}_\kappa(m,\mu)$ need not be a probability measure, even if μ is. This motivates the following definition.

Definition 3. For a guided process with backward kernel $\tilde{\kappa}$ we have

$$\mathcal{B}_{\tilde{\kappa}}(h) = (m, \tilde{\kappa}h), \quad \text{where} \quad m(x,y) = \frac{h(y)}{(\tilde{\kappa}h)(x)}.$$

If $\varpi \geq 0$ and μ is a probability measure, then

$$\mathcal{F}_\kappa(m, \varpi \cdot \mu)(\mathrm{d}y) = (\varpi w_\kappa(m,\mu)) \cdot \nu(\mathrm{d}y)$$

with the weight $w_\kappa(m,\mu)$ and probability measure ν defined by

$$\begin{aligned} \nu(\mathrm{d}y) &= w_\kappa^{-1}(m,\mu) \int \frac{h(y)}{(\tilde{\kappa}h)(x)} \mu(\mathrm{d}x)\kappa(x,\mathrm{d}y) \quad \text{and} \\ w_\kappa(m,\mu) &= \iint m(x,y)\kappa(x,\mathrm{d}y)\mu(\mathrm{d}x) = \int \frac{(\kappa h)(x)}{(\tilde{\kappa}h)(x)} \mu(\mathrm{d}x). \end{aligned} \tag{14}$$

[20] Compatibility of κ and h is implicitly assumed.

[21] Again, compatibility of κ, m and μ is implicitly assumed.

[22] This really corresponds to our earlier approximation $\tilde{p}$ for p. We will have a bit more flexibility here, as we will not require $\tilde{\kappa}$ to be a Markov kernel, but rather a kernel.

[23] The indices κ and $\tilde{\kappa}$ reflect the true forward dynamics and approximate dynamics that are used in computing the BIF respectively. Hence, $\tilde{\kappa}$ takes the role of $\tilde{p}$ used earlier in our description.

Note this definition is consistent with our previous definition of $\mathcal{F}$.

JOINT APPLICATION OF THE BACKWARD- AND FORWARD MAPS can be written as

$$F(\kappa, \tilde{\kappa}) = \langle \mathcal{F}_\kappa \mid \mathcal{B}_{\tilde{\kappa}} \rangle.$$

Two kernels κ_1 and κ_2 can be composed to $\kappa_1\kappa_2$ and applied in parallel as $\kappa_1 \otimes \kappa_2$. It turns out that

$$\begin{aligned} F(\kappa_1\kappa_2, \tilde{\kappa}_1\tilde{\kappa}_2) &= F(\kappa_1, \tilde{\kappa}_1) \cdot F(\kappa_2, \tilde{\kappa}_2) \\ F(\kappa_1 \otimes \kappa_2, \tilde{\kappa}_1 \otimes \tilde{\kappa}_2) &= F(\kappa_1, \tilde{\kappa}_1) \otimes F(\kappa_2, \tilde{\kappa}_2) \end{aligned} \tag{15}$$

I haven't told you about $\cdot$ and $\otimes$ on the right-hand-side. That is in the paper! Also be careful with interpreting $\otimes$: while we use the same symbol on the left- and right-hand-side, in the former case it is parallel application of Markov kernels but in the latter case denoting product measure.

THIS IS THE BEGINNING OF A STORY where the forward evolution of the Markovian process on the DAG is written as parallel/serial composition of Markov kernels. To each forward kernel we specify a backward kernel $\tilde{\kappa}$, which by the way need not necessarily be Markov. Then each element κ in this composition gets replaced with $F(\kappa, \tilde{\kappa})$ in ABFFG. That's it. Hence: "all" that needs to be implemented is

1. the forward and backward map;
2. the compositionality rules appearing in Equation (15).

Of course, we additionally need a dictionary which tells us in which order to compose in the forward evolution.

IF YOU ARE FAMILIAR WITH REVERSE-MODE AUTOMATIC DIFFERENTIATION (AD) you may have noted similarities. Indeed, the compositional structure here is essentially the category of optics proposed for AD.

7 *Continuous time transitions over an edge*

In many settings, the natural modelling framework is to assume that the transition over an edge is in fact the result of evolving a continuous time process over some time interval. [24] Thus suppose along an edge the transition is the result of running a continuous-time Markov process X over the interval $[0, T]$. Conditioning the process on its value at time T corresponds to a change of measure, details follow shortly. We closely follow the exposition in the paper by Palmowski and Rolski from 2002 [25], which we denote PR2002.

[24] Phylogenetics is one example, where a Brownian motion or finite-state continuous time Markov process pops up.

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

WARNING: this section is necessarily mathematically more demanding, as continuous-time Markov processes are inherently more complicated than their discrete-time counterpart.

ASSUME X_t IS MARKOV PROCESS on a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}, \mathbb{P})$ having extended generator $\mathcal{L}$ with domain $\mathcal{D}(\mathcal{L})$. [26] For a strictly positive function f define

$$E^f(t) = \frac{f(X_t)}{f(X_0)} \exp\left(-\int_0^t \frac{(\mathcal{L}f)(X_s)}{f(X_s)}\,\mathrm{d}s\right).$$

If h is such that $E^h(t)$ is a martingale, then it is called an exponential martingale and then h is called a good function. As $\mathbb{E}E^h(t) = \mathbb{E}E^h(0) = 1$ this martingale can be used to define a change of measure.

Under this change of measure, the process X_t is typically again Markovian with *nicer* properties.[27] For a probability measure $\mathbb{P}$ we denote its restriction to $\mathcal{F}_t$ by $\mathbb{P}_t$. The main result of PR2002 (Theorem 4.2) says the following: if h is a good function and $\bar{\mathbb{P}}_t$ is defined by

$$\frac{\mathrm{d}\bar{\mathbb{P}}_t}{\mathrm{d}\mathbb{P}_t} = E^h(t)$$

then under $\bar{\mathbb{P}}_t$ the process X_t is a Markov process with extended generator

$$\bar{\mathcal{L}}f = \frac{1}{h}\left[\mathcal{L}(fh) - f\mathcal{L}h\right]. \tag{16}$$

Moreover, $\mathcal{D}(\mathcal{L}) = \mathcal{D}(\bar{\mathcal{L}})$. Note that if h is harmonic, i.e. $\mathcal{L}h = 0$, then we have the simple expression $E^h(t) = h(X_t)/h(X_0)$.

HOW DO WE KNOW h IS A GOOD FUNCTION (meaning that $E^h(t)$ is a martingale)? First, if we define

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

then by Lemma 3.1 in PR2002, $\{D^f(t),\, t \geq 0\}$ is a local martingale if and only if $\{E^f(t),\, t \geq 0\}$ is a local martingale.[28] The *local* martingale can be strengthened to true martingale under certain extra conditions on h (sufficient conditions are given in Proposition 3.2 in PR2002).

NOW IT IS TIME TO APPLY THESE RESULTS. To this end, we will apply the change-of-measure to the space-time-process (t, X_t), which has infinitesimal generator $\mathcal{A} = \partial_t + \mathcal{L}$. To condition the process X on $\{X_T = x_T\}$ we take the specific choice

$$h(t, x) = p(t, x; T, x_T),$$

[26] Recall that a Markov process is (under certain technical conditions) characterised by its infinitesimal generator $\mathcal{L}_t$. i.e.

$$(\mathcal{L}_t f)(x) = \lim_{h \downarrow 0} t^{-1}\mathbb{E}\left[f(X_{t+h}) - f(X_t) \mid X_t = x\right],$$

for all f in the domain of $\mathcal{L}$ (which is part of the definition and defined by those f for which the above limit exists).

[27] The key example of "nicer" for us is that the process is conditioned on a future event.

[28] This requires $f \in \mathcal{D}(\mathcal{L})$ but additionally f needs to satisfy integrability conditions. We refer to the paper for details.

where p denotes the transition density of X, evolving from x at time t to x_T at time T. It is well known that for this choice of h we have $\mathcal{A}h = 0$, which is simply Kolmogorov's backward equation.[29] Define the measure $\mathbb{P}_t^\star$ by

[29] Put differently, $(t,x) \mapsto h(t,x)$ is space-time harmonic.

$$\frac{\mathrm{d}\mathbb{P}_t^\star}{\mathrm{d}\mathbb{P}_t}(X) = E^h(t) = \frac{h(t,X_t)}{h(0,X_0)}.$$

Using (16) we can find the extended generator under $\mathbb{P}_t^\star$ to be

$$\mathcal{L}^\star f = \frac{1}{h}\left[\mathcal{L}(fh) - f\mathcal{L}h\right], \tag{17}$$

where f depends on (t,x). Carrying out this computation in concrete examples reveals for example that

- if X_t is a diffusion process, then $X_t^\star$ is also a diffusion process with an extra term added to the drift parameter;
- if X_t is a Poisson process of constant intensity, then $X_t^\star$ is a non-homogeneous Poisson process.

What does the change of measure imply? I claim that under $\mathbb{P}^\star$ the process X is conditioned on the event $\{X_T = x_T\}$. To see this, take $t_0 < t_1 < \cdots < t_n < t < T$, assume the process is started at x_0 and consider

$$\begin{aligned}
\mathbb{E}^\star\left[g(X_{t_1},\ldots,X_{t_n},X_t)\right] &= \mathbb{E}\left[E^h(t)g(X_{t_1},\ldots,X_{t_n},X_t)\right] \\
&= \int g(x_1,\ldots,x_n,x_t)\frac{p(t,x_t;T,x_T)}{p(t_0,x_0;T,x_T)}p(t_i,x_i;t,x_t)\prod_{i=1}^{n}p(t_{i-1},x_{i-1},t_i,x_i)\,\mathrm{d}x_1\cdots\mathrm{d}x_n\,\mathrm{d}x_t \\
&= \mathbb{E}\left[g(X_{t_1},\ldots,X_{t_n},X_t)\mid X_T = x_T\right]
\end{aligned}$$

So far so good, but the problem is of course that only in very specific cases the transition densities p are known. Therefore, in general h is unknown and at first sight the preceding does not seem to be of any help. However, suppose that there is a Markov process $\tilde{X}_t$ with space-time generator $\tilde{\mathcal{A}}$ and tractable $\tilde{h}$ satisfying $\tilde{\mathcal{A}}\tilde{h} = 0$. Let the measure $\mathbb{P}_t^\circ$ be defined by

$$\frac{\mathrm{d}\mathbb{P}_t^\circ}{\mathrm{d}\mathbb{P}_t}(X) = E^{\tilde{h}}(t).$$

Therefore

$$\frac{\mathrm{d}\mathbb{P}_t^\star}{\mathrm{d}\mathbb{P}_t^\circ}(X) = \frac{h(t,X_t)}{h(0,X_0)}\frac{\tilde{h}(0,X_0)}{\tilde{h}(t,X_t)}\exp\left(\int_0^t \frac{(\mathcal{A}\tilde{h})(s,X_s)}{\tilde{h}(s,X_s)}\,\mathrm{d}s\right).$$

The term in the exponential can be simplified slightly since we have

$$\mathcal{A}\tilde{h} = (\partial t + \mathcal{L})\tilde{h} = (\mathcal{L} - \tilde{\mathcal{L}})\tilde{h}.$$

Again, using (16) we can find the extended generator under $\mathbb{P}_t^\circ$ to be

$$\mathcal{L}^\circ f = \frac{1}{h}\left[\mathcal{L}(fh) - f\mathcal{L}h\right]. \tag{18}$$

LET'S SUMMARISE SOME OF OUR FINDINGS. By a change of measure from $\mathbb{P}$ to $\mathbb{P}^\star$ the Markov process X_t can be conditioned. Moreover, the expression for $\mathcal{L}^\star$ in (17) can be used to identify the dynamics of the process under $\mathbb{P}^\star$.

Unfortunately, the h-function required for $\mathbb{P}^\star$ is usually intractable and hence we take an approximation $\tilde{h}$ to h. This $\tilde{h}$ can be used in the same way for an exponential change of measure to define $\mathbb{P}^\circ$. The process X_t under $\mathbb{P}^\circ$ is tractable and its dynamics can be identified using (18). Finally, the likelihood ratio $\mathrm{d}\mathbb{P}^\star / \mathrm{d}\mathbb{P}^\circ$ is known in closed form. This quantity can be used to correct for the discrepancy between $\mathbb{P}^\star$ and $\mathbb{P}^\circ$ in Monte-Carlo methods such as importance sampling, sequential Monte Carlo and Markov Chain Monte-Carlo. [30]

[30] Note that the annotation by $\circ$ and $\star$ is consistent with our earlier use of these symbols for transition densities.

THIS IS THE BASIC IDEA. There are definitely subtle things that need to be taken care of: most importantly, we need to assess the behaviour of the likelihood ratio as $t \uparrow T$. For certain classes of Markov processes, under certain extra conditions, it can be shown that [31]

$$\frac{\mathrm{d}\mathbb{P}^\star_T}{\mathrm{d}\mathbb{P}^\circ_T}(X) = \frac{\tilde{h}(0, X_0)}{h(0, X_0)} \exp\left(\int_0^T \frac{(\mathcal{A}\tilde{h})(s, X_s)}{\tilde{h}(s, X_s)} \, \mathrm{d}s \right).$$

[31] This is analogous to the expression in Equation (8).

Only $h(0, X_0)$ is still intractable, but as it shows up as a multiplicative constant in the denominator that turns out to be harmless.

8 *Example: Stochastic Differential Equations on a tree*

To conclude, let's consider a toy example with an SDE on a directed tree. 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 $\tilde{h}$;
2. the process X° is defined by applying Doob's h-transform using $\tilde{h}$.

This means that on segments where the process evolves as a diffusion process, the process X°, characterised by its extend generator $\mathcal{L}^\circ$ as specified in (18), is run forward. For discrete-transitions, the process evolves according to the transition densities p° as specified in (7).

What $\tilde{h}$ to use? It should be tractable and this tractability should be preserved in the backward filtering steps, starting from the leaves back to the root vertex. I'll illustrate with the setting depicted in Figure 4.

If we assume $v_i \mid x_t \sim N(L_i x_t, \Sigma_i)$, then

$$\tilde{h}_{t \to v_i}(x) = \varphi(v_i; L_i x, \Sigma_i), \qquad i = 1, 2.$$

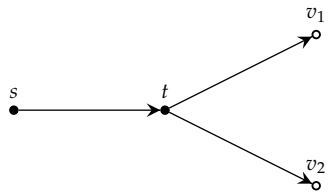


Figure 4: Part of a tree. On $s \to t$ a continuous-time Markov process evolves. Observations are at leaf-vertices v_1 and v_2.

Because of Gaussianity, we can write

$$\tilde{h}_{t\to v_i}(x) = \exp\left(-c_i + F_i' x - \frac{1}{2} x' H_i x\right) \tag{19}$$

for triplets (c_i, F_i, H_i), with c_i scalar valued, F_i vector valued and H_i matrix valued. [32] Now at vertex t we have the fusion-step yielding

$$\tilde{h}_t(x) = \prod_{i=1}^{2} \tilde{h}_{t\to v_i}(x)$$

which can be interpreted as collecting all messages at vertex t from its children. Clearly, $\tilde{h}_t$ can be represented by the triplet $(c_1 + c_2, F_1 + F_2, H_1 + H_2)$. Now suppose that the branch connecting vertices s and t represents the evolution of a continuous time process X on the time-interval $[s, t]$.[33] On this segment, we define $\tilde{h}$ by solving on $(s, t]$

$$(\partial_u + \tilde{\mathcal{L}}_u)\tilde{h} = 0, \qquad \tilde{h}(t, \cdot) = \tilde{h}_t(\cdot). \tag{20}$$

where $\tilde{\mathcal{L}}_u f(x) = \sum_i \tilde{b}_i(u, x)\partial_i f(x) + \sum_{i,j} \tilde{a}_{i,j}(u)\partial_{i,j} f(x)$. This is the infinitesimal generator of the process evolving according to the linear SDE

$$\mathrm{d}\tilde{X}_t = (B(t)\tilde{X}_t + \beta(t))\,\mathrm{d}t + \tilde{\sigma}(t)\,\mathrm{d}W_t \tag{21}$$

where $\tilde{b}(u, x) = B(u)x + \beta(u)$ and $\tilde{a} = \tilde{\sigma}\tilde{\sigma}'$. Solving the partial differential equation in (20) is known as the Cauchy problem. With the specific choice of a linear SDE the nice thing is that since $\tilde{h}_t$ is of the form (19), then for $u \in (s, t]$ we have $\tilde{h}(u, x) = \exp\left(c_u + F_u' x - \frac{1}{2} x' H_u x\right)$. Hence, the functional form of $\tilde{h}$, where it is represented by a triplet (c, F, H) is preserved. Moreover,

$$\begin{aligned}
\mathrm{d}H(u) &= \left(-B(u)'H(u) - H(u)B(u) + H(u)\tilde{a}(u)H(u)\right)\,\mathrm{d}u,\\
\mathrm{d}F(u) &= \left(-B(u)'F(u) + H(u)\tilde{a}(u)F(u) + H(u)\beta(u)\right)\,\mathrm{d}u,\\
\mathrm{d}c(u) &= \left(\beta(u)'F(u) + \frac{1}{2}F(u)'\tilde{a}(u)F(u) - \frac{1}{2}\mathrm{tr}\left(H(u)\tilde{a}(u)\right)\right)\,\mathrm{d}u.
\end{aligned} \tag{22}$$

see for instance [34], Theorem 2.5. [35] In this way, $\tilde{h}$ can be defined recursively on the whole tree, starting from the leaves all the way back towards the root. This constitutes the backwards filtering step. The main computational work consists of solving the ODEs in (22). This operation scales quadratically in the dimension of the diffusion process. Improved scaling can be obtained in case of sparsity in B, β and/or $\tilde{\sigma}$.

For the forward guiding step, we start from the root and evolve the process on "continuous-time" segments under the law $\mathbb{P}^\circ$. From (18) we can identify that X° is a diffusion process satisfying the SDE

$$\mathrm{d}X_t^\circ = (b(t, X_t^\circ) + a(t, X_t^\circ)(F(t) - H(t)X_t^\circ))\,\mathrm{d}t + \sigma(t, X_t^\circ)\,\mathrm{d}W_t,$$

[32] In fact, we have $c_i = -\log \varphi(v_i; 0, \Sigma_i)$, $F_i = v_i'\Sigma_i^{-1}L_i$ and $H_i = L_i'\Sigma_i^{-1}L_i$.

[33] There is a slight abuse of notation here, as s and t both denote a vertex and time.

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

[35] This is just backward filtering of linear SDE: a problem which has been solved decades ago.

which can easily be forward simulated using Euler-discretisation (or more sophisticated SDE-solvers).

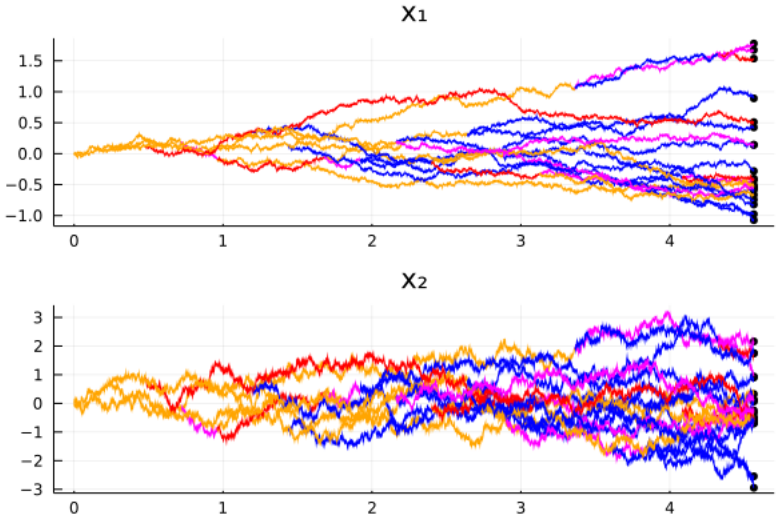
Numerical example using `MitosisStochasticDiffEq.jl`

We illustrate the methods described in here using an example of an SDE on a tree. Frank Schäfer gave a 3-minute talk about this a JuliaCon2021 `https://www.youtube.com/watch?v=rie7MTvPpIs`. The forward model is as follows: on each branch of the tree the process evolving according to the SDE [36]

$$\mathrm{d}X_t = \tanh\,.\left(\begin{bmatrix} -\theta_1 & \theta_1 \\ \theta_2 & -\theta_2 \end{bmatrix} X_t\right)\mathrm{d}t + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\mathrm{d}W_t.$$

[36] The "." appearing in the drift means that the tanh function is applied coordinatewise.

Now forward simulating from this model on given tree gives rise to the following paths:



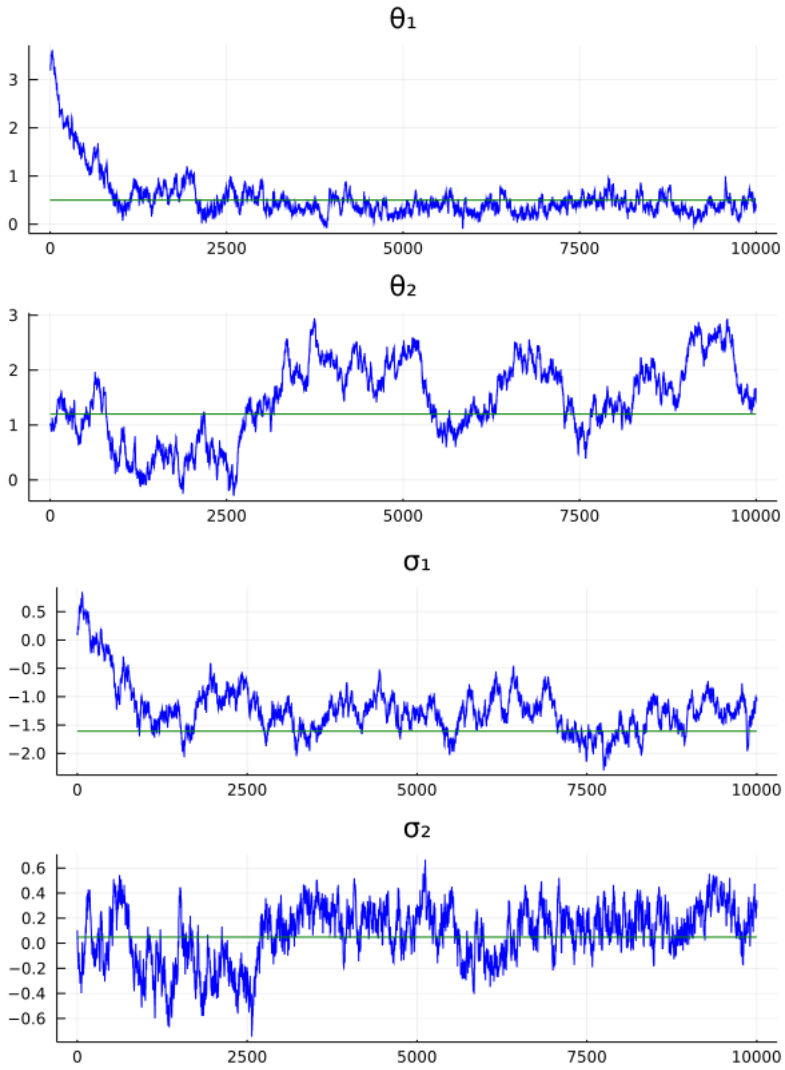
We assume, as throughout, that only the values at the leaf-vertices are observed. Assume the tree-structure itself is known. We aim to estimate the parameters $\boldsymbol{\theta} := (\theta_1, \theta_2, \sigma_1, \sigma_2)$. Note that a standard Kalman-filter cannot be used due to the nonlinearity in the drift. 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 (21), with [37]

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

[37] We choose the diffusivity of $\tilde{X}$ to match that of X. This is crucial in case the extrinsic noise level approaches zero.

[38] Here are traceplots after running the algorithm for 10_000 iterations

[38] The figures here are meant to illustrate the potential of the method. The code for producing the figures in this example is on the Github repository of the `MitosisStochasticDiffEq.jl`-package.



One thing which makes this problem mildly difficult is that there is no strong nonlinearity in the drift. If that were the case, the paper *Continuous-discrete smoothing of diffusions*[39] gives a host of methods to deal with this setting, essentially choosing the process $\tilde{X}$ in a more advanced fashion.

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

9 *Online talk*

I also tried to explain this in an online talk:

`https://www.youtube.com/watch?v=XjBO4GSc0i8`

ACKNOWLEDGEMENT: Thanks to Frank Schäfer (University of Basel) and Stefan Sommer (University of Copenhagen) for providing detailed feedback on earlier versions that helped improving this paper.

Comments are welcome.

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2007. ISBN 978-0-387-31073-2.

T.M. Liggett. *Continuous Time Markov Processes: An Introduction*. Graduate studies in mathematics. American Mathematical Society, 2010. ISBN 9780821884195.

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

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

Frank van der Meulen and Moritz Schauer. Automatic backward filtering forward guiding for markov processes and graphical models, 2021.